sensors

MDPI

Article

5G-AKA-FS: A 5G Authentication and Key Agreement Protocol for Forward Secrecy

Ilsun You [1], Gunwoo Kim [1], Seonghan Shin [2,*], Hoseok Kwon [1], Jongkil Kim [3] and Joonsang Baek [4]

[1] Department of Financial Information Security, Kookmin University, Seoul-si 02707, Republic of Korea; isyou@kookmin.ac.kr (I.Y.); gguakim22@kookmin.ac.kr (G.K.); hoseok1997@kookmin.ac.kr (H.K.)
[2] Cyber Physical Security Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan
[3] Department of Cyber Security, Ewha Womans University, Seoul-si 03760, Republic of Korea; jongkil@ewha.ac.kr
[4] School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia; baek@uow.edu.au
* Correspondence: seonghan.shin@aist.go.jp; Tel.: +81-3-3599-8001

Abstract: 5G acts as a highway enabling innovative digital transformation and the Fourth Industrial Revolution in our lives. It is undeniable that the success of such a paradigm shift hinges on robust security measures. Foremost among these is primary authentication, the initial step in securing access to 5G network environments. For the 5G primary authentication, two protocols, namely 5G Authentication and Key Agreement (5G-AKA) and Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA′), were proposed and standardized, where the former is for 3GPP devices, and the latter is for non-3GPP devices. Recent scrutiny has unveiled vulnerabilities in the 5G-AKA protocol, exposing it to security breaches, including linkability attacks. Moreover, mobile communication technologies are dramatically evolving while 3GPP has standardized Authentication and Key Management for Applications (AKMA) to reuse the credentials, generated during primary authentication, for 5G network applications. That makes it so significant for 5G-AKA to be improved to support forward secrecy as well as address security attacks. In response, several protocols have been proposed to mitigate these security challenges. In particular, they tried to strengthen security by reusing secret keys negotiated through the Elliptic Curve Integrated Encryption Scheme (ECIES) and countering linkability attacks. However, they still have encountered limitations in completing forward secrecy. Motivated by this, we propose an augmentation to 5G-AKA to achieve forward security and thwart linkability attacks (called 5G-AKA-FS). In 5G-AKA-FS, the home network (HN), instead of using its static ECIES key pair, generates a new ephemeral key pair to facilitate robust session key negotiation, truly realizing forward security. In order to thoroughly and precisely prove that 5G-AKA-FS is secure, formal security verification is performed by applying both BAN Logic and ProVerif. As a result, it is demonstrated that 5G-AKA-FS is valid. Besides, our performance comparison highlights that the communication and computation overheads are intrinsic to 5G-AKA-FS. This comprehensive analysis showcases how the protocol effectively balances between security and efficiency.

Keywords: 5G security; 5G-AKA; forward secrecy (FS); standard compatibility

Citation: You, I.; Kim, G.; Shin, S.; Kwon, H.; Kim, J.; Baek, J. 5G-AKA-FS: A 5G Authentication and Key Agreement Protocol for Forward Secrecy. *Sensors* **2024**, *24*, 159. https://doi.org/10.3390/s24010159

Academic Editors: Rafal Kozik, Michal Choras and Marek Pawlicki

Received: 5 November 2023
Revised: 15 December 2023
Accepted: 22 December 2023
Published: 27 December 2023

1. Introduction

Since the first deployment of the fifth generation (5G) mobile networks in 2019, 5G has rapidly become a mainstream mobile network worldwide. According to the Global System for Mobile Communications Association (GSMA), the 5G adoption rate reached 43.1% in the United States in Q4 2022 [1]. 5G mobile networks and telecommunication standards have been developed to meet the various demands of modern telecommunication. In particular, it is designed to support enhanced features such as Enhanced Mobile BroadBand (EMBB),

Massive Machine-Type Communications (MMTC), and Ultra-Reliable and Low-Latency Communications (URLLC) [2].

In order to ensure the seamless delivery of high-quality services to users through these three features, 5G demands heightened security compared to its predecessors in mobile networks. Therefore, the 3rd Generation Partnership Project (3GPP) consortium, which is in charge of the standardization of 5G mobile networks, has standardized an essential 5G security architecture and procedures as well as several security features [3,4]. Figure 1 depicts the 3GPP 5G security architecture.

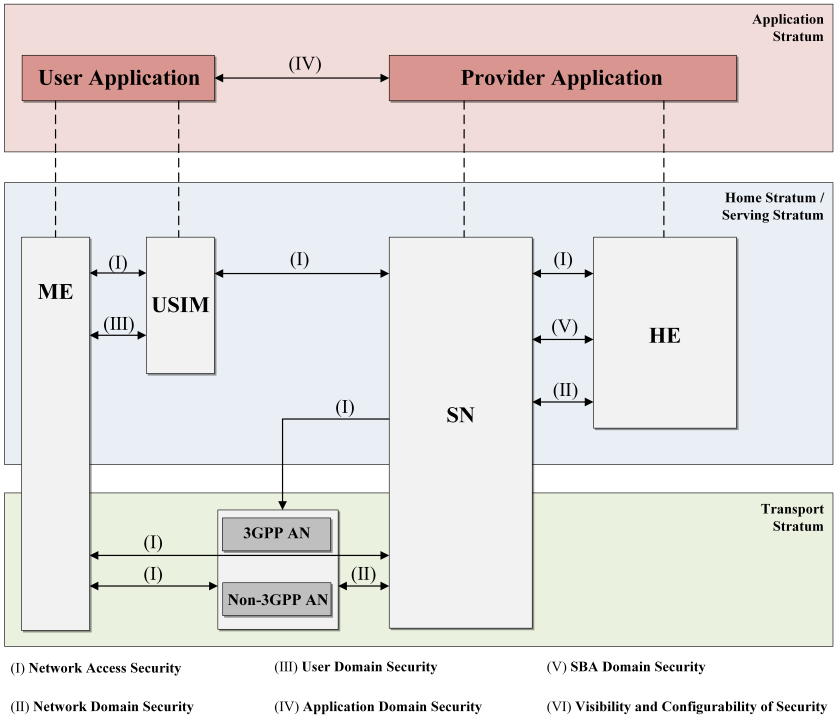

Figure 1. 3GPP 5G security architecture.

Notably, in 5G, primary authentication emerges as a cornerstone, representing the initial security checkpoint for accessing 5G network environments. For the 5G primary authentication, two protocols have been adopted as standards: 5G Authentication and Key Agreement (5G-AKA) and Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA′) [3,5]. The former caters to 3GPP devices, while the latter is tailored for non-3GPP devices.

The 5G primary authentication protocols, i.e., 5G-AKA and EAP-AKA′, allow a user equipment (UE) (e.g., a mobile phone) and a home network (HN) (e.g., the network of a service provider) to authenticate each other and exchange key materials (e.g., anchor keys) to protect entire subsequent 5G communications. Note that they have been enhanced and markedly differentiated from their previous version (i.e., Evolved Packet System Authentication and Key Agreement (EPS-AKA) [4] and Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) [6]). The most distinctive feature of those security protocols in the 5G network is using a Subscriber Concealed Identifier (SUCI) that can be explained as a Subscriber Permanent Identifier (SUPI) in the encrypted format. In the previous generation, authentication was performed by transmitting the user identification information, International Mobile Subscriber Identity (IMSI), without encryption. On the other hand, in 5G, SUPI, which is a UE's identifier, is encrypted into SUCI using a key derived through the Elliptic Curve Integrated Encryption Scheme (ECIES) before transmission to address identifier exposure [7–10]. Despite these efforts, 5G-AKA still remains vulnerable to various types of attacks [7,9–14]. Ref. [15] described vulnerabilities for 5G-AKA through formal verification and analysis. The authors

showed that there still exist attack scenarios against 5G-AKA. In addition, refs. [10,16] pointed out that privacy problems of users may occur since 5G-AKA is susceptible to linkability attacks. Furthermore, ref. [17] presented shortcomings of 5G-AKA, including the lack of support for forward secrecy, also known as perfect forward secrecy.

Meanwhile, the 5G network environments face the following new challenges:

- As the advancement of 5G technology and the proliferation of 5G network applications continue at a rapid pace, the emergence of new security threats and attacks becomes more pronounced, thereby necessitating the establishment of elevated security prerequisites.
- Thanks to the Authentication and Key Management for Applications (AKMA) [18], the credentials generated through 5G primary authentication can be reused for application authentication in the 5G network environments; that is, the master session key negotiated during the 5G primary authentication is applied to derive an application key that allows a UE to authenticate itself to AKMA-based applications smoothly and efficiently.

Evidently, the security robustness of the existing 5G primary authentications falls short of addressing the aforementioned concerns. As a result, it becomes imperative to bolster the security framework with robust public key-based measures and ensure the implementation of forward secrecy.

Regarding EAP-AKA′, there has been a standardization effort aimed at enhancing the protocol to incorporate support for forward secrecy (known as EAP-AKA′-FS) [19]. On the 5G-AKA side, there are existing works proposed to support unlinkability and forward secrecy, such as [3,16,17,20,21]. In particular, those 5G-AKA enhancements attempted to reuse the ECIES shared key, which is used to protect the UE's identifier in the initiation phase. However, in such an approach, if an adversarial security event happens in HN, forward secrecy is not guaranteed for the session key.

In this paper, we propose a 5G-AKA-Forward Secrecy (5G-AKA-FS) protocol that supports forward secrecy and unlinkability together to solve the limitations of the existing studies and maximize the efficiency of authentication in 5G networks. The proposed protocol accomplishes forward secrecy and unlinkability by introducing an additional ECIES-based ephemeral key pair generation within HN. The main contributions of this paper are summarized as follows:

- The 5G-AKA-FS protocol is designed to concurrently support forward secrecy and unlinkability.
- We analyze the latest studies proposed to improve the vulnerabilities of 5G-AKA and provide a solid comparison of security attributes between our 5G-AKA-FS and the latest studies.
- We conduct a rigorous security verification of the proposed protocol using formal verification (BAN Logic [22] and ProVerif [23]).
- Performance evaluation is thoroughly carried out by measuring overhead in terms of computation and communication.

