WCDCAnalyzer: Scalable Security Analysis of Wi-Fi Certified Device Connectivity Protocols

Zilin Shen
Purdue University
shen624@purdue.edu

Imtiaz Karim
University of Texas at Dallas
imtiaz.karim@utdallas.edu

Elisa Bertino
Purdue University
bertino@purdue.edu

***Abstract*—The Wi-Fi Alliance has developed several device connectivity protocols—such as WI-FI DIRECT, WI-FI EASY-CONNECT, and WI-FI EASYMESH—that are integral to billions of devices worldwide. Given their widespread adoption, ensuring the security and privacy of these protocols is critical. However, existing research has not comprehensively examined the security and privacy aspects of these protocols' designs. To address this gap, we introduce WCDCAnalyzer (Wi-Fi Certified Device Connectivity Analyzer), a formal analysis framework designed to evaluate the security and privacy of these widely used Wi-Fi Certified Device Connectivity Protocols. One of the significant challenges in formally verifying the WI-FI DIRECT protocol is the scalability problem caused by the state explosion resulting from the protocol's large scale and complexity, which leads to an exponential increase in memory usage. To address this challenge, we develop a systematic decomposition method following the *compositional reasoning* paradigm and integrate it into WCDCAnalyzer. This allows WCDCAnalyzer to automatically decompose a given protocol into several sub-protocols, verify each sub-protocol separately, and combine the results. Our design is a practical application of compositional reasoning based on rigorous foundations, and we provide detailed algorithms showing how this reasoning approach can be applied to cryptographic protocol verification. Using WCDCAnalyzer, we analyze these protocols and newly discover 10 vulnerabilities, including authentication bypass, privacy leakage, and DoS attacks. The vulnerabilities and associated practical attacks have been validated on commercial devices and acknowledged by the Wi-Fi Alliance.**

I. INTRODUCTION

Wi-Fi is a widely used wireless technology that enables devices to connect to the Internet and communicate with one another. It operates according to the 802.11 family of standards [1], providing the protocols for Wireless Local Area Networks (WLANs). Apart from WPA (Wi-Fi Protected Access) protocols, the Wi-Fi Alliance has also designed several device connectivity protocols for various environments, including the Internet of Things (IoT) environment. We refer to these protocols as **WCDC** (**W**i-Fi **C**ertified **D**evice **C**onnectivity) protocols.[1] These protocols are designed to interact with Wi-Fi Protected Access (WPA) protocols and play a crucial role in enabling devices to connect and communicate with each other and with applications across the Internet. Although the Wi-Fi Alliance designs both WPA and WCDC protocols, they differ significantly in their pairing processes. Some WCDC protocols, such as WI-FI DIRECT [2], have been widely adopted and are included in Android and Linux systems [3], [4] and have been deployed in over 3 billion devices [3]. For such a widely deployed protocol, it is crucial to ensure the security and privacy of the protocol design.

Prior works and problem. In terms of prior research on the security and privacy of WCDC protocols, there is a notable lack of preliminary efforts to analyze the security and privacy of these protocols. This gap can be attributed to several factors. First, the complexity of protocols like WI-FI DIRECT poses significant challenges for analysis. Second, these protocols are relatively newer compared to WPA, leading to less scrutiny in academic research. In a broader context, previous work [5] adopts formal analysis of WPA2 four-way and group-key handshakes. Therefore, in this paper, we pose the following research question: *Is it possible to develop a formal analysis framework to investigate the security and privacy properties of WCDC protocols comprehensively?*

To resolve this question, we design and develop WCDCAnalyzer, the first *comprehensive formal analysis* framework focusing on WI-FI DIRECT [2], WI-FI EASYCONNECT [6], and WI-FI EASYMESH [7]. These protocols serve different purposes, such as initial pairing or efficient data transfer. Since the initial pairing step plays a key role in the overall security, we choose these three protocols because they incorporate such processes.

