VWAnalyzer: A Systematic Security Analysis Framework for the Voice over WiFi Protocol

Hyunwoo Lee
lee3816@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Imtiaz Karim
karim7@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Ninghui Li
ninghui@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Elisa Bertino
bertino@purdue.edu
Purdue University
West Lafayette, Indiana, USA

ABSTRACT

In this paper, we evaluate the security of the Voice over WiFi (VoWiFi) protocol by proposing the VWAnalyzer framework. We model five critical procedures of the VoWiFi protocol and deploy a model-based testing approach to uncover potential design flaws. Since the standards of the VoWiFi protocol contain underspecifications that can lead to vulnerable scenarios, VWAnalyzer explicitly deals with them. Unlike prior approaches that do not consider the underspecifications, VWAnalyzer adopts a systematic approach that constructs diverse and viable scenarios based on the underspecifications and substantially reduces the number of possible scenarios. Then the scenarios are verified against security properties. VWAnalyzer automatically generates 960 viable scenarios to be analyzed among 10,368 scenarios (91% decrease) from the initial models. We demonstrate the effectiveness of VWAnalyzer by verifying 38 properties and uncovering 3 new attacks. Notable among our findings is the *denial-of-cellular-connectivity attack*, due to insecure handover that disconnects the user through both VoWiFi and VoLTE. To ensure that the exposed attacks pose real threats and are indeed realizable in practice, we have validated the attacks in a real-world testbed. We also report several implementations issues that were uncovered during the testbed evaluation.

CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**.

KEYWORDS

Voice over WiFi, WiFi calling, mobile network security

ACM Reference Format:
Hyunwoo Lee, Imtiaz Karim, Ninghui Li, and Elisa Bertino. 2022. VWAnalyzer: A Systematic Security Analysis Framework for the Voice over WiFi Protocol. In *Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security (ASIA CCS '22), May 30–June 3, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3488932.3517425

ASIA CCS '22, May 30–June 3, 2022, Nagasaki, Japan
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9140-5/22/05.
https://doi.org/10.1145/3488932.3517425

1 INTRODUCTION

Voice over WiFi (VoWiFi) [6], a.k.a, WiFi-Calling, is a voice service that uses a WiFi-access network to improve mobile coverage. VoWiFi is widely adopted since using WiFi does not require extra applications, additional charges, or additional logins. Moreover, as a user does not even need to know whether the user is making a WiFi call, VoWiFi is transparent and user-friendly. Therefore, the usage of VoWiFi keeps rising, and the increasing numbers of public WiFi hotspots will keep pushing its usage. For example, the number of total public hotspots is expected to grow four-fold from 2017 (124M) to 2022 (549M) [15]. Following a similar trend, the market of VoWiFi is also rapidly increasing. According to market research [24], the global VoWiFi market size is projected to reach $22,801.2 million by 2030, up from $2,035.3 million in 2019. As such, VoWiFi has an increasing presence in our "mobile" life. Such widespread adoption thus requires strong security.

Problem scope. The main goal of this paper is to analyze the VoWiFi protocol based on its specifications. The protocol is extremely complex and stateful with protocols across multiple layers and consists of several critical procedures including evolved Packet Data Gateway (ePDG) discovery, User Equipment (UE) authentication and authorization, channel maintenance, making/receiving VoWiFi calls, and handover from VoWiFi to VoLTE. Furthermore, the protocol specification contains ambiguities, underspecifications due to the intricate protocol details written in natural languages [1, 2, 4, 5, 9, 23, 25, 27, 29, 36]. Therefore, in this work, we aim to analyze all the protocols and procedures included in the VoWiFi protocol. Note that although the protocol contains "WiFi" in its name, the security of WiFi is out of scope in the VoWiFi specifications.

Prior research. The analysis of such a large-scale protocol is challenging and requires a systematic approach. Existing security analyses on VoWiFi [11, 32, 38] covered only parts of the protocol, and none of them develops a systematic framework. Prior work has shown that formal verification is useful in assessing the security of large-scale and real-world protocols like 4G LTE [21] or 5G NR [12, 17, 22]. However, none of them tackle the question of underspecifications in the protocols and use their best interpretation of the standards for modeling. These prior methodologies are, therefore, not applicable to the VoWiFi protocol that contains multiple underspecifications. Prior work [26, 30] has shown that a number of vulnerabilities are induced by underspecified protocols, and developers do not always select the most secure option in their implementations when they encounter underspecifications. This raises the following research question: *Is it possible to formally verify the aforementioned critical procedures of the VoWiFi protocol, which contains underspecifications?*

Challenges. To conduct formal verification of the VoWiFi protocol, we need to address the following challenges: (C1) *Underspecification*: The VoWiFi protocol contains multiple underspecifications. It is mostly due to the specifications being written in natural languages and the lack of formal specification or formal implementation for the VoWiFi protocol. Therefore, the implementations can be different and deviate from each other. (C2) *Closed system*: VoWiFi implementations are proprietary. Therefore, we can only rely on the related specifications to develop the model of the protocol. (C3) *Untrusted entities*: VoWiFi introduces untrusted Internet entities, such as the Domain Name System (DNS) and WiFi Access Points (APs), which exposes a UE to Internet vulnerabilities when making/receiving calls.

Addressing the challenges. To address the challenges, (C1) while modeling the protocols, we consider underspecifications in detail and generate variants of the model whenever we find them. For example, if there are underspecifications about the entity that has to initiate a certain communication, we develop multiple models, one for each entity that could start this communication. The state space can easily explode as the number of such models is exponential in the number of underspecifications. To constrain the state space, we define and utilize a systematic algorithm that substantially reduces the number of models. To address the challenge (C2), we carefully read through VoWiFi-related specifications including the 3GPP specifications [1, 2, 4], the conformance test suite [5], and the RFC documents [9, 23, 25, 27, 29, 36] that the 3GPP specifications refer to. To address the challenge (C3), we develop models for all the entities related to the VoWiFi protocol. We model a total of 3 entities including the DNS system and also implement the network address translator (NAT) function of WiFi AP in the network channels.

Methodology. We propose VWAnalyzer–a formal verification framework that can systematically consider underspecifications in the protocols. To analyze the VoWiFi protocol, we first construct the VoWiFi model, as Finite State Machines (FSMs), by consulting the specifications. As the protocol standards contain underspecifications, the FSMs have multiple choices based on the underspecifications. Then, VWAnalyzer systematically picks the different choices and builds multiple variant models that lead to diverse scenarios, which consist of three entities that run different model variants of the VoWiFi protocol. The number of variant scenarios can easily explode due to the exponential relation with the number of underspecifications. However, not all the scenarios are viable. For instance, if one entity has to initiate a certain communication, the other entity should not start the communication but should respond to it. To tackle this, we develop a systematic algorithm and implement it in a tool that takes the initial FSMs with multiple choices due to underspecifications, constraints based on protocol entities and execution, and automatically generates the viable scenarios for verification.

Once the scenarios are constructed, each scenario is verified according to the properties extracted from the specifications by combining the reasoning powers of a general-purpose model checker and a cryptographic protocol verifier. The usage of two verifiers is inspired by previous works [21, 22, 26], as it enables to reason about all the desired properties including rich temporal properties such as safety, liveliness, and correspondence.

Implementation. Our initial VoWiFi FSM has a total of 111 states and 170 transitions. The scenario generation tool creates 960 distinct viable scenarios from the initial FSM, utilizing 23 constraints. It reduces the number of scenarios by 91%, from the case where all the possible variants (10,368) are generated from the underspecifications. We instantiate VWAnalyzer with an infinite-state model checker (i.e., nuXmv [14]) and a cryptographic protocol verifier (i.e., Tamarin [33]) and verify the scenarios with 38 properties.

Findings. Using VWAnalyzer, we have carried out a detailed analysis of VoWiFi. VWAnalyzer uncovered three new protocol design flaws that can lead to attacks on real devices. Furthermore, while testing the design flaws in the testbed, we found several implementation issues as well. Notable among our findings is the *denial-of-cellular-connectivity attack* that allows an adversary to force the device to be in a disruptive handover state, where it is not able to connect to the network both through VoWiFi and VoLTE. Coupled with the implementation flaws, other attacks can surreptitiously deny the VoWiFi service making a victim unable to make or receive calls.

Contributions. In summary, we make the following contributions:

- We propose VWAnalyzer, a model-based formal verification framework that deals with underspecifications, to analyze the security of the VoWiFi protocol.
- We build a full-stack VoWiFi protocol model from the underlying layers, such as IP, to the application layer, such as Session Initiation Protocol (SIP) [36]. Also, we develop algorithms and automated tools to systematically reduce the number of scenarios generated due to underspecifications. To the best of our knowledge, we are the first to perform a fine-grained formal analysis of the VoWiFi protocol standards. We release the source codes of the model, the properties, and related tools to support further research in this area.[1]
- Using VWAnalyzer we discover one denial-of-cellular-connectivity attack and two denial-of-VoWiFi-service attacks based on 2 insecure underspecifications.
- We validate our findings in our testbed and report two implementation issues that we found during the evaluation.

2 VOWIFI PRIMER

This section provides a brief introduction to the VoWiFi protocol and the procedures in the protocol. Unlike the existing mobile technologies, including LTE, where a UE directly accesses the mobile network via a base station, a UE that wants to use the VoWiFi service utilizes an untrusted access network in the Internet infrastructure to connect to the mobile network. To integrate the two different networks, an *evolved Packet Data Gateway* (ePDG) is introduced

[1] The source codes can be found at https://github.com/vwanalyzer.

