Cryptanalysis of TRUTH: fixing a protocol to make it 5G-compliant

J. Munilla
Dpt. of Telecommunication Engineering
University of Málaga
Málaga, 29071, Spain
Email: jmf@uma.es

M. Burmester
Department of Computer Science
Florida State Unviersity
Tallahassee, Florida, 32304, USA.
Email: burmester@cs.fsu.edu

P. Caballero-Gil
Dept. of Computer Science and AI
University of La Laguna
La Laguna, Tenerife, Spain
Email: pcaballe@ull.edu.es

2024 International Symposium on Networks, Computers and Communications (ISNCC) | 979-8-3503-6491-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISNCC62547.2024.10758950

***Abstract*—Trust and security are critical deployment requirements for Industrial Internet of Things (IIoT) networks. A recent protocol, called TRUTH, integrates security mechanisms for authentication and privacy alongside a Dempster-Shafer based trust model, to assess the trustworthiness of collected data. We analyze this protocol and show that it does not comply with the implementation requirements of the 5G standard, and that flaws in the identification of involved parties hinder its adoption by 5G devices. We show how to fix this protocol and tailor it for the specific 5G implementation constraints, by enhancing its computational and communication performance. Finally, we show how to prove the security of this fixed protocol (TRUTH+) using the Tamarin automated verification tool.**

I. Introduction

The 5th-generation (5G) of cellular networks, developed by the 3rd Generation Partnership Project consortium (3GPP), was designed to support IIoT applications. Specifically, 5G offers the Massive Machine-Type Communications (mMTC) service. For such applications, security is a critical requirement, however achieving it poses significant challenges.

Soleymani et al. recently presented a Trust and Authentication scheme [1], called TRUTH, that combines a trust model based on evidence theory (Dempster-Shafer Theory (DST)) with a privacy-preserving authentication scheme. This ensures the trustworthiness and reliability of data collected by sensors, maintaining privacy and integrity of data while in transit. However, despite its security claims, in this paper we shall show that it deviates significantly from the security requirements for devices as specified by the 3GPP standard [2], and that it has functional flaws that prevent its implementation in practical IIoT 5G networks. We propose fixes that align with the security goals, architecture, and requirements of the 5G standard. This paper targets implementation technicians who need to be aware of possible security vulnerabilities of IIoT protocols, and 5G-6G and beyond experts seeking a comprehensive understanding of mobile security and methodologies for crafting protocols to comply with security standards.

The rest of the paper is organized as follows: Section II provides an overview of 5G security procedures, Section III describes the TRUTH protocol, and Section IV analyses it, highlighting its main shortcomings. Section V describes a 5G-compliant version TRUTH+. We summarize our conclusions in Section VI.

For reference, Table I below lists some of the general notation employed in our protocol descriptions.

TABLE I
General Notation

Notation	Description
$\oplus$	Exclusive-or operation
$\|\|$	Concatenation
SK-x	Private key of user x for an asymmetric cryptosystem
PK-x	Public key of user x for an asymmetric cryptosystem
$\mathrm{E}_{\text{PK-}x}(M)$	Encryption of M using the public key of user x
$\mathrm{E}_{\text{PK-}x}(M, r)$	Randomized encryption of M using the nonce r
$\mathrm{D}_{\text{SK-}x}(M)$	Decryption of M using the private key of x
$\mathrm{S}_{\text{SK-}x}(M)$	Signature on M using the private key of user x
$\mathrm{V}_{\text{PK-}x}(S, M)$	Signature verification on M using the public key of x
h(M)	Hash of M
SQx-y	Sequence Number of users x and y stored by x
$K_{x/y}$	Long-term symmetric key shared by x and y
$V \leftarrow x$	Value x is assigned to V
$M \Rightarrow V$	V is obtained from M
TS	Timestamp

II. 5G Standard Security

A. Security Architecture

The security architecture of 5G systems is outlined in the Technical Specifications 33.501 of the 3GPP consortium [2] (referred to as TS 33.501 henceforth). This architecture is segmented into two domains: subscriber and network. The subscriber domain encompasses the User Equipment (UE), whereas the network domain has two elements: the Home Network (HN) and the Serving Network (SN). SN connects the UE with the HN, providing wireless access through its base stations, called Next Generation NodeBs (gNBs). SN and HN may belong to the same operator or different operators, as is the case in roaming scenarios. The UE contains the Mobile Equipment (ME) of the subscriber, commonly a smartphone or IoT device, equipped with a Universal Subscriber Identity Module (USIM). The USIM has cryptographic capabilities and stores the subscriber's credentials provided by the network operator.

The communication between UE and gNB is referred to as the Access Stratum (AS), while communication between UE and HN is termed Non-Access Stratum (NAS). In the 5G threat model, it is assumed that UE and SN are connected over an

untrusted wireless channel, while SN and HN communicate through a trusted channel. Adversarial capabilities are commonly modeled using the Dolev–Yao (DY) threat model [3], where active adversaries control the network and can intercept, inject, manipulate or drop messages.

B. 5G-AKA protocol

Prior to a UE being able to communicate securely, TS 33.501 mandates an authentication process, irrespective of the service or network access they request (agnostic access network). The purpose of this primary authentication is to enable mutual authentication between UE and the network, and generate an anchor key K_{SEAF}, used to derive session keys for securing communication channels. For primary authentication, TS 33.501 proposes the 5G Authentication and Key Agreement protocol, 5G-AKA (or alternatively, the legacy EAP-AKA).

