Received December 31, 2019, accepted January 18, 2020, date of publication January 27, 2020, date of current version February 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969474

Formal Analysis of 5G EAP-TLS Authentication Protocol Using Proverif

JINGJING ZHANG[1,2], LIN YANG[2], WEIPENG CAO[3], AND QIANG WANG[2]

[1]College of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210007, China
[2]National Key Laboratory of Science and Technology on Information System Security, Institute of System Engineering, Chinese Academy of Military Science, Beijing 100039, China
[3]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Corresponding author: Qiang Wang (wenjunwang.nudt@gmail.com)

This work has been supported by the National Key Laboratory of Science and Technology on Information System Security.

ABSTRACT As a critical component of the security architecture of 5G network, the authentication protocol plays a role of the first safeguard in ensuring the communication security, such as the confidentiality of user data. EAP-TLS is one of such protocols being defined in the 5G standards to provide key services in the specific IoT circumstances. This protocol is currently under the process of standardization, and it is vital to guarantee that the standardized protocol is free from any design flaws, which may result in severe vulnerabilities and serious consequences when implemented in real systems. However, it is still unclear whether the proposed 5G EAP-TLS authentication protocol provides the claimed security guarantees. To fill this gap, we present in this work a comprehensive formal analysis of the security related properties of the 5G EAP-TLS authentication protocol based on the symbolic model checking approach. Specifically, we build the first formal model of the 5G EAP-TLS authentication protocol in the applied pi calculus, and perform an automated security analysis of the formal protocol model by using the ProVerif model checker. Our analysis results show that there are some subtle flaws in the current protocol design that may compromise the claimed security objectives. To this end, we also propose and verify a possible fix that is able to mitigate these flaws. To the best of our knowledge, this is the first thorough formal analysis of the 5G EAP-TLS authentication protocol.

INDEX TERMS Authentication protocol, 5G network, formal verification, model checking, applied pi calculus, ProVerif, EAP-TLS.

I. INTRODUCTION

As an indispensable infrastructure, mobile networks have evolved over several generations in the past decades. With the latest 5G network, both the subscribers and the carriers expect an increase of network throughput, as well as stronger security guarantees. Among all security guarantees, authentication and key agreement is of primary interest, which provides a fundamental mechanism to establish a secured communication channel.

Authentication and key agreement is usually achieved by executing an authentication protocol between the subscriber and the network. In the 5G network, three different authentication protocols are defined in the related 3GPP documents, including the 5G AKA (Authentication and Key Agreement) protocol [1], the EAP-AKA$'$ protocol [1] and the 5G EAP-TLS protocol [1], [2]. The first two protocols are based on the shared key cryptography (with minor differences in the way to derive session keys), and the last one is based on the public key cryptography. Though they all aim at providing mutual authentication of subscribers and networks, different protocols are used to provide services in different circumstances, for instance, the EAP-TLS protocol is defined for the subscriber authentication in limited use cases, such as private networks or IoT environments.

Currently, these protocols are in the process of standardization. They are mainly developed in the form of an RFC, which is an informal-language (usually English) document that provides extensive guidance for protocol engineers, but is nonetheless ambiguous and broadly open to interpretations. The ambiguities of informal protocol designs are sources of many severe security vulnerabilities in the implementations, as reported in [3], [4]. A useful mechanism for resolving these ambiguities and validating the correctness of the protocol

The associate editor coordinating the review of this manuscript and approving it for publication was Miguel López-Benítez.

design is to perform formal analysis, where we first build a mathematical model of the protocol in a formal language and then analyze whether the formal protocol model meets the required security properties. One prominent approach to performing the formal analysis of security protocols is the symbolic model checking [5]. Since the pioneer work [6], which found a design flaw of Needham-Schroeder protocol using this technique, symbolic model checking of security protocols has been an active research area, and it has been recognized as a powerful technique to formal analysis of the design of security protocols [7]–[13], and applied to some real-life protocols, such as TLS [14]. Research on model checking of security protocols is beyond the scope of this work. We refer to the article [5] for a detailed introduction to this field.

In our work, we employ the ProVerif [15] model checker to perform the formal analysis. ProVerif takes a formal protocol model specified in the applied pi calculus [13], [16] as input, and automatically checks whether this model meets certain security properties in the presence of a malicious attacker. Due to the fact that model checking of security protocol is undecidable in general [5], ProVerif may not terminate in some cases. In case of termination, ProVerif is able to tell whether the specified security properties are satisfied or not, and if some properties are violated, counterexamples will be generated to demonstrate the violations.

Generally speaking, ProVerif relies on the symbolic model of cryptography and the Dolev-Yao attacker model [17]. The protocol messages are abstracted by terms and the cryptographic primitives are abstracted by function symbols and assumed to be perfect (i.e., unbreakable). The algebraic properties of cryptographic primitives are described by equational relations over function symbols. The protocol logic is then modelled by a process model, which is further encoded as a set of horn clauses for formal reasoning, and the security properties are specified as reachability or correspondence properties of the formal model. Security analysis then boils down to a horn clause unification problem [18]. The main strengths of ProVerif are the followings. First, it provide a powerful modelling mechanism to describe a wide range of cryptographic primitives by using rewrite rules and equations. It also supports various security properties, including strong and weak secrecy, authentication, and some observational equivalence properties. It is also able to handle an unbounded number of parallel protocol executions, which is crucial for detecting subtle attacks, such as the man-in-the-middle. Finally, it can automatically generate the counterexamples in the form of protocol executions, when verification shows that security properties are violated.

We remark that our work is inspired by the related works [19], [20], where the authors have analyzed the security properties of the 5G AKA protocol (and its variant EAP-AKA$'$) based on the protocol analyzer TAMARIN [21]. However, as far as we know, there is no formal analysis of the 5G EAP-TLS protocol, and it is still unclear whether the current 5G EAP-TLS protocol design meets the claimed security properties as stated in the 3GPP documents. The analysis results of the 5G AKA protocol in [19], [20] cannot be applied to our case, because the 5G EAP-TLS protocol differs from the 5G AKA protocol significantly in both cryptographic primitives that are used and the way to derive session keys. Moreover, we take a different modeling framework based on a process calculus that is specific for security protocols, while their modeling is based on the term rewriting rules [22].

To this end, we make the following contributions in this work:

1) We construct the first formal model of the 5G EAP-TLS authentication protocol in the applied pi calculus, which is a formal language for security protocols. We also elicit the set of security properties from the informal standardization documents and encode them as analyzable queries in the formal model. We remark that coming up with the formal protocol model and encoding the security properties are tricky, since it requires a deep understanding of the protocol logic and the possible behavior of the attack.
2) We perform a comprehensive formal analysis of the 5G EAP-TLS protocol based on the ProVerif model checker. Our analysis results reveal several weaknesses and design flaws in the current protocol, which as a result break the intended authentication properties. The demonstrating counterexamples are produced and analyzed to identify the root causes of this flaws.
3) We also propose a possible fix of the current 5G EAP-TLS authentication protocol, and verify that the fix satisfies all the required properties. To the best of our knowledge, this is the first thorough formal analysis of the 5G EAP-TLS authentication protocol.

The remainder of this paper is organized as follows. In Section II, we review the most related works. In Section III, we present the syntax and semantics of applied pi calculus. In Section IV, we give a detailed introduction to the 5G EAP-TLS protocol. In Section V, we present the formal protocol model as well as the security properties. In Section VI, we report the verification results. Finally, in Section VII we conclude this paper and outline the future work.

II. RELATED WORKS

In this section, we review the most related works on formal analysis of 5G authentication protocols and TLS protocols.

First of all, regarding the formal analysis of 5G authentication protocols, it is a relatively new topic and most of the work focus on the analysis of the 5G AKA protocol and its variant EAP-AKA$'$. A detailed comparison of the formal models of the related 5G authentication protocols is illustrated in Table1.

In [19], the authors have modeled and analyzed the 5G AKA protocol and its security properties using TAMARIN. In their model, they merge the two main components (serving network and home network) into a single network entity. They have found authentication problems due to the lack of

TABLE 1. Comparison of the formal models of 5G authentication protocols.

Protocol	5G AKA				5G AKA$'$	5G EAP-TLS	
Article	[19]	[20]	[27]	[24]	[19]	[28]	This paper
Cryptographic primitives	Shared key cryptography	Shared key cryptography	Shared key cryptography	Shared key cryptography	Shared key cryptography	Public key cryptography	Public key cryptography
Modeling entities	UE,SEAF, AUSF	UE,SEAF, AUSF,ARPF	UE,SEAF, AUSF	UE,AUSF	UE,SEAF, AUSF	UE,SEAF, AUSF	UE,SEAF, AUSF,ARPF
Model checker being used	TAMARIN	TAMARIN	TAMARIN	-	TAMARIN	Scyther	ProVerif
Modeling language	Multiset rewriting rules	Multiset rewriting rules	Multiset rewriting rules	Bana-comon logic	Multiset rewriting rules	Role scripts	Applied pi calculus
Security Properties	Confidentiality of session key, SUPI and SQN; Authentication of each entity	Confidentiality of session key, SUPI and SQN; Authentication of each entity	Confidentiality of SQN	Unlinkability between UE and AUSF	Confidentiality of session key, SUPI and SQN; Authentication of each entity	Confidentiality of session key and SUPI; Authentication of each entity	Confidentiality of session key and SUPI; Authentication of each entity and session key
Threat model	Dolev-Yao model and compromised components	Dolev-Yao model and compromised components	Dolev-Yao model	Customize model	Dolev-Yao model and compromised components	Dolev-Yao model	Dolev-Yao model

integrity protection for service network identities. In addition, they also model and analyze the EAP-AKA$'$ protocol, which employs the elliptic curve integrated encryption scheme and uses identity hiding to guarantee user privacy. Later in [23], the authors propose a novel version of the 5G AKA protocol to overcome all the currently identified weaknesses in [19].