The rest of the paper is organized as follows. Section 2 explores the existing enhancements of 5G-AKA, while Section 3 describes the preliminaries used in this paper. Moving on to Section 4, we present a detailed description of the proposed protocol, followed by the formal security analysis using BAN Logic and ProVerif in Section 5. Assessing performance, Section 6 carries out a comparative evaluation of security properties between the proposed and existing protocols, along with measured overhead. Finally, Section 7 provides the conclusion.

2. Related Works

3GPP has defined two AKA protocols, namely 5G-AKA and Extensible Authentication Protocol (EAP)-AKA′, for primary authentication in 5G networks. The purpose of the AKA protocols is to establish mutual authentication between the UE and the HN. In 2G to 4G networks, UE's identifier IMSI is publicly exposed, thus resulting in privacy issues.

However, in 5G-AKA, the subscriber identity information, known as SUPI, is encrypted into SUCI, which is then used for authentication while solving the privacy problem.

5G-AKA enhances authentication and key exchange in addition to introducing subscriber identity protection (e.g., SUCI) for privacy. However, as it is developed based on EPS-AKA, it inherits existing security vulnerabilities. Ref. [15] conducted a formal verification and analysis of the 5G-AKA protocol using the Mixed Strand Space model, revealing vulnerabilities within the protocol. Their study also presented 21 attack scenarios specific to 5G-AKA and highlighted security features not supported by 5G-AKA. Furthermore, ref. [17] identified shortcomings, such as the lack of support for forward secrecy through an analysis of 5G-AKA. In this regard, improved protocols, including [3,16,17,20,21], have been proposed to address such vulnerabilities in 5G-AKA.

SUCI-AKA, proposed by [20], aims to achieve forward secrecy for the anchor key K_{SEAF}. For this goal, when generating the master session key K_{AUSF}, the protocol reuses the shared key k_{HN}, which is exchanged through ECIES to encrypt SUPI. In more details, the sequence number SQN is replaced with k_{HN} during the generation of K_{AUSF}. Such an approach enhances the security of the anchor and sub-session keys derived from K_{AUSF} while still allowing the UE to verify if the received messages and their message authentication code (MAC) are fresh. However, SUCI-AKA can not support forward secrecy because the anchor key K_{SEAF} is compromised when the long-term key k, shared between the UE and the HN, and the HN's private key sk_{HN} are leaked.

5G-IPAKA, proposed by [17], aims to provide mutual authentication between the UE and the SN, enhanced security for the anchor key and authentication vector (AV), and key confirmation. This protocol tries to provide forward secrecy by applying the ECIES secret k_{HN} to generate the anchor key K_{SEAF}, from which sub-sessions keys are then derived. Moreover, it lets the HN send K_{SEAF} to the SN before the UE authentication. In this way, 5G-IPAKA enables the UE to authenticate the SN by verifying the SN's message authentication code computed with K_{SEAF} in addition to the HN. Similar to this, the UE is authenticated to the SN through its message authentication code. As a result, 5G-IPAKA achieves mutual authentication between the SN and the UE and between the HN and the UE while supporting key confirmation. However, if k and sk_{HN} are leaked, an attacker can reconstruct the anchor key K_{SEAF} for malicious purposes while breaking forward secrecy. Furthermore, active attacks by malicious SNs are also possible since the HN delivers K_{SEAF} to the SN without authenticating the UE. Finally, this protocol leads to compatibility issues with Subscriber Identity Modules (SIMs) by proposing a structure that deviates from the existing standard specification.

5GAKA-LCCO, proposed by [21], aims to improve high communication and computation overheads as well as address SUCI replay attacks present in 5G-IPAKA. In this protocol, the SN first creates the random number and timestamp $RAND_{SN}$ and T_{SN}, and sends to the UE these values, which are then applied to generate the key block with the long-term shared key k and the ECIES secret k_{HN}. Note that the generated key block is not only used to compute the UE's SUCI but also to authenticate the UE to the HN. Upon a receipt of the new SUCI, the HN decrypts it into the corresponding SUPI, counts on its timestamp T_{HN} to validate the received T_{SN}, and authenticates the UE. In such a way, the authentication process is optimized to have one round trip, reducing the computation and communication overhead. Also, the HN utilizes timestamps to prevent the SUCI replay and Denial-of-Service (DoS) attacks on itself while enhancing the security of the session keys by deriving K_{AUSF} from sk_{HN}, k, $RAND_{SN}$, and T_{SN}. However, it should be noted that when generating K_{AUSF}, both k and sk_{HN} are utilized. Therefore, if k and sk_{HN} are leaked, an attacker can recover K_{AUSF} and conduct subsequent malicious attacks. Furthermore, to address SUCI replay attacks, 5GAKA-LCCO introduces freshness to SUCI by utilizing T_{SN}. However, this approach requires time synchronization, which may pose challenges in situations such as roaming. Consequently, the use of T_{SN} is not desirable for mobile telecommunication scenarios. Furthermore, 5GAKA-LCCO exhibits an unconventional protocol flow compared to the 5G-AKA standard, and the differences in the *Authenticate SIM command* can lead

to compatibility issues, particularly with Legacy Universal Subscriber Identity Modules (USIMs), potentially resulting in backward compatibility problems.

5G-AKA $'$, introduced by [16], focuses on addressing linkability attacks by reusing the ECIES secret k_{HN}, which is used to protect SUPI in the initial step. In this protocol, the HN encrypts its randomly generated number *RAND* into $RAND'$ with k_{HN}, then sending to the UE the encrypted result instead of *RAND* along with the authentication token AUTN. At this point, it is worth noting that since the UE trusts the freshness of k_{HN}, it also trusts the freshness of $RAND'$. Therefore, if successfully decrypting $RAND'$ with k_{HN}, the UE can trust the freshness of its received AUTN, thereby detecting the message replay attack prior to arriving at the `Sync_Failure` while defending against the linkability attack. In spite of such a successful defense against the linkability attack, this protocol is vulnerable to active attacks by malicious SNs because it allows the HN to send K_{SEAF} to the SN without authentication to the UE. More importantly, 5G-AKA$'$ fails to achieve forward secrecy because the old anchor keys can be recovered if the long-term key k shared between the UE and the HN is leaked.

3. Preliminaries

3.1. Notations

Table 1 shows abbreviations and notations to be used throughout this paper.

Table 1. Abbreviations and notations.

	Meanings
HN	Home Network
UE (ME, SIM)	User Equipment (Mobile Equipment, Subscriber Identity Modules)
SN	Serving Network
SUPI	SUbscriber Permanent Identifier
SUCI	SUbscriber Concealed Identifier
KEM	Key Encapsulation Mechanism
DEM	Data Encapsulation Mechanism
AMF	Access Management Function
AUSF	Authentication Server Function
SEAF	SEcurity Anchor Function
k	A permanent key shared between UE and HN
K_{AUSF}	A master session key derived from 5G-AKA-FS
K_{SEAF}	An anchor key derived from 5G-AKA-FS
k_{UE}	UE's shared key established by ECIES-KEM
k_{HN}	HN's shared key established by ECIES-KEM
(PK_{HN}, sk_{HN})	HN's ECIES public-private key pair where $PK_{HN} = sk_{HN} \cdot G$
ID_{SN}	Unique identifier of SN
ID_{HN}	Unique identifier of HN
SQN_{UE}	UE's sequence number
SQN_{HN}	HN's sequence number
$RAND$	HN's challenge message
AUTH	Mutual AUTHentication
SKE	Secure Key Exchange
LUCS	Legacy USIM Compatibility Support
LBA	LinkaBility Attack
AMS	Active attack by Malicious SN
FS	Forward Secrecy

3.2. Elliptic Curve Integrated Encryption Scheme

The Elliptic Curve Integrated Encryption Scheme (ECIES) [24,25] is a well-known hybrid encryption scheme consisting of a Key Encapsulation Mechanism (KEM) and a Data Encapsulation Mechanism (DEM) where messages of arbitrary length can be encrypted. This scheme is a key component of 5G-AKA.

The ECIES-KEM has the following three algorithms:

- **KeyGen**(pp): On input of a public parameter pp, the algorithm outputs a public-private key pair (PK, sk) such that $PK = sk \cdot G$, where pp is an elliptic curve parameter standardized in `secp256r1` [26], and $G \in pp$ is a base point.
- **Encap**(PK): On input of a public key PK, the algorithm generates an ephemeral public-private key pair (R, r) such that $R = r \cdot G$, and then outputs a ciphertext $C_0 = R$ and a shared key $k_s = \mathsf{KDF}(r \cdot PK)$, where KDF is a key derivation function.
- **Decap**(sk, C_0): On input of a ciphertext C_0 and a private key sk, the algorithm outputs the shared key $k_s = \mathsf{KDF}(sk \cdot C_0)$.

The ECIES-DEM has the following two algorithms:

- **SEnc**(k_s, M): On input of a key k_s and a message M, the algorithm first parses k_s as $k_1||k_2$, computes $C_1 = \mathsf{ENC}(k_1, M)$ and $C_2 = \mathsf{MAC}(k_2, C_1)$, and then outputs (C_1, C_2), where ENC is an encryption part of a symmetric encryption scheme and MAC is a message authentication code.
- **SDec**$(k_s, (C_1, C_2))$: On input of a ciphertext (C_1, C_2) and a key k_s, the algorithm first parses k_s as $k_1||k_2$. If $C_2 \neq \mathsf{MAC}(k_2, C_1)$, it outputs $\perp$. Otherwise, the algorithm outputs $M = \mathsf{DEC}(k_1, C_1)$, where DEC is a decryption part of a symmetric encryption scheme.