The 5G-AKA protocol utilizes several cryptographic mechanisms: the exclusive-or operation for one-time pad encryption, an asymmetric encryption scheme (based on elliptic curves or proprietary to the HN) with public key PK-HN, and private key SK-HN, stored in HN, a key derivation function KDF, based on the hash function SHA256, and seven one-way keyed authentication and key generation functions: f1, f2, f3, f4, f5, and f1*, f5*. The standard does not specify an implementation for these keyed functions but only that they must be cryptographically secure and mutually independent; the 3GPP consortium developed an example, called the MILENAGE algorithms [4].

To execute the protocol, the USIM stores privately:

- The Subscription Permanent Identifier (SUPI), which uniquely identifies UE.
- A long-term secret key K, which is different for each subscriber.
- The public key PK-HN of HN.
- A sequence number SQ_{UE}, incremented with each protocol execution to prevent replay attacks.

HN stores the private key SK-HN of the asymmetric cryptosystem and also, for each subscriber, the SUPI and its own version of the sequence number, SQ_{HN}. Randomized asymmetric encryption is used to conceal the subscriber identity, defined by $\text{SUCI} = \text{E}_{\text{PK-HN}}(\text{SUPI}, r_e)$, with r_e a random bitstring. In practice, this asymmetric scheme works more like a Key Encapsulation Mechanism (KEM). Specifically, one of the two ECIES (Elliptic Curve Integrated Encryption Scheme) profiles described in the 3GPP specifications is utilized to generate ephemeral keys. These in turn, are used to derive new keys that encrypt the SUPI using an AES-based symmetric cipher. Thus, the SUPI is never sent in the clear to prevent the "International Mobile Subscriber Identity" (IMSI) catcher attacks [5].

Additionally, a Global Unique Temporary Identifier (GUTI) can be used to identify the UE [6]. This is a temporary identity assigned to UE which replaces the SUCI, thus avoiding a public key encryption and a random number generation.

Fig. 1 describes the flows of the 5G-AKA protocol, which has three phases:

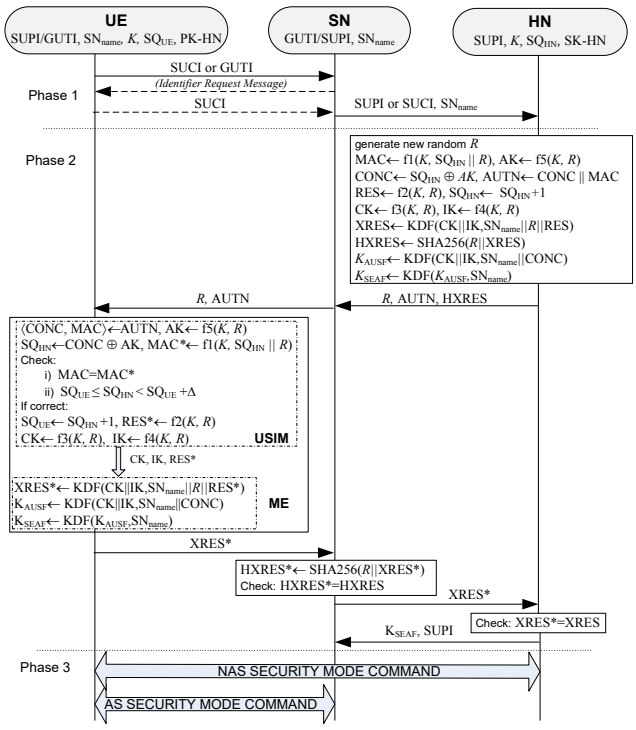

Fig. 1. 5G-authentication and key agreement (AKA).

Phase 1. UE initiates the protocol by identifying itself. If UE does not have a valid GUTI, then it computes and sends a SUCI to SN, that relays it to HN.

Phase 2. Challenge-response authentication. Upon receiving an authentication request, HN generates a random number R, used as a challenge, and an authentication value AUTN. Upon UE receiving $\{R,\ \text{AUTN}\}$, USIM retrieves SQ_{HN}, computes the MAC, and checks that: (*i*) the MAC is correct and therefore the message is authentic; (*ii*) the sequence number SQ_{HN} has not been replayed ($\text{SQ}_{\text{HN}} \geq \text{SQ}_{\text{UE}}$), and is within a specified range ($\text{SQ}_{\text{HN}} < \text{SQ}_{\text{UE}} + \Delta$), to prevent de-synchronization attacks by forcing the counter to wrap around. If *i*) fails, then a "MAC failure message" is sent. If *i*) is correct but *ii*) fails, then a "Synchronization failure message" is sent along with a re-sync token to inform HN of the value SQ_{UE} in a hidden way (masked with f5* and authenticated with f1*). Otherwise, *i*) and *ii*) are correct, USIM updates SQ_{UE}, computes RES* (using f2), CK (using f3), IK (using f4), and forwards these to ME. ME then derives the anchor key K_{SEAF}, and computes and sends XRES* to SN. SN verifies the hashed value of the response, and if correct, forwards it to HN. Finally, HN verifies the response and if correct, sends K_{SEAF} to SN.

Phase 3. Security mode command procedure. A successful 5G-AKA ends with the derivation of the anchor key K_{SEAF} by both HN and UE. The standard does not specify any additional key confirmation query for K_{SEAF} so that the authentication is *implicit* and only confirmed when a "Security Mode Command Procedure" is executed correctly [7], [8]. This

ensures that the session anchor keys computed by both parties agree and verifies the security capabilities of UE to prevent bidding-down attacks.