In [20], the authors have modeled all key components of the 5G AKA protocol (i.e., the user equipment, the serving network and the home network) according to the definition in the 3GPP specification document. Their model is more fine-grained, which includes the modeling of the core network channel between the serving network and the home network, and the modeling of dishonest participants, compared to the model in [19]. They discover an attack that exploits a potential race condition and additionally show that solving the race condition for the honest case does not necessarily prevent the attack. They also propose fixes and prove that these fixes can prevent the attack and then report their findings to 3GPP.

In [24], the authors have investigated the privacy properties of the 5G AKA authentication protocol, in the Bana-Comon logic [25], [26], which is an extension of the first order logic. They discover a novel de-synchronization attack against a modified version of the AKA protocol (i.e. PRIV-AKA), even though it has been claimed secure. They also propose a fix of this weakness, and prove that the fixed protocol guarantees the privacy properties.

In [27], the authors have found a new logical vulnerability in the specifications of all aforementioned variants of AKA. They claimed that the protection mechanism of the sequence number (SQN) can be defeated under specific replay attacks due to its use of Exclusive-OR (XOR) and a lack of randomness.

Regarding the formal analysis of the TLS protocol, we review the most related ones. In [14], the authors have developed a symbolic model of the TLS 1.3 specification (draft 21), which considers all the possible interactions of the available handshake modes. They prove the majority of the specified security requirements using the TAMARIN prover [21], and uncover a behaviour that may lead to security problems in applications that assume that TLS 1.3 provides strong authentication guarantees.

In [29], the authors present composition theorems for security protocols, to compose a key exchange protocol and a symmetric key protocol that uses the exchanged key. Their results rely on the computational model of cryptography and are stated in the framework of the model checker CryptoVerif [30]. They support key exchange protocols that guarantee injective or non-injective authentication. They also allow random oracles shared between the composed protocols. They declare that it is the first composition theorems for key exchange stated for a computational protocol verification tool, and also the first to allow such flexibility. As a case study, they apply their composition theorems to a proof of TLS 1.3 Draft-18 and have formally proved it.

In [31], the authors present a novel model framework that accounts for all recent attacks on TLS, including those relying on weak cryptographic. They use ProVerif to evaluate various modes and drafts of TLS 1.3 culminating in the first symbolic analysis of Draft-18 and the first composite analysis of TLS 1.3+1.2. Their analyses uncover both known and new vulnerabilities that influenced the final design of Draft-18. Some of the features they have studied no longer appear in the protocol, but they believe that the results of the analysis are still useful to the next generations as a warning to protocol designers and developers who may try to reintroduce these problematic features in the future.

In [32], the authors look at the implementations of the protocol, rather than the design. They have presented a thorough analysis of commonly used TLS implementations using a systematic approach called protocol state fuzzing. They use the state machine learning, which relies only on black box testing, to infer a state machine of the protocol implementation and then they perform a manual analysis of the state machines obtained to check if the implementation is consistent with the specification or not. They have analyzed the most commonly used TLS implementations and discovered new flaws.

In [33], the authors have formalized and analyzed a variant of the signal protocol for a series of security goals using both ProVerif [15] and CryptoVerif [30]. They have also implemented the signal protocol in ProScript, which is a new domain specific language for writing cryptographic protocol code. The implementation in ProScript can be executed within JavaScript programs and also automatically translated to a readable model in the applied pi calculus. Their analysis uncover several weaknesses of the protocol, including previously unreported replay and key compromise impersonation attacks. Furthermore, they have also implemented the fixes and verified the security of the fixed version.

In [34], the authors perform a wide-angle analysis of the 5G radio access network (RAN) security architecture and procedures and its potential deployment challenges as a result of the proposed 5G security framework. It is not to provide a comprehensive analysis of the security of 5G network layers and elements, but rather to assess the critical challenges of the current 5G security specifications with an outlook at future network deployments. Their study highlights a number of potential insecure protocol edge cases and limitations that result from infeasible requirements or assumptions. A survey on security and privacy of 5G technologies is available in [35].

Finally, we remark that this work is partially based on our previous publication [28], where we have modeled and analyzed the 5G EAP-TLS authentication protocol using Scyther [36]. We have refined and extended our previous work in several directions. First of all, we take a more expressive formal language that is able to describe the user defined functions and equality tests in this work. This language is more precise than Scyther in modeling the behavior of the protocol. Secondly, we construct a more fine-grained formal model of the 5G EAP-TLS authentication protocol, which includes the modeling of all key components and channels of the protocol. While, the model in [28] only takes into account two entities, i.e., the user equipment and the network. We consider the serving network and the home network to be a single entity, and we do not distinguish the key modules of the home network. Finally, we perform a more comprehensive analysis of the security requirements, and discuss our key findings by pointing out the causes of each weakness. We also propose a fix that is verified to be secure.

TABLE 2. Syntax for terms.

$M,N ::=$	terms
a, b, c, k, m, n, s	names
x, y, z	variables
$(M_1, ..., M_k)$	tuple
$h(M_1, ..., M_k)$	constructor/destructor
$M = N$	term equality
$M <> N$	term inequality
$M \&\& M$	conjunction
$M \| \| M$	disjunction
$\text{not}(M)$	negation

TABLE 3. Syntax for process.

$P,Q,R ::=$	processes
0	null process
$P \| \| Q$	parallel composition
$!P$	replication
$new\ n : t; P$	name restriction
$in(M, x : t); P$	message input
$out(M, N); P$	message output
$if\ M\ then\ P\ else\ Q$	conditional
$let\ x = M\ in\ P\ else\ Q$	term evaluation
$R(M_1, ..., M_n)$	macro usage

III. PRELIMINARY ON APPLIED PI CALCULUS

In this section, we present both the syntax and the semantics of the applied pi calculus [13], [16], which is a formal language for security protocol modeling and popularized by the ProVerif [15] model checker.

The basic grammar of the terms used in applied pi calculus is presented in Table2. A term could be a name representing a channel or data item. A term could also be a variable, or a tuple of terms $(M_1, \ldots, M_k)$. Terms constructed by constructor/destructor application are denoted by $h(M_1, \ldots, M_k)$, where k is the arity of function h. They are used to represent function applications, such as encryption or decryption. Specifically, we give the function applications with a boolean sort. Term $M = N$ ($M <> N$) represents equality (inequality) tests respectively. Notice that both equality and inequality work modulo an *equational theory* [37]. Term M && N is for boolean conjunction, and $M||M$ for the boolean disjunction, and $not(M)$ for the boolean negation.

Behavior is modeled by processes as shown in Table3. The null process 0 represents a process of doing nothing. $P \mid\mid Q$ is the parallel composition of processes P and Q, which is used to represent participants running in parallel. The replication $!P$ is the infinite composition of P (i.e. $P \mid P \mid \ldots$), which is often used to capture an unbounded number of sessions of a protocol. Name restriction $new\ \ n : t; P$ binds the name n of type t inside process P. The introduction of restricted names (or private names) is useful to capture both fresh random numbers (modeling nonces and keys, for example) and private channels. Communication is captured by message input and message output. The process $in(M, x : t); P$ awaits a message of type t from channel M and then behaves as P with the received message bound to the variable x, that is, every free occurrence of x in P refers to the message received.

TABLE 4. Syntax for pattern matching.

$T ::=$	patterns
$x : t$	typed variable
x	variable without explicit type
$(T_1, ..., T_n)$	tuple
$= M$	equality test

The process $out(M, N)$; P is ready to send term N on channel M and then run P. In both of these cases, we may omit P when it is the null process 0. The conditional *if* M *then* P *else* Q is standard: it runs P when the boolean term M evaluates to true, and it runs Q when M evaluates to some other value. For convenience, conditionals may be abbreviated as *if* M *then* P, when Q is the null process. When statement *let* $x = M$ *in* P *else* Q is encountered during process execution, there are two possible outcomes. If the evaluation of term M under destructors does not fail (that is, M is equivalent to another term without any destructors under equational theory), then x is bound to term M and the P branch is taken. Otherwise, the Q branch is taken. Similarly, it may be abbreviated as *let* $x = M$ *in* P when Q is the null process. Finally, we have $R(M_1, \ldots, M_n)$, denoting the use of the macro R with terms $M_1, \ldots, M_n$ as arguments.

We also include pattern matching (Table4) as supported by ProVerif. The variable pattern $x : t$ matches any term of type t and binds the matched term to x. The variable pattern x is similar but can be used only when the type of x can be inferred from the context. The tuple pattern $(T_1, \ldots, T_n)$ matches tuples $(M_1, \ldots, M_n)$ where each component $M_i (i \in \{1, \ldots, n\})$ is recursively matched with T_i. Finally, the pattern $= M$ matches any term N where $M = N$.

As in the applied pi calculus, terms are subject to an equational theory. Identifying an equational theory with its signature Σ, we write $\Sigma \vdash M = N$ for an equality modulo the equational theory, and $\Sigma \vdash M \neq N$ an inequality modulo the equational theory. (We write $M = N$ and $M \neq N$ for syntactic equality and inequality,respectively.) The equational theory is defined by a finite set of equations $\Sigma \vdash M_i = N_i$, where M_i and N_i are terms that contain only constructors and variables. The equational theory is then obtained from this set of equations by reflexive, symmetric, and transitive closure, closure by substitution (for any substitution σ, if $\Sigma \vdash M = N$ then $\Sigma \vdash \sigma M = \sigma N$), and closure by context application (if $\Sigma \vdash M = N$ then $\Sigma \vdash M'\{M/x\} = M'\{N/x\}$, where $\{M/x\}$ is the substitution that replaces x with M).