4. A 5G-AKA Protocol for Forward Secrecy

In this section, we propose a 5G-AKA protocol for forward secrecy (for short, 5G-AKA-FS) that is compatible with the current 3GPP standards [3], and the proposed protocol is shown in Figure 2. Before executing the 5G-AKA-FS protocol, UE holds $(k, PK_{HN}, SUPI, SQN_{UE})$ secretly and HN stores $(k, sk_{HN}, ID_{HN}, SQN_{HN})$ secretly. Also, SN stores ID_{SN}. We denote by $\mathsf{H}_{\text{SHA-256}}$ the SHA-256 cryptographic hash function.

4.1. The Initiation Phase: Step 1

In this phase, UE sends its SUPI as an encrypted form using ECIES [24,25] with HN's public key PK_{HN}. Correspondingly, HN decrypts SUCI with its private key sk_{HN}. Upon successful completion of Step 1, we proceed to Step 2 by selecting the 5G-AKA-FS among various methods. Before choosing the authentication method, the Step 1 process unfolds as follows:

4.1.1. Step 1.1 (UE)

Step 1.1 (UE)

Inputs. With HN's public key PK_{HN}, UE executes the followings:
The Protocol:

1. Generate an ephemeral private-public key pair (r, R) such that $R = r \cdot G$
2. With PK_{HN}, compute a ciphertext $C_0 = R$ and a key $k_{UE} = \mathsf{KDF}(r \cdot PK_{HN})$
3. Parse the shared key k_{UE} as $k_1||k_2$
4. Compute $C_1 = \mathsf{ENC}(k_1, SUPI)$ and $C_2 = \mathsf{MAC}(k_2, C_1)$
5. Set $SUCI \leftarrow (C_0, C_1, C_2)$

Outputs. UE sends $(SUCI)$ to SN.

4.1.2. Step 1.2 (SN)

Step 1.2 (SN)

Inputs. SN receives $(SUCI)$ from UE.
Outputs. SN sends $(SUCI, ID_{SN})$ to HN.

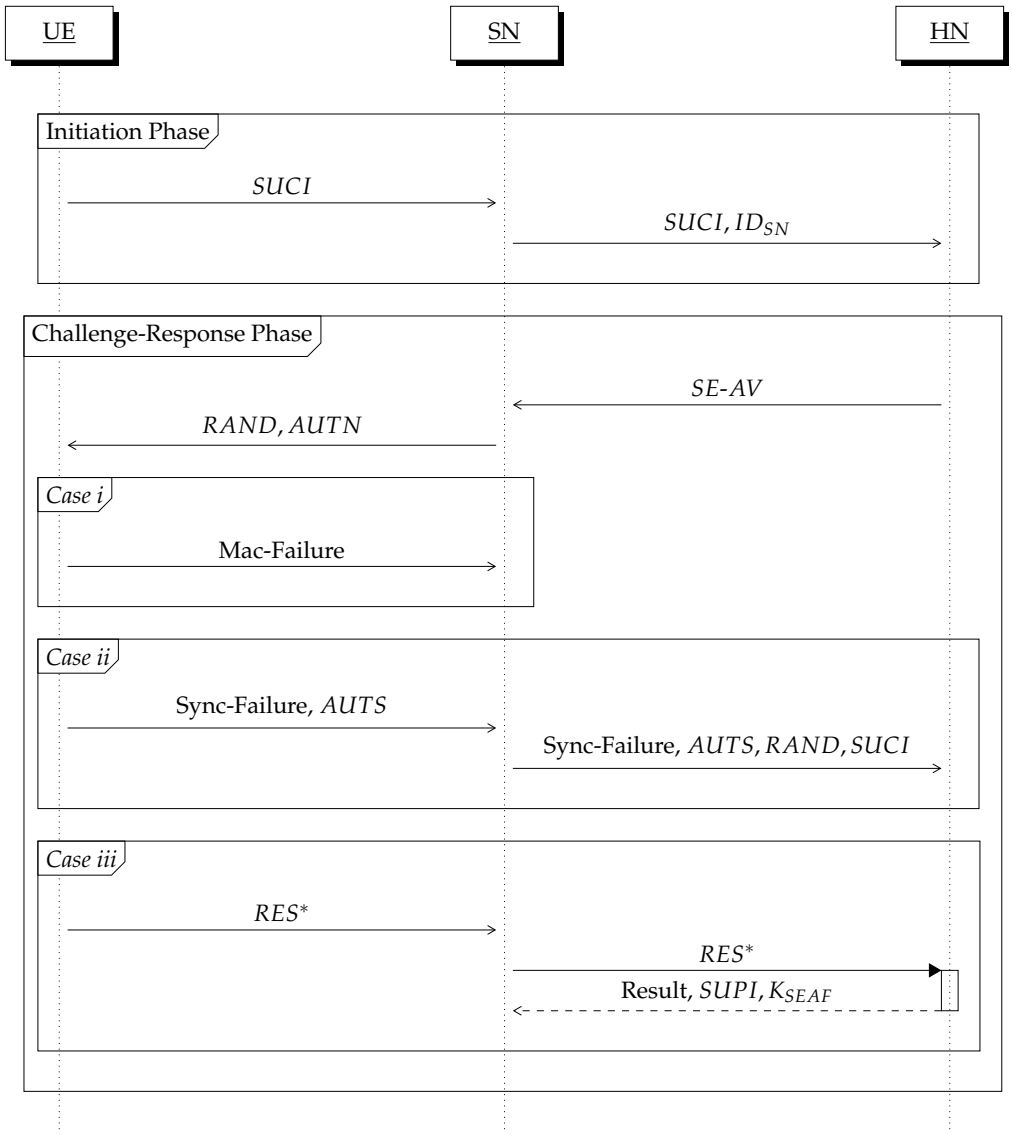

Figure 2. The 5G-AKA-FS protocol.

4.1.3. Step 1.3 (HN)

Step 1.3 (HN)

Inputs. Upon receiving $(SUCI, ID_{SN})$ from SN, HN executes the followings:
The Protocol:

1. Compute a shared key $k_{HN} = \mathsf{KDF}(sk_{HN} \cdot C_0)$
2. Parse the shared key k_{HN} as $k_1||k_2$
3. Retrieve the corresponding k and SQN_{HN} from its database

Outputs. HN outputs $\perp$ if $C_2 \neq \mathsf{MAC}(k_2, C_1)$. Otherwise, it outputs $SUPI = \mathsf{DEC}(k_1, C_1)$.

4.2. The Challenge-Response Phase: Step 2

In this phase, UE and HN authenticate each other via a challenge-response method and establish anchor keys (i.e., K_{SEAF}) together with SN. A key idea is that we set a Diffie–Hellman public key Y as a random challenge $RAND$, and a Diffie–Hellman key DHK is used as a key material for forward secrecy. This phase uses a series of HMAC-SHA-256 cryptographic key derivation functions $\mathsf{f}_1, \mathsf{f}_2, \mathsf{f}_3, \mathsf{f}_4, \mathsf{f}_5, \mathsf{f}_1^*$, and f_5^*, as specified by TS 33.501 [3] (see also Figure 3).

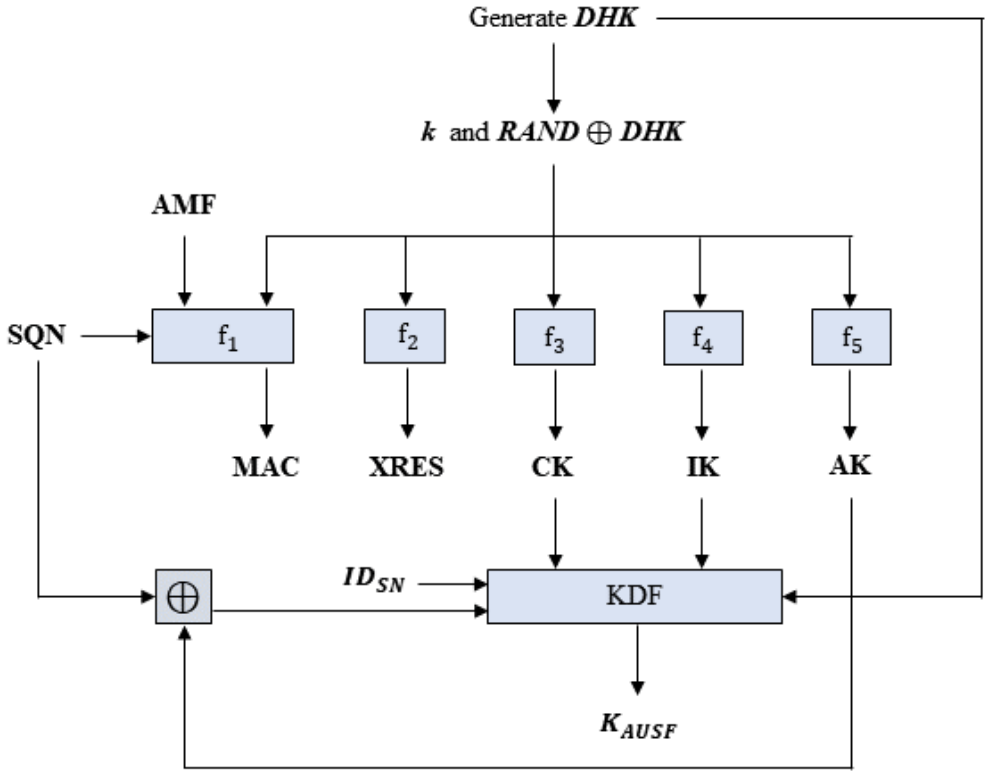

Figure 3. Computations of $MAC, XRES, CK, IK, AK$, and K_{AUSF} in 5G-AKA-FS.