Along with authentication and privacy, 5G-standard security goals include: confidentiality of K_{SEAF} even if the attacker learns session keys established in other sessions (previous or subsequent) and unlinkability against passive adversaries. It is important to note here that the Linkability of AKA Failure Messages (LFM) attacks, which enable tracing a specific tag using the "failure messages" in case of failed authentication, do not compromise this security goal, since these attacks rely on the presence of an active adversary [9].

III. TRUTH Protocol

TRUTH [1] combines a trust model and an AKE[1] protocol to achieve: authentication, data integrity and confidentiality, anonymity and data trustworthiness. In this paper, we focus on the latter, setting aside the trust model.

This protocol utilizes a network layered architecture consisting of users U, cloud servers CS and gateways GW. Users seek access to information collected by sensor nodes (SNs). Users do not communicate directly with sensors; instead, they request information from cloud servers (CS). These servers then contact appropriate gateways that access the sensors. A communication channel is established between U and GW to exchange this information. In addition to these parties, the protocol involves a trusted authority (TA) responsible for distributing cryptographic material among all parties through a secure channel. Communication between U, CS, and GW adheres to the 5G communication standard and is susceptible to Dolev-Yao threats (Section II-A). The TA and the CS are presumed to be fully trusted entities [1, Section III.A].

This protocol comprises three phases: an *initialization* phase, a *user pseudonym generation* phase, and an *authentication* phase.

In the initialization phase, TA generates the system parameters of an elliptic curve cryptosystem of order q, a prime, and a one-way hash function $h : \{0,1\}^* \leftarrow Z_q^*$. Then TA distributes securely private keys to U, CS and GW: SK-U, SK-CS and SK-GW, and publishes the corresponding public keys: PK-U, PK-CS and PK-GW. Next TA chooses a master key ψ and provides a password PWD_U to each user. In this architecture, CS and GW use pseudonyms $\text{CID}_\text{CS} = h(\text{ID}_\text{CS}||\text{SK-CS}||\psi)$, $\text{GID}_\text{GW} = h(\text{ID}_\text{GW}||\text{SK-GW}||\psi)$, respectively, computed using their real identities: ID_CS and ID_GW. It is worth noting that, although not explicitly described, TA must share the master key ψ with CS (and also with GW if GID_GW is not precomputed), since it is necessary, as explained later, for the execution of Step 2.

In the user pseudonym generation phase, the user with real identity ID_U and password PWD_U, generates a pseudonym UID_U, and sends $\{\text{ID}_\text{U}, \text{PWD}_\text{U}, \text{UID}_\text{U}\}$ to TA. TA then computes $\text{SID}_\text{U} = h(\text{UID}_\text{U}||\psi)$, and sends to U and the relevant cloud servers: $\{\text{UID}_\text{U}, \text{SID}_\text{U}\}$. These pseudonyms will be valid for a limited time, after which, new ones should be generated.

Then the third phase commences. Fig. 2 sketches the flows involved in this AKE process of the TRUTH scheme among a user U_i, a cloud server CS_k, and a gateway GW_j. This phase includes the following steps:

Step 1. U_i chooses a random number r_u and computes:
$B_1 \leftarrow r_u \oplus \text{h}(\text{UID}_{\text{U}_\text{i}} \| \text{SID}_{\text{U}_\text{i}})$, $B_2 \leftarrow \text{h}(r_u) \oplus \text{UID}_{\text{U}_\text{i}}$
$B_3 \leftarrow \text{h}(\text{SID}_{\text{U}_\text{i}} \| r_u \| \text{UID}_{\text{U}_\text{i}})$, $M_1 \leftarrow \{B_1, B_2, B_3, \text{TS1}\}$, and $\text{S}_{\text{SK-U}_\text{i}}(M_1)$, a digital signature.
U_i then submits to CS_k the encryption:
1) $F_1 \leftarrow \text{E}_{\text{PK-CS}_\text{k}}(M_1, \text{S}_{\text{SK-U}_\text{i}}(M_1))$.

Step 2. CS_k decrypts F_1 and then verifies the signature and the correctness of the timestamp (this should be within an acceptable window ΔT). If both are correct, CS_k computes $\text{SID}^*_{\text{U}_\text{i}} \leftarrow \text{h}(\text{UID}_{\text{U}_\text{i}}||\psi)$, $r_u^* \leftarrow B_1 \oplus \text{h}(\text{UID}_{\text{U}_\text{i}} \| \text{SID}^*_{\text{U}_\text{i}})$ and $B_3^* \leftarrow h(\text{SID}^*_{\text{U}_\text{i}} \| r_u^* \| \text{UID}_{\text{U}_\text{i}})$, and checks that: $B_3^* \stackrel{?}{=} B_3$. If this is correct, CS_k generates r_{cs} and r_{sk}, computes:
$B_4 \leftarrow r_{cs} \oplus h(\text{GID}_{\text{GW}_\text{j}})$, $B_5 \leftarrow r_u^* \oplus h(\text{GID}_{\text{GW}_\text{j}} \| \text{CID}_{\text{CS}_\text{j}})$,
$B_6 \leftarrow h(\text{GID}_{\text{GW}_\text{j}} \| r_{cs} \| \text{CID}_{\text{CS}_\text{j}})$, $B_7 \leftarrow r_{sk} \oplus r_{cs} \oplus r_u$,
$M_2 \leftarrow \{B_2, B_4, B_5, B_6, B_7, \text{TS2}\}$, and $\text{S}_{\text{SK-CS}_\text{k}}(M_2)$.
and submits to GW_j the encrypted request,
2) $F_2 \leftarrow E_{\text{PK-GW}_\text{j}}(M_2), S_{\text{SK-CS}_\text{k}}(M_2))$.