Challenges. The formal analysis of the WCDC protocols, especially WI-FI DIRECT, with state-of-the-art verification tools such as Tamarin [8], presents one critical roadblock: *scalability*. This is due to the large size and complexity of the WI-FI DIRECT protocol. In formal analysis, the complexity of computation typically scales quickly with the size of the protocol and the model, particularly for cryptographic protocols operating in adversarial environments [9]. Scalability poses a significant challenge in analyzing and verifying systems as they grow in size. The primary reason is the state explosion problem [10], where the number of possible system states increases exponentially, resulting in a large state space and excessive memory consumption. An approach to addressing

[1] "WCDC" is a collective term used to denote protocols developed by the Wi-Fi Alliance for various connection environments.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-99919276-8-0
https://dx.doi.org/10.14722/ndss.2026.231049
www.ndss-symposium.org

scalability is to use abstraction techniques [11], but this method risks overlooking critical vulnerabilities. Other previous approaches have proposed segment-based analysis [12] and module-based analysis [13]. However, these approaches focus solely on the modular construction of models rather than modular verification. On the whole, designing a scalable formal analysis framework for WI-FI DIRECT poses a non-trivial challenge due to the large size and complexity of the protocol.

Solution outline. *Compositional reasoning* is a known theoretical paradigm for solving scalability problems [14]. In the context of cryptographic protocol verification, a theoretical study has provided soundness guarantees for compositional reasoning under the requirement of *disjoint cryptographic primitives* [15]. However, this approach has not been applied to verify real-world, complex wireless communication protocols. Therefore, in this work, we explore the practicality of this paradigm in designing a *compositional verification* method to address the computational scalability challenge of WI-FI DIRECT's formal analysis. The *compositional verification* method allows us to automatically break down complex protocols into several smaller, manageable sub-protocols, which can be verified independently and then composed to verify the overall system's correctness. Similarly, the security properties are decomposed into individual, local properties. Each local property is then verified within its respective sub-protocol. Finally, these separate verifications are combined to obtain the overall verification results. Designing an automatic decomposition process and integrating it into *Tamarin* poses several critical challenges (discussed in detail in section III). We address these challenges through the development of WCDCAnalyzer.

Our approach. The WCDCAnalyzer workflow consists of several steps. First, similar to any automated protocol formal analysis framework [8], [16], we manually analyze the protocol specifications and extract the Protocol State Machines (PSMs) of the WCDC protocols. Similarly, we extract security properties of the protocols. Then, these protocols in the format of PSMs are automatically decomposed into several *sub-protocols*. Just like any wireless communication, the WI-FI DIRECT protocol consists of multiple step-by-step protocol processes. We utilize this inherent divisional property to decompose it into several sequential sub-protocols. To automatically decompose the protocol, we utilize the intuition of finding Strongly Connected Components (SCCs) [17]. SCCs are directed graphs with a path from every state (vertex) to every other state in that component. Identifying SCCs in the protocol provides a *natural and automatic* process to decompose the original protocol into sequential sub-protocols. After identifying the SCCs, it is necessary to ensure that the requirement of *disjoint cryptographic primitives* is met to maintain the soundness of the decomposition and compositional verification method [15]. To achieve this, adjacent SCCs that share the same cryptographic primitives are combined to guarantee that each sub-protocol has disjoint cryptographic primitives, thereby satisfying the theoretical requirements for soundness. Then, like the protocols, the properties of the whole protocol are decomposed into several isolated properties corresponding to these sub-protocols. Subsequently, the isolated properties are verified to obtain individual results, which are combined to obtain the final verification results. We then focus on the counterexamples of the final results, through which we discover new potential attacks.