4.2.1. Step 2.1 (HN)

Step 2.1 (HN)

Inputs. Using SQN_{HN}, k, and DHK, HN generates an Authentication Vector $SE\text{-}AV = (RAND, AUTN, HXRES^*)$ as follows:
The Protocol:

1. Generate an ephemeral private-public key pair (y, Y) such that $Y = y \cdot G$
2. Compute a Diffie–Hellman key $DHK = y \cdot C_0$
3. Set $RAND \leftarrow Y$ as a challenge (The 128-bit randomness of $RAND$ is guaranteed since y is randomly chosen from the 128-bit key space.)
4. Compute $MAC \leftarrow \mathsf{f}_1(k, SQN_{HN}, AMF, RAND \oplus DHK)$ and an anonymous key $AK \leftarrow \mathsf{f}_5(k, RAND \oplus DHK)$
5. Set $AUTN \leftarrow (AK \oplus SQN_{HN}, AMF, MAC)$
6. Compute $CK \leftarrow \mathsf{f}_3(k, RAND \oplus DHK)$ and $IK \leftarrow \mathsf{f}_4(k, RAND \oplus DHK)$
7. Compute expected responses $XRES \leftarrow \mathsf{f}_2(k, RAND \oplus DHK)$, $XRES^* \leftarrow \mathsf{KDF}(CK||IK, ID_{SN}||RAND||XRES)$, and $HXRES^* \leftarrow \mathsf{LEFT}(128, \mathsf{H}_{\text{SHA-256}}(RAND||XRES^*))$
8. Derive $K_{AUSF} \leftarrow \mathsf{KDF}(CK||IK, ID_{SN}||AK \oplus SQN_{HN}||DHK)$ and $K_{SEAF} \leftarrow \mathsf{KDF}(K_{AUSF}, ID_{SN})$
9. Increase SQN_{HN} by 1 (i.e., $SQN_{HN} \leftarrow SQN_{HN} + 1$)
10. Set $HE\text{-}AV \leftarrow (RAND, AUTN, XRES^*, K_{AUSF})$ and $SE\text{-}AV \leftarrow (RAND, AUTN, HXRES^*)$

Outputs. HN sends $(SE\text{-}AV)$ to SN.

4.2.2. Step 2.2 (SN)

Step 2.2 (SN)

Inputs. SN receives $(SE\text{-}AV)$ from HN and stores $(RAND, HXRES^*)$.
Outputs. SN sends $(RAND, AUTN)$ to UE.

4.2.3. Step 2.3 (UE)

Step 2.3 (UE)

Inputs. Upon receiving $(RAND, AUTN)$ from SN, UE executes the following steps:

1. In the UE, the ME forwards the received $RAND$ and $AUTN$ to the SIM
2. The SIM computes a Diffie–Hellman key $DHK = r \cdot RAND$
3. Run the SIM card command $\texttt{AUTHENTICATE}(RAND, DHK, AUTN)$
4. Compute $AK \leftarrow \mathsf{f}_5(k, RAND \oplus DHK)$
5. Parse $AUTN$ as $(CONC, AMF, MAC)$
6. De-conceal $SQN_{HN} \leftarrow AK \oplus CONC$
7. Check $\mathsf{f}_1(k, SQN_{HN}, AMF, RAND \oplus DHK) = MAC$
   - If this check does not pass, the SIM card returns $\perp$ and then UE sends a failure message `Mac_Failure` to SN (see ***Case i***)
   - Otherwise, proceed to the next step
8. Check $SQN_{UE} < SQN_{HN} < SQN_{UE} + \Delta$ (The first condition $SQN_{UE} < SQN_{HN}$ ensures the freshness of $(RAND, AUTN)$. Also, the second condition $SQN_{HN} < SQN_{UE} + \Delta$, which is optional in the non-normative Annex C of TS 33.102 [27], prevents a wrap-around of SQN_{UE}. For example, if Δ is too small (i.e., $\Delta = 2$), an attacker can make a synchronization failure by sending $SUCI$ computed by the attacker with a fake SUPI. After this attack, the honest UE and HN can no longer authenticate each other. In TS 33.102 [27], a recommended value of Δ is 2^{28} so as to decrease the synchronization failure rate.)
   - If this check does not pass, the SIM card computes $MAC^* \leftarrow \mathsf{f}_1^*(k, SQN_{UE}, AMF, RAND \oplus DHK)$ and returns $AUTS \leftarrow (AK^* \oplus SQN_{UE}, AMF, MAC^*)$ where $AK^* \leftarrow \mathsf{f}_5^*(k, RAND \oplus DHK)$. Then, UE re-synchronizes with HN by sending a failure message `Sync_Failure` and $AUTS$ to SN (see ***Case ii***)
   - Otherwise, proceed to the next step
9. Set $SQN_{UE} \leftarrow SQN_{HN}$
10. Compute $CK \leftarrow \mathsf{f}_3(k, RAND \oplus DHK)$ and $IK \leftarrow \mathsf{f}_4(k, RAND \oplus DHK)$
11. Compute $RES \leftarrow \mathsf{f}_2(k, RAND \oplus DHK)$ and SIM returns RES, CK, and IK to ME
12. The ME calculates $RES^* \leftarrow \mathsf{KDF}(CK||IK, ID_{SN}||RAND||RES)$ and derives $K_{AUSF} \leftarrow \mathsf{KDF}(CK||IK, ID_{SN}||CONC||\ DHK)$ and $K_{SEAF} \leftarrow \mathsf{KDF}(K_{AUSF}, ID_{SN})$
13. The UE returns (K_{SEAF}, RES^*)

Outputs. The UE stores K_{SEAF} and sends RES^* to SN (see ***Case iii***)

Case i **(The SIM card returns $\perp$):** The UE sends a failure message `Mac_Failure` to SN.

Case ii **(The SIM card returns $AUTS$):** The UE re-synchronizes with HN by sending a failure message `Sync_Failure` and $AUTS$ to SN. Upon receiving $(\texttt{Sync\_Failure}, AUTS)$, SN sends $(\texttt{Sync\_Failure}, AUTS, RAND, SUCI)$ to HN. Then, HN parses $AUTS$ as $(AK^* \oplus SQN_{UE}, AMF, MAC^*)$, and de-conceals $SQN_{UE} \leftarrow AK^* \oplus SQN_{UE} \oplus \mathsf{f}_5^*(k, RAND \oplus DHK)$. Next, HN checks its authenticity by comparing $MAC^* = \mathsf{f}_1^*(k, SQN_{UE}, AMF, RAND \oplus DHK)$. If the check holds, HN re-sets SQN_{HN} by $SQN_{UE} + 1$ (i.e., $SQN_{HN} \leftarrow SQN_{UE} + 1$).

Case iii **(The SIM card returns (K_{SEAF}, RES^*)):** The UE stores K_{SEAF} and sends (RES^*) to SN. Upon receiving (RES^*), SN computes a hashed value $HRES^* \leftarrow \mathsf{LEFT}(128, \mathsf{H}_{\text{SHA-256}}(RAND||RES^*))$ and checks its validity by comparing $HRES^* = HXRES^*$. If $HRES^* = HXRES^*$, SN forwards (RES^*) to HN. Next, HN authenticates UE by comparing $RES^* = XRES^*$. If $RES^* = XRES^*$, HN sends its result and $(SUPI, K_{SEAF})$ to SN. The SN continues the protocol only if both checks hold, and aborts the protocol otherwise. When all checks pass, UE and SN communicate with session keys derived from anchor keys (i.e., K_{SEAF}) in the subsequent 5G procedures. According to TS 33.501 [3], UE and SN should confirm the keys agreed and the identities of each other implicitly through the successful use of keys in subsequent procedures, which can be expressed by a key-confirmation round trip with K_{SEAF}.

5. Formal Verification

To verify the security of the protocol, various formal verification methods are used as shown in Figure 4. Among several formal verification methods, the proposed protocol is verified through two methods: BAN Logic and ProVerif. BAN Logic is a representative method of Modal Logic and one of the widely used formal verification methods. With its decisiveness on the result, it can be fully trusted once the verification process is correct. However, each verification method has its pros and cons. BAN Logic does not consider dishonest reasoning. In other words, it cannot detect the attack of malicious participants. Thus, for precise verification results, we have included ProVerif as the second verification tool. ISO29129-1, a document for protocol verification framework, guides to formally verify the protocol using the automated prover [28]. ProVerif is one of the state-of-the-art verification tools that meets the guidelines. It can formally verify the protocol in an unbounded session environment and detect malicious attacks on the protocol. By using two complementary verification methods, BAN Logic and ProVerif, we have come up with reliable verification results.

	Model Checking	Theorem Proving	Modal Logic
Symbolic	NRL FDR AVISPA SCYTHER ProVerif AVISPA (TA4SP)	Isabelle/HOL	BAN Logic SVO Logic Rubin Logic
Crypto graphic	CryptoVerif	BPW(in Isabelle/HOL) Game-based Security Prof (in Coq) **Unbounded**	

Figure 4. Types of formal verification.

5.1. Formal Verification via BAN Logic

The results of BAN Logic are driven by the idealization, assumption, goals, and derivation phase. The notations and rules of BAN Logic used in the above process are shown in Table 2 and 3. Excluding messages that are not encrypted, only messages protected by secret keys or secret information between communication participants, such as encryption, digital signature, and message authentication code are expressed. Second, in the Assumption step, preconditions for communication, such as network environment and home, are defined in a form that can be applied in BAN Logic. Third, in the goal step, the security goal to be required in the proposed protocol is defined. Finally, in the derivation step, it shows a series of processes to derive the security attributes defined in the goals through the BAN Logic rule. The results verified through BAN Logic in this paper are as follows.

Table 2. Notations of BAN Logic.