We show the formal semantics in Table 5. The definition consists of two parts. First, we define the semantics of expressions: the relation $D \Downarrow U$ means that the closed expression D evaluates to the closed may-fail term U, which may be a closed term M or the constant *fail*. The first two rules of the definition of $\Downarrow$ express that a closed term M and *fail* evaluate to themselves; The third rule deals with function application. It first evaluates the arguments of the function $D_1, \ldots D_n$ to $U_1, \ldots U_n$ respectively. Then, it applies the j-th rewrite rule of h, $h(U'_{j,1}, \ldots, U'_{j,n}) \to U'_j$, instantiated with the substitution σ, so $h(U_1, \ldots, U_n) = h(\sigma U'_{j,1}, \ldots, \sigma U'_{j,n})$ reduces into $\sigma U'_j$. The last line checks that rewrite rules before the j-th cannot be applied.

TABLE 5. Operational semantics.

$D \Downarrow U$
$fail \Downarrow fail$
$h(D_1, ..., D_n) \Downarrow \sigma U'_j$ if and only if
$D_1 \Downarrow U_1, ..., D_n \Downarrow U_n$,
def(h) consists of the rewrite rules
$h(U'_{i,1}, ..., U'_{i,n}) \to U'_i$ for $i \in \{1, ..., k\}$,
$\sigma U'_{j,1} = U_1, ...\sigma U'_{j,n} = U_n$, and
for all $i \leq j$, for all σ', $\sigma' U'_{i,1} \neq U_1, ..., \sigma' U'_{i,n} \neq U_n$.

$E, \mathcal{P} \subseteq \{0\} \to E, \mathcal{P}$	(Nil)
$E, \mathcal{P} \subseteq \{P \vert Q\} \to E, \mathcal{P} \subseteq \{P, Q\}$	(Par)
$E, \mathcal{P} \subseteq \{!P\} \to E, \mathcal{P} \subseteq \{P, !P\}$	(Repl)
$(\mathcal{N}_{pub}, \mathcal{N}_{priv}), \mathcal{P} \subseteq \{new\ a; P\} \to$ $(\mathcal{N}_{pub}, \mathcal{N}_{priv} \subseteq \{a'\}), \mathcal{P} \subseteq \{P\{a'/a\}\}$ where $a' \notin \mathcal{N}_{pub} \subseteq \mathcal{N}_{priv}$	(Res)
$E, \mathcal{P} \subseteq \{out(N, M); Q, in(N, x); P\} \to$ $E, \mathcal{P} \subseteq \{Q, P\{M/x\}\}$	(I/O)
$E, \mathcal{P} \subseteq \{let\ x = D\ in\ P\ else\ Q\} \to$ $E, \mathcal{P} \subseteq \{P\{M/x\}\}$ if $D \Downarrow M$	(Eval 1)
$E, \mathcal{P} \subseteq \{let\ x = D\ in\ P\ else\ Q\} \to$ $E, \mathcal{P} \subseteq \{Q\}$ if $D \Downarrow fail$	(Eval 2)
$E, \mathcal{P} \subseteq \{if\ true\ then\ P\ else\ Q\} \to$ $E, \mathcal{P} \subseteq \{P\}$	(Cond 1)
$E, \mathcal{P} \subseteq \{if\ M\ then\ P\ else\ Q\} \to$ $E, \mathcal{P} \subseteq \{Q\}$ if $M \neq true$	(Cond 2)

Second, we present the semantics of processes, by the reduction of semantic configurations. A semantic configuration is a pair $(E, \mathcal{P})$ where the environment E is a pair of two finite sets of names $(\mathcal{N}_{pub}, \mathcal{N}_{priv})$ and $\mathcal{P}$ is a finite multiset of closed processes. The set $\mathcal{N}_{pub}$ contains the public names, the set $\mathcal{N}_{priv}$ contains the private names, and the multiset of processes $\mathcal{P}$ contains the processes currently running. The configuration $((\{a_1, \ldots, a_n\}, \{b_1, \ldots, b_m\}), \{P_1, \ldots, P_k\})$ corresponds intuitively to the process **new** b_1;...**new** b_m; $(P_1 | \ldots | P_k)$. A configuration $((\mathcal{N}_{pub}, \mathcal{N}_{priv}), \mathcal{P})$ is valid when $\mathcal{N}_{pub}$ and $\mathcal{N}_{priv}$ are disjoint and $fn(\mathcal{P}) \subseteq \mathcal{N}_{pub} \cup \mathcal{N}_{priv}$. We only consider valid configurations. The reduction relation $\to$ on semantic configurations is defined in Table 5. The reduction rules define the semantics of each language construct. The rule Nil removes processes 0, since they do nothing. The rule Par expands parallel compositions. The rule Repl creates an additional copy of a replicated process; since this rule can be applied again on the resulting configuration, it allows creating an unbounded number of copies of the replicated process. The rule Res creates a fresh name a', substitutes it for a, and adds it to the private names $\mathcal{N}_{priv}$. The fresh name a' is required to occur neither in $\mathcal{N}_{priv}$, which contains the initial private free names as well as any fresh name created by a previous application of Res, nor in $\mathcal{N}_{pub}$, which contains the public free names. The rule (I/O) allows communication between processes. The message M is sent by the output and received by the input in the variable x,

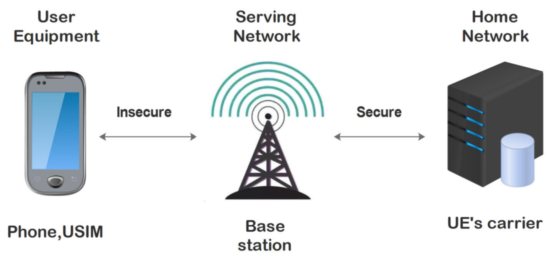


FIGURE 1. **The 5G network architecture.**

provided the output and input channels are equal. The rules (Eval 1) and (Eval 2) define the semantics of expression evaluations. They evaluate D. In case of success, (Eval 1) runs P with the result M of the evaluation substituted for x. In case of failure, (Eval 2) runs Q. The rules (Cond 1) and (Cond 2) define the semantics of conditionals. When M is true, (Cond 1) runs P. When M is different from true, (Cond 2) runs Q.

Given the protocol process P_0, it is usually running in parallel with an adversary process Q during the verification. In this case, the initial configuration is $(\mathcal{N}_{pub}, \mathcal{N}_{priv}), P_0, Q$.

IV. THE 5G EAP-TLS AUTHENTICATION PROTOCOL

In this section, we present a detailed review of the 5G EAP-TLS authentication protocol according to the 3GPP document TS 33.501 v15.4.0 [1].

The typical architecture of 5G network is shown in Fig. 1. It consists of three main entities: the user equipment (UE), the home network (HN) and the serving network (SN). The user equipment represents the subscriber's device (e.g., mobile phone) that is connected to the network. The home network is the subscriber's carrier and the key entity that is responsible for the user authentication. The serving network is where the user equipment may attach directly (e.g. the wireless network of a base station). Notice that in this work we distinguish between the home network, which is the network that the user signs up with, and the serving network, which is the actual base station that the mobile phone connects to. In some cases, the home network can be the same as the serving network, while in some other cases, such as the roaming scenario, they are different.

The user equipment usually communicates with the serving network through a wireless channel, which is public and under the attack of malicious users. Thus, it is reasonable to consider it as a insecure channel. In contrast, the channel between the home network and the serving network is private and internal to the network operators. this channel can be assumed to be secure, as in the related works on the analysis of 5G AKA protocols [19], [20].

In the following, we elaborate the key modules of the three entities that are relevant to the 5G EAP-TLS authentication protocol. In the user equipment, the main module is the Universal Subscriber Identity Module (USIM), which contains the following two key relevant elements:

- the Subscription Permanent Identifier ($SUPI$), that is used as a unique and permanent subscriber identity;
- the public asymmetric key pk_{HN} of the corresponding home network.

This module also contains a long-term symmetric key K, which is used as a shared secret between subscribers and their corresponding home networks, and a counter, called sequence number SQN. However, there are not used in the 5G EAP-TLS authentication protocol, but in the 5G AKA protocol.

The key module of the serving network is the Security Anchor Function (SEAF), which acts as a 'middle-man' during the authentication process between the user equipment and its home network. It can be viewed as a transparent authenticator by forwarding all messages between the user equipments and the home network. In contrast to the 5G AKA protocol, the serving network is not involved in the computation of the 5G EAP-TLS authentication process, but as a lower-level routing component.

The home network is the most sophisticated one, and contains the following key modules:

- the Authentication Server Function (AUSF), which performs authentication with the user equipment. When making the decision on authentication, it invokes a backend service for computing the authentication data and keying materials.
- the Unified Data Management (UDM) module, which is an entity that hosts functions related to data management. Two key functions are the Authentication Credential Repository and Processing Function (ARPF) and the Subscription Identifier De-concealing Function (SIDF). ARPF selects an authentication method based on the subscriber identity and configured policy and then computes the authentication data and keying materials for the AUSF. SIDF is responsible for decrypting the messages from the subscribers and also retrieving the subscriber's identity SUPI.

A. NORMAL EXECUTION OF THE PROTOCOL

When it is selected as the authentication method by the home network, the 5G EAP-TLS protocol is carried out between the user equipments and the home network through the serving network. We model the protocol specified in the 3GPP document TS 33.501 version 15.4.0 [1]. We would also like to remark that constructing the formal model of the 5G EAP-TLS protocol is not an easy task, given the sheer complexity of the specification document, which spans hundreds of pages. Moreover, the informal nature of the security requirements makes the modeling and analysis even harder. The detailed steps of the 5G EAP-TLS authentication protocol are depicted in Fig. 2.

1) In the beginning, the user equipment initiates a connection request, and forwards the encryption of $SUPI$ and a random number R_{UE} to the serving network. The encryption is denoted by $SUCI$, and the symbol $aenc(\cdot, pk_{UDM})$ represents the asymmetric encryption of the first element using the public key of UDM.