Results. We instantiate WCDCAnalyzer with Tamarin [8]– a widely used formal analysis tool used to analyze a wide range of protocols [2], [6], [7]. Using WCDCAnalyzer, we identify 10 newly discovered vulnerabilities, including authentication bypass, privacy leakage, and Denial-of-Service (DoS) attacks. These attacks enable an attacker to track a victim device's location, impersonate it, or force the device to drop its connection. We verify these attacks on 19 commercial devices in total. The validation demonstrates the high-impact issues we have identified. Later, during the testing, we also found an implementation issue. In addition to uncovering these vulnerabilities, we formally verify that certain protocols satisfy their intended security properties under realistic assumptions, assuring their correctness. Furthermore, our experiments demonstrate that WCDCAnalyzer substantially reduces the time and memory consumption in the property verification process while verifying WCDC protocols with Tamarin.

Contributions. In summary, the contributions of our paper are as follows:

- We propose the WCDCAnalyzer framework for formally analyzing three WCDC protocols. To the best of our knowledge, we are the first to systematically analyze the security and privacy of these protocols' specifications.
- To successfully verify these WCDC protocols, we design and develop an automatic decomposition methodology to address the scalability problem of WI-FI DIRECT formal analysis. As part of this methodology, we design systematic algorithms and provide a theoretical basis.
- We newly discover 10 specification vulnerabilities from the verification. All the 19 commercial devices are impacted by the vulnerabilities.

Responsible disclosure and acknowledgment. The vulnerabilities have been responsibly disclosed to the Wi-Fi Alliance and all affected vendors. *The Wi-Fi Alliance has acknowledged the vulnerabilities* and said that they will be resolved in the next version of the standards. Most of the vendors have replied that they will revise the implementations based on the standard changes by Wi-Fi Alliance.

Open-source. Artifacts related to this project's formal verification and the decomposition methodology are open-sourced at https://github.com/Zilinlin/WCDCAnalyzer/.

II. BACKGROUND

This section provides the background on three WCDC protocols relevant to WCDCAnalyzer, introduces the Tamarin prover, presents an overview of the protocol composition problem, and defines the key concepts used in WCDCAnalyzer.

A. WCDC Protocols

WI-FI DIRECT, EASYCONNECT, and EASYMESH are the WCDC protocols with pairing and connection processes.

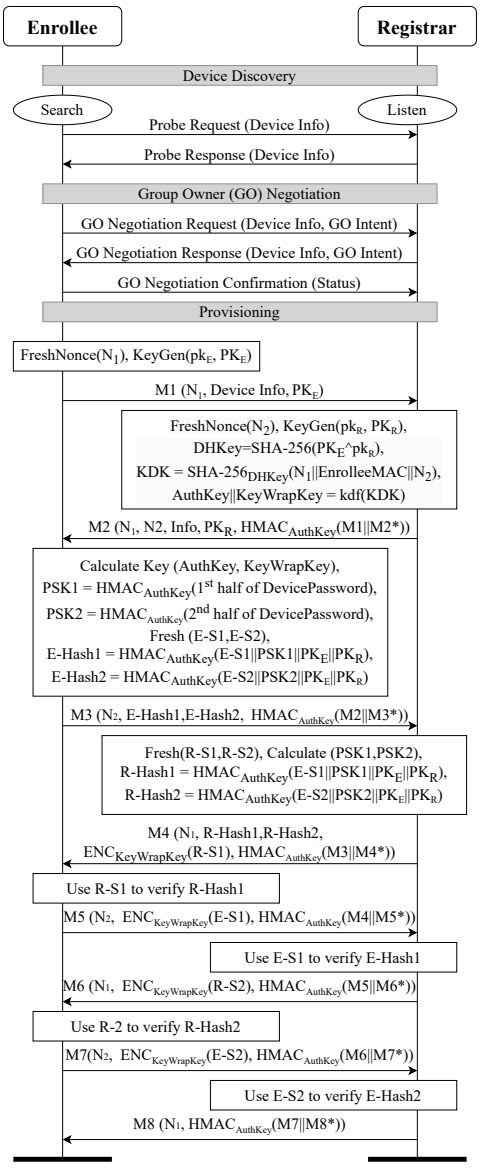

Fig. 1. The eight-message flow includes the messages from M1 to M8 of the WSC provisioning protocol. The key exchange uses Diffie-Hellman key exchange and cryptography suite to secure communications. For authentication, a hash value of the device password is used to verify the identity of the communicating devices.