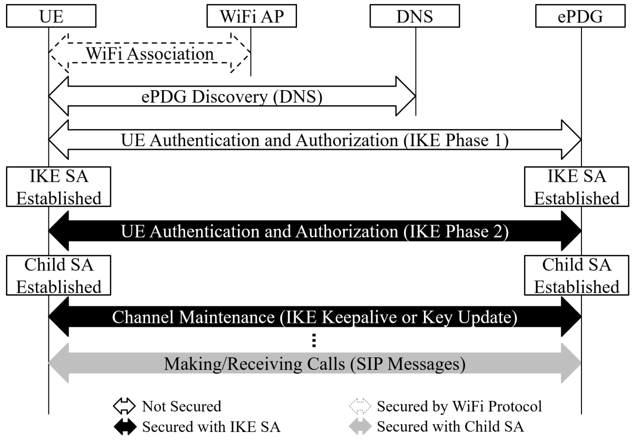

Figure 1: Channel establishment between a UE and an ePDG. To enable VoWiFi, a UE begins with the WiFi association, followed by the ePDG discovery procedure by using the DNS protocol. Then, the UE executes the IKEv2 protocol to establish the channel with the ePDG while authenticating itself with the EAP-AKA protocol. Finally, the UE can make or receive a call over WiFi using the SIP protocol.

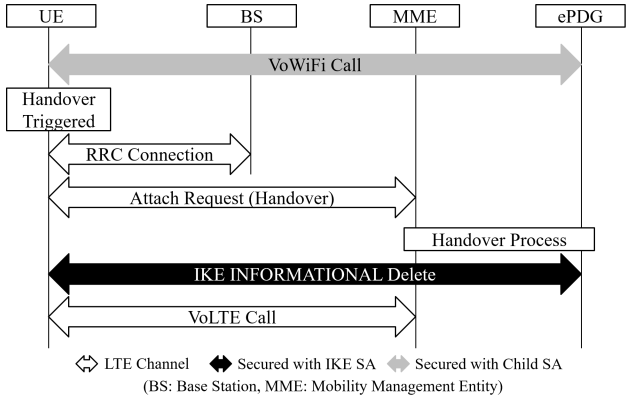

Figure 2: Handover from VoWiFi to VoLTE. When a UE decides to execute a handover from VoWiFi to VoLTE during a WiFi call, it initiates the process by sending the `Attach Request` message with indication of the "handover". Once the UE attaches to the LTE core network, the ePDG sends the `IKE Delete` message to the UE to inform that the handover is completed.

between the Internet and the mobile network as an *entity*. We use the term *entity* to describe participants of the VoWiFi protocol including ePDG, UE, DNS. Since the Internet is considered untrusted, the security of VoWiFi primarily depends on protecting the channel between the UE and the ePDG. For this purpose, the IPsec suite that aims to provide security between two IP peers is used [29]. In what follows, we present an overview of the key procedures of VoWiFi (see Figure 1 and Figure 2).

WiFi association. When a user turns on the WiFi interface, the UE searches available WiFi access points (APs) and establishes an association with one that sends a high-strength signal. The protocol used in this phase is an out-of-scope of the VoWiFi specification. Instead, the specification assumes that a WiFi security protocol, such as WPA2-PSK [10], is used. Once the WiFi association is completed, the WiFi AP assigns an IP address to the UE.

ePDG discovery. A UE discovers the ePDG by requesting the address from the DNS server with the ePDG Fully Qualified Domain Name, which is specified in 3GPP TS 23.003 [1] and is provisioned in the UE. The DNS responds to the UE with the IP address of the ePDG.

UE authentication and authorization. The UE authentication and authorization procedure aims to authenticate a UE and authorize the UE to use its available service. It includes the IPsec channel establishment between the UE and an ePDG. In IPsec, two IP peers establish security associations (SA) [29] that consist of the security algorithms and keys between them. To this end, the Internet Key Exchange version 2 (IKEv2) protocol [27] is used. In IKEv2, an *initiator* begins with the protocol by sending the first message to a *responder*. There are several IKE *exchanges* that always consist of a *request* message followed by a *response* message in the protocol[2]. Each message includes a series of *payloads*. For example, an `IKE_SA_INIT` request message contains a `Nonce` payload, a `Security Association` payload, and other payloads. Any entity can send a request message, and the other should reply with the response message. If one entity does not receive the corresponding response message for a particular period (i.e., a retransmission timeout interval), the entity retransmits the same request message to the other.

The protocol starts with the `IKE_SA_INIT` exchange between a UE and an ePDG. The exchange includes nonce values, algorithms to be used, Diffie-Hellman public keys, and other notifications, such as the NAT detection payloads. After the exchange, both peers can set up the IKE SAs that contain two encryption keys and two authentication keys used in opposite directions, respectively. Each entity stores the SA in its SA database and looks it up with the initiator's SPI, the responder's SPI, and the related IP addresses. Next, the UE and the ePDG exchange several `IKE_AUTH` exchanges. The main purpose of the exchanges is to authenticate the UE and to establish confidentiality keys (CKs) and integrity keys (IKs) used for the Encapsulating Security Payload (ESP) packets [28] by using the Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) protocol [9]. These keys are included in the Child SA. To this end, the UE sends its International Mobile Subscriber Identity (IMSI) or Temporary Mobile Subscriber Identity (TMSI) to the Authentication Authorization and Accounting (AAA) server via the ePDG and solves the AKA-Challenge from the AAA server. Also, the UE can optionally send a `CertificateRequest` payload to request an ePDG's certificate and authenticate the ePDG with it. After the UE is authenticated, they exchange the authentication (`AUTH`) values that bind shared keys with the `IKE_SA_INIT` messages they sent in the first phase

[2] In this paper, we use the term *IKE exchange* to refer to both a request message and the corresponding response message. We use the term *IKE message* to indicate either of them.

in order to avoid a message modification attack. Finally, the IKE channel is established.

Channel maintenance. To confirm that the IKE channel is alive, the UE and the ePDG periodically exchange *IKE keepalive* messages. They are in the form of IKE `INFORMATIONAL` exchanges containing only 16-byte padding values. Also, either the UE or the ePDG can renew the keys of the child SA in this procedure.

We note two properties of the IKE messages. First, the IKE messages are exchanged independently of making and receiving calls. The process of the IKE channel establishment is initiated after a UE is associated with a particular WiFi AP. Second, all the exchanges after the first phase of the IKE protocol are encrypted and integrity-protected. Also, each message includes its message ID in its payload to provide anti-replay protection.

Making/receiving calls. To make and receive calls, a UE executes the Session Initiation Protocol (SIP) [25, 36] with an IP Multimedia Subsystem (IMS) server [7] via an ePDG. To secure the SIP messages between a UE and an ePDG, the messages are exchanged in the form of ESP packets that provide encryption, integrity-protection, and anti-replay protection.

Handover from VoWiFi to VoLTE. When a UE decides to execute a handover from VoWiFi to VoLTE during a WiFi call, the UE initiates the handover process by sending the `Attach Request` with Request Type indicating "Handover" to the Mobility Management Entity (MME). Note that `Attach Request` includes IMSI as an identifier. Once the UE attaches to the LTE core network, the network directs the ePDG to disconnect the IKE channel with the UE. Then, the ePDG sends the IKE `INFORMATIONAL` request message with the `Delete` payload to the UE, which may respond with the response message. The ePDG sends the Session Termination Request (STR) to the AAA server to remove the VoWiFi session. Finally, the AAA server responds with the Session Termination Answer (STA) to confirm the request.

3 OVERVIEW OF VWANALYZER

In this section we define the protocol finite state machine (FSM) and describe the threat model that we assume. Then, we provide the formal definitions used in our analysis, followed by a high-level overview of VWAnalyzer.

3.1 Protocol Finite State Machine

The protocol execution can be described as a finite state machine (FSM) defined as follows:

Definition 1 (finite state machine)**.** A FSM m is defined as a 5-tuple $(\Sigma, \Gamma, \mathcal{S}, s_0, \mathcal{T})$, where Σ and Γ indicate non-empty sets of conditions and actions respectively, $\mathcal{S}$ denotes a finite set of states, $s_0 \in \mathcal{S}$ is an initial state, and $\mathcal{T}$ represents a set of transitions in $\mathcal{S}$. In detail, let $t = (s_{in}, \sigma, s_{out}, \gamma)$ be a transition in $\mathcal{T}$; then if m is in state s_{in} and the condition σ is true, then action γ is executed and m moves to state s_{out}. We use $m.\Sigma$, $m.\Gamma$, $m.\mathcal{S}$, $m.s_0$, and $m.\mathcal{T}$ to indicate each element of m, and $t.s_{in}$, $t.\sigma$, $t.s_{out}$, and $t.\gamma$ for $t \in \mathcal{T}$.

The VoWiFi protocol consists of multiple protocols, each of which might define more than one roles (e.g., a client or a server). We model all of them. For instance, there is an initiator and a responder in the IKE protocol; thus, we generate two FSMs for an IKE initiator and an IKE responder. On the other hand, we build only one FSM for the IP protocol since there is no specific role defined in it. If we need to specify a protocol or a role that the FSM captures, we append a subscript to a variable. For example, we use $m_{\texttt{ike,initiator}}$ to denote an FSM of the IKE protocol for an IKE initiator. $m_{\texttt{ip}}$ represents an FSM related to the IP protocol.