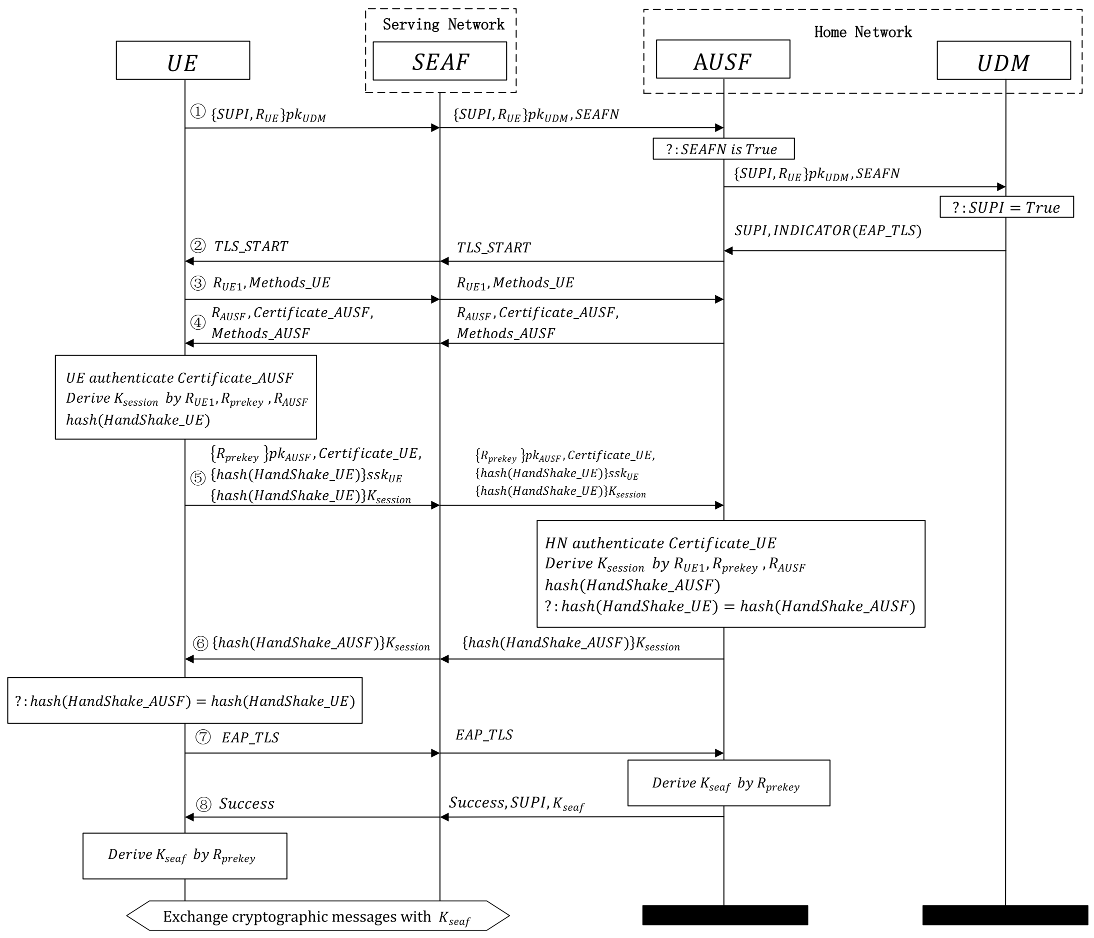

Serving Network
SEAF
Home Network
UE
AUSF
UDM
① {SUPI, R_UE}pk_UDM
{SUPI, R_UE}pk_UDM, SEAFN
?: SEAFN is True
{SUPI, R_UE}pk_UDM, SEAFN
?: SUPI = True
SUPI, INDICATOR(EAP_TLS)
② TLS_START
TLS_START
③ R_UE1, Methods_UE
R_UE1, Methods_UE
④ R_AUSF, Certificate_AUSF, Methods_AUSF
R_AUSF, Certificate_AUSF, Methods_AUSF
UE authenticate Certificate_AUSF
Derive K_session by R_UE1, R_prekey, R_AUSF
hash(HandShake_UE)
⑤ {R_prekey}pk_AUSF, Certificate_UE, {hash(HandShake_UE)}ssk_UE {hash(HandShake_UE)}K_session
{R_prekey}pk_AUSF, Certificate_UE, {hash(HandShake_UE)}ssk_UE {hash(HandShake_UE)}K_session
HN authenticate Certificate_UE
Derive K_session by R_UE1, R_prekey, R_AUSF
hash(HandShake_AUSF)
?: hash(HandShake_UE) = hash(HandShake_AUSF)
⑥ {hash(HandShake_AUSF)}K_session
{hash(HandShake_AUSF)}K_session
?: hash(HandShake_AUSF) = hash(HandShake_UE)
⑦ EAP_TLS
EAP_TLS
Derive K_seaf by R_prekey
⑧ Success
Success, SUPI, K_seaf
Derive K_seaf by R_prekey
Exchange cryptographic messages with K_seaf

FIGURE 2. The 5G EAP-TLS authentication protocol.

2) The serving network would start the authentication procedure when receiving the message *SUCI*, and forwards *SUCI* together with his name *SEAFN* to the home network.
3) The AUSF module of the home network will check whether the name *SEAFN* is the legal serving network's name. If the check passes, then AUSF sends the message to UDM, where the function SIDF is invoked to decrypt *SUCI* to obtain the subscriber's identity *SUPI*. UDM then checks if the obtained *SUPI* is a legal identity of the subscriber.
4) If the above check passes, the UDM module sends a response consisting of *SUPI* and the selected authentication method (denoted by $INDICATOR(EAP\_TLS)$) to AUSF. In this case, it indicates the 5G EAP-TLS protocol.
5) Then AUSF sends the message TLS_START to the user equipment via the serving network to signal the starting of the EAP-TLS authentication procedure.
6) The user equipment generates a new nonce R_{UE1} and sends it to SEAF with the information about its supported algorithms $Methods\_UE$. The SEAF will forward this message directly to AUSF.
7) The AUSF responds a message to the user equipment via SEAF, which contains a nonce R_{AUSF}, the home network certificate $Certificate\_AUSF$ and the information about its supported algorithms $Methods\_AUSF$.
8) When receiving the message $Certificate\_AUSF$, the user equipment first verifies the validity of this certificate. In case of verification success, the user equipment generates a new nonce R_{prekey}, which is called pre-master key, and derives a session key $K_{session}$ using R_{prekey}, R_{UE1} and R_{AUSF}. We refer to [2] for the details of this derivation. Then the user equipment computes the hash $hash(HandShake\_UE)$ of the previous handshaking messages (i.e. the messages in steps ②, ③ and ④). The user equipment forwards to the serving network SEAF the following messages: the encryption $\{R_{prekey}\}pk_{AUSF}$ of the pre-master key using the public key of AUSF, the encryption $\{hash(HandShake\_UE)\}K_{session}$ of the hash using the derived session key, and the signature

$\{hash(HandShake\_UE)\}ssk_{UE}$ with its own private signing key ssk_{UE}. Then SEAF forwards the above message to AUSF directly.

9) AUSF then checks the certificate of the user equipment. If it is valid, it decrypts $\{R_{prekey}\}pk_{AUSF}$ to obtain the pre-master key, and computes the session key $K_{session}$ together with the previous nonces R_{UE1} and R_{AUSF}. Then it obtains the hash $hash(HandShake\_UE)$ by decrypting $\{hash(HandShake\_UE)\}K_{session}$, and compares to the value in the signature $\{hash(HandShake\_UE)\}$ ssk_{UE} using the UE's public signing key spk_{UE}. AUSF also computes the hash $hash(HandShake\_AUSF)$ of his previous handshaking messages, and compares to the one received from the user equipment $hash(HandShake\_UE)$. If they are equal, AUSF then encrypts this hash with the session key $K_{session}$ and sends it back to the user equipment.
10) The user equipment decrypts the message and gets the hash $hash(HandShake\_AUSF)$, and compares to its own hash $hash(HandShake\_UE)$. If they are equal, the user equipment claims the success of authentication, and sends the message EAP_TLS to SEAF, which further forwards to AUSF.
11) When AUSF receives the message EAP_TLS, it generates a new key K_{seaf} based on the pre-master key R_{prekey} [2] and sends it to SEAF together with the identity of the user equipment *SUPI* and a *Success* message.
12) SEAF forwards the *Success* message to the user equipment, concluding the authentication procedure on the server-side. On receiving this message, the user equipment generates the key K_{seaf} in the same way as AUSF. The key K_{seaf} will then be used to secure the subsequent communications between the user equipment and the networks.

B. REQUIRED SECURITY PROPERTIES

The 3GPP document TS 33.501 describes the security requirements in an informal manner. We now present the parts that directly affect the EAP-TLS protocol. For each requirement, we point out the relevant text in the original document.

We cite the authentication and authorization requirements in Fig.3. The subscription authentication and UE authorization requirements shown in Fig.3 indicates that both the serving network and the home network should be able to authenticate the identity of the subscriber, such that only honest subscribers can have access to the networks. Then as the serving network authentication and serving network authorization requirements indicate, the subscribers must have the assurance that authentication can only be successful with the serving network authorized by their home network, such that a serving network cannot fake authentication requests with the home network for subscribers who are not attached to one of its base stations. In other words, the 5G EAP-TLS authentication protocol shall provide mutual authentication between subscribers and their home networks. The authentication is achieved when they agree on the identity of each other and also on the pre-master key used for deriving session keys. We thus interpret the requirements as the following authentication properties:

A1 Both the home network and the subscriber should agree on the identity of each other after successful termination.
A2 Both the home network and the subscriber should agree on the pre-master key R_{prekey} after successful termination.

We cite the confidentiality requirements in Fig.4 and Fig.5. We also assume that the user and the network have completed the exchange of their certificates and the public keys, which is generally carried out by means of off-band establishment or public key presetting.