Wi-Fi Direct [2], *a.k.a.* Peer-to-Peer (P2P), allows Wi-Fi devices to connect directly to each other without the need for a central network. We focus on the P2P discovery process, which helps devices find each other quickly and set up a direct connection. P2P discovery consists of *Device Discovery* and *Group Formation*. *Device Discovery* facilitates the exchange of device information between two P2P devices. *Group Formation* determines which device will become the P2P Group Owner and establish a new P2P Group. The *Group Formation* process includes two steps: *Group Owner (GO) negotiation* and *Provisioning*. The provisioning process uses the Wi-Fi Simple Configuration (WSC) [18] method. Provisioning includes an eight-message flow (labeled M1 to M8) to verify the identity of the devices (see Fig. 1). We analyze the whole P2P discovery process as it plays a crucial role in discovering and connecting P2P devices. On the side of the WSC provisioning process, there are two main configuration methods: ❶ *PushButton Configuration:* The user presses a button on one device to start the authentication process. ❷ *PIN-Based Configuration:* the user enters an 8-digit PIN displayed on one device on another device. This PIN acts as the password required for authentication.

Wi-Fi EasyConnect [6], *a.k.a.* the Device Provisioning Protocol (DPP), offers a secure and user-friendly way to connect Wi-Fi devices. Our primary interest lies in the DPP bootstrapping and authentication processes used for the initial pairing between devices. This protocol uses public bootstrapping keys obtained through QR code scanning for bootstrapping. Then a three-message exchange is executed for mutual authentication between the *Initiator* and *Responder*. Furthermore, Elliptic Curve Cryptography (ECC) is used for cryptographic functions, and the Diffie-Hellman method is used for key exchange. The process for DPP bootstrapping and authentication is detailed in Fig. 5 in the appendix.

Wi-Fi EasyMesh [7] is a standard approach for multiple-AP home Wi-Fi networks, which extends Wi-Fi coverage and enhances performance throughout homes or small offices. The WI-FI EASYMESH provisioning method is the same as the one used in the Device Provisioning Protocol (DPP). There are three entities in the WI-FI EASYMESH protocol: *Enrollee*, *Multi-AP Agent*, and *Multi-AP Configurator*. The *Multi-AP Agent* acts as a proxy in the middle to relay the messages between *Enrollee* and *Multi-AP Configurator*.

B. Tamarin Prover

The Wi-Fi Device Connectivity protocols are cryptographic protocols, that is, communication protocols that use cryptographic functions to achieve security goals such as secrecy and authentication. These protocols have non-finite state concepts such as nonces and keys, encryption, and decryption functions [9]. Specialized tools for analyzing cryptographic protocols include Tamarin [8], ProVerif [16], and DeepSec [19]. Among these, Tamarin is an advanced tool for facilitating automated symbolic analysis of security protocols, which has been widely used in formal analysis of critical protocols, such as TLS 1.3, 5G, and WPA2 [5], [20]–[28]. Moreover, Tamarin ensures correctness by guaranteeing that no false attacks are reported upon termination [29]. Given these advantages, we select Tamarin as our analysis tool. To formalize the security protocol with Tamarin, ultimately, we represent the protocol as a collection of multiset rewriting rules. Once the entire protocol is specified using multiset rules, we use first-order logic formulas [30] to represent security properties. These properties encompass authentication, secrecy, and more, serving as metrics for verifying the security of the protocol. We

introduce practical steps for constructing models and writing properties of Tamarin in Appendix A-A.