Notation	Meaning
$P \mid\equiv X$	P believes the message X
$P \triangleleft X$	P receives the message X
$P \mid\sim X$	P previously sent the message X
$P \Rightarrow X$	P has authority over X
$\#(X)$	The message X is fresh
$\langle X \rangle_K$	X is combined with a secret K
$\{X\}_K$	X is encrypted with a key K
$P \overset{K}{\longleftrightarrow} Q$	K is a secret key shared between P and Q
$\overset{K}{\mapsto} P$	K is the public key of P
$P \overset{K}{\rightleftharpoons} Q$	K is a shared secret between P and Q

Table 3. Rules of BAN Logic.

Rule	Formula
Message Meaning Rule (MM)	$\frac{P \mid\equiv P \overset{K}{\longleftrightarrow} Q,\ P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$ $\frac{P \mid\equiv P \overset{K}{\rightleftharpoons} Q,\ P \triangleleft \langle X \rangle_K}{P \mid\equiv Q \mid\sim X}$ $\frac{P \mid\equiv \overset{K}{\mapsto} Q,\ P \triangleleft \{X\}_{Q^{-1}}}{P \mid\equiv Q \mid\sim X}$
Nonce Verification Rule (NV)	$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$
Jurisdiction Rule (JR)	$\frac{P \mid\equiv Q \Rightarrow X,\ P \mid\equiv Q \mid\sim X}{P \mid\equiv X}$
Freshness Rule (FR)	$\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X, Y)}$
Decomposition Rule (DR)	$\frac{P \triangleleft (X, Y)}{P \triangleleft X}$
Belief Conjunction Rule (BC)	$\frac{P \mid\equiv X,\ P \mid\equiv Y}{P \mid\equiv (X, Y)}$ $\frac{P \mid\equiv Q \mid\equiv (X, Y)}{P \mid\equiv Q \mid\equiv X}$ $\frac{P \mid\equiv Q \mid\sim (X, Y)}{P \mid\equiv Q \mid\sim X}$
Hash Rule (HR)	$\frac{P \mid\equiv Q \mid\sim H(X), P \triangleleft X}{P \mid\equiv Q \mid\sim X}$
Diffie–Hellman Rule	$\frac{P \mid\equiv Q \mid\sim \overset{g^Y}{\longmapsto} Q,\ P \mid\equiv \overset{g^X}{\longmapsto} P}{P \mid\equiv P \overset{g^{XY}}{\longleftrightarrow} Q}$ $\frac{P \mid\equiv Q \mid\sim \overset{g^Y}{\longmapsto} Q,\ P \mid\equiv \overset{g^X}{\longmapsto} P}{P \mid\equiv P \overset{g^{XY}}{\rightleftharpoons} Q}$

5.1.1. The Initiation Phase: Step 1

The idealization form of the Initiation Phase of the protocol is shown below:

$$UE \rightarrow HN: \langle \left\{ \text{SUPI}, UE \overset{k_2}{\rightleftharpoons} HN \right\} \rangle_{k_1} \tag{1}$$

We added Assumptions (2) to (5) to verify through BAN Logic.

$$HN \mid\equiv UE \overset{k_1}{\rightleftharpoons} HN \tag{2}$$

$$HN \mid\equiv \#(UE \overset{k_1}{\rightleftharpoons} HN) \tag{3}$$

$$HN \mid\equiv UE \overset{k_2}{\leftrightarrow} HN \tag{4}$$

$$HN \mid\equiv \#(UE \overset{k_2}{\leftrightarrow} HN) \tag{5}$$

Technically, all the aforementioned assumptions could be invalidated as the HN inherently lacks trust in the UE's public key PK_{UE}. However, in the case of 5G AKA, since it was designed to tolerate a replay attack or Man-in-the-Middle (MITM) attack on the public key, we followed the standard and added the above assumption to continue this analysis.

Therefore, we set a security goal that HN believes that UE believes in SUPI, and derived this goal through the derivation step.

$$HN \mid\equiv UE \ \mid\equiv \text{SUPI} \tag{6}$$

from (1), and we derive

$$HN \triangleleft \langle \left\{ \text{SUPI}, \ UE \overset{k_2}{\rightleftharpoons} HN \right\}_{k_2}, UE \overset{k_1}{\rightleftharpoons} HN \rangle_{k_1} \quad \text{by (1)} \tag{7}$$

$$HN \mid\equiv UE \mid\sim \left(\left\{ \text{SUPI}, UE \overset{k_2}{\rightleftharpoons} HN \right\}_{k_2}, \ UE \overset{k_1}{\rightleftharpoons} HN \right) \quad \text{by (7), (2), MM} \tag{8}$$

$$HN \mid\equiv UE \mid\equiv \left\{ \text{SUPI}, \ UE \overset{k_2}{\rightleftharpoons} HN \right\} \quad \text{by (8), (3), FR, NV, BC} \tag{9}$$

$$HN \mid\equiv UE \mid\equiv \text{SUPI} \quad \text{by (9), (4), MM, (5), FR, NV, BC} \tag{10}$$

According to the derivation above, the proposed security protocol can achieve the security goal (6) stated in the goal. This means that the HN believes that UE believes about SUPI, in the initiation phase of the proposed protocol.

5.1.2. The Challenge-Response Phase: Step 2

The idealization form of the Challenge-Response Phase of the protocol is shown below:

$$HN \rightarrow UE: \overset{Y}{\mapsto} HN, \ \{SQN_{HN}\}_{AK}, \ \langle SQN_{HN}, \ AMF, \overset{Y}{\mapsto} HN, \ UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN, \ UE \overset{K_{SEAF}}{\longleftrightarrow} HN \rangle_k \tag{11}$$

$$UE \rightarrow SN: RES^* \tag{12}$$

$$UE \rightarrow HN: \langle ID_{SN}, \overset{Y}{\mapsto} HN, UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN, UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \rangle_k \tag{13}$$

$$HN \rightarrow SN: \left\{ \text{SUPI}, UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF, \left(UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \right\}_{K_{N12}} \tag{14}$$

Unlike HN in (14), in (13), due to no knowledge on k, SN treats RES^* as an unrecognized simple value. We added assumptions from (15)–(26) to derive the security goal.

$$UE \mid\equiv UE \overset{K}{\longleftrightarrow} HN \tag{15}$$

$$UE \mid\equiv \overset{X}{\mapsto} UE \tag{16}$$

$$UE \mid\equiv \#(SQN_{HN}) \tag{17}$$

$$UE \mid\equiv \#(K_{SEAF}) \tag{18}$$

$$SN \mid\equiv \overset{Y}{\mapsto} HN \tag{19}$$

$$SN \mid\equiv H\left(RES^*, \overset{Y}{\mapsto} HN\right) \tag{20}$$

$$HN \mid\equiv UE \overset{K}{\rightleftharpoons} HN \tag{21}$$

$$HN \mid\equiv \#(\overset{Y}{\mapsto} HN) \tag{22}$$

$$SN \mid\equiv SN \overset{K_{N12}}{\longleftrightarrow} HN \tag{23}$$

$$HN \mid\equiv \#(SN \overset{K_{N12}}{\longleftrightarrow} HN) \tag{24}$$

$$SN \mid\equiv HN \Rightarrow UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \tag{25}$$

$$SN \mid\equiv HN \Rightarrow \left(UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \tag{26}$$

K_{SEAF} is derived based on the ECIES ephemeral key pair performed once again in HN, and the UE can also derive K_{SEAF} through the ECIES process. Therefore, since the UE can trust that K_{SEAF} is fresh, (18) is added. Also, (19) and (20) are added because SN receives $RAND = \overset{Y}{\mapsto} HN$ and $HXRES^*$ from HN via secure channel and counts on HN.

$$UE \mid\equiv HN \mid\equiv \overset{Y}{\mapsto} HN \tag{27}$$

$$UE \mid\equiv UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \tag{28}$$

$$UE \mid\equiv HN \mid\equiv UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \tag{29}$$

$$UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} HN \tag{30}$$

$$UE \mid\equiv HN \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \tag{31}$$

$$HN \mid\equiv UE \mid\equiv UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \tag{32}$$

$$HN \mid\equiv UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} HN \tag{33}$$

$$SN \mid\equiv HN \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \tag{34}$$

$$SN \mid\equiv UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \tag{35}$$

from (11), we derive

$$UE \triangleleft \overset{Y}{\mapsto} HN \quad \text{by (11)} \tag{36}$$

Here, given the ephemeral ECIES public key of HN, UE derives $DHK = x \cdot Y = x \cdot y \cdot G$ and then computes $AK = f_5(k, Y \bigoplus (x \cdot y \cdot G))$. At this point, despite no trust in Y, UE can be sure that AK is a good key shared between HN and itself. This is because AK is derived by inputting k and x where k is only known to both UE and HN and x is its own fresh ephemeral ECIES private key. As a result, UE gains the belief (37) as follows.

$$UE \mid\equiv UE \overset{AK}{\longleftrightarrow} HN \qquad \text{by (36), (15), (16)} \tag{37}$$

$$UE \triangleleft \{SQN_{HN}\}_{AK} \qquad \text{by (11)} \tag{38}$$

$$UE \mid\equiv HN \mid\equiv SQN_{HN} \qquad \text{by (38), (37), MM, (20), NV} \tag{39}$$

Note that after recovery, SQN_{HN} is checked for its freshness. For this procedure, we add the assumption (17). Now, UE possesses the valid SQN_{HN}.

$$UE \triangleleft \langle SQN_{HN},\ AMF,\ \overset{Y}{\mapsto} HN,\ UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \rangle_k \quad \text{by (11)} \tag{40}$$

$$UE \triangleleft HN \mid\sim \left(SQN_{HN},\ AMF,\ \overset{Y}{\mapsto} HN,\ UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \quad \text{by (40), (15), MM} \tag{41}$$

$$UE \mid\equiv HN \mid\equiv \left(SQN_{HN},\ AMF,\ \overset{Y}{\mapsto} HN,\ UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \quad \text{by (41), (18), FR, NV} \tag{42}$$

$$UE \mid\equiv HN \mid\equiv \overset{Y}{\mapsto} HN \quad \text{by (11)} \tag{43}$$

Even if *RAND* and *AUTN* are replayed, a linkability attack does not occur in the UE. Because freshness is verified based on the key derived through ECIES rather than SQN_{HN} during *MAC* check, when a replay attack occurs, the `Sync_Failure` step is not performed, and `MAC_Failure` is executed to stop authentication, so unlinkability is supported.