Though the TS 33.501 document does not explicitly specify the secrecy of the pre-master key R_{prekey} or the session

5.1.2 Authentication and Authorization
The 5G system shall satisfy the following requirements.
Subscription authentication: The serving network shall authenticate the Subscription Permanent Identifier (SUPI) in the process of authentication and key agreement between UE and network.
Serving network authentication: The UE shall authenticate the serving network identifier through implicit key authentication.
UE authorization: The serving network shall authorize the UE through the subscription profile obtained from the home network. UE authorization is based on the authenticated SUPI.
Serving network authorization by the home network: Assurance shall be provided to the UE that it is connected to a serving network that is authorized by the home network to provide services to the UE. This authorization is 'implicit' in the sense that it is implied by a successful authentication and key agreement run.
Access network authorization: Assurance shall be provided to the UE that it is connected to an access network that is authorized by the serving network to provide services to the UE. This authorization is 'implicit' in the sense that it is implied by a successful establishment of access network security. This access network authorization applies to all types of access networks.

FIGURE 3. Authentication and authorization (from [1] p.21).

5.1.3 Confidentiality
The following security features are provided with respect to confidentiality of data on the network access link:
cipher key agreement: the property that the MS and the SN agree on a cipher key that they may use subsequently;
confidentiality of user data: the property that user data cannot be overheard on the radio access interface;

FIGURE 4. Confidentiality (from [38] p.15).

5.1.1 User identity confidentiality
The following security features related to user identity confidentiality are provided:
user identity confidentiality: the property that the permanent user identity (IMSI) of a user to whom a services is delivered cannot be eavesdropped on the radio access link;

FIGURE 5. User identity confidentiality (from [38] p.14).

key $K_{session}$, the cipher key agreement and confidentiality of user data shown in Fig.4 strongly implies the confidentiality of the session key $K_{session}$, which takes the pre-master key R_{prekey} as the seed. Moreover, the same session key $K_{session}$ should never be established twice. This will be analyzed as part of injective agreement properties on the established key $K_{session}$ for different pairs of parties. In addition, the user identity confidentiality shown in Fig.5 implies that SUPI should be considered sensitive and must remain secret since it uniquely identifies users.

In details, we interpret the requirements as the following secrecy properties.

S1 The attacker cannot obtain the identity *SUPI* of an honest subscriber.
S2 The attacker cannot obtain the pre-master key R_{prekey} of an honest subscriber.
S3 The attacker cannot obtain the session key $K_{session}$ of an honest subscriber.

V. FORMAL MODEL OF THE 5G EAP-TLS AUTHENTICATION PROTOCOL

In this section, we present the formal model of the 5G EAP-TLS protocol in applied pi calculus, and then we present the formalization of the intended security properties.

A. SYMBOLIC PROTOCOL MODEL

1) ATTACKER MODEL AND THE PERFECT CRYPTOGRAPHY ASSUMPTION

We consider all protocol participants (i.e., user equipments and networks) are honest. They all behave faithfully according to the protocol specification. we also consider the existence of a malicious attacker, that has full control over the network. The attacker can send, intercept, forge and replay messages on public channels. However, the attacker's ability to intercept the messages is restricted under the *perfect cryptography assumption*, i.e. the attacker can only decrypt an encrypted message if and only if he possesses the right key. This attacker model is the so-called Dolev-Yao model [17].

In the perfect cryptography assumption, we interpret the cryptographic primitives as symbolic functions. Below we specify the symmetric encryption by a binary constructor *senc*, which takes two arguments of type *bitstring* and *key* respectively, and returns the encryption of type *bitstring*. The symmetric decryption is specified by a constructor *sdec*. The arithmetic property of encryption and decryption is characterized by the equation $sdec(senc(m, k), k) = m$, which means that given the cipher text $senc(m, k)$, decrypting with the same key k would return the message m. While decrypting with some other key would not be successful.

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

For asymmetric encryption and decryption, we define a unary constructor *pk* to capture the notion of constructing a key pair, which takes a private key of type *skey* as an argument and returns a public key of type *pkey*. Similarly, encryption and decryption are captured by operators *aenc* and *adec* respectively.

```
fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall m:bitstring,sk:skey; adec(aenc(m,pk(sk)),sk)=m.
```

In a similar manner to the asymmetric encryption, digital signature makes use of a pair of keys. One is the private key of type *sskey* for signing the message, and the other is the public key of type *spkey* for checking the signature. We use the constructor *spk* to map public keys to the corresponding private keys. The constructor *sign* for constructing signatures with the private key is standard. The destructor *checksign* checks the signature with the public key and returns the message when the signature is correct.

```
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall m: bitstring, k: sskey;
checksign(sign(m,k),spk(k))=m.
```

Additionally, we define a hash function, and a type converter function for modeling purposes. The hash function is represented as a constructor *h* that takes three input messages and returns the digest of these messages. The type converter is simply a special data constructor *b2k*, that takes an input of type *bitstring* and returns a value of type *key*.

```
fun h(bitstring,bitstring,bitstring): bitstring.
fun b2k(bitstring):key.
```

According to our previous discussion, UE communicates with SEAF through a wireless channel, which is public and under the attack of malicious users. Thus, we use the public channel $c1$ to represent the channel between UE and SEAF. In contrast, the channel between AUSF and SEAF, as well as AUSF and UDM, are private and internal to the network operators. Therefore, we declare the private channel $c2$ to represent the channel between SEAF and AUSF and the private channel $c3$ to represent the channel between AUSF and UDM.

```
free c1:channel.
free c2:channel [private].
free c3:channel [private].
```

2) USER EQUIPMENT PROCESS

It first declares a nonce *Rue*, then encrypts both *SUPI* and *Rue* with the public key of UDM, and outputs the cipher-text on a public channel $c1$. We remark that this public channel is accessible to attackers. Then the UE process waits for an input message (bound to variable *Startx* on channel $c1$. Afterwards, UE generates and outputs a new nonce *Rue*1 on channel $c1$, and waits for an input of the form (*Rausfx*, *CertAUSFx*). When receiving this message, UE checks if the second element of this tuple is a signature of the home network's certificate *CertAUSF*. If it is valid, UE generates a nonce *Rprekey* representing the pre-master key, and computes the hash of *Rue*1, *Rausfx* and *Rprekey*, which is converted to a key *Ksession* using the type converter *b_to_k*. UE then computes the encryption *senc*(*HSUE*, *Ksession*), the signature *sign*(*HSUE*, *sskUE*) of the handshaking messages *HSUE*, and the encryption *aenc*(*Rprekey*, *pkAUSF*) of the pre-master key *Rprekey* using the public key *pkAUSF* of the home network. After sending these messages on channel $c1$ together with the user's certificate *CertUE*, UE waits for an input *HSAUSFx* and decrypts it using the key *Ksession*. If the decryption returns the right handshaking messages (i.e., it equals to *HSUE*), then UE sends the message *EAPM*, which is used to inform AUSF that the authentication has been passed. and waiting the success message *SUCMx* from the network. Additionally, we insert events *acceptsUE*(*Rue*1), *termUE*(*Rausfx*) and *sendPrek*(*prekey*) in the process to encode the authentication properties.

```
let UE(pkAUSF:pkey,pkUDM:pkey,spkAUSF:spkey,
spkUE:spkey,sskUE:sskey) =
new Rue:bitstring;
out(c1,aenc((SUPI,Rue),pkUDM));
in(c1,Startx:bitstring);
new Rue1:bitstring;
out(c1,Rue1);
in(c1,(Rausfx:bitstring,CertAUSFx:bitstring));
let (=CertAUSF) = CertAUSFx in
new Rprekey:bitstring;
let prekey = Rprekey in
let a = h(Rue1,Rausfx,prekey) in
let Ksession = b_to_k(a) in
let HSUE = (Startx,Rue1,Rausfx,CertAUSFx) in
event acceptsUE(Rue1);
event sendPrek(prekey);
out(c1,(aenc(prekey,pkAUSF),CertUE,
sign(HSUE,sskUE),senc(HSUE,Ksession)));
in(c1,HSAUSFx:bitstring);
let (=HSUE) = sdec(HSAUSFx,Ksession) in
new EAPM:bitstring;
out(c1,EAPM);
in(c1,SUCMx);
event termUE(Rausfx).
```

3) SERVING NETWORK PROCESS

We treat the serving network as a lower-level routing component and model its behavior using the following *SEAF* process. Its main functionality is to forward the messages between the user equipment and the home network in a transparent manner. Moreover, in our trust model, the connection between the serving network and home network is assumed to be secure and inaccessible to attackers. Thus, we declare this channel (i.e., the channel $c2$ in the process) to be private. On the contrary, the channel between the user equipment and the serving network is public and accessible to attackers (i.e., the channel $c1$ declared in the following process).

```
let SEAF(pkAUSF:pkey,pkUDM:pkey,spkAUSF:spkey,
spkUE:spkey) =
in(c1,x1:bitstring);
out(c2,(x1,SEAFN));
in(c2,x2:bitstring);
out(c1,x2);
in(c1,x3:bitstring);
out(c2,x3);
in(c2,(x4:bitstring,x5:bitstring));
out(c1,(x4,x5));
in(c1,(x6:bitstring,x7:bitstring,
x8:bitstring,x9:bitstring));
out(c2,(x6,x7,x8,x9));
in(c2,x10:bitstring);
out(c1,x10);
in(c1,EAPMx);
out(c2,EAPMx);
in(c2,SUCMx);
out(c1,SUCMx).
```

4) HOME NETWORK PROCESSES

We model the home network using two processes, the *AUSF* process and the *UDM* process. In AUSF process, it first receives an input message (*SUPIx*, *SEAFNx*) on channel $c2$ from *SEAF*, and checks whether the *SEAFNx* is the legal serving network name *SEAFN*. If the check passes, it forwards this message to UDM on the private channel $c3$. Then AUSF receives a message *Startx* on channel c3 from UDM and forwarding this message to SEAF. Then AUSF receives a message and bounds to the variable *Rue*1*x* and generates a new nonce *Rausf* and outputs it together with its certificate. AUSF then waits for a message that consisting of four elements (bounding to the variable y, *CertUEx*, t and z respectively). AUSF checks if *CertUEx* is the UE's certificate, and if it is the case, it bounds the decryption of y to the variable *prekeyx*. AUSF then computes the hash of *Rue*1*x*, *Rausf* and *prekeyx*, and converts this hash to a key *Ksessionx*. Afterwards, AUSF will decrypt the input message z using key *Ksessionx* and check the signature of t using the public key *spkUE* of UE. Both checks should return the handshaking messages denoted by the tuple *HSAUSF*. If the checks succeed, AUSF will output the encryption of *HSAUSF* using the key *Ksessionx*. And then AUSF waits to send the success message *SUCM* after receiving the message *EAPMx*. Similarly, we add event *acceptPrek*(*prekeyx*), *acceptsAUSF*(*Rausf*) and *termAUSF*(*Rue*1*x*) for checking authentication properties.