C. Protocol Composition

The *protocol composition* problem assesses the security of multiple protocols composed together. Our work uses this approach to assess the security of combining sub-protocols derived from a larger protocol. The protocol composition theorem holds for any cryptographic primitive that can be modeled using equational theories, provided that the cryptographic functions of the composed protocols, such as encryption and hash functions, are disjoint [15]. The previous result, albeit in theory, demonstrates that in the Dolev-Yao model [31], if two protocols use disjoint cryptographic primitives, their composition is secure as long as the individual protocols are secure, even if they share data [15]. We leverage the theory of *protocol composition* to design a practical WCDC verification framework — WCDCAnalyzer — and establish the soundness of the compositional verification component. Since it is evident that if any sub-protocol is insecure, the entire protocol is also insecure, we rely on the *protocol composition* result to ensure that if all sub-protocols are secure, the entire protocol remains secure under the stated *disjoint cryptographic primitives* requirements.

An important concept in protocol composition, which we use in the design of WCDCAnalyzer, is the concept of *synchronization phase* [15], Let $\mathcal{P}_i$ and $\mathcal{P}_j$ be two sub-protocols of a given protocol $\mathcal{P}$, we denote the fact that $\mathcal{P}$ first runs $\mathcal{P}_i$ and then runs $\mathcal{P}_j$ as $\mathcal{P}_i \cdot \mathcal{P}_j$. The *synchronization phase* is a phase in the execution of $\mathcal{P}$ in which upon successfully completing the execution of $\mathcal{P}_i$, all the protocol roles involved in the execution of $\mathcal{P}_i$ reach a synchronized state. For example, let a and b be two roles involved in the execution of $\mathcal{P}_i$, and let the expression $\mathcal{P}_i = \mathcal{P}_i^a | \mathcal{P}_i^b$ denotes that the two protocol roles run in parallel. Then the synchronization phase is the phase when $\mathcal{P}_i$ has successfully completed its execution, both $\mathcal{P}_i^a$ and $\mathcal{P}_i^b$ have finished the execution. In the case of WI-FI DIRECT, the synchronization phase ensures that once $\mathcal{P}_1$ completes, both the *Registrar* and the *Enrollee* have successfully finished the group owner negotiation.

D. Formalizations

We now introduce the definitions and concepts used in WCDCAnalyzer. In a protocol, messages sent by the protocol roles are expressed as *terms* built from variables, constants, and function symbols. We assume to have the pairwise disjoint sets $Fresh$, Var, and $Func$ of freshly generated terms, variables, and function names. The definition of *term* is as follows:

Definition 1 (Term)**.**

$$Term ::= Fresh \mid Var \mid (Term, Term) \mid Func(Term)$$

Elements of the set $Func$ can be used to model all the cryptographic functions, such as hash, encryption, and decryption functions. We let $\mathbf{E} = \{l_i = r_i\}_{i \in 1,\ldots,n}$ be an equational theory, where l_i, r_i are terms. We say that two terms s and t are equal in the equation theory $\mathbf{E}$. A classic example of equational theory is the modeling of symmetric encryption. *dec* and *enc* are functions of arity two and represent the decryption and encryption operator. *m* represents some plaintext message and *k* represents the key. Then the equational theory here is $\mathbf{E}_{enc} = dec(enc(m, k), k) = m$. The protocols operate in an environment controlled by a Dolev-Yao-style [31] adversary, which can intercept, read, and modify the messages. A protocol *role* can engage in the following actions:

Definition 2 (Actions)**.**

$$Action ::= fresh(t) | assign(t) | send(t) | recv(t)$$

t means term here. The actions include generating a fresh variable, assigning a term, and sending or receiving a message. Assigning a term refers to the operation $Term := Func(Term)$, where a term is assigned either a variable or the result of a function applied to one or more variables. For example, $x =_E senc(m, k)$ indicates that x is assigned the value of m encrypted with the key k. We use a to represent a single action, A to represent a set of actions, and $\mathcal{A}$ to denote a collection of sets of actions. Then, we introduce the definition of a protocol state machine:

Definition 3 (Protocol State Machine)**.** A protocol state machine is defined as a tuple $\mathcal{P} = (\mathcal{A}, S, s_0, \delta, F)$ where: S is the set of states, $s_0 \in S$ is the initial state, $F \subseteq \mathcal{S}$ is a non-empty set of final states, $\mathcal{A}$ is collection of action sets that induce transitions between states, and $\delta : S \times \mathcal{A} \to S$ is the state transition function.