Step 3. GW_j decrypts F_2, verifies the signature and the correctness of the timestamp. If both are correct, it computes: $r_{cs}^* \leftarrow B_4 \oplus h(\text{GID}_{\text{GW}_\text{j}})$, $r_u^{**} \leftarrow B_5 \oplus h(\text{GID}_{\text{GW}_\text{j}} \| \text{CID}_{\text{CS}_\text{k}})$, $\text{UID}^{**}_{\text{U}_\text{i}} \leftarrow B_2 \oplus h(r_u^{**})$ and $B_6^* \leftarrow h(\text{GID}_{\text{GW}_\text{j}} \| r_{cs}^* \| \text{CID}_{\text{CS}_\text{k}})$, and checks if $B_6^* \stackrel{?}{=} B_6$. If so, CS_k is authenticated. Then GW_j generates r_{gw}, computes:
$B_8 \leftarrow r_{gw} \oplus h(\text{CID}_{\text{CS}_\text{k}})$, $B_9 \leftarrow h(\text{CID}_{\text{CS}_\text{k}} \| r_{gw} \| \text{GID}_{\text{GW}_\text{j}})$,
$M_3 \leftarrow \{B_8, B_9, \text{TS3}\}$, and $r_{sk}^* \leftarrow B_7 \oplus r_{cs}^* \oplus r_u^{**}$
and submits to CS_k the encrypted request,
3) $F_3 \leftarrow \text{E}_{\text{PK-CS}_\text{k}}(M_3, \text{S}_{\text{SK-GW}_\text{j}}(M_3))$.

GW_j is now ready to compute the session key $\text{SK} \leftarrow h(\text{UID}^{**}_{\text{U}_\text{i}} \| r_u^{**} \| r_{cs}^* \| r_{gw} \| r_{sk}^* \| \text{CID}_{\text{CS}_\text{k}} \| \text{GID}_{\text{GW}_\text{j}})$.

Step 4. CS_k decrypts F_3, and verifies the signature and timestamp. If both are correct, it computes: $r_{gw}^* \leftarrow B_8 \oplus h(h(\text{ID}_{\text{CS}_\text{k}} \| \text{SK-CS}_k \| \psi))$ and $B_9^* \leftarrow h(\text{CID}_{\text{CS}_\text{k}} \| r_{gw}^* \| \text{GID}_{\text{GW}_\text{j}})$, and checks if $B_9^* \stackrel{?}{=} B_9$. If so, GW_j is authenticated and CS_k computes:
$B_{10} \leftarrow h(r_u) \oplus \text{GID}_{\text{GW}_\text{j}}$, $B_{11} \leftarrow r_{gw}^* \oplus h(\text{SID}^*_{\text{U}_\text{i}})$
$B_{12} \leftarrow h(\text{UID}_{\text{U}_\text{i}} \| r_{cs} \| r_{gw} \| \text{GID}_{\text{GW}_\text{j}})$,
$M_4 \leftarrow \{B_4, B_7, B_{10}, B_{11}, B_{12}, \text{TS4}\}$, and $S_{\text{SK-CS}_\text{k}}(M_4))$
and submits to U_i the encrypted response,
4) $F_4 \leftarrow E_{\text{PK-U}_\text{i}}(M_4, S_{\text{SK-CS}_\text{k}}(M_4))$.

Step 5. U_i decrypts F_4 and verifies the signature and timestamp. If correct, U_i computes: $\text{GID}^*_{\text{GW}_\text{j}} \leftarrow B_{10} \oplus h(r_u)$, $r_{gw}^{**} \leftarrow B_{11} \oplus h(\text{SID}^*_{\text{U}_\text{i}})$, $r_{cs}^{**} \leftarrow B_4 \oplus h(\text{GID}^*_{\text{GW}_\text{j}})$, $B_{12}^* \leftarrow h(\text{UID}_{\text{U}_\text{i}} \| r_{cs}^{**} \| r_{gw}^{**} \| \text{GID}^*_{\text{GW}_\text{j}})$, and checks if $B_{12}^* \stackrel{?}{=} B_{12}$. If so, CS_k (and consequently GW_j) is authenticated. Then U_i computes $r_{sk}^{**} \leftarrow B_7 \oplus r_{cs}^{**} \oplus r_u$ and: $\text{SK} \leftarrow h(\text{UID}_{\text{U}_\text{i}} \| r_u \| r_{cs}^{**} \| r_{gw}^{**} \| r_{sk}^{**} \| \text{CID}_{\text{CS}_\text{k}} \| \text{GID}^*_{\text{GW}_\text{j}})$.

[1]The acronym AKE is employed here for consistency with the original paper. Both AKA and AKE essentially denote the same process.