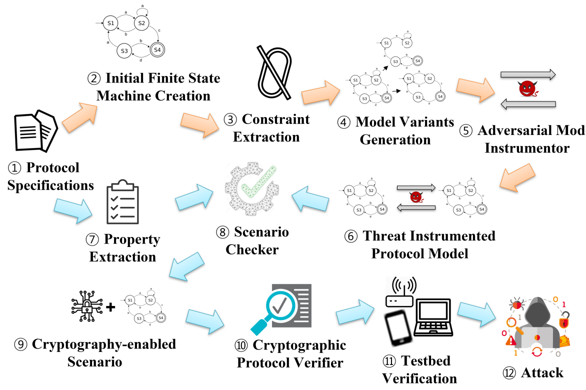
① Protocol Specifications
② Initial Finite State Machine Creation
③ Constraint Extraction
④ Model Variants Generation
⑤ Adversarial Model Instrumentor
⑥ Threat Instrumented Protocol Model
⑦ Property Extraction
⑧ Scenario Checker
⑨ Cryptography-enabled Scenario
⑩ Cryptographic Protocol Verifier
⑪ Testbed Verification
⑫ Attack

Figure 3: Overview of the VWAnalyzer framework. VWAnalyzer consists of two main phases – 1) scenario construction (orange arrow) and 2) scenario verification (blue arrow). The first phase builds scenarios consisting of three entities with their protocol stacks, while the second phase analyzes the scenarios by employing the protocol verification technique.

In our model, each entity runs several FSMs and exchanges messages with other entities to progress with the procedures. We refer to the medium where a message is sent by an entity and received by another entity as a *channel*. In detail, FSMs defined for one entity communicate with the FSMs defined for another entity with two unidirectional channels.

3.2 Threat Model

Channels are exposed to a Dolev-Yao adversary [18] that can observe, modify, or drop any packet and inject new ones, and also impersonate a legitimate entity. Moreover, the adversary is computationally bounded. That is, the adversary cannot break cryptographic primitives and can decrypt a message only if it has the decryption key.

3.3 Challenges with Underspecification

The most challenging problem in designing VWAnalyzer is to deal with underspecifications and build scenarios to be analyzed. There are three challenges:

(C1) Definition of underspecifications. The meaning of *underspecifications* can be ambiguous. Therefore, we need to have a precise definition of underspecifications to deal with them.

(C2) Possible scenario generation. Once an underspecified issue is identified, the next step is to construct possible scenarios for the issue. We need to have a relevant way to generate them.

Algorithm 1 Model Variant Generation Algorithm

```
Input: Finite State Machine $m \in \mathcal{M}$
Output: Model Variants $\mathcal{MV}(m)$
▷ Let $\mathcal{T}_{s,\sigma}$ be a set of transitions such that for $t \in m.\mathcal{T}$, $t.s_{in} \in m.\mathcal{S}$ and $t.\sigma \in m.\Sigma$.
▷ Let $\mathcal{T}'$ be a set of $\mathcal{T}_{s,\sigma}$.
for $t$ in $m.\mathcal{T}$ do
    Insert $t$ in $\mathcal{T}_{t.s_{in},t.\sigma}$
end for
▷ Let $C$ be a set of combinations by picking up transitions one by one from $\mathcal{T}_{t.s_{in},t.\sigma} \in \mathcal{T}'$.
for $c$ in $C$ do
    Define a variant $m' = (m.\Sigma, m.\Gamma, m.\mathcal{S}, m.s_0, c)$
    Insert $m'$ in $\mathcal{MV}(m)$
end for
```

(C3) Exponential growth of possible scenarios. The number of possible scenarios exponentially increases whenever we deal with certain underspecification with their possible choices. Therefore, we need an approach to limit the explosive growth of scenarios to make our approach scalable.

3.4 Addressing the Challenges

To address (C1), we first provide the formal definition of underspecifications based on our FSM definition. Informally, an FSM describing a protocol is underspecified if the protocol allows the FSM to move to different states or perform various actions from the same state and conditions.

Definition 2 (underspecification)**.** Let m be an FSM. We say that m is *underspecified* if $m.\mathcal{T}$ contains transitions t_i and t_j with $i \neq j$, such that: (i) $t_i.s_{in} = t_j.s_{in}$ and $t_i.\sigma = t_j.\sigma$ and (ii) $t_i.s_{out} \neq t_j.s_{out}$ or $t_i.\gamma \neq t_j.\gamma$.

Based on the definition of underspecifications, we define a set of *model variants* of an FSM m, denoted as $\mathcal{MV}(m)$, and an algorithm to generate this set to address (C2).

Definition 3 (model variants)**.** Let m be an FSM. We say that m generates *model variants* $\mathcal{MV}(m)$ – a set of variant FSMs that remove underspecifications from m, by selecting only one transition for a given state and condition, according to Algorithm 1.

To address (C3), we first define a *protocol stack*. Each entity runs its protocol stack, that is, a set of FSMs. The protocol stack consists of the PHY/MAC, IP, UDP, IKE, SIP, and DNS layer. Those are built by picking each FSM from the corresponding model variants of the protocols (see Algorithm 2). We say that *the protocol stacks are different* if any of their FSMs are different. Then, we define a *scenario* as a set of three protocol stacks of three different entities, i.e., a UE, an ePDG, and a DNS; thus, a scenario is defined in the form of a 3-tuple. We say that *the scenarios are different* if any of their protocol stacks are different.

With increasing underspecifications, the number of model variants rapidly grows, which in turn exponentially increases the different scenarios. For instance, if one of the FSMs in the UE protocol stack and one of the FSMs in the ePDG protocol stack have two underspecifications with two different choices respectively, it generates a total of 16(= 2 × 2 × 2 × 2) scenarios. To reduce the number of different scenarios that rely on the number of model variants, we apply *constraints* when generating scenarios. The first group of constraints is based on which FSMs a particular entity can run. For example, a UE should not execute an FSM of a DNS server since it is always a DNS client in the VoWiFi protocol. Another group of constraints helps to select the reasonable FSMs in the context of the protocol execution. For instance, if a UE's IKE FSM contains the states and the transitions related to sending the IKE keepalive request message, the ePDG's IKE FSM should have the states and the transitions to process it and send the response message.

Algorithm 2 Protocol Stack Generation Algorithm

```
Input: Model Variants $\mathcal{MV}(m_{\mathsf{phymac}})$, $\mathcal{MV}(m_{\mathtt{ip}})$, $\mathcal{MV}(m_{\mathtt{ike}})$,
    $\mathcal{MV}(m_{\mathtt{sip}})$, $\mathcal{MV}(m_{\mathsf{dns}})$ and Constraints $C$
Output: Protocol stacks $\mathcal{P}_{\mathsf{ue}}$, $\mathcal{P}_{\mathsf{epdg}}$, and $\mathcal{P}_{\mathsf{dns}}$
for $x$ in $\{\mathsf{ue}, \mathsf{epdg}, \mathsf{dns}\}$ do
    for $a$ in $\mathcal{MV}(m_{\mathsf{phymac}})$, $b$ in $\mathcal{MV}(m_{\mathtt{ip}})$,
        $c$ in $\mathcal{MV}(m_{\mathsf{udp}})$, $d$ in $\mathcal{MV}(m_{\mathtt{ike}})$,
        $e$ in $\mathcal{MV}(m_{\mathtt{sip}})$, $f$ in $\mathcal{MV}(m_{\mathsf{dns}})$ do
        $\pi = (a, b, c, d, e, f)$
        if $\pi$ does not violate $C$ then
            Insert $\pi$ into $\mathcal{P}_x$
        end if
    end for
end for
```

Algorithm 3 Scenario Construction Algorithm

```
Input: Initial FSMs and Constraints $C$
Output: Scenarios $\mathcal{S}$
for $x$ in $\{\mathsf{phymac}, \mathtt{ip}, \mathsf{udp}\}$ do
    $\mathcal{MV}_x$ = ModelVariantsGeneration($m_x$)
end for
$\mathcal{MV}_{\mathtt{ike}}$ = ModelVariantsGeneration($m_{\mathtt{ike,initiator}}$)
  ∪ ModelVariantsGeneration($m_{\mathsf{ike,responder}}$)
for $x$ in $\{\mathtt{sip}, \mathtt{dns}\}$ do
    $\mathcal{MV}_x$ = ModelVariantsGeneration($m_{x,\mathtt{client}}$)
      ∪ ModelVariantsGeneration($m_{x,\mathsf{server}}$)
end for
for $x$ in $\{\mathsf{ue}, \mathsf{epdg}, \mathsf{dns}\}$ do
    $\mathcal{P}_x$ = ProtocolStackGeneration($\mathcal{MV}(m)$, $C$)
end for
for $u$ in $\mathcal{P}_{\mathsf{ue}}$, $e$ in $\mathcal{P}_{\mathsf{epdg}}$, $d$ in $\mathcal{P}_{\mathsf{dns}}$ do
    $s = (u, e, d)$
    if $s$ does not violate $C$ then
        Insert $s$ into $\mathcal{S}$
    end if
end for
```

3.5 High Level Description of VWAnalyzer

VWAnalyzer consists of two main phases – 1) scenario construction and 2) scenario verification. The first phase builds scenarios consisting of three entities with their protocol stacks. To build the scenarios, we first investigate the protocol specifications and manually design FSMs. We call these FSMs *initial FSMs*. As a general model checker is not able to reason about cryptographic constructs, the cryptographic assumptions of the FSMs are abstracted, and the plain-text version of each message is modeled. At this point, the FSMs may have underspecifications. In other words, the FSMs can have more than one transition on the same state and conditions. To reduce the number of scenarios, we extract constraints from the specifications. The scenario construction algorithm (refer to Algorithm 3) uses the initial FSMs and constraints to automatically create the possible scenarios for verification. First, each initial FSM m is divided into several FSMs that have only one transition on the same state and conditions, resulting in $\mathcal{MV}(m)$– *model variants* of m. Second, *protocol stacks* are generated for the entities (i.e., ue, epdg, and dns) by picking FSMs one by one from the model variants. The protocol stacks for the entities are used to create the *scenarios*. At this point, the constraints are used to reduce the number of scenarios. If a scenario violates any constraints, it is removed from consideration, and the viable scenarios that comply with the constraints are moved to the next phase.