In addition, UE obtains the indirect belief on $\overset{Y}{\mapsto} HN$, which helps this protocol to defend against the Man-in-The-Middle attacks. Based on (43), UE proceeds to gain the direct belief on $UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN$ as follows.

$$UE \mid\equiv UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \qquad \text{by (43), (16), DH} \tag{44}$$

$$UE \mid\equiv HN \mid\equiv\ UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \qquad \text{by (42), BC} \tag{45}$$

UE derives K_{SEAF} with the values k, SQN_{HN}, Y , $x \cdot y \cdot G$, ID_{SN}. In particular, based on (15), (17), (43), and (44), UE can arrive at the following belief.

$$UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} HN \qquad \text{by (25), (18), (43), (44)} \tag{46}$$

$$UE \mid\equiv HN \mid\equiv\ UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \qquad \text{by (42), BC} \tag{47}$$

At this point, it is confirmed from (44) to (47) that HN is authenticated to UE.
from (12), and we derive

$$SN \triangleleft RES^* \quad \text{by (12)} \tag{48}$$

$$SN \mid\equiv RES^* \quad \text{by (48), (18), (19), HR} \tag{49}$$

Based on (49), SN proceeds this protocol by forwarding RES^* to HN.
from (13), and we derive

$$HN \triangleleft \langle ID_{SN}, \overset{Y}{\mapsto} HN, UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN, UE \overset{K_{SEAF}}{\longleftrightarrow} HN \rangle_k \quad \text{by (13)} \tag{50}$$

$$HN \mid\equiv UE \mid\equiv \left(ID_{SN}, \overset{Y}{\mapsto} HN, UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN, UE \overset{K_{SEAF}}{\longleftrightarrow} HN \right)$$
$$\text{by (50), (21), MM, (22), FR, NV} \tag{51}$$

$$HN \mid\equiv UE \mid\equiv UE \overset{x \cdot y \cdot G}{\longleftrightarrow} HN \quad \text{by (51), BC} \tag{52}$$

$$HN \mid\equiv UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} HN \quad \text{by (51), BC} \tag{53}$$

Based on (51), UE and HN mutually authenticate
from (14), and we derive

$$SN \triangleleft \left\{ \text{SUPI}, UE \overset{K_{SEAF}}{\longleftrightarrow} HN, \left(UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \right\}_{K_{N12}} \quad \text{by (14)} \tag{54}$$

$$SN \mid\equiv HN \mid\equiv \left(\text{SUPI}, UE \overset{K_{SEAF}}{\longleftrightarrow} HN, \left(UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \right)$$
$$\text{by (54), (23), MM, (24), FR, NV} \tag{55}$$

$$SN \mid\equiv HN \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \quad \text{by (55), BC} \tag{56}$$

$$SN \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \quad \text{by (56), (25), JR} \tag{57}$$

$$SN \mid\equiv HN \mid\equiv \left(UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \right) \quad \text{by (55), BC} \tag{58}$$

$$SN \mid\equiv UE \mid\equiv UE \overset{K_{SEAF}}{\longleftrightarrow} SEAF \quad \text{by (58), (26), JR} \tag{59}$$

5.2. Formal Verification via ProVerif

5.2.1. Implementations and designs

In this section, we formally verify the 5G-AKA-FS protocol using a well-known formal verification tool, ProVerif [23]. We designed the processes of User Equipment (UE), Serving Network (SN), and Home Network (HN) respectively at **UE**, **SN** and **HN** in ProVerif to verify its security properties. Our implementation also contains the process protocol that concludes the proof.

We implemented the processes under Dolev–Yao's attacker model, where the attacker can access the encrypted data only if it has the correct key to decrypt them. We largely adopt the settings that the implementation of Damir et al. [29] is based on. The differences between ours and Damir et al. are as follows:

1. First, we newly define the symbolic process DHkey for the perfect forward secrecy, that is the Diffie–Hellman Key exchange protocol, in the implementation.

2. Although our settings are similar to Damir et al. [29], the actual internal processes are different as the internal processes of UE, SN, and HN are fairly different from those of Damir et al.'s protocol. We implement those differences in our implementation.

Under Dolev–Yao's attacker model, we simplify some cryptographic primitives as symbolic functions. Particularly, the Diffie–Hellman key exchange protocol can be implemented as a symbolic function as follows:

DH Key Exchange protocol

```
type pubKey.
type secKey.
...
fun pk(secKey):pubKey.
fun DHkey(secKey,pubKey):bitstring.
equation forall sk1:secKey, sk2:secKey;
DHKey(sk2,pk(sk1))=DHKey(sk1,pk(sk2)).
```

For the symmetric key encryption/decryption, we utilized a symbolic function defined in [29] to implement the protocols in our 5G-AKA-FS. Symmetric key encryption/decryption means that a message m is encrypted by senc using a secret key n. Then, the encrypted message senc(m,n) only can be decrypted using sdec with the same key. It is defined as follows:

Encryption/Decryption

```
fun senc(bitstring,bitstring):bitstring.
reduc forall m:bitstring,n:bitstring;
sdec(senc(m,n),n)=m.
```

5.2.2. Protocol Process

We processed our protocol for the verification. We define the public key of HN using its corresponding private key, skHN, and the public key is broadcasted to all public channels as it is a public parameter. We composited infinite replication of UE, SN and HN processes in parallel for protocol processes.

Protocol Process

```
process
new skHN :secKey;
new uk :bitstring;
new idHN :bitstring;
new SNname :bitstring;
new SQNUE :bitstring;
new delta :bitstring;
new SUPI :bitstring;
let pkHN = pk(skHN) in out(usch1, (pkHN));
out(usch2, (pkHN));
out(usch3, (pkHN));
(!UE(SUPI,idHN,pkHN,uk,SNname,SQNUE,delta)|
!SN(SNname)|!HN(skHN,idHN))
```

5.2.3. Assertions

Using ProVerif, we can verify if the adversary can access the security parameters by reaching the state where those parameters are available. In our protocol, the private key of HN (skHN), a Long-term key at UE/HN (uk), and the Long-term identity (SUPI) were queried to verify their secrecy.

Secrecy of Parameters

```
query attacker(skHN).
query attacker(uk).
query attacker(SUPI).
```

The sequence of the protocol can also be verified by modeling how the messages are processed and exchanged among UE, SN and HN. Those can be verified using assertions consisting of events that represent message processing and delivery. We utilize the same naming rules for events with [29]. Therefore, for a query x, we formally define events as follows:

- *event UESendReqSN(·):* UE sends a connection request with the required parameters to SN in Step 1.1.
- *event SNRecReqUE(·):* SN receives a connection request with the required parameters from UE in Step 1.2.
- *event SNSendReqHN(·):* SN sends a connection request with the required parameters to HN in Step 1.2.
- *event HNRecReqSN(·):* HN receives the result of a connection request with the other relevant parameters from SN in Step 1.3.
- *event HNSendResSN(·):* HN sends the result of a connection request with the other relevant parameters to SN in Step 2.1.
- *event SNRecResHN(·):* SN receives the result of a connection request with the other relevant parameters from HN in Step 2.2.
- *event SNSendResUE(·):* SN sends the result of a connection request with the other relevant parameters to HN in Step 2.2.
- *event UERecResSN(·):* UE receives the result of a connection request with the other relevant parameters from SN in Step 2.3.
- *event UESendConSN(·):* UE sends authentication parameters to SN in *Case iii* in Step 2.3.
- *event SNSendConHN(·):* SN forwards (RES^*) to HN in *Case iii* in Step 2.3.
- *event SNRecConUE(·):* SN receives (RES^*) from UE *Case iii* in Step 2.3.
- *event HNRecConSN(·):* HN receives (RES^*) from SN *Case iii* in Step 2.3.
- *event HNSendSUPISN(·):* HN sends ($SUPI, K_{SEAF}$) to SN in *Case iii* in Step 2.3.
- *event SNRecSUPIHN(·):* SN receives ($SUPI, K_{SEAF}$) from HN in *Case iii* in Step 2.3.

Using the events described above, we can formally verify the hypothesis "event A ==> event B", which means that, without the execution of event B, event A cannot be executed. For example, the first two lines of the following ProVerif code of Process Verification imply that SN sends a connection request with the required parameters to HN only if HN receives the result of a connection request with the other relevant parameters from SN.

The following are the all events we verified in the implementation.

Process Verification

```
query a:bitstring,b:bitstring;
event(HNRecReqSN(a)) ==> event(SNSendReqHN(b)).
query a:bitstring,b:bitstring;
event(SNRecResHN(a)) ==> event(HNSendResSN(b)).
query a:bitstring,b:bitstring;
event(UERecResSN(a)) ==> event(SNSendResUE(b)).
query a:bitstring,b:bitstring;
event(SNRecConUE(a)) ==> event(UESendConSN(b)).
query a:bitstring,b:bitstring;
event(HNRecConSN(a)) ==> event(SNSendConHN(b)).
query a:bitstring,b:bitstring;
event(SNRecSUPIHN(a)) ==> event(HNSendSUPISN(b)).
```

5.2.4. Verification Results

We execute our ProVerif code on the ProVerif online demo website. As a result of the verification, we can conclude that the attacker cannot access the security keys in any state of the process. Moreover, all the sequences of executions of UE, SN, and HN are verified as defined in the protocol.