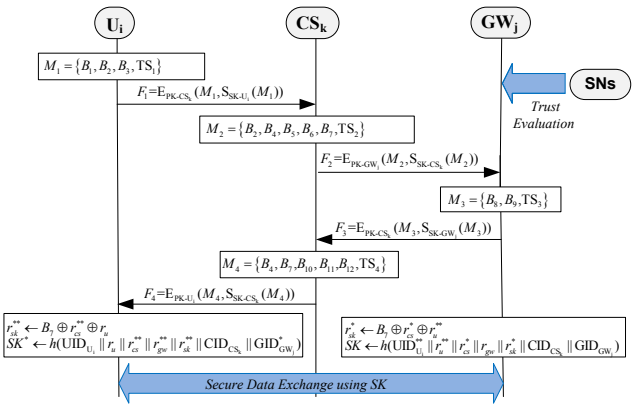

Fig. 2. Flows of the TRUTH Authenticated Key Exchange protocol.

IV. Cryptanalysis of TRUTH

The TRUTH protocol has design flaws and imposes hardware requirements that diverge significantly from the original 5G-AKA protocol.

a) User Identification. In each step of this protocol, the sender digitally signs a message with its private key, encrypting both the message and signature with the receiver's public key: $F_t = \mathrm{E}_{\mathrm{PK}_{\mathrm{Receiver}}}(M_t, \mathrm{S}_{\mathrm{SK}_{\mathrm{Sender}}}(M_t))$, with $t = 1, 2, 3, 4$. Thus in Step 2, upon receiving F_1, $\mathrm{CS_k}$ decrypts it using its private key, and verifies the signature using the public key of $\mathrm{U_i}$. The problem here is that $\mathrm{CS_k}$ faces uncertainty about the sender's identity (it *does not know who the sender is*, it could be any user). Consequently, $\mathrm{CS_k}$ cannot determine which public key to use for verifying the signature. Although $\mathrm{CS_k}$ might attempt to acquire this information from the decrypted messages B_1, B_2, B_3, this is not possible without the pseudonym $\mathrm{UID_{U_i}}$. Similarly, $\mathrm{CW_j}$ encounters the same issue upon receiving F_2. The only possible solution seems to be verifying the digital signature using the public key of *every possible sender*, until a match is found where $\mathrm{V}_{\mathrm{PK}-x}(\mathrm{S}_{\mathrm{SK}_{\mathrm{Sender}}}(M_t), M_t) = \mathrm{TRUE}$, for a possible sender x (users in Step 2 and cloud servers in Step3). However, this approach, although possible, is highly vulnerable to DoS attacks; an adversary sending a "dummy flow" could trigger numerous signature verifications (asymmetric computations). It is evident therefore that the protocol must be modified to include sender identification information (as 5G-AKA does).

b) Asymmetric encryption & digital signing. The use of asymmetric encryption and digital signing incurs substantial computation time and load, which 5G aims to minimize. Asymmetric computations can incur costs thousands of times greater than those associated with symmetric computations [10]. In the context of IIoT, where a huge number of devices are deployed, reducing hardware complexity, computation workload and communication overhead becomes essential.

c) Redundant symmetric/asymmetric authentication. In the exchanged encryptions, $\mathrm{U_i}$, $\mathrm{CS_k}, \mathrm{GW_j}$, and again $\mathrm{CS_k}$, are authenticated by checking the correctness of messages B_3, B_6, B_9 and B_{12}. However, it is uncertain if this is needed for authentication, since the verification of the digital signatures already authenticates the sender, and the use of timestamps guarantees freshness. Notably, timestamps are *absent* in the B_x messages, thus, these messages, in isolation and without the subsequent asymmetric operations, cannot serve for party authentication. It follows that this combined use of symmetric and asymmetric authentication increases the overall computation workload without offering additional benefits. The protocol should be changed to remove this hybrid authentication and clearly define the authentication conditions [11]

d) Timestamps vs sequence numbers. Employing timestamps presents practical challenges, particularly in environments like the IIoT, where synchronization among numerous intermittently connected devices is notably difficult. In these cases, the use of sequence numbers is preferable (as 5G-AKA does).

e) Master key sharing & pseudonym updating. Sharing the master key ψ, used in generating user pseudonyms, with every $\mathrm{CS_k}$ poses a risk of compromising the entire system if any component is compromised. To mitigate this risk, it is advisable to avoid such widespread key sharing. Moreover, managing pseudonym updates by the TA, distributing fresh pseudonyms to involved parties, can be inefficient and challenging, especially when numerous devices are intermittently offline. Instead, employing dynamic pseudonyms, derived from previous or stored information, would offer a more efficient approach. This method allows for the concealment of permanent identifiers, mirroring the approach seen in the 5G standard's use of SUCIs and GUTIs.

Finally, it is noteworthy that TRUTH establishes an infrastructure assuming transitive trust: users delegate the selection of their service provider to a trusted third party (a cloud server), which, in turn, chooses trustworthy gateways/sensors for them. In some cases, trust cannot be assumed to be transitive, and the infrastructure and the protocol should be redesigned.