The second phase analyzes the scenarios by employing the protocol verification technique utilized by prior work [21, 22]. It combines the reasoning powers of a general-purpose model checker and a cryptographic protocol verifier following the counterexample guided abstraction refinement principle (CEGAR) [16]. In the CEGAR framework, the verifier takes each scenario and the properties as input. For each property, the verifier checks if the scenario satisfies it. It provides a counterexample in case of the property violation, which is either due to a spurious case or an error in the specification. Before applying the CEGAR-based analysis, we add a Dolev-Yao adversary to scenarios, resulting in threat-instrumented scenarios. Then, we verify the resultant scenarios against the security properties using the general-purpose model checker. If the model checker approves the property, then a scenario is verified against it. However, when the model checker reports that a scenario violates the property, it returns a counterexample. At this point, the counterexample can be spurious due to the abstraction of cryptographic properties. To resolve this, we use a cryptographic protocol verifier to check if the counterexample adheres to the cryptographic assumptions. If the cryptographic verification fails, we add invariants to the property to discard this counterexample. We repeat this process until the verification completes or the verifier outputs a realizable counterexample. The counterexample is then tested in a real-world testbed.

4 DETAIL OF VWANALYZER

We now present detailed descriptions of the major components of VWAnalyzer (see Figure 3).

4.1 Scenario Construction

Initial FSMs creation and constraint extraction. We model the protocols – PHY/MAC, IP, UDP, IKE (initiator), IKE (responder), SIP (client), SIP (server), DNS (client), and DNS (server) – of the VoWiFi protocol as FSMs defined according to Definition 1. In this step, we include all the possible transitions that can occur due to underspecifications in the standards. We refer to the FSMs resulting from this step as *initial FSMs*. Moreover, from the specifications we extract the constraints utilized later on for reducing the exponential growth of scenarios.

There are two types of constraints. The first type of constraint is about the role that an entity can play. For example, a UE participates in the VoWiFi protocol as an IKE initiator, an SIP client, and a DNS client. An ePDG can act as an IKE responder and an SIP server. However, it does not need to run the DNS protocol. These constraints reduce the number of possible FSMs that an entity should pick. The second type of constraint is about the protocol execution. If one entity runs an FSM that sends a certain message, the receiving entity should run an FSM that receives and processes this message. For example, when both the UE's IKE FSM and ePDG's IKE FSM include the action of sending the IKE keepalive request message, the scenario is not viable as the protocol flow cannot progress anymore. We utilize these constraints to reduce the number of scenarios to be verified by removing such non-viable scenarios.

Model variants & scenario generation. Once the initial FSMs are constructed, we generate model variants, protocol stacks for different entities, and various scenarios related to underspecifications.

(1) Generating model variants: Algorithm 1 describes how to create model variants from the initial FSMs that have underspecifications. We classify transitions based on their initial states (s_{in}) and the conditions (σ). Then, several FSMs are generated by picking transitions one-by-one from the transition sets. For example, let assume that there are two different transitions $t_1 = (s_{in}, \sigma, s_{out_1}, \gamma_1)$ and $t_2 = (s_{in}, \sigma, s_{out_2}, \gamma_2)$ where $s_{out_1} \neq s_{out_2}$. The two transitions are included in the same set, that is, $\mathcal{T}_{s_{in},\sigma}$. From the set, two different models – one with only t_1 and the other with t_2 – are built. Finally, we have model variants (i.e., $\mathcal{MV}(m_{\mathsf{phymac}})$, $\mathcal{MV}(m_{\mathtt{ip}})$, $\mathcal{MV}(m_{\mathsf{udp}})$, $\mathcal{MV}(m_{\mathtt{ike}})$, $\mathcal{MV}(m_{\mathtt{sip}})$, and $\mathcal{MV}(m_{\mathsf{dns}})$) for the initial FSMs for the six protocols.

(2) Generating different protocol stacks: A set of FSMs implements a protocol stack for an entity. Due to all the combinations among model variants, there can be numerous protocol stacks. For example, a UE can always send the IKE keepalive request message or a UE may always receive the IKE keepalive request message. Therefore, there are two different models (say, $m^1_{\mathtt{ike}}$, $m^2_{\mathtt{ike}}$, and $m^3_{\mathtt{ike}}$) in the model variants $\mathcal{MV}(m_{\mathtt{ike}})$. To generate different protocol stacks, we generate all the combinations from the model variants. Then, we remove the combinations that violate the constraints to reduce the number of protocol stacks.

(3) Generating different scenarios: A scenario is defined with three protocol stacks for a UE, an ePDG, and a DNS, respectively. Since a UE and an ePDG have several protocol stacks, there are many scenarios that consist of different UE's and ePDG's protocol stacks. While generating the scenarios, we remove ones that do not follow the constraints. Note that the scenarios are generated automatically following Algorithm 1, Algorithm 2, and Algorithm 3, which takes the initial FSMs and the constraints as inputs and return the viable scenarios.

Adversarial FSM instrumentor. The main role of this component is to combine our FSMs with an adversary. As mentioned, the adversary controls the channels between entities. Each channel consists of two unidirectional sub-channels (i.e., not a single bidirectional channel), corresponding to two separate FSMs that perform forward, insertion, drop, and modification of IP packets. We model two different sub-channels (or FSMs) to capture diverse adversarial scenarios (e.g., an adversary only attacking in one direction).

Threat instrumented protocol scenario. Finally, the channels are integrated between entities in each scenario, resulting in threat instrumented protocol scenarios.

4.2 Scenario Verification

Scenario checker and Cryptographic protocol verifier. The scenario checker is used to verify if the threat instrumented protocol scenario satisfies the properties. The scenario checker reports counterexamples that violate the properties in the execution of the FSMs. Since we do not consider cryptographic assumptions at this point, the counterexamples are fed to the cryptographic protocol verifier to distinguish the counterexamples that adhere to the cryptographic assumptions from spurious ones. If any counterexample turns out to be spurious, we add an invariant to the property in order not to generate the same counterexample. Finally, we obtain logical attacks.

Testbed verification. We also conduct a testbed evaluation for the discovered counterexamples to see if the counterexamples are validated in a real-world testbed.

5 RUNNING EXAMPLE

This section illustrates how our framework generates scenarios with a concrete example (see Figure 4). We only consider two entities, a UE and an ePDG, and only one underspecification for brevity. The IKE keepalive exchange used for the channel maintenance procedure is described in [27] as:

> The types of subsequent exchanges (after `IKE_SA_INIT` and `IKE_AUTH`) are `CREATE_CHILD_SA` and `INFORMATIONAL`. ⋯ An `INFORMATIONAL` request with no payloads (other than the empty `Encrypted` payload required by the syntax) is commonly used as a check for liveness.

Initial FSM creation. The leftmost graph in Figure 4 shows a part of the initial FSM for the keepalive procedure; this part is the same for the IKE initiator and responder ($m_{\mathtt{ike,initiator}}$ and $m_{\mathtt{ike,responder}}$). There are three states, namely `Channel Idle`, `Keepalive Requester`, and `Keepalive Responder`. We note that the protocol allows for two different output states on the transitions (red lines) from the state `Channel Idle` and for the condition (`keepalive_start`). The reason is that the standard does not specify who initiates the exchange; thus, either the UE or the ePDG can send the keepalive request message.

Constraint extraction. We extract the following two constraints:

- "if one has the transition from `Channel Idle` to `Keepalive Requester`, the other should have the transition from `Channel Idle` to `Keepalive Responder`"
- "if one has the transition from `Channel Idle` to `Keepalive Responder`, the other should have the transition from `Channel Idle` to `Keepalive Requester`."

Generating model variants. We generate two different FSMs from the initial FSM. Following Algorithm 1, the initial FSM is split into two FSMs each of which has one red line respectively ($m^1_{\mathtt{ike}}$ and $m^2_{\mathtt{ike}}$). These two FSMs are members of a model variant of the initial FSM, i.e., $\mathcal{MV}(m_{\mathtt{ike}})$.

Generating different protocol stacks. Since we assume that there is no underspecification in all the protocols except the IKE protocol, then $|\mathcal{MV}_{\mathtt{phymac}}| = |\mathcal{MV}_{\mathtt{ip}}| = |\mathcal{MV}_{\mathtt{udp}}| = |\mathcal{MV}_{\mathtt{sip}}| = |\mathcal{MV}_{\mathtt{dns}}| = 1$ and $|\mathcal{MV}_{\mathtt{ike}}| = 2$. That is, an entity can select two variants from the IKE model variants and the other protocols are the same. Therefore, there are two protocol stacks for the UE:

- $\mathtt{UE}_1 = (m_{\mathtt{phymac}}, m_{\mathtt{ip}}, m_{\mathtt{udp}}, m^1_{\mathtt{ike}}, m_{\mathtt{sip}}, m_{\mathtt{dns}})$
- $\mathtt{UE}_2 = (m_{\mathtt{phymac}}, m_{\mathtt{ip}}, m_{\mathtt{udp}}, m^2_{\mathtt{ike}}, m_{\mathtt{sip}}, m_{\mathtt{dns}})$

The same process is followed for the ePDG, and due to the same underspecification, there will be two protocol stacks for the ePDG ($\mathtt{ePDG}_1$ and $\mathtt{ePDG}_2$).