Accordingly, in the UDM process, it receives the input message $(x, SEAFNx)$ on channel $c3$, and then decrypts x using its private key, which returns a tuple $(SUPIx, Ruex)$. The UDM process continues if the first element of the tuple is the identity *SUPI* of UE. Then UDM forwards the a starting information *start* to AUSF to initiate the handshaking.

```
let AUSF(skAUSF:skey,pkUE:pkey,spkUE:spkey,
sskAUSF:sskey,spkAUSF:spkeyg) =
in(c2,(SUPIx:bitstring,SEAFNx:bitstring));
if SEAFNx = SEAFN then
out(c3,(SUPIx,SEAFNx));
in(c3, Startx:bitstring);
out(c2, Startx);
in(c2,(Rue1x:bitstring));
new Rausf:bitstring;
out(c2,(Rausf,CertAUSF));
in(c2,(y:bitstring,CertUEx:bitstring,t:bitstring,
z:bitstring));
let (=CertUE) = CertUEx in
let prekeyx = adec(y,skAUSF) in
let b = h(Rue1x,Rausf,prekeyx) in
let Ksessionx = b_to_k(b) in
new HSAUSF:bitstring;
let HSAUSF = (Startx,Rue1x,Rausf,CertAUSF) in
let (=HSAUSF) = sdec(z,Ksessionx) in
let (=HSAUSF) = checksign(t,spkUE) in
event acceptPrek(prekeyx);
event acceptsAUSF(Rausf);
out(c2,senc(HSAUSF,Ksessionx));
in(c2,EAPMx);
new SUCM:bitstring;
out(c2,SUCM);
event termAUSF(Rue1x).

let UDM(skUDM:skey) =
in(c3,(x:bitstring,SEAFNx:bitstring));
let (SUPIx:bitstring,Ruex:bitstring) = adec(x,skUDM) in
if SUPIx = SUPI then
new start:bitstring;
out(c3,start).
```

5) PROTOCOL PROCESS

It first generates the private keys for asymmetric encryption and signature, and outputs the corresponding public keys on channel $c1$, $c2$ and $c3$. The *SEAFN* is the serving network name that is used in the derivation of the anchor key. It is also broadcasted on the public channel and accessible to attackers. Then the protocol process is the parallel composition of infinite replication of UE, SEAF, AUSF and UDM processes.

```
process
new skUE:skey;
new skAUSF:skey;
new skUDM:skey;
new sskAUSF:sskey;
new sskUE:sskey;
new SEAFN:bitstring;
let pkUE = pk(skUE) in out(c1,pkUE);
out(c2,pkUE);out(c3,pkUE);
let pkAUSF = pk(skAUSF) in out(c1,pkAUSF);
out(c2,pkAUSF);out(c3,pkAUSF);
let pkUDM = pk(skUDM) in out(c1,pkUDM);
out(c2,pkUDM);out(c3,pkUDM);
let spkAUSF = spk(sskAUSF) in out(c1,spkAUSF);
out(c2,spkAUSF);out(c3,spkAUSF);
let spkUE = spk(sskUE) in out(c1,spkUE);
out(c2,spkUE);out(c3,spkUE);
out(c1,SEAFN);out(c2,SEAFN);out(c3,SEAFN);
( (!UE(pkAUSF,pkUDM,spkAUSF,spkUE,sskUE)) |
(!SEAF(pkAUSF,pkUDM,spkAUSF,spkUE)) |
(!AUSF(skAUSF,pkUE,spkUE,sskAUSF,spkAUSF))
| (!UDM(skUDM)))
```

B. SECURITY PROPERTY SPECIFICATION

In ProVerif, a fact is modeled as a ground term. To prove the secrecy of a term M, ProVerif essentially solves a reachability problem, i.e., whether the attacker can reach a state where the term M is available. In this work, ProVerif takes the following queries in the protocol model to check the secrecy of the user's identity *SUPI* and the pre-master key *Rprekey*:

$$query\ \ attacker(SUPI)$$
$$query\ \ attacker(prekey)$$

Authentication properties are captured by correspondence assertions, which can express the relationships between events in the form "if some event has been executed in the protocol, then some other event has been previously executed." In ProVerif, events are of the form $event\ \ e(M_1, \ldots, M_n)$, and the query of a correspondence assertion is

$$query\ \ x_1 : t_1, \ldots, x_n : t_n;\ event(e(M_1, \ldots, M_j)) \Rightarrow event(e'(N_1, \ldots, N_k))$$

where terms $M_1, \ldots, M_j$, $N_1, \ldots, N_k$ are built by applying constructors to variables $x_1, \ldots, x_n$. The query is satisfied if, for each occurrence of event $e(M_1, \ldots, M_j)$, there is a previous execution of event $e'(N_1, \ldots, N_k)$. There is also a stronger variant of correspondence assertion, where can capture the one-to-one relationship between events. They are often called injective correspondence assertions, which take the form:

$$query\ \ x_1 : t_1, \ldots, x_n : t_n;\ inj - event(e(M_1, \ldots, M_j)) \Rightarrow inj - event(e'(N_1, \ldots, N_k))$$

Informally, this correspondence asserts that, for each occurrence of the event $e(M_1, \ldots, M_j)$, there is a distinct earlier occurrence of the event $e'(N_1, \ldots, N_k)$. This differs from the previous correspondence assertions in that no single event $e'(N_1, \ldots, N_k)$ can map to two more event $e(M_1, \ldots, M_j)$.

In this work, we declare the following events in the formal model:

- *event acceptsUE*(x), indicating the user believes that she has accepted to run the protocol with the home network and the supplied parameter;

- *event acceptsAUSF*(x), indicating the home network believes that she has accepted to run the protocol with the client and the supplied parameter;
- *event termUE*(x), indicating the user believes that she has terminated a protocol run using the given parameter;
- *event termAUSF*(x), indicating the home network believes that she has terminated a protocol run using the given parameter;
- *event sendPrek*(x), indicating the user believes that she has transmitted a pre-key to the home network given as the parameter;
- *event acceptPrek*(x), indicating the home network believes that she has accepted a pre-key from the user given as the parameter.

Therefore, we consider the following correspondence assertions to prove authentication properties.

- $query\ x : bitstring; inj - event(termAUSF(Rue1x)) \Rightarrow inj - event(acceptsUE(Rue1))$.
- $query\ x : bitstring; inj - event(termUE(Rausfx)) \Rightarrow inj - event(acceptsAUSF(Rausf))$.
- $query\ x : key; inj - event(acceptPrek(prekeyx)) \Rightarrow inj - event(sendPrek(prekey))$.

Intuitively, the first assertion means that whenever the network terminates a protocol run, there exists a user who has accepted to run with the network. Notice that we use injective correspondence assertion to prevent the cases where a single client session is authenticated to multiple server sessions. This may happen when attackers could replay the client's messages. The meaning of the last two assertions is defined similarly.

VI. VERIFICATION RESULTS AND DISCUSSIONS

We have verified whether the formal protocol model satisfies the security properties listed in Section.IV. We take ProVerif 2.00[1] as the verification engine, and the experiments are carried out in a PC equipped with the Windows 10 professional OS and Intel Core i5-7300HQ with 2.5GHz CPU and 8.0GB RAM. The overall time to verify the properties is about three seconds.

The results are presented in Table 6. As the results show, all the secrecy properties (i.e. S1, S2 and S3) are satisfied, while the agreement properties (i.e. A1 and A2) are violated. ProVerif can generate counterexamples for the violated properties. For the simplicity of the presentation, we do not report the tedious outputs of ProVerif in this paper but elaborate on the counterexamples and possible fixes. Notice that in all counterexamples we mark the bogus messages from the attacker in red. The protocol model and results are publicly available in the Github repository.[2]

A. VIOLATION OF PROPERTY A1

This property requires that when the participants terminate a successful protocol execution, they should agree on the identity of each other to achieve mutual authentication. However, our analysis shows that in the current 5G EAP-TLS protocol, this property is violated in that the user cannot authenticate the identity of the home network. The counterexample is shown in Fig.6. It shows that an attacker is able to impersonate the home network and establish a connection with the user. However, the user is unaware of whether she is connected to a legal home network or an attacker. We elaborate on the feasibility of this counterexample in the following.

TABLE 6. Verification results.