We assume that an action set $A = \{a_1, a_2, \ldots, a_n\}$ defines the actions that cause transitions between states. For example, a transition between two states may involve multiple actions, such as receiving and sending a message, grouped in one action set. The collection of all action sets for the transitions in the protocol state machine $\mathcal{P}$ is denoted by $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$.

Definition 4 (Term Set of a Protocol)**.** The *term set* of the protocol $\mathcal{P}$, denoted by Terms($\mathcal{P}$), is defined as the set of all terms that appear in the actions within the protocol. Formally:

$$\text{Terms}(\mathcal{P}) = \bigcup_{a \in \mathcal{A}} \{t_i \mid a = \textit{action}(t_1, \ldots, t_n),\ t_i \text{ appears in } a\}$$

In the previous definition, the term *action* is used with the meaning of general action, such as sending and receiving a message.

Definition 5 (Cryptographic Primitives in a Protocol)**.** The *cryptographic primitives* of a protocol $\mathcal{P}$, denoted by $CryptoPrims(\mathcal{P})$, are defined as the set of all cryptographic primitives used in the actions within the protocol. Formally:

$$CryptoPrims(\mathcal{P}) = \bigcup_{a \in \mathcal{A}} \{c \mid c \text{ is used in } a\},$$

where $\mathcal{A}$ represents all actions in $\mathcal{P}$, and c is a cryptographic primitive used in an action a.

Most cryptographic primitives are associated with *assign* actions of the form $Term := Func(Term)$. A cryptographic primitive *c* is considered used in an action if the function $Func$ is a cryptographic operation belonging to c. For example,

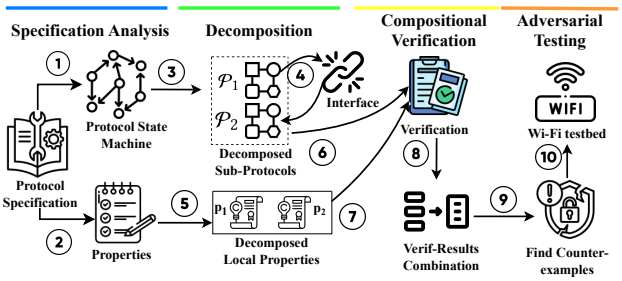

Fig. 2. The fundamental workflow of WCDCAnalyzer consists of four components, each represented in a different color.

the function *senc* (symmetric encryption) belongs to the *symmetric-encryption* cryptographic primitive.

III. Workflow and Challenges

In this section, we discuss the computational scalability issue that we encountered during the formal analysis, outline the workflow of WCDCAnalyzer, and highlight the key challenges faced in its design.

A. Computational Scalability Issue

The formal analysis of Wi-Fi Direct is infeasible due to the associated memory requirements. From our experiments with a machine equipped with 192 cores and 2 TB of RAM, the verification process is terminated after a few hours due to memory exhaustion. There can be multiple reasons for the memory explosion. Since each state represents the global system's status at a given time, a state is essentially a snapshot of the system's memory. As a result, the size of the state space increases exponentially with the size of the memory [9]. For example, the Wi-Fi Direct protocol involves numerous messages. Each message generates a *sending* event and a corresponding *receiving* event, with more than 50 terms sent to a Dolev-Yao-controlled public channel. As the number of messages increases, the state space grows exponentially, resulting in rapid expansion of memory usage. This increase directly contributes to the computational scalability issue.