Generating different scenarios. Now we generate the scenarios based on different protocol stacks for entities. Since there are two UEs and two ePDGs, we can generate four scenarios. However, because of the constraints extracted above, the number of the scenarios is reduced. That is:

- If `UE` runs $m^1_{\mathtt{ike}}$, then `ePDG` should run $m^2_{\mathtt{ike}}$
- If `UE` runs $m^2_{\mathtt{ike}}$, then `ePDG` should run $m^1_{\mathtt{ike}}$

The two other scenarios are discarded as non-viable. We thus obtain two scenarios: $s_1 = (\mathtt{UE}_1, \mathtt{ePDG}_2)$ and $s_2 = (\mathtt{UE}_2, \mathtt{ePDG}_1)$. Then, the scenario verification phase is run only with those two scenarios.

6 IMPLEMENTATION

This section describes the implementation of the components of VWAnalyzer.[3]

6.1 Scenario Construction

Protocol specifications. We design the initial FSMs for each protocol layer based on the 3GPP specifications [2, 4], the conformance test suites [5], and the related RFC documents [25, 27, 36] that the 3GPP specifications point to.

We focus on the procedures used for the UE to get serviced by the core network. Therefore, we model five procedures – ePDG discovery, UE authentication and authorization, channel maintenance, making/receiving VoWiFi calls, and handover from VoWiFi to VoLTE – with three entities, namely: a UE, an ePDG, and the DNS system. Since all the components in the core network send messages via the ePDG, we consider the ePDG as the core network. We introduce the DNS system to capture the ePDG discovery procedure. We also consider a WiFi AP in the channels since it changes the IP addresses as NAT.

[3] We release our source code of the models, the constraints, the properties, the automatic scenario generation tools, and the testbed used in the analysis at https://github.com/vwanalyzer.

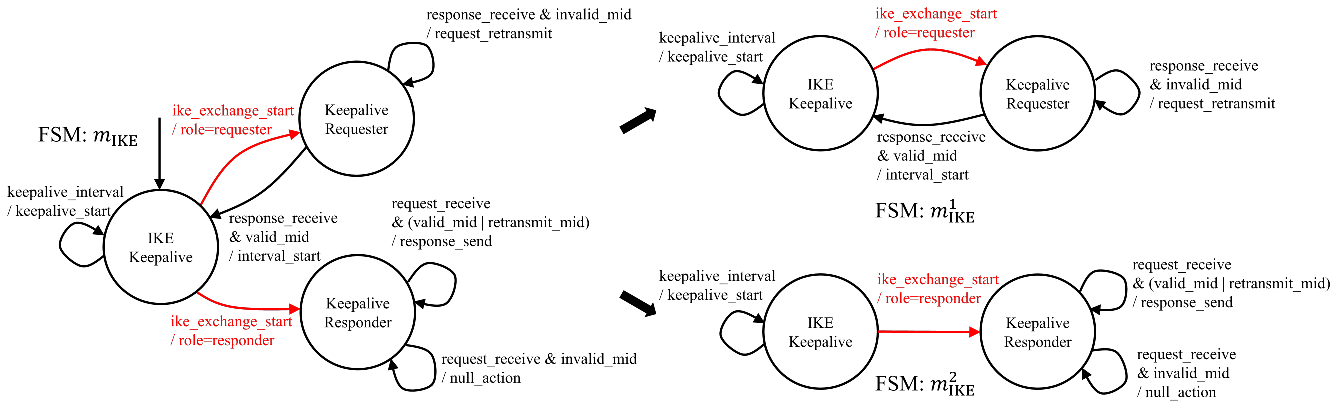

Figure 4: Example of a model variant generation. The FSM that has an underspecification (red lines) is split into two FSMs each of which has only one red line respectively. Note that we simplify an FSM of the IKE protocol to illustrate the example.

Initial FSMs and constraint extraction. We implement the full protocol stack of VoWiFi covered by the specification. Since the security of VoWiFi mainly relies on the IPsec suite, we model the IP layer and the above layers in detail based on the behavior related to the VoWiFi protocol. On the other hand, we model the PHY/MAC layer used to select the interface of the entity to capture the multi-interface (e.g., WiFi and LTE interfaces). Based on the specification, we model our FSMs using a DOT graph description language and visualizing them with Graphviz [19]. Finally, we build nine FSMs: $m_{\texttt{phymac}}$, $m_{\texttt{ip}}$, $m_{\texttt{udp}}$, $m_{\texttt{ike,initiator}}$, $m_{\texttt{ike,responder}}$, $m_{\texttt{sip,initiator}}$, $m_{\texttt{ike,responder}}$, $m_{\texttt{dns,client}}$, and $m_{\texttt{dns,server}}$. From the specifications, we also extract 23 constraints. We summarize the list of the constraints in Table 1.

Scenario generation. We implement a Python script following Algorithm 3 to generate model variants, entities, and scenarios. The script takes the initial FSMs and constraints as inputs and automatically outputs the viable scenarios. It has 975 lines of code.

Threat instrumented scenario generation. The scenarios generated from the previous step are then instrumented to include a Dolev-Yao adversary model, creating a threat instrumented protocol scenario in smv format [14]. For this, we extend the open-source tool from 5GReasoner [22] to automatically combine all the FSMs from the different protocol stacks together to create the threat instrumented scenario. The threat instrumented scenario generator has 593 lines of code.

6.2 Scenario Verification

Property extraction. The types of properties that we check include *authenticity* (i.e., to prevent impersonation), *availability* (i.e., to prevent denial-of-service attacks), *integrity* (i.e., to detect tampered messages), and *anti-replay protection* (i.e., to prevent replay attacks) that the protocol provide. We finally extract, formalize, and verify 38 properties in total from the specifications.

Scenario checker and cryptographic protocol verifier. We use nuXmv [14] as a scenario checker and Tamarin [33] as a cryptographic protocol verifier. For running all the scenarios simultaneously, we write a script that runs all the scenarios with the property list in nuXmv and saves the counterexamples. Tamarin is used to prove the feasibility of the counterexample. If the cryptographic assumptions of the counterexample are verified, we conclude that the counterexample is a possible logical attack.

Testbed verification. Once we discover a possible logical attack by the scenario checker and the cryptographic protocol verifier, we test it with a real-world testbed. We build a testbed consisting of a UE, a WiFi-AP, a DNS server, and a proxy. The proxy is used for multiple purposes. All the experiments are conducted with 5 VoWiFi-enabled UEs of different vendors targeting three different carriers. We implement an adversary application and run it over a WiFi-AP.

7 EVALUATION

In this section, we evaluate VWAnalyzer to answer the following research questions:

- **RQ1. Underspecification and scenarios:** How many scenarios does VWAnalyzer generate with underspecifications? How much VWAnalyzer's scenario construction algorithm reduces the number of scenarios?
- **RQ2. New findings:** What kind of counterexamples and attacks does VWAnalyzer uncover?
- **RQ3. Existing attacks:** What and how many existing attack scenarios can be detected by VWAnalyzer?

7.1 RQ1. Underspecification and Scenarios

We model 10 underspecifications in the initial FSMs. With such underspecifications, we generate 178 model variants from the IKE initiator FSM and the IKE responder FSM, 5 model variants from the SIP client FSM and the SIP server FSM, and 2 model variants from the DNS client FSM and the DNS server FSM. Other FSMs related to the underlying protocols (from PHY/MAC to UDP) do not generate any model variants. All the entities should run the underlying

Related Entities	Description
Constraints for entities	
UE	A UE has the WiFi and LTE interfaces
	A UE runs the IP protocol
	A UE runs the UDP protocol
	A UE runs the IKE protocol as an initiator
	A UE runs the SIP protocol as a client
	A UE runs the DNS protocol as a client
ePDG	An ePDG has the ethernet interface
	An ePDG runs the IP protocol
	An ePDG runs the UDP protocol
	An ePDG runs the IKE protocol as a responder
	An ePDG runs the SIP protocol as a server
DNS	A DNS has the ethernet interface
	A DNS runs the IP protocol
	A DNS runs the UDP protocol
	A DNS does not run the IKE protocol
	A DNS does not run the SIP protocol
	A DNS runs the DNS protocol as a server
Constraints for protocol execution	
UE - ePDG	If an ePDG's `IKE_AUTH` includes a certificate payload, an AUTH payload, and an eap-request-aka-challenge payload, a UE should be able to process the payloads
	If an ePDG's `IKE_AUTH` includes an AUTH payload and an eap-request-aka-challenge payload, a UE does not need to process the certificate payload
	If an ePDG's `IKE_AUTH` includes an eap-request-aka-challenge payload, a UE does not need to process a certificate and an AUTH payloads
	If a UE initiates the keepalive request message, an ePDG should be able to process it and send the response message
	If an ePDG initiates the keepalive request message, a UE should be able to process it and send the response message

Table 1: Constraints applied in the scenario construction algorithm.

protocols, but they may not execute some of the application layer protocols. For instance, ePDG is not involved in the ePDG discovery procedure, it does not need to run the DNS protocol. After applying the constraints about the role that an entity plays to the protocol stacks, we get 16 protocol stacks for UEs, 648 protocol stacks for ePDGs, and one protocol stack for the DNS server, respectively. It leads to 10,368 scenarios (16 × 648 × 1 = 10,368). We finally acquire 960 scenarios (91% decrease) with the constraints about the protocol execution.

7.2 RQ2. New Findings

VWAnalyzer reports 3 new availability attacks.

7.2.1 Denial-of-Cellular-Connectivity Attack. With this attack, the UE fails to perform the handover procedure and temporarily breaks the underlying connections for both VoLTE and VoWiFi. It results in a call drop if the victim UE was having a call.