The summary of the verification results is as follows:

Verification Summary

```
Query not attacker(skHN[]) is true.
Query not attacker(uk[]) is true.
Query not attacker(SUPI[]) is true.
Query event(HNRecReqSN(a)) ==> event(SNSendReqHN(b)) is true.
Query event(SNRecResHN(a)) ==> event(HNSendResSN(b)) is true.
Query event(UERecResSN(a)) ==> event(SNSendResUE(b)) is true.
Query event(SNRecConUE(a)) ==> event(UESendConSN(b)) is true.
Query event(HNRecConSN(a)) ==> event(SNSendConHN(b)) is true.
Query event(SNRecSUPIHN(a)) ==> event(HNSendSUPISN(b)) is true.
```

6. Comparative Analysis

In this section, we conduct a comparative assessment aimed at evaluating the effectiveness of the proposed protocol by considering security requirements as well as the computational and communication overheads. For a starter, six protocols are compared against six security requirements to assess the degree of security they each offer. This analysis showcases that the proposed protocol delivers robust security measures in comparison to the others.

Next, we proceed to determine the computational overhead linked to each protocol. This involves scrutinizing the amount of cryptographic operations required within each security protocol and quantifying the computational overhead through Python. We also evaluate the communication overhead by closely examining the message dimensions described in the 3GPP standard document [3]. For that, the transmitted messages are not only inspected, but also their bit sizes are calculated to measure the communication overhead caused by each protocol.

6.1. Security Analysis

The proposed security protocol is compared to the existing protocols based on six security requirements: 5G Network and UE's Mutual Authentication (AUTH), Secure Key Exchange (SKE), Legacy USIM Compatibility Support (LUCS), LinkaBility Attack (LBA), Active attack by Malicious SN (AMS), and Forward Secrecy for K_{SEAF} (FS). The security requirements that each protocol satisfies are shown in Table 4.

According to the table, all protocols satisfy the requirements for AUTH as well as SKE. However, LUCS is not supported by 5G-IPAKA and 5GAKA-LCCO. 5G-IPAKA introduces a different structure, which may result in compatibility issues. Similarly, 5GAKA-LCCO deviates from the standard by utilizing T_{SN} and follows an unconventional protocol flow, potentially leading to compatibility problems with previous versions, i.e., reverse compatibility problems.

LBA is an attack related to compromising a UE's location privacy. 5GAKA-LCCO eliminates the process of synchronization failure by reducing round-trip, thus preventing LBA. In the case of 5G-AKA′, k_{HN} is reused instead of SQN, which enables it to confirm the freshness of MAC and address LBA. On the other hand, 5G-AKA-FS can defend against LBA because it computes DHK in HN and reflects this key when generating values required for authentication.

In 5G-IPAKA and 5G-AKA′, active attacks by malicious SN are possible because the HN delivers K_{SEAF} to the SN without authentication of the UE. 5G-AKA, SUCI-AKA,

and 5G-AKA′ derive the anchor key K_{SEAF} through the long-term key k, so FS is not supported when k is leaked. 5G-IPAKA and 5GAKA-LCCO used not only k but also HN's private key sk_{HN} to support FS. However, if sk_{HN} is compromised, FS for K_{SEAF} in the past is not achieved.

Table 4. Comparison in terms of security requirements with existing protocols.

Protocol	Security Requirements					
	AUTH	**SKE**	**LUCS**	**LBA**	**AMS**	**FS**
5G-AKA [3]	◯	◯	◯	×	◯	×
SUCI-AKA [20]	◯	◯	◯	×	◯	×
5G-IPAKA [17]	◯	◯	×	×	×	×
5GAKA-LCCO [21]	◯	◯	×	◯	◯	×
5G-AKA′ [16]	◯	◯	◯	◯	×	×
EAP-TLS1.3	◯	◯	×	◯	◯	◯
EAP-AKA′-FS	◯	◯	◯	◯	◯	◯
5G-AKA-FS	◯	◯	◯	◯	◯	◯

◯: Support; ×: Not support;

6.2. Overhead Analysis

To compare the trade-off between security and resource consumption, we have compared the proposed protocol computation and communication overhead. SUCI-AKA, 5G-IPAKA, 5GAKA-LCCO, and 5G-AKA′ are other protocols improvised based on 5G-AKA. However, they do not provide complete FS. EAP-TLS1.3 and EAP-AKA′-FS are both representative protocols on the mobile network field and provide complete FS. The test results provide respectful data for a trade-off between security and resource consumption.

6.2.1. Computation Overhead

Table 5 summarizes the environment used in the experiment. The computation overhead for each protocol was measured by conducting 5000 test runs using the cryptography library in Python 3.10.11.

Table 5. Experimental environments.

Operating System	Windows 11
CPU	12th Gen Intel(R) Core(TM) i7-12700KF
GPU	NVIDIA GeForce RTX 3070 Ti
RAM	DDR4 64.0GB
Program Language	Python 3.10.11
Library	Cryptography

As can be seen in a test result shown in Figure 5, 5G-AKA-related protocols have minor differences with 5G-AKA computation overhead. However, these protocols do not completely provide FS. The proposed protocol, with a computation overhead of 6.75 ms, provides FS and has resistance against LBA and AMS. Moreover, as can be seen in comparison with EAP-TLS1.3 and EAP-AKA′-FS, the increase in computation overhead for providing FS in the proposed protocol is low.

The majority of the computation overhead is incurred by ECDH (Elliptic Curve Diffie–Hellman) and digital signature. To provide FS, the protocol must generate a fresh key every session, which 5G-AKA does not do on the HN side, so an increase in overhead to provide FS on protocols is mostly caused by adding fresh ECDH in the key generation phase. In EAP-AKA′-FS, HN and UE generate fresh ECDH keys in the challenge-response phase. However, the proposed protocol, with the reuse of the fresh ECDH key generated in the initiation phase. In Step 1.1, the proposed protocol computation overhead is optimized. Moreover, unlike EAP-TLS1.3, by a succession of 5G-AKA architecture, the proposed protocol does not require a digital signature. These are the reasons the proposed protocol has lower computation overhead than EAP-TLS1.3 and EAP-AKA′-FS. Considering that

our proposed protocol provides strong security, this level of computational overhead is deemed acceptable. It represents a trade-off that we are willing to accept to prioritize robust security.

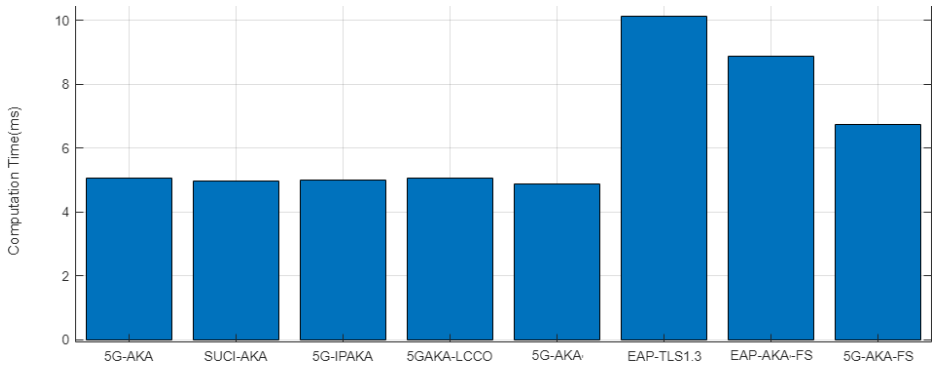

Figure 5. Total Computation Overhead.

6.2.2. Communication Overhead

Communication overhead refers to the total message size that needs to be transferred in the network for a specific purpose. Based on the 3GPP 33.501 specification, we have analyzed the total size of the messages that are exchanged through the primary authentication phase. Table 6 refers to the size of the messages that consist of the 5G primary authentication phase.

Table 6. 5G-AKA message size.

Message	SUCI	SNN(SN-Name)	5G HE AV	5G SE AV	AUTN	SUPI
Bits	536	256	640	384	128	60

The combination of upper messages concludes the total communication overhead of the 5G-AKA. Table 7 gives the total communication overhead of 5G-AKA, improved protocols, and for practical comparison EAP-TLS1.3 and EAP-AKA′-FS.

Table 7. Communication Overhead.

Protocol	**UE (Bits)**	**Core (Bits)**	**Total (Bits)**
5G-AKA	920	3112	4032
SUCI-AKA	1048	3368	4416
5G-IPAKA	1112	2268	4480
5G-LCCO-AKA	984	3176	4224
5G-AKA′	920	3112	4032
EAP-TLS 1.3	5520	5264	10,784
EAP-AKA′-FS	2068	3500	5568
5G-AKA-FS	1048	3368	4416

For one protocol to offer additional security properties and inherit the structure of 5G-AKA, it is most likely to use additional messages to fulfill that purpose. As a result, 5G-AKA-FS leads to a higher communication overhead than 5G-AKA. However, as compensation for this sacrifice, i.e., additional overhead, the proposed protocol offers forward secrecy by newly generating the HN's ephemeral ECDH public key and using it as RAND in the initialization phase. Note that in 5G-AKA RAND is a 128-bit random challenge, but in 5G-AKA-FS the RAND challenge is replaced with the HN's ephemeral ECDH public key, which is 256-bit. This results in 5G-AKA-FS having 384-bit higher communication overhead than 5G-AKA. With only 384-bit extra messages traveling through the network,

5G-AKA-FS achieves resistance against LBA and AMS and provides FS. Among 5G-AKA-related protocols, 5G-AKA-FS is the only improvised protocol that offers all three security properties. Furthermore, despite its increase, the overhead of 5G-AKA-FS remains lower than that of the EAP-TLS-1.3 protocol and EAP-AKA′-FS. The analysis results indicate that the addition of 384 extra bits for three crucial security properties are reasonably justifiable trade-offs.