B. Workflow

The fundamental workflow of WCDCAnalyzer's protocol analysis is shown in Fig. 2. The workflow consists of four components: specification analysis, decomposition, compositional verification, and adversarial testing. Like any other formal analysis framework, WCDCAnalyzer's (steps 1 and 2) involve analyzing the protocol specification to construct the Protocol State Machine (PSM), which captures the protocol's behaviors and states. Additionally, these steps include extracting the security properties essential for verification. Then, to handle the computational scalability issue, the *decomposition* component automatically breaks down the protocol, with dependent terms precisely handled (steps 3 and 4). The *compositional verification* component verifies the *local properties* of each sub-protocol individually, and the results are combined to produce a final result (steps 6, 7, and 8). The protocol is considered secure only *if all local properties are verified*. Conversely, if any local property is falsified, the protocol's *global property* is also considered falsified. These counterexamples of falsified security properties are then manually analyzed and adversarially tested on a Wi-Fi testbed to validate the results (step 9, 10).

C. Challenges

We now discuss the challenges that were tackled to design WCDCAnalyzer.

(C1) **How can protocol $\mathcal{P}$ be decomposed into several sub-protocols?** Addressing this challenge requires answering two key questions: (i) Where can the protocol be decomposed into sub-protocols? (ii) How can the protocol be automatically decomposed into these sub-protocols?

(S1) **We use the *Strongly Connected Components (SCC)* algorithm to automate protocol decomposition into sub-protocols. We then ensure that the sub-protocols have *disjoint cryptographic primitives* to meet the theoretical requirements.** Just like most of the wireless communication protocols, the WCDC protocols consist of multiple step-by-step protocol processes. We utilize this natural divide of the protocols to decompose them into sub-protocols. For instance, in the case of Wi-Fi Direct, it is designed with distinct protocol processes, such as bootstrapping, authentication, and configuration. These processes are natural candidates for dividing the protocols since they are sequential. Therefore, in the PSM, transitions between processes always proceed from the end state of one process to the start state of the next, emphasizing the directionality of protocol operations and ensuring that all transitions are unidirectional. Thus, the problem of decomposing the PSMs involves identifying these sequential processes within the PSM. To achieve this, our design leverages algorithms for identifying Strongly Connected Components (SCCs) in graphs [17]. By definition, transitions between adjacent SCCs are unidirectional. Identifying SCCs within the PSM naturally decomposes a large PSM into several sequential processes. To automate it, we utilize Tarjan's algorithm [32]. Furthermore, all decomposed sub-protocols operate within the *synchronization phase*. This ensures that once a sub-protocol is completed, all involved protocol roles have finished execution. Since the soundness of the decomposition method is based on the requirement of *disjoint cryptographic primitives*, after identifying the SCCs, we do a simple static analysis pass to verify whether they meet this requirement. If all SCCs have disjoint cryptographic primitives, they are directly used as sub-protocols. However, if certain SCCs share cryptographic primitives—such as both utilizing *symmetric-encryption*—we merge these SCCs into a single sub-protocol to satisfy the requirement.

(C2) **How to manage dependent terms between sub-protocols?** Decomposing a large protocol into sub-protocols introduces the challenge of managing dependencies between them. For example, in Wi-Fi Direct, the *DeviceInfo* generated in the first sub-protocol is required by subsequent sub-protocols for correct execution. Addressing this challenge is non-trivial, as it requires first identifying such dependencies

and then propagating the necessary information forward based on the context established in the earlier sub-protocol.

(S2) **To address this challenge, we introduce the notion of an *interface*.** The interface bridges sub-protocols by enabling the transfer of dependent terms through a designated interface state and its associated transitions. Positioned at the start of a sub-protocol, it ensures consistent propagation of terms from previous phases. To detect and manage such dependencies, we propose a novel approach based on verifying secrecy properties in the previous sub-protocol. If the secrecy of a term is maintained, it is treated as freshly generated; if falsified, we explicitly expose it through an output action to reflect the knowledge of the attacker. This mechanism ensures semantic consistency across sub-protocols while requiring careful analysis of interphase information flow, a subtle and technically challenging task.