Security property	Result
A1. Both the home network ($AUSF$) and the subscriber (UE) should agree on the identity of each other after successful termination.	False
A2. Both the home network ($AUSF$) and the subscriber (UE) should agree on the pre-master key (R_{prekey}) after successful termination.	False
S1. The adversary must not be able to obtain the $SUPI$ of an honest subscriber.	True
S2. The adversary must not be able to obtain the pre-master key (R_{prekey}) of an honest subscriber.	True
S3. The adversary must not be able to obtain the session key ($K_{session}$) of an honest subscriber.	True

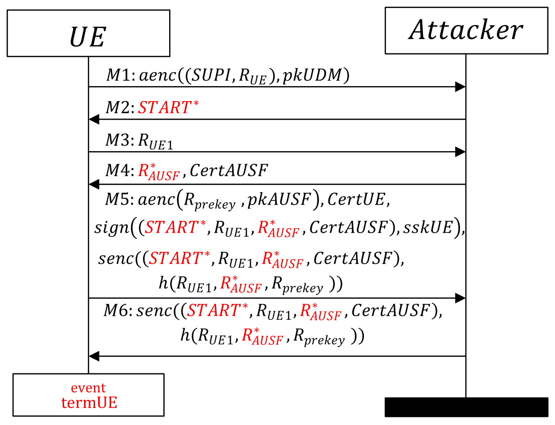

UE
Attacker
$M1: aenc((SUPI, R_{UE}), pkUDM)$
$M2: START^*$
$M3: R_{UE1}$
$M4: R^*_{AUSF}, CertAUSF$
$M5: aenc(R_{prekey}, pkAUSF), CertUE, sign((START^*, R_{UE1}, R^*_{AUSF}, CertAUSF), sskUE), senc((START^*, R_{UE1}, R^*_{AUSF}, CertAUSF), h(R_{UE1}, R^*_{AUSF}, R_{prekey}))$
$M6: senc((START^*, R_{UE1}, R^*_{AUSF}, CertAUSF), h(R_{UE1}, R^*_{AUSF}, R_{prekey}))$
event termUE

FIGURE 6. The counterexample for property A1.

When the user outputs the encryption of identity *SUPI* and random number R_{UE}, the attacker replies with a *START* message, which can be obtained by caching the previous communications of the home network. Since it is not required to check the legality of this message, the user will continue executing the protocol and outputs a new random number R_{UE1} in plaintext. Then the attacker will forge the reply (R^*_{AUSF}, *CertAUSF*) with his own random number R^*_{AUSF} and the home network's certificate, which is publicly available. We remark that from the user's point of view, whether the random number R^*_{AUSF} is from the attacker or the home network is unpredictable. Upon receiving this message from the attacker, the user will then compute the signature and encryption of the handshaking messages in the same way as before, and wait for the encryption of the handshaking messages from the network's side. However, the attacker can simply intercept the user's output and reply with the

[1]https://prosecco.gforge.inria.fr/personal/bblanche/proverif/
[2]https://github.com/bxk2008/5G-tls-protocol-analysis.git

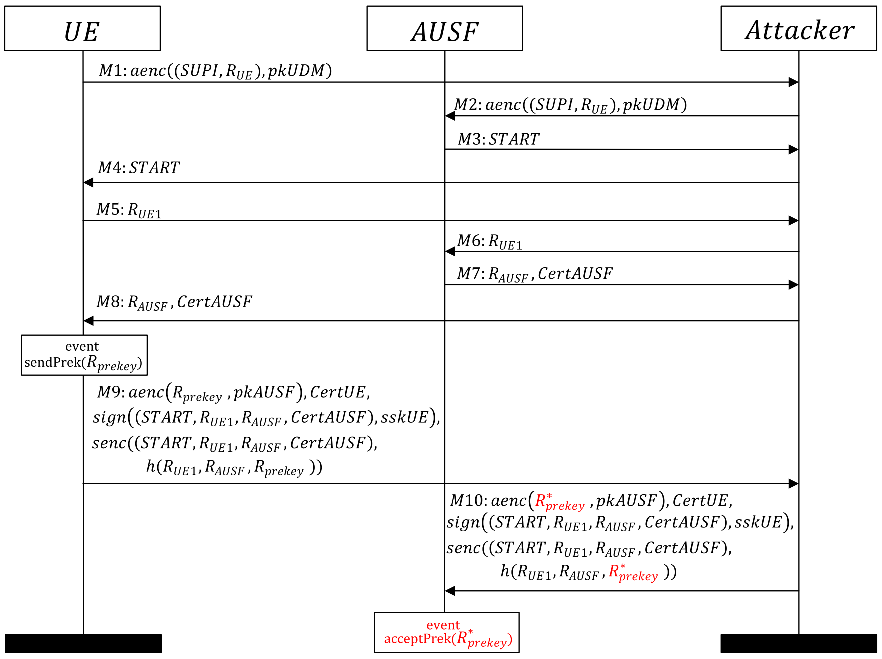

UE
AUSF
Attacker
M1: aenc((SUPI, R_UE), pkUDM)
M2: aenc((SUPI, R_UE), pkUDM)
M3: START
M4: START
M5: R_UE1
M6: R_UE1
M7: R_AUSF, CertAUSF
M8: R_AUSF, CertAUSF
event sendPrek(R_prekey)
M9: aenc(R_prekey, pkAUSF), CertUE, sign((START, R_UE1, R_AUSF, CertAUSF), sskUE), senc((START, R_UE1, R_AUSF, CertAUSF), h(R_UE1, R_AUSF, R_prekey))
M10: aenc(R*_prekey, pkAUSF), CertUE, sign((START, R_UE1, R_AUSF, CertAUSF), sskUE), senc((START, R_UE1, R_AUSF, CertAUSF), h(R_UE1, R_AUSF, R*_prekey))
event acceptPrek(R*_prekey)

FIGURE 7. The counterexample for property A2.

user's encryption. The user is unable to tell whether the reply is from the attacker or the home network. And at the end of this execution, the user believes that she has established an authenticated connection with the home network.

B. VIOLATION OF PROPERTY A2

This property requires that after the successful execution of the protocol, both the user and the home network should agree on the pre-master key that is used to generate the session key. The counterexample that demonstrates this property violation is shown in Fig.7. This is a typical man-in-the-middle attack. In the beginning, the attacker simply intercepts and forwards messages (i.e. the messages $M1$ and $M5$ from the user, and the messages $M3$ and $M7$ from the home network) between the user and the home network. When receiving the message $M9$, the attacker would replace the encryption of the user's pre-master key *Rprekey* with the encryption of his own key *Rprekey**, and forge a reply message $M10$ to the home network. This forgery is possible since the attacker is caching all the handshaking messages between the user and the home network, and the home network cannot tell if the pre-master key *Rprekey** is from the attacker or not. Thus, when the execution terminates, the user believes that she has established a pre-master key *Rprekey* with the home network (i.e. event $sendPrek(prekey)$), However, the home network believes that it is the key *Rprekey** being established (i.e. event $acceptPrek(prekey^*)$).

C. LESSONS LEARNED AND A POSSIBLE fix

Regarding the violation of property A1, the main reason is that the messages to be checked are transmitted

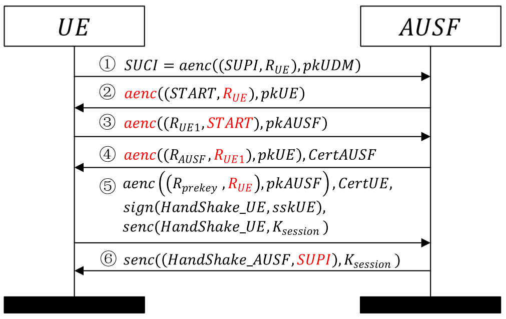

UE
AUSF
① SUCI = aenc((SUPI, R_UE), pkUDM)
② aenc((START, R_UE), pkUE)
③ aenc((R_UE1, START), pkAUSF)
④ aenc((R_AUSF, R_UE1), pkUE), CertAUSF
⑤ aenc((R_prekey, R_UE), pkAUSF), CertUE, sign(HandShake_UE, sskUE), senc(HandShake_UE, K_session)
⑥ senc((HandShake_AUSF, SUPI), K_session)

FIGURE 8. The revised 5G EAP-TLS protocol.

in plaintext. The attacker can easily forge the plaintext handshaking messages between the user and the home network. As for the second case, the main reason is the lack of the challenge-response mechanism between the user and the home network. Neither the user nor the home network is aware of whether the messages are sent one-to-one by the correct counterparts or not. When a user or a home network receives a message, he should be able to verify that the message is indeed from the legitimate role with which it communicates at this point in time. For example, when the user sends the message $SUCI$ to the home network, the user should be able to check that the following message $START$ is received from the home network, which has received the previous message $SUCI$. This is achieved by including the random number R_{UE} generated by the user in the reply, since only the legal home network can decrypt the first message to obtain R_{UE}. As a possible fix, we propose a revised 5G EAP-TLS authentication design between UE and AUSF in Fig.8. Since the serving network is not involved in the computation of the 5G EAP-TLS authentication process, we simplify the presentation of the protocol design into two parties. The fix is based on asymmetric cryptography and the newly added elements are marked in red. Our analysis shows that this revised protocol satisfies all the security properties.

Remarks: In [20], the authors discover an attack that exploits a potential race condition, where allowing sessions for different users to be confused causes the violation of both secrecy and authentication properties. However, their work is based on the older version of the document (TS 33.501 V0.7.0). We do not find this attack in the latest version.

VII. CONCLUSION AND FUTURE WORK

In this work, we investigate the security properties of 5G EAP-TLS authentication protocol that is being standardized by 3GPP for the next-generation cellular network. Our analysis is based on a formal symbolic approach, in which we model the protocol and its security properties in the applied pi calculus and carry out the analysis using model checker ProVerif. Our analysis reveals several design flaws and counterexamples are reported to show the possibilities of these flaws. We need to point out that these flaws are found on the formal model level, which however would also exist in the real systems if the protocol is implemented as it is. We also proposed several strategies to repair these vulnerabilities and proved that the revised protocol satisfies all the security properties.

Regarding the disadvantages of the current work, we would like to remark that the analysis results by ProVerif are based on the symbolic protocol model, where we assume the cryptography is perfect, and we do not take into account the computational strengths of the primitives. Though, this assumption is of theoretical research interest, it is too strong to be practical. Thus, if the underlying cryptographic primitives are broken, the protocol would also be faulty, even though it is proven correct and secure on the symbolic model.