V. A 5G-compliant version: TRUTH+

We present a 5G-complaint version of TRUTH, TRUTH+, that addresses the security concerns raised in the previous section. The main modifications involve:

- *SUPI/GUTI pseudonyms for sender identification.*
- *Limited use of asymmetric encryption:* used only when strictly necessary, as in 5G-AKA for identifying purposes, and not for authentication.
- *Authentication via shared symmetric keys:* in line with 5G-AKA, authentication relies on a proof of knowledge of shared keys.
- *Sequence Numbers instead of Timestamps.*

A. TRUTH+ Description

The real identities of the parties ID_{U_i} (SUPI in 5G), ID_{CS_k} and ID_{GW_j} are never transmitted in the clear. Instead, either GUTI or SUCI is employed based on flag values

that distinguish normal or completed previous communications ($flag = 0$) and uncompleted previous communications ($flag = 1$). Initially, $flag_{\mathrm{U_i}} = flag_{\mathrm{CS_k}} = 1$, and the parties share symmetric keys $K_{\mathrm{U_i/CS_k}}$, between between $\mathrm{U_i}$ and $\mathrm{CS_k}$, and $K_{\mathrm{CS_k/GW_j}}$, between and $\mathrm{CS_k}$ and $\mathrm{GW_j}$. Additionally, the parties share loosely synchronized sequence numbers, each party storing its own version (akin to 5G-AKA). Fig. 3 shows the flows of TRUTH+, including the computation by the different parties. It comprises the following steps:

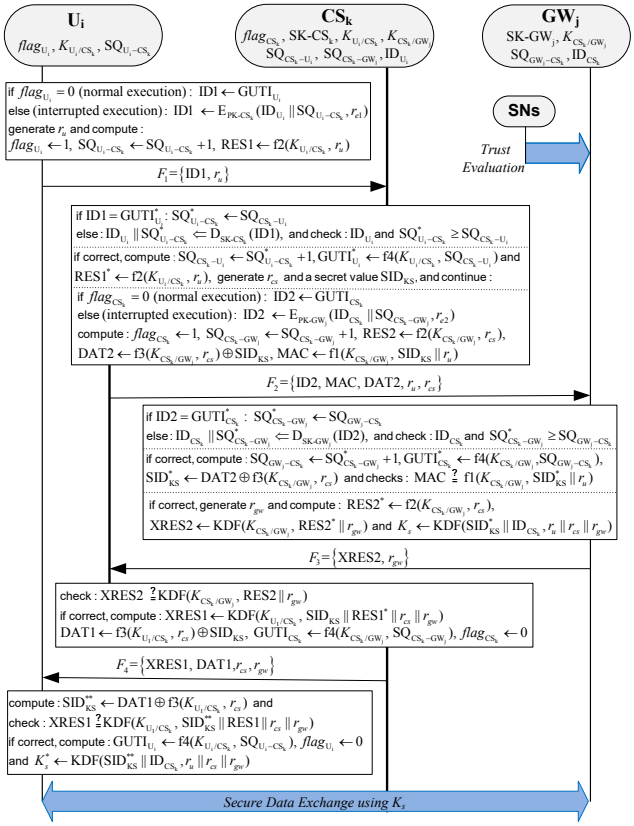

Fig. 3. TRUTH+: an example of 5G-compliant version of TRUTH.

Step 1. $\mathrm{U_i}$ creates a log-in request (F_1) depending on the value of its flag:

if $flag_{\mathrm{U_i}} = 0$: $\mathrm{ID1} \leftarrow \mathrm{GUTI_{U_i}}$,
else: $\mathrm{ID1} \leftarrow \mathrm{E_{PK\text{-}CS_k}}(\mathrm{ID_{U_i}} \| \mathrm{SQ_{U_i\text{-}CS_k}}, r_{e1})$.

If $flag_{\mathrm{U_i}} = 1$, the asymmetric encryption includes the sequence number (as proposed in [12] for 5G-AKA). Note that this hardly increases the computational complexity of the asymmetric computation, since, as explained in Section II, this operates as a KEM. Then, it chooses a nonce r_u, updates $flag_{\mathrm{U_i}}$ and $\mathrm{SQ_{U_i\text{-}CS_k}}$, computes RES1 and sends to $\mathrm{CS_k}$:

1) $F1 = \{\mathrm{ID1}, r_u\}$.

Step 2. Upon receiving F_1, $\mathrm{CS_k}$ retrieves $\mathrm{ID_{U_i}}$ and $\mathrm{SQ^*_{U_i\text{-}CS_k}}$ from ID1. If $\mathrm{GUTI^*_{U_i}}$ is received, the sequence numbers are synchronized. Otherwise, this is retrieved from the randomized asymmetric encryption using its private key $\mathrm{SK\text{-}CS_k}$, and must be greater than the last previously accepted sequence number. Note that here there is no need to check that this number is *within a certain range* (Δ) because the probability of an adversary computing a valid message ID1 that includes the secret value $\mathrm{ID_{U_i}}$ along with a new valid sequence number is negligible. If a valid user and sequence number are identified, CS_k updates its version of the sequence number, according to the received value, $\mathrm{SQ_{CS_k\text{-}U_i}} \leftarrow \mathrm{SQ^*_{U_i\text{-}CS_k}} + 1$ and $\mathrm{GUTI^*_{U_i}} \leftarrow \mathrm{f4}(K_{\mathrm{U_i/CS_k}}, \mathrm{SQ_{CS_k\text{-}U_i}})$, generates a nonce r_{cs} and a secret value (unique for each session) $\mathrm{SID_{KS}}$, computes $\mathrm{RES1}^*$, ID2 depending on the value of $flag_{\mathrm{CS_k}}$, RES2, DAT2, MAC, updates $flag_{\mathrm{CS_k}}$ and $\mathrm{SQ_{CS_k\text{-}GW_j}}$, and sends to $\mathrm{GW_j}$:

2) $F2 = \{\mathrm{ID2}, \mathrm{MAC}, \mathrm{DAT2}, r_u, r_{cs}\}$.