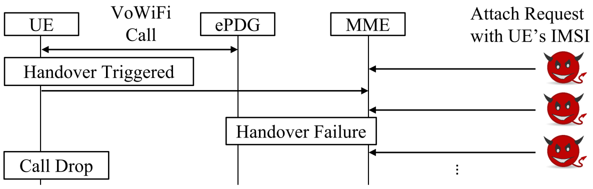

Figure 5: Denial-of-cellular-connectivity attack

Scenario description We find the attack from all the scenarios where a UE initiates the handover procedure and finalizes it after receiving the IKE `Delete` payload from the ePDG.

Detection and attack scenarios. With the above models, we discover the attack by the property: *"If a handover from VoWiFi to VoLTE is triggered, the handover should be done seamlessly."* We observe one counterexample CE from nuXmv, where an adversary keeps sending the `Attach Request` packets. The counterexample forces the UE to get stuck in a loop and prevents the completion of the handover. To validate the adversary's capability of forging an `Attach Request` packet, we leverage Tamarin which shows that forging such packets is possible, thus validating the feasibility of the attack.

Root cause. The root cause of the attack is mainly due to the protocol design of the handover procedure. The specification [2] only states: *"The UE sends an Attach Request to the MME with Request Type indicating 'Handover' Attach."* However, `Attach Request` message does not have any integrity protection [3]; thus, an adversary can spoof the message to the MME.

Attack threat model. An adversary requires to be in the radio range of the victim and to know the victim's IMSI in advance, which can be acquired with the IMSI catcher [37]. The adversary sends `Attach Request` packets with the victim's IMSI to MME through a software-defined radio (SDR) – which is very cheap ($300 USD). To make the victim UE initiate the handover between VoWiFi and VoLTE, the adversary can leverage the deauthentication attack [11] to abort the WiFi connection.

Testbed verification. Our SDR device (USRP B210) in the testbed keeps sending `Attach Request` to the MME. When we trigger the handover procedure on the victim UE, we find that the connection between the UE and the core network is dropped. The UE does not receive the IKE `Delete` payload that is the final message from the core network when the handover is completed.

Attack implication. When triggering the handover, the expectation from the specification is that the handover should be seamless to the user [2]. However, using this attack, an adversary can stop the seamless handover from VoWiFi to VoLTE, heavily hampering the resilience of cellular communication. After the attack, for around two seconds, the victim reaches a state where the UE is disconnected from the core network both through VoWiFi and VoLTE. After then, the victim is able to connect to the VoLTE. However, the attack can be run in a loop to keep the UE in a disruptive handover state, where it cannot connect to the core network.

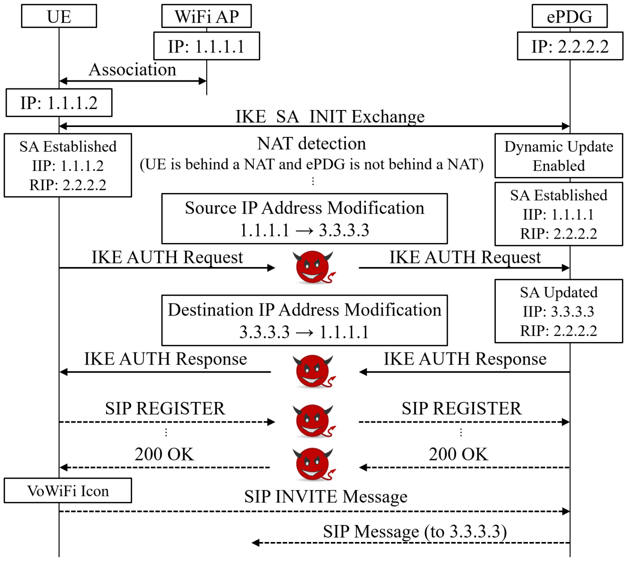

Figure 6: Denial-of-VoWiFi-Service by dynamic update (CE1)

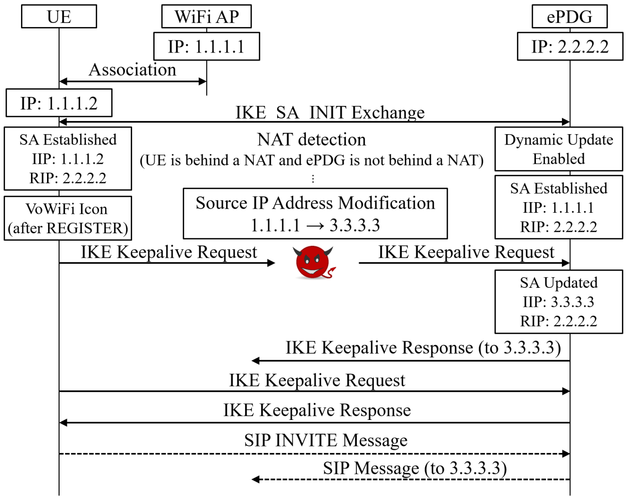

Figure 7: Denial-of-VoWiFi-Service by Dynamic Update (CE2)

7.2.2 *Denial-of-VoWiFi-Service through Dynamic Update.*

This attack makes a victim UE unable to use the VoWiFi service by breaking the communication with the ePDG without the UE's knowledge. In other words, the victim cannot make or receive a call via VoWiFi.

Scenario description. We uncover the attack from two different scenarios. In those scenarios, the dynamic update is not applied to the ESP packets (only to the IKE messages). However, the two scenarios are different in that a UE initiates the keepalive message in one scenario, while an ePDG does in the other scenario.

Attack threat model. The attacker has the capability to set up a fake WiFi AP, which can be done using previous attacks such as the evil-twin attack [8, 31, 34, 35]). Once a fake WiFi AP makes a victim UE associate with it, the AP can monitor all the IKE packets exchanged between the UE and the ePDG and selectively change the source IP address of the packets. We recognize that the threat model for this attack is strong. However, in the next section (see Section 8), we show that due to an implementation issue in most of the VoWiFi implementations, the same root cause can be exploited by a weaker adversary that only requires to be in the radio range of the victim and does not need to deploy a fake WiFi AP to carry out this attack.

Detection and attack scenarios. With the above models, we uncover the attack with regard to the property: "*If an ePDG that has enabled the dynamic update mechanism receives a packet with a different address, the sender would be one that the ePDG establishes the SA with.*" We observe two counterexamples CE1 and CE2 from nuXmv, where an ePDG updates the address to the adversary's IP address and port number. We describe them in what follows:

- **Counterexample 1 (CE1, see Figure 6):** an adversary intercepts the last `IKE_AUTH` request message, modifies the source IP address of the message to the IP address of a proxy introduced by the adversary, and sends it to the ePDG. The proxy receives the response message from the ePDG if it updates the IP address. The proxy forwards it to the adversary. Then, the adversary modifies the destination IP address to the IP address of the UE and forwards it to the UE. After that, the UE exchanges several SIP messages with the ePDG to register itself to the IMS server. The adversary sends these messages again while modifying their IP addresses and forwards the replies that the proxy receives to the UE.
- **Counterexample 2 (CE2, see Figure 7):** After a UE is registered with the IMS server, an adversary intercepts the first IKE keepalive request message from a UE, changes the source IP address of the message to the different IP address, and sends the message to the ePDG. Due to the dynamic update mechanism, the ePDG updates the address of the UE and sends the response message to the updated address. The UE cannot receive the response message, so it retransmits the request message. Although the received message is validated, the ePDG does not update the address, but the ePDG sends the response message to the source IP address of the request message (i.e., the UE); thus, the UE receives it and does not retransmit the request message.

We verify that the packet spoofed by the adversary does not violate the cryptographic assumption by using Tamarin [33].

Root cause. The vulnerability lies in the dynamic update mechanism in IKEv2 [27]. The dynamic update mechanism allows an entity to remap the IP addresses or the port number associated with the IKE SA when it receives the MAC-validated IKE packet from a different address. The mechanism is enabled for the entity not behind a NAT if the other entity is behind a NAT. Therefore, the ePDG usually enables the mechanism in VoWiFi. In order to understand whether an entity or the other is behind a NAT, there is the NAT detection mechanism in the `IKE_SA_INIT` exchange of IKEv2. To this end, the two entities exchange a `NAT_DETECTION_SOURCE_IP` payload and a `NAT_DETECTION_DESTINATION_IP` payload. The former contains a SHA-1 digest of the initiator's SPI, the responder's SPI, the source IP address, and the source port number. The latter involves the SHA-1 digest with the two SPIs, the destination IP address and the port number. The initiator and the responder include the two payloads in their `IKE_SA_INIT` messages, respectively. Then, the receiving entity calculates the expected digest values of the two payloads from the sending entity and compares them with the actual values. The sending or receiving entity determines the other is behind a NAT by detecting mismatch of `NAT_DETECTION_SOURCE_IP` and `NAT_DETECTION_DESTINATION_IP` values, respectively.

In the VoWiFi scenario, the ePDG may receive the IKE message from a different IP address or port number because of removing the NAT mapping or the change of the NAT IP address. The ePDG updates the address if the message is MAC-validated. Afterward, it sends the response message, including retransmissions to the updated address. Since the IP address of the packet is not integrity-protected and independent of the IKE message, the message is always validated even if the IP address is changed. Therefore, an adversary can make an ePDG update the address associated with a particular SA to its intended IP address. Although the SA exists, the ePDG can no longer retrieve the SA from the UE packets if the IP address is also used to find the SA. Or, the ePDG can fetch it only with the SPIs, but sends the response messages to the changed IP address; thus, the UE cannot receive the messages from the ePDG.

Testbed verification. We implement both attack scenarios and run them on our testbed. We note two observations.

First, all UEs are vulnerable to the CE1 attack because all the ePDGs dynamically update the address with IKE messages but not with ESP packets. Therefore, after the address is dynamically updated with the IKE message, the ePDG does not receive any ESP packets from UEs.