In the future, we would like to go one step further to investigate the correctness of the protocol implementations, with respect to the specification. The idea is that ensuring the security of the protocol design is not enough, and we need also to ensure that the implementation of the protocol state machine is secure. One possible technique to achieve this goal is the protocol state fuzzing technique [32]. We will also extend the current work to the computational cryptography model, where the probability of breaking cryptographic primitives is taken into account. In addition, we would like to use some machine learning techniques such as neural networks [39] and online learning algorithms [40] to automate the process of protocol validation.

REFERENCES

[1] *Security Architecture and Procedures for 5G System*, document 3GPP, TS 33.501, v15.4.0.2019, Mar. 2019

[2] *The Transport Layer Security (TLS) Protocol Version 1.2*, document RFC5246, Aug. 2008.

[3] D. Basin, C. Cremers, K. Miyazaki, S. Radomirovic, and D. Watanabe, "Improving the security of cryptographic protocol standards," *IEEE Secur. Privacy*, vol. 13, no. 3, pp. 24–31, May 2015.

[4] L. Hirschi, R. Sasse, and J. Dreier, "Security issues in the 5G standard and how formal methods come to the rescue," ERCIM News, Paris, France, Tech. Rep., Apr. 2019, vol. 2019, no. 117. [Online]. Available: https://ercim-news.ercim.eu/en117/special/security-issues-inthe-5g-standard-and-how-formal-methods-come-to-the-rescue

[5] D. Basin, C. Cremers, and C. Meadows, "Model checking security protocols," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds. Cham, Switzerland: Springer, 2018, pp. 727–762, doi: 10.1007/978-3-319-10575-8_22.

[6] G. Lowe, "An attack on the Needham-Schroeder public-key authentication protocol," *Inf. Process. Lett.*, vol. 56, no. 3, pp. 131–133, Nov. 1995.

[7] J. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using Murϕ," in *Proc. IEEE Symp. Secur. Privacy*, Nov. 2002, pp. 141–151.

[8] D. X. Song, "Athena: A new efficient automatic checker for security protocol analysis," in *Proc. 12th IEEE Comput. Security Found. Workshop*, Jan. 2003, pp. 192–202.

[9] E. M. Clarke, S. Jha, and W. Marrero, "Verifying security protocols with Brutus," *ACM Trans. Softw. Eng. Methodol.*, vol. 9, no. 4, pp. 443–487, Oct. 2000.

[10] A. Armando and L. Compagna, "SATMC: A SAT-based model checker for security protocols," in *Logics in Artificial Intelligence*, J. J. Alferes and J. Leite, Eds. Berlin, Germany: Springer, 2004, pp. 730–733.

[11] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: A symbolic model checker for security protocols," *Int. J. Inf. Secur.*, vol. 4, no. 3, pp. 181–208, Jun. 2005.

[12] V. Cortier, S. Delaune, and P. Lafourcade, "A survey of algebraic properties used in cryptographic protocols," *J. Comput. Secur.*, vol. 14, no. 1, pp. 1–43, Feb. 2006.

[13] B. Blanchet, "Modeling and verifying security protocols with the applied pi calculus and ProVerif," *Found. Trends Privacy Secur.*, vol. 1, nos. 1–2, pp. 1–135, 2016.

[14] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Dallas, TX, USA, Oct./Nov. 2017, pp. 1773–1788.

[15] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proc. 14th IEEE Comput. Secur. Found. Workshop*, Aug. 2005, pp. 82–96.

[16] M. Abadi, B. Blanchet, and C. Fournet, "The applied pi calculus: Mobile values, new names, and secure communication," *J. ACM*, vol. 65, no. 1, pp. 1:1–1:41, Oct. 2017.

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

[18] B. Blanchet, "Using Horn clauses for analyzing security protocols," in *Formal Models and Techniques for Analyzing Security Protocols* (Cryptology and Information Security Series), vol. 5, V. Cortier and S. Kremer, Eds. Amsterdam, The Netherlands: IOS Press, 2011, pp. 86–111.

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

[20] C. Cremers and M. Dehnel-Wild, "Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion," in *Proc. 26th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2019, pp. 1–15.

[21] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *Computer Aided Verification,*, N. Sharygina and H. Veith, Eds. Berlin, Germany: Springer, 2013, pp. 696–701.

[22] D. Basin, C. Cremers, J. Dreier, and R. Sasse, "Symbolically analyzing security protocols using tamarin," *ACM SIGLOG*, vol. 4, no. 4, pp. 19–30, Oct. 2017.

[23] 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. 64040–64052, 2019.

[24] A. Koutsos, "The 5G-AKA authentication protocol privacy," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Stockholm, Sweden, Jun. 2019, pp. 464–479.

[25] G. Bana and H. Comon-Lundh, "Towards unconditional soundness: Computationally complete symbolic attacker," in *Proc. 1st Int. Conf. Princ. Secur. Trust (ETAPS)*, Tallinn, Estonia, Mar./Apr. 2012, pp. 189–208.

[26] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Scottsdale, AZ, USA, Nov. 2014, pp. 609–620.

[27] R. Borgaonkar, L. Hirschi, S. Park, and A. Shaik, "New privacy threat on 3G, 4G, and upcoming 5G AKA protocols," *Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 108–127, 2019.

[28] J. Zhang, Q. Wang, L. Yang, and T. F. and, "Formal verification of 5G-EAP-TLS authentication protocol," in *Proc. 4th IEEE Int. Conf. Data Sci. Cyberspace (DSC)*, Hangzhou, China, Jun. 2019, pp. 503–509.

[29] B. Blanchet, "Composition theorems for cryptoverif and application to TLS 1.3," in *Proc. 31st IEEE Comput. Secur. Found. Symp. (CSF)*, Oxford, U.K., Jul. 2018, pp. 16–30.

[30] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 4, pp. 193–207, Oct. 2008.

[31] K. Bhargavan, B. Blanchet, and N. Kobeissi, "Verified models and reference implementations for the TLS 1.3 standard candidate," in *Proc. IEEE Symp. Secur. Privacy (SP)* 2017, San Jose, CA, USA, May 2017, pp. 483–502.

[32] J. de Ruiter and E. Poll, "Protocol state fuzzing of TLS implementations," in *Proc. 24th USENIX Secur. Symp., USENIX Secur.*, vol. 15, Washington, DC, USA, Aug. 2015, pp. 193–206.

[33] N. Kobeissi, K. Bhargavan, and B. Blanchet, "Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Paris, France, Apr. 2017, pp. 435–450.

[34] R. P. Jover and V. Marojevic, "Security and protocol exploit analysis of the 5G specifications," *IEEE Access*, vol. 7, pp. 24956–24963, 2019.

[35] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5G technologies: Potential solutions, recent advancements and future directions," *IEEE Commun. Surveys Tuts.*, to be published.

[36] C. J. F. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols," in *Computer Aided Verification*, A. Gupta and S. Malik, Eds. Berlin, Germany: Springer, 2008, pp. 414–418.

[37] H. Comon-Lundh, S. Delaune, and J. K. Millen, "Constraint solving techniques and enriching the model with equational theories," in *Formal Models and Techniques for Analyzing Security Protocols* (Cryptology and Information Security Series), vol. 5, V. Cortier and S. Kremer, Eds. Amsterdam, The Netherlands: IOS Press, 2011, pp. 35–61.

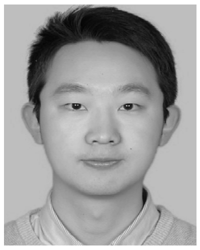
[38] *3G Security; Security Architecture*, document TS 33.102, v11.5.1, 3GPP, Jun. 2013.

[39] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, Jan. 2018, doi: 10.1016/j.neucom.2017.08.040.

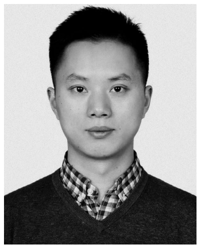
[40] W. Cao, J. Gao, Z. Ming, S. Cai, and Z. Shan, "Fuzziness-based online sequential extreme learning machine for classification problems," *Soft. Comput.*, vol. 22, no. 11, pp. 3487–3494, Jun. 2018, doi: 10.1007/s00500-018-3021-4.

JINGJING ZHANG received the B.S. degree in information security from Information Science and Technology University, Beijing, China, and the M.S. degree in electronics and communication engineering from the Beijing University of Posts and Telecommunications, Beijing. He is currently pursuing the Ph.D. degree with the College of Command and Control Engineering, Army Engineering University of PLA. He has been an exchange student with the National Key Laboratory of Science and Technology on Information System Security, Institute of Systems Engineering, Beijing, Since 2018. His research topic is mainly about formal analysis of security protocols from the perspective of design models and implementations.

LIN YANG received the B.S. and M.S. degrees in automation and the Ph.D. degree in communication and electronic system from the National University of Defense Technology, Changsha, Hunan, in 1995 and 1998, respectively. He is currently a Researcher Fellow with the National Key Laboratory of Science and Technology on Information System Security, Institute of Systems Engineering, Beijing. His research interests include computer security, information system security, network security, trusted computing, moving target defense, security protocol analysis, and big-data security.



WEIPENG CAO received the Ph.D. degree in computer science from Shenzhen University, in June 2019. Since 2017, he has been serving as a Visiting Scholar with the School of Engineering and Computer Science, University of the Pacific, Stockton, CA, USA. He is currently an Associate Researcher with the College of Computer Science and Software Engineering, Shenzhen University. His research interests include machine learning and deep learning.



QIANG WANG received the bachelor's and master's degree from the National University of Defense Technology, in 2010 and 2012, respectively, and the Ph.D. degree from EPFL, Switzerland, in 2017, where he has been a key person in the development of the component-based embedded system design framework BIP. His current research interest focuses on the formal verification techniques and tools for safety and security critical systems.