Step 3. Upon receiving F_2, $\mathrm{GW_j}$ retrieves $\mathrm{ID_{CS_k}}$ and $\mathrm{SQ^*_{CS_k\text{-}GW_j}}$ from ID2. If a valid cloud server and sequence number are identified, $\mathrm{GW_j}$ updates its sequence number, according to the received value, $\mathrm{SQ_{GW_j\text{-}CS_k}} \leftarrow \mathrm{SQ^*_{CS_k\text{-}GW_j}} + 1$ and $\mathrm{GUTI^*_{CS_k}} \leftarrow \mathrm{f4}(\mathrm{K_{CS_k/GW_j}}, \mathrm{SQ_{GW_j\text{-}CS_k}})$. Then, it retrieves $\mathrm{SID^*_{KS}} \leftarrow \mathrm{DAT2} \oplus \mathrm{f3}(K_{\mathrm{CS_k/GW_j}}, r_{cs})$ and checks: $\mathrm{MAC} \stackrel{?}{=} \mathrm{f1}(K_{\mathrm{CS_k/GW_j}}, \mathrm{SID^*_{KS}} \| r_u)$. If correct, $\mathrm{GW_j}$ generates a nonce r_{gw}, which identifies the session, computes $\mathrm{RES2}^*$, XRES2 and the session key K_s, and sends back to $\mathrm{CS_k}$:

3) $F3 = \{\mathrm{XRES2}, r_{gw}\}$.

Step 4. Upon receiving F_3, $\mathrm{CS_k}$ checks that: $\mathrm{XRES2} \stackrel{?}{=} \mathrm{KDF}(K_{\mathrm{CS_k/GW_j}}, \mathrm{RES2} \| r_{gw})$. If correct, it updates $\mathrm{GUTI_{CS_k}} \leftarrow \mathrm{f4}(K_{\mathrm{CS_k/GW_j}}, \mathrm{SQ_{CS_k\text{-}GW_j}})$, sets $flag_{\mathrm{CS_k}} \leftarrow 0$, computes XRES1, DAT1, and sends to $\mathrm{U_i}$:

4) $F4 = \{\mathrm{XRES1}, \mathrm{DAT1}, r_{cs}, r_{gw}\}$.

Step 5. Upon receiving F_4, user $\mathrm{U_i}$ computes: $\mathrm{SID^{**}_{KS}} \leftarrow \mathrm{DAT1} \oplus \mathrm{f3}(K_{\mathrm{U_i/CS_k}}, r_{cs})$, and checks the received response: $\mathrm{XRES1} \stackrel{?}{=} \mathrm{KDF}(K_{\mathrm{U_i/CS_k}}, \mathrm{SID^{**}_{KS}} \parallel \mathrm{RES1} \parallel r_{cs} \parallel r_{gw})$. If this is correct, $\mathrm{U_i}$ updates $\mathrm{GUTI_{U_i}} \leftarrow \mathrm{f4}(K_{\mathrm{U_i/CS_k}}, \mathrm{SQ_{U_i\text{-}CS_k}})$, sets $flag_{\mathrm{U_i}} \leftarrow 0$ and computes its version of the session key:

$$K^*_s \leftarrow \mathrm{KDF}(\mathrm{SID^{**}_{KS}} \| \mathrm{ID_{CS_k}}, r_u \| r_{cs} \| r_{gw}).$$

B. TRUTH+ Analysis

As in 5G, TRUTH+ guarantees (implicit) mutual authentication between $\mathrm{GW_j}$ and $\mathrm{U_i}$ with the confirmed use of the session key K_S, which cannot be retrieved by an adversary without knowing the session secret value $\mathrm{SID_{KS}}$, generated and shared secretly by $\mathrm{CS_k}$ with $\mathrm{GW_j}$ and $\mathrm{U_i}$ using $K_{\mathrm{CS_k/GW_j}}$ and $K_{\mathrm{U_i/CS_k}}$, respectively. The confidentiality of the session key K_s is kept even if the adversary learns previous or subsequent session keys since $\mathrm{CS_k}$ generates independent (unlinkable) values $\mathrm{SID_{KS}}$ for each session. The identities of the parties remain anonymous since they are never sent in clear; ID values are used to identify the parties. ID computation depends on the session state (indicated by the flag bit): asymmetric IDs (SUCI) are only used with interrupted sessions while symmetric IDs (GUTI) are used with completed ones. For $\mathrm{U_i}$, a session is completed if $\mathrm{CS_k}$ is authenticated with the reception of XRES1; the computation of which involves $K_{\mathrm{U_i/CS_k}}$ and the fresh value r_u. Likewise, $\mathrm{CS_k}$ authenticates $\mathrm{GW_j}$ with XRES2, which involves $K_{\mathrm{CS_k/GW_j}}$ and the fresh value r_{gw}. If a valid GUTI is employed, then the sender's sequence number

(for that message) coincides with that stored by the receiver. Otherwise, the receiver checks that this is not being replayed by checking that the received sequence number has not been previously used. Note that the probability of an adversary to forge valid ID with larger values of SQ is negligible, as it requires knowing the secret value ID. As a consequence, the adversary cannot force the counter to wrap around and it makes unnecessary to check that SQ is within a certain range Δ. Note that as with 5G-AKA, an active adversary can intercept the initial flow F_1 and replay it later. CS_k would accept this replay and send a valid F_4 to the adversary. However, without knowledge of the key K_{U_i/CS_k}, the adversary is unable to compute the session key, thereby preventing the attack from progressing further. Finally, TRUTH+ guarantees the unlinkability of the parties since ID values are never repeated; GUTIs are updated every time the session is completed and if they are not, randomized asymmetric encryption is used. All messages include fresh values generated by the parties to prevent adversaries from forcing recognizable computations.