Second, even if the UE model is the same, the results are different depending on the carriers. We find that in the communications of two carriers out of three, the UE always sends the keepalive request messages. However, in one carrier, the ePDG usually initiates the keepalive exchanges. The UE only sends the keepalive request message when the UE does not receive it from the ePDG for a certain period. Thus, to perform the CE2 attack, the adversary should drop all the keepalive request messages from the ePDGs before finding the keepalive request message from the UE.

Attack implication. Due to the attack, the victim is not able to send and receive any calls. Moreover, the VoWiFi and VoLTE icons are not affected, making the attack surreptitious. The effective time of the attack depends on the IKE keepalive request message interval that varies depending on different UEs. The DoS time ranges from 30 seconds to 2 minutes after each attack. However, the attack can be run in a loop to provide continuous surreptitious DoS.

7.2.3 Denial-of-VoWiFi-Service through IKE Keepalive.

Like the previous attack, this attack makes a victim UE unable to use the VoWiFi service by dropping the keepalive message in-between the UE and the ePDG.

Scenario description. We discover the attack from the scenarios where the ePDG initiates the keepalive messages.

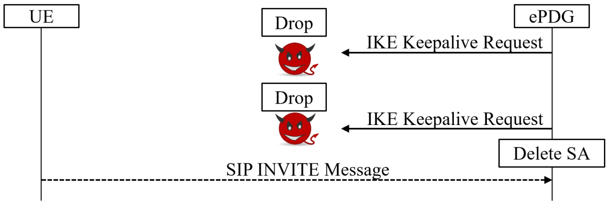

Figure 8: Denial-of-VoWiFi-Service by IKE Keepalive (CE3)

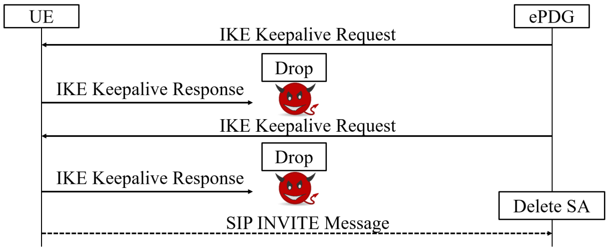

Figure 9: Denial-of-VoWiFi-Service by IKE Keepalive (CE4)

Detection and attack scenarios. With the models, we detect the attack by checking the following property: "*If one of the peers deletes the SA, then the other has deleted the SA.*" We observe two counterexamples CE3 and CE4, where an ePDG deletes its SA while a UE does not. We describe them in the following:

- **Counterexample 3 (CE3, see Figure 8):** an adversary drops the IKE keepalive request message from the ePDG. The ePDG retransmits the request message if it does not receive the message for a certain time interval. The ePDG finally thinks the UE is not alive and deletes the associated SA, while the UE still considers using the VoWiFi service.
- **Counterexample 4 (CE4, see Figure 9):** this attack scenario is similar to the scenario of CE3, but the adversary drops the IKE keepalive response message from the UE. The result of the attack is the same as CE4.

What is required for the adversary is to drop packets. Since it does not violate any cryptographic assumption, we confirm these attacks as realizable counterexamples.

Root cause. The root cause is that there is no way for a UE to be aware of the attack if an ePDG initiates the keepalive messages.

Attack threat model. The adversary has to drop the IKE packet. To this end, the adversary can use a fake WiFi AP in-between the UE and the ePDG.

Testbed verification. Similar to the experiment of CE1 and CE2, we implement CE3 and CE4. Then, we experiment on five different UEs; three have the same model for the different carriers, and the other two have different models. We check if a UE is able to make or receive a call and if a VoWiFi indicator icon is turned on during the attack.

We find that in one carrier, the ePDG usually initiates the IKE keepalive request. When the attacker drops either of the IKE keepalive request from the ePDG or the IKE keepalive response from the UE, the ePDG removes the security association and then it cannot process packets from the UE; thus, the UE cannot make or receive calls while a VoWiFi indicator icon still appears.

We note that the corresponding UE also initiates the IKE keepalive exchange if it does not receive any IKE keepalive request from the ePDG for some period. It determines how long the attack lasts. The period ranges from 2 minutes to 10 minutes after the keepalive request from ePDG or the keepalive response from UE is dropped.

Attack implication. The effect of the attack on the victim is similar to the previous attack. However, unlike CE1 and CE2, the ePDG removes the security association executing CE3 and CE4. Therefore, the UE can know the ePDG is not alive if it sends the IKE keepalive request message because the ePDG cannot respond to it. After then, the UE sends the `IKE_SA_INIT` request message to establish a new IKE channel; thus, the attacker can have another chance to perform CE1, CE2, CE3, and CE4.

7.3 RQ3. Existing Attacks

In addition to those new attacks, VWAnalyzer also discovers existing attacks from prior work [11, 32]. We summarize such attacks detected by VWAnalyzer in Table 2. We also describe the reason why VWAnalyzer can or cannot detect them.

8 IMPLEMENTATION ISSUES

This section discusses two implementation issues that we find during the testbed evaluation.

ePDG behind a NAT enables dynamic update. We find that the ePDGs of two carriers enable the dynamic update mechanism even if they turn out to be behind a NAT through the NAT detection mechanism. Due to this implementation flaw, a weaker adversary can perform the denial-of-VoWiFi-attack without deploying a fake WiFi AP. The only requirement is that the attacker has to be in the radio range of the victim. The attack procedure is as follows:

- The adversary performs a DNS spoofing attack to make the UE connect to it. From the UE's DNS query, the adversary can learn the target ePDG server and connect to the ePDG.
- The UE sends IKE messages to the adversary that forwards them to the ePDG. The adversary also forwards the messages from the ePDG to the UE.
- Now the adversary performs any of the attack steps of CE1, CE2, CE3, or CE4.

Note that from the UE's perspective the adversary is the endpoint where it connects. Thus, the UE includes the adversary's address in the `NAT_DESTINATION_IP` payload. When the ePDG receives it through the adversary, the ePDG determines that it is behind a NAT since it cannot evaluate the value of the payload. As specified by the standards, the ePDG should not enable the dynamic update mechanism if it is behind a NAT; however, we find that many ePDGs do, this leading to potential DoS attacks.

No ePDG sends its certificate. With the testbed evaluation, we find that none of the ePDGs sends the certificate. This allows an adversary to perform the IMSI catcher attack. The adversary can simply perform the DNS spoofing attack to pretend itself as the ePDG. Then, it can receive the first `IKE_AUTH` message that contains the identity of a UE. We find that all of the UEs in our testbed send their IMSI (not the TMSI) as the identity. Although this attack is already discussed in [11], they only mention that it is due to the lack of mutual authentication. However, we find that TS 33.402 [4] specifies mutual authentication as follows:

> • For untrusted access, UE and the ePDG shall perform mutual authentication during the IPsec tunnel establishment between the UE and the ePDG.
> • Public key signature-based authentication with certificates, as specified in RFC 5996, shall be used to authenticate the ePDG.

It means that there is a gap between the specification and the practice. Instead of using the public key signature-based authentication, the ePDG authentication in practice relies on the `EAP_ONLY_-AUTHENTICATION` mechanism [20], which performs the mutual authentication between a UE and an ePDG at the last `IKE_AUTH` exchange that contains the authentication messages. In this approach, the IMSI catcher attack is unavoidable. Although the current approach in the specification addresses the issue, the current practice shows that no implementation complies with the specification.

9 DISCUSSION

Manual effort. Since there is no formal description of the protocol, VWAnalyzer requires manual effort to build the initial FSMs. The manual process includes reading specifications with many pages, designing states and possible transitions, extracting the properties to be verified, and listing the constraints of the protocol. Although it takes significant effort, it is a one-time intervention. As discussed before, once our initial FSMs are built, we can automatically generate plausible scenarios resulting from them.

Soundness and completeness of VWAnalyzer. Similar to previous testing frameworks [11, 21, 22, 38], VWAnalyzer cannot achieve completeness. Since we abstract the target to be analyzed when building the model, leading to loss of information, it is not possible to uncover all the issues of the protocol. However, VWAnalyzer is sound. In case VWAnalyzer uncovers an issue, then it is a definite issue in the protocol specification.

Ethics consideration. We conduct our testbed verification in a confined setup only affecting our controlled devices. We do not send any flooding traffic toward carriers or malformed messages, which may affect the cellular infrastructure (e.g., parsing memory corruption). We only perform the experiments with valid messages within their constraints. Our purpose is to prove the effectiveness of the VWAnalyzer and to validate our findings, but not cause any damages.

Responsible disclosure. Our analysis reveals several types of protocol underspecification and implementation issues in the VoWiFi protocol, some of which may result in Denial-of-Service attacks for the VoWiFi service. Considering the sensitive nature of our findings, we responsibly disclose the vulnerabilities to the affected vendors in a coordinated way and discuss fixes to mitigate the identified issues.

Attacks	Description	Reference	Detected	Remarks
IMSI Catcher	UE exposes its IMSI to a fake ePDG	[11]	✓	Our threat model captures the DNS spoofing attack
ePDG discovery failure attack	A DNS reply contains an incorrect address of ePDG, resulting in the connection failure	[11]	✓	Our threat model captures the DNS packet manipulation
IKE_SA_INIT failure attack	The IKE_SA_INIT from UE includes inappropriate cipher suites, causing the ePDG to refuse the connection	[11]	✓	Our threat model captures the IKE message forgery
Deauthentication frame attack	The UE that receives deauthentication frames drops the ongoing calls	[11]		Our threat model does not capture the link layer attack
Stealthy call DoS attack	Sending an INVITE message without acknowledging any provisional responses causes a call DoS to a victim	[32]	✓	Our model captures the SIP protocol

Table 2: Existing attacks detected by VWAnalyzer.