7. Further Discussions

The proposed protocol has both advantages on security properties and overheads. However, implementing 5G-AKA-FS presents another challenge. While it is USIM compatible and does not require hardware exchange, software updates are required on both the UE and the 5G Core. Given that the 5G network serves as the infrastructure for connecting a massive number of devices, testing, and simulation are essential before the updates.

As for security properties, the proposed 5G-AKA-FS protocol has several limitations. For example, our protocol does not provide resistance to key compromise impersonation attacks since an attacker who obtains a permanent key k from HN can easily impersonate UE. Also, the proposed protocol does not guarantee security against ephemeral key leakage (e.g., due to side-channel attacks) because the exposure of r breaks UE anonymity. Additionally, the security of the proposed protocol relies on the safety of the cryptographic key exchange function ECDH. However, the emergence of quantum computing poses a threat to the security of legacy cryptographic algorithms including ECDH. Therefore, future research on quantum-resistant AKA using Post-Quantum Cryptography is required.

8. Conclusions

In this paper, we proposed an improved 5G-AKA (5G-AKA-FS) protocol that provides UE unlinkability and forward secrecy, and is compatible with the 3GPP standard. Implementing the proposed protocol will require the update on UE and 5G Core. However, since it is compatible with the original 5G-AKA it only needs minor software updates. Also, we proved that the 5G-AKA-FS protocol is valid by using formal verification tools BAN Logic and ProVerif. Moreover, we compared the security properties and computation/communication overheads of our 5G-AKA-FS to those of the other existing protocols, including 5G-AKA, SUCI-AKA, 5G-AKA′, 5G-IPAKA , 5GAKA-LCCO, EAP-TLS 1.3, and EAP-AKA′-FS. This comparative analysis demonstrates that the proposed protocol effectively maintains a balance between security and efficiency.

Author Contributions: Conceptualization, I.Y. and S.S.; methodology, I.Y., S.S., J.K. and J.B.; software, G.K., H.K. and J.K.; validation, I.Y., G.K. and J.K.; formal analysis, I.Y., G.K., J.K. and J.B.; investigation, I.Y. and S.S.; writing—original draft preparation, I.Y., G.K., S.S., H.K. and J.K.; writing—review and editing, I.Y., G.K., S.S., H.K., J.K. and J.B.; visualization, G.K. and H.K.; supervision, I.Y.; funding acquisition, I.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2022-00207416, A study on PQC optimization and security protocol migration to neutralize advanced quantum attacks in Beyond 5G-based next-generation IoT computing environments, 100%).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. European 5G Performance Trails its International Peers. Available online: https://www.gsma.com/membership/resources/european-5g-performance-trails-its-international-peers/(accessed on 10 December 2023).
2. Henry, S.; Alsohaily, A.; Sousa, E.S. 5G is real: Evaluating the compliance of the 3GPP 5G new radio system with the ITU IMT-2020 requirements. *IEEE Access* **2020**, *8*, 42828–42840. [CrossRef]
3. 3GPP. Security Architecture and Procedures for 5G System TS33.501 v18.2.0. Technical Report, The 3rd Generation Partnership Project. 2023. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169 (accessed on 10 December 2023).
4. 3GPP. 3GPP System Architecture Evolution (SAE)—Security Architecture (Release 17) TS33.401 V17.4.0. Technical Report, The 3rd Generation Partnership Project. 2023. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296 (accessed on 10 December 2023)
5. Arkko, J.; Lehtovirta, V.; Eronen, P. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). Technical Report, 2009. Available online: . (accessed on 10 December 2023). [CrossRef]
6. Arkko, J.; Haverinen, H. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). Technical Report. 2006. Available online: https://www.rfc-editor.org/rfc/rfc4187 (accessed on 10 December 2023).
7. Jover, R.P.; Marojevic, V. Security and protocol exploit analysis of the 5G specifications. *IEEE Access* **2019**, *7*, 24956–24963. [CrossRef]
8. Kunz, A.; Zhang, X. New 3GPP security features in 5G phase 1. In Proceedings of the 4st IEEE Conference on Standards for Communications and Networkin (CSCN '18), Paris, France, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. [CrossRef]
9. Khan, R.; Kumar, P.; Jayakody, D.N.K.; Liyanage, M. A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions. *IEEE Commun. Surv. Tutorials* **2019**, *22*, 196–248. [CrossRef]
10. Liu, F.; Peng, J.; Zuo, M. Toward a secure access to 5G network. In Proceedings of the 17st IEEE Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '18)/ 12th IEEE Conference On Big Data Science And Engineering (TrustCom/BigDataSE '18) , New York, NY, USA, 1–3 August 2018; pp. 1121–1128. [CrossRef]
11. Hussain, S.R.; Echeverria, M.; Karim, I.; Chowdhury, O.; Bertino, E. 5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol. In Proceedings of the 26st ACM SIGSAC Conference on Computer and Communications Security (CCS'19), London, UK, 11–15 November 2019; ACM Press: New York, NY, USA, 2019; pp. 669–684. [CrossRef]
12. Ferrag, M.A.; Maglaras, L.; Argyriou, A.; Kosmanos, D.; Janicke, H. Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes. *J. Netw. Comput. Appl.* **2018**, *101*, 55–82. [CrossRef]
13. Basin, D.; Dreier, J.; Hirschi, L.; Radomirovic, S.; Sasse, R.; Stettler, V. A formal analysis of 5G authentication. In Proceedings of the 25st ACM SIGSAC Conference on Computer and Communications Security (CCS'18), Toronto, Canada, 15–19 October 2018; ACM Press: New York, NY, USA, 2018; pp. 1383–1396. [CrossRef]
14. Ahmad, I.; Shahabuddin, S.; Kumar, T.; Okwuibe, J.; Gurtov, A.; Ylianttila, M. Security for 5G and beyond. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3682–3722. [CrossRef]
15. Xiao, Y.; Gao, S. Formal verification and analysis of 5G AKA protocol using mixed strand space model. *Electronics* **2022**, *11*, 1333. [CrossRef]
16. Wang, Y.; Zhang, Z.; Xie, Y. Privacy-Preserving and Standard-Compatible AKA Protocol for 5G. In Proceedings of the 30th USENIX Security Symposium (USENIX Security '21), Online, 11–13 August 2021; USENIX Association: Vancouver, BC, Canada, 2021; pp. 3595–3612. Available online: https://www.usenix.org/conference/usenixsecurity21/presentation/wang-yuchen (accessed on 10 December 2023).
17. Xiao, Y.; Wu, Y. 5G-IPAKA: An improved primary authentication and key agreement protocol for 5g networks. *Information* **2022**, *13*, 125. [CrossRef]
18. 3GPP. Authentication and Key Management for Applications (AKMA) Based on 3GPP Credentials in the 5G System (5GS) TS 33.535 (Release 18). Technical Report, The 3rd Generation Partnership Project, 2023. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3690 (accessed on 10 December 2023).
19. Arkko, J.; Norrman, K.; Mattsson, J.P. Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS). Internet-Draft draft-ietf-emu-aka-pfs-11, Internet Engineering Task Force. 2023. Available online: https://datatracker.ietf.org/doc/draft-ietf-emu-aka-pfs/11/ (accessed on 10 December 2023).
20. Køien, G.M. The SUCI-AKA Authentication Protocol for 5G Systems. In Proceedings of the 13rd NISK Conference on Norwegian Information Security (NISK'20), Online, 23–25 Nobember 2020. Available online: https://ojs.bibsys.no/index.php/NIK/article/view/885 (accessed on 10 December 2023).
21. Xiao, Y.; Gao, S. 5GAKA-LCCO: A secure 5G authentication and key agreement protocol with less communication and computation overhead. *Information* **2022**, *13*, 257. [CrossRef]
22. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. *ACM Trans. Comput. Syst. (TOCS)* **1990**, *8*, 18–36. [CrossRef]
23. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. ProVerif 2.04: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial. Proverif User Manual, 2021. Available online: https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf (accessed on 10 December 2023).

24. Shoup, V. A Proposal for an ISO Standard for Public Key Encryption. Cryptology ePrint Archive, Paper 2001/112. 2001. Available online: https://eprint.iacr.org/2001/112 (accessed on 10 December 2023).
25. Research, C. SEC 1: Elliptic Curve Cryptography. Standards for Efficient Cryptography. 2009. Available online: https://www.secg.org/sec1-v2.pdf (accessed on 10 December 2023).
26. Research, C. SEC 2: Recommended Elliptic Curve Domain Parameters. Standards for Efficient Cryptography, 2010. Available online: https://www.secg.org/sec2-v2.pdf (accessed on 10 December 2023).
27. 3GPP. 3G Security Architecture TS 33.102 (Release 16). Technical Report, The 3rd Generation Partnership Project. 2020. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2262 (accessed on 10 December 2023).
28. ISO. ISO/IEC 29128-1:2023 Information Security, Cybersecurity and Privacy Protection—Verification of Cryptographic Protocols—Part 1: Framework. Technical Report, the International Organization for Standardization, 2023. Available online: https://standards.iteh.ai/catalog/standards/iso/5a8c7c4d-434f-4816-a4e0-b33660dd311c/iso-iec-29128-1-2023 (accessed on 10 December 2023).
29. Damir, M.T.; Meskanen, T.; Ramezanian, S.; Niemi, V. A Beyond-5G Authentication and Key Agreement Protocol. In Proceedings of the Network and System Security—16th International Conference, NSS 2022, Denarau Island, Fiji, 9–12 December 2022; Berlin/Heidelberg, Germany, 2022; Volume 13787, pp. 249–264. [CrossRef]