Tables II and III show the improvement in computational and communication performance. For the computational cost, we focus on symmetric and asymmetric computations, disregarding other simple operations such as exclusive-or and concatenation. For the communication cost, we assume that the output of the asymmetric encryption is at least the length of the input message, and, referencing [1] and TS 33.501, the following lengths are assumed for the different blocks in the messages: 128 bits for the outputs of 5G functions f1-f5, 160 bits for the outputs of hash functions and random numbers, 128 bits for ID, 256 bits for the signatures and KDF, and 32 bits for the timestamps and sequence numbers.

TABLE II
COMPUTATIONAL COST

Scheme	U_i Step 1	CS_k Step 2	GW_j Step 3	CS_k Step 4	U_i Step 5	Total
TRUTH	2A+3S	4A +6S	4A+6S	4A+5S	2A+5C	16A+25S
TRUTH+	0/1A+1S	0/2A+5S	0/1A + 6S	4S	4S	0/4A+20S

S: symmetric computation, A: asymmetric computation.

TABLE III
COMMUNICATION COST (IN BYTES)

Scheme	F1	F2	F3	F4	Total
TRUTH	96	136	76	136	444 Bytes
TRUTH+	36	88	52	88	264 Bytes

For security verification, the Tamarin verification tool has been used [7]. We have developed rules (initialization, key leakages, protocol rules...) and lemmas (executability, secrecy, agreements...) to check the security of the protocol along with specific restrictions to prevent the identification problem detected in TRUTH. In particular, as Tamarin Prover deals cryptographic primitives as function symbols and not as algorithms, the verification check of the identity and the extraction of the sequence number from the asymmetric randomized encryption have been modelled by splitting it into three different terms:

$$<\mathrm{E_{PK}}(\mathrm{ID}\,\|\,\mathrm{SQ}, re)>\Leftrightarrow<\mathrm{E_{PK}}(\mathrm{ID})\oplus re, \mathrm{E_{PK}}(\mathrm{SQ})\oplus re, \mathrm{E_{PK}}(re)>$$

and verifying the identity using the restriction as follows: "Eq(ID, adec(term1 XOR adec(term3, skS), skS)), Verified($S, ID)]", where skS is the private key of the asymmetric scheme of the verifying party S (!Pk($S, pk(∼skS))).

VI. CONCLUSIONS

This paper analyzes a recently published protocol, TRUTH, demonstrating a flaw in the identification of the sender and significant divergences with the design criteria of the 5G standard. Furthermore, the extensive reliance on asymmetric cryptography, the use of synchronized timers, and master key sharing raises serious concerns regarding the practical deployment of the protocol. Finally, we propose fixes that address these issues and reduce the communication and computation workloads.

ACKNOWLEDGMENT

This research was supported by the BIOSIP Research Group (TIC-251), and partly funded by the projects: "Massive AI for the Open RadIo b5G/6G network (MAORI)", PID2022-138933OB-I00: ATQUE, funded by MCIN/AEI/10.13039/501100011033/FEDER, EU, and by NSF under Grant 2146354.

REFERENCES

[1] S. A. Soleymani, S. Goudarzi, M. H. Anisi, H. Cruickshank, A. Jindal, and N. Kama, "Truth: Trust and authentication scheme in 5g-iiot," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 880–889, 2023.

[2] *Security architecture and procedures for 5G system*, document TS 33.501, V16.8.0, 3GPP, Sep. 2021.

[3] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[4] *3G Security; Specification of the MILENAGE algorithm set: an example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 5: Summary and results of design and evaluation*, document TR 35.909, V16.0.0, 3GPP, Jul. 2020.

[5] A. Shaik, R. Borgaonkar, J.-P. Seifert, N. Asokan, and V. Niemi, "Practical attacks against privacy and availability in 4G/LTE," in *Proc. 23nd Annual Netw. Distrib. System Secur. Symp. (NDSS)*, Feb. 2016, doi: 10.14722/ndss.2016.23236.

[6] *Numbering, addressing and identification*, document TS 23.003, V17.3.0, 3GPP, Sep. 2021.

[7] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *Proc. AMC SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Oct. 2018, pp. 1383–1396.

[8] D. Segura, J. Munilla, E. J. Khatib, and R. Barco, "5g early data transmission (rel-16): Security review and open issues," *IEEE Access*, vol. 10, pp. 93 289–93 308, 2022.

[9] H. Khan and K. M. Martin, "A survey of subscription privacy on the 5G radio interface - The past, present and future," *J. Inf. Secur. Appl.*, vol. 53, Aug. 2020, Art. no. 102537, doi: 10.1016/j.jisa.2020.102537.

[10] S. Patonico and A. Braeken, "Identity-based and anonymous key agreement protocol for fog computing resistant in the canetti–krawczyk security model," *Wireless Networks*, vol. 29, 07 2019.

[11] J. Munilla, M. Burmester, and R. Barco, "An enhanced symmetric-key based 5G-AKA protocol," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108373, doi: 10.1016/j.comnet.2021.108373.

[12] A. Braeken, M. Liyanage, P. Kumar, and J. Murphy, "Novel 5g authentication protocol to improve the resistance against active attacks and malicious serving networks," *IEEE Access*, vol. 7, pp. 64 040–64 052, 2019.