10 RELATED WORK

Attacks on VoWiFi. Baek *et al.* [11] present the IMSI privacy attack and the availability attacks. Their attacks are based on the use of a fake IPsec server to catch the IMSI sent from the UE during the execution of the IKE protocol. Also, they use a WiFi deauthentication frame to block a WiFi-call. Xie *et al.* [38] show that VoWiFi devices can select insecure WiFi networks and are vulnerable to ARP spoofing attacks. Furthermore, they demonstrate that an adversary can easily infer events (e.g., call being made) by performing traffic analysis over packets transmitted between a UE and an ePDG. However, their main findings are related to the security of WiFi, which is out of scope of our work, because we aim to analyze the VoWifi protocol. Lu *et al.* [32] uncover vulnerabilities in the IMS call service and show how to use ghost calls to launch a stealthy call DoS attack against a targeted victim device. Such previous work focuses on specific parts of the VoWiFi protocol and does not develop a systematic framework to identify vulnerabilities. On the other hand, our main focus is to provide a systematic framework for a comprehensive analysis of the VoWiFi protocol.

Systematic methodologies to analyze protocols. Hussain *et al.* propose LTEInspector [21] and 5GReasoner [22] that analyze the control planes protocols of 4G and 5G core networks. Both frameworks provide formal models of the control planes and verify security properties based on the counterexample-guided abstraction-refinement principle (CEGAR) [16]. Our VWAnalyzer also relies on the CEGAR framework, but we not only focus on the specification but also deal with underspecification issues. In their work they analyze only one protocol model; however, in the case of VoWiFi, there can be multiple scenarios due to underspecifications. We systematically generate and verify the different possible scenarios resulting from underspecifications. Kim *et al.* [30] developed LTEFuzz, a stateless dynamic testing framework that analyzes the control components in operational LTE networks. They set security properties, evaluate whether the protocols verify the properties, and evaluate the operational core networks with automatically generated test cases. Unlike LTEFuzz, VWAnalyzer uses a stateful approach; thus, it can capture vulnerabilities after multiple state changes. Basin *et al.* [13] conduct a comprehensive formal verification of the 5G AKA protocol. They discuss underspecified security goals and assumptions, and consider the worst-case scenarios in the assumptions of their analysis. Our work differs from theirs in that we focus on underspecifications of the protocol behavior that may lead to insecure implementations or scenarios.

11 CONCLUSION AND FUTURE WORK

We present VWAnalyzer, a framework to analyze the VoWiFi protocol. Our framework handles different scenarios that can be possible due to underspecifications. To this end, we model the full stack of the VoWiFi protocol with 9 initial FSMs. Once we get the initial FSMs, we automatically generate model variants from each FSM due to the underspecifications and build protocol stacks for entities. We report three protocol design issues uncovered by VWAnalyzer. In future, we plan to apply this framework to other protocols that contain underspecifications.

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their suggestions to improve our paper. This research was supported by NSF under grant 2112471.

REFERENCES

[1] 3GPP TS 23.003. 2013. Numbering, addressing and identification Release 17. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729. (Accessed on 07/14/2021).

[2] 3GPP TS 23.402. 2019. Architecture enhancements for non-3GPP accesses Release 16. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=850. (Accessed on 07/21/2021).

[3] 3GPP TS 24.301. 2019. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1072. (Accessed on 03/06/2022).

[4] 3GPP TS 33.402. 2020. Security aspects of non-3GPP accesses Release 16. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2297. (Accessed on 07/17/2021).

[5] 3GPP TS 36.523-1. 2021. Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Packet Core (EPC); User Equipment (UE) conformance specification; Part 1: Protocol conformance specification Release 16. https://www.etsi.org/deliver/etsi_ts/136500_136599/13652301/16.08.00_60/ts_13652301v160800p.pdf. (Accessed on 07/20/2021).

[6] 3GPP. 2020. Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3 (Release 17). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1073. (Accessed on 07/07/2021).

[7] 3GPP. 2021. IP Multimedia Subsystem (IMS); Stage 2. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=821. (Accessed on 07/08/2021).

[8] Mayank Agarwal, Santosh Biswas, and Sukumar Nandi. 2018. An efficient scheme to detect evil twin rogue access point attack in 802.11 Wi-Fi networks. *International Journal of Wireless Information Networks* 25, 2 (2018), 130–145.

[9] J Arkko and H Haverinen. 2006. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). https://

datatracker.ietf.org/doc/html/rfc4187. (Accessed on 07/08/2021).

[10] IEEE Standards Association et al. 2007. IEEE 802.11i-2004 - IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements. https://standards.ieee.org/standard/802_11i-2004.html. (Accessed on 07/08/2021).

[11] Jaejong Baek, Sukwha Kyung, Haehyun Cho, Ziming Zhao, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. 2018. Wi not calling: Practical privacy and availability Attacks in Wi-Fi calling. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 278–288.

[12] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1383–1396. https://doi.org/10.1145/3243734.3243846

[13] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A formal analysis of 5G authentication. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 1383–1396.

[14] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. 2014. The nuXmv symbolic model checker. In *International Conference on Computer Aided Verification*. Springer, 334–342.

[15] Cisco. 2019. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022. https://s3.amazonaws.com/media.mediapost.com/uploads/CiscoForecast.pdf. (Accessed on 07/08/2021).

[16] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*. Springer, 154–169.

[17] C. Cremers and Martin Dehnel-Wild. 2019. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. https://doi.org/10.14722/ndss.2019.23394

[18] Danny Dolev and Andrew C. Yao. 1983. On the security of public key protocols. *IEEE Transactions on information theory* 29, 2 (1983), 198–208.

[19] John Ellson, Emden R Gansner, Eleftherios Koutsofios, Stephen C North, and Gordon Woodhull. 2004. Graphviz and dynagraph—static and dynamic graph drawing tools. In *Graph drawing software*. Springer, 127–148.

[20] Pasi Eronen, H Tschofenig, and Y Sheffer. 2010. An Extension for EAP-Only Authentication in IKEv2. https://datatracker.ietf.org/doc/html/rfc5998. (Accessed on 07/16/2021).

[21] Syed Rafiul Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. 2018. LTEInspector: A systematic approach for adversarial testing of 4G LTE. In *Network and Distributed Systems Security (NDSS) Symposium 2018*.

[22] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 2019. 5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 669–684.

[23] Ari Huttunen, Brian Swander, Victor Volpe, Larry DiBurro, and Markus Stenberg. 2005. UDP encapsulation of IPsec ESP packets. https://datatracker.ietf.org/doc/html/rfc3948. (Accessed on 07/08/2021).

[24] P&S Intelligence. 2020. VoWiFi (Voice Over Wi-Fi) Market Research Report: By Voice Client, Technology, Architecture, Device Type, End User - Global Industry Analysis and Growth Forecast to 2030. https://www.psmarketresearch.com/market-analysis/voice-over-wifi-vowifi-market. (Accessed on 07/08/2021).

[25] A Johnston, S Donovan, R Sparks, C Cunningham, and K Summers. 2003. Session Initiation Protocol (SIP) Basic Call Flow Examples. https://datatracker.ietf.org/doc/html/rfc3665. (Accessed on 07/08/2021).

[26] Imtiaz Karim, Syed Rafiul Hussain, and Elisa Bertino. 2021. ProChecker: An Automated Security and Privacy Analysis Framework for 4G LTE Protocol Implementations. In *2021 IEEE 41th International Conference on Distributed Computing Systems (ICDCS)*. IEEE.

[27] Charlie Kaufman, Paul Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. 2010. Internet key exchange protocol version 2 (IKEv2). https://datatracker.ietf.org/doc/html/rfc5996. (Accessed on 07/08/2021).

[28] S Kent. 2005. IP Encapsulating Security Payload (ESP). https://datatracker.ietf.org/doc/html/rfc4303. (Accessed on 07/08/2021).

[29] S. Kent and K. Seo. 2005. Security Architecture for the Internet Protocol. https://datatracker.ietf.org/doc/html/rfc4301. (Accessed on 07/08/2021).

[30] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. 2019. Touching the untouchables: Dynamic security analysis of the LTE control plane. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1153–1168.

[31] Joonhee Lee, Hyunwoo Lee, Jongheon Jeong, Doowon Kim, and Ted Taekyoung Kwon. 2021. Analyzing Spatial Differences in the TLS Security of Delegated Web Services. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 475–487.

[32] Yu-Han Lu, Chi-Yu Li, Yao-Yu Li, Sandy Hsin-Yu Hsiao, Tian Xie, Guan-Hua Tu, and Wei-Xun Chen. 2020. Ghost calls from operational 4G call systems: IMS vulnerability, call DoS attack, and countermeasure. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[33] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*. Springer, 696–701.

[34] Diogo Mónica and Carlos Ribeiro. 2011. Wifihop-mitigating the evil twin attack through multi-hop detection. In *European Symposium on Research in Computer Security*. Springer, 21–39.

[35] Chunyi Peng, Chi-Yu Li, Hongyi Wang, Guan-Hua Tu, and Songwu Lu. 2014. Real threats to your data bills: Security loopholes and defenses in mobile data charging. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 727–738.

[36] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. 2002. SIP: Session Initiation Protocol. https://datatracker.ietf.org/doc/html/rfc3261. (Accessed on 07/08/2021).

[37] Daehyun Strobel. 2007. IMSI catcher. *Chair for Communication Security, Ruhr-Universität Bochum* 14 (2007).

[38] Tian Xie, Guan-Hua Tu, Chi-Yu Li, Chunyi Peng, Jiawei Li, and Mi Zhang. 2018. The dark side of operational Wi-Fi calling services. In *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–1.