(C3) **How to verify the properties across multiple sub-protocols?** Properties derived from specifications and broader security requirements are defined over the entire protocol and are referred to as *global properties*. To verify them at the sub-protocol level, we require a systematic approach to map each global property to the relevant sub-protocols.

(S3) **We address this challenge by decomposing the global properties into local properties, each corresponding to a specific sub-protocol.** We split a global property into multiple local properties, enabling separate verification for each sub-protocol. A *local property* applies only to one sub-protocol, making it directly verifiable. For example, consider a protocol $\mathcal{P}$ that is decomposed into two sub-protocols, $\mathcal{P}_1$ and $\mathcal{P}_2$. If a global property p spans both $\mathcal{P}_1$ and $\mathcal{P}_2$, it is decomposed into two local properties, p_1 for $\mathcal{P}_1$ and p_2 for $\mathcal{P}_2$, using the property decomposition method. Verification then proceeds as follows. p_1 ensures that p holds during $\mathcal{P}_1$'s execution, and p_2 ensures that p holds during $\mathcal{P}_2$'s execution. Finally, once all local properties corresponding to a global property have been verified, their individual results are combined to determine the validity of the global property. The global security property is considered verified if all local properties are verified. For example, in WI-FI DIRECT, the executability property is decomposed to ensure that group owner negotiation completes successfully in $\mathcal{P}_1$, and provisioning proceeds correctly in $\mathcal{P}_2$.

IV. DETAILED DESIGN

In this section, we delve into the details of WCDCAnalyzer.

A. Specification Analysis

We extract the Protocol State Machine (PSM) of the Wi-Fi Device Connectivity protocols, which spans over 1,000 pages of documentation, including WPS (Wi-Fi Protected Setup) protocol documentation [18]. We derive the security properties based on both the protocol specification and broader security requirements. These properties fall into key categories: executability, secrecy, authentication, and privacy. Each property is initially expressed in natural language to aid understanding. Detailed formalizations are provided in Appendix A-B.

❶ **Executability:** Executability ensures that a protocol $\mathcal{P}$ can execute successfully. In natural language, it is described as: "Actions $Action_1, Action_2, \ldots, Action_n$ can be executed during a run", where these actions form the execution trace. For ease of representation, we write $Executable(\mathcal{P})$ to denote the executability of protocol $\mathcal{P}$. For example, in the WI-FI EASYCONNECT protocol, executability covers the sequence of actions, including bootstrapping, and the exchange of request and response messages between the two devices involved in the setup.

❷ **Secrecy:** Secrecy ensures that a variable v is not known to the attacker. We write $Secret(v, \mathcal{P})$ to denote v's secrecy during the execution of protocol $\mathcal{P}$. For example, in the WI-FI DIRECT protocol, a secrecy property ensures that the private key used by *enrollee* is not revealed to the attacker.

❸ **Authentication:** Authentication ensures that whenever $Role_1$ and $Role_2$ accept each other's identities at the end of a run, their records of the run match. This property is described as: "If $Role_1$ accepts $Role_2$'s identity, then $Role_2$ must have been running the authentication process". We use $Auth(Role_1, Role_2, \mathcal{P})$ to denote $Role_2$ is authenticated from $Role_1$ point of view during the execution of protocol $\mathcal{P}$. For example, in the WI-FI DIRECT protocol, authentication means that if the *Registrar* completes the provisioning process, then the *Enrollee* must have participated in the provisioning beforehand.

❹ **Privacy:** Privacy ensures that a device cannot be traced by an attacker across multiple sessions. This property is described as: "The attacker cannot link protocol runs to the same device identity." We follow established definitions of *passive* and *active* privacy as introduced in [33]. We write $Privacy(id, \mathcal{P})$ to denote that the identity id remains unlinkable during the execution of protocol $\mathcal{P}$. For example, in the WI-FI DIRECT protocol, passive privacy requires that the UUID should not be observable by the attacker, as it can be linked to a specific device across sessions.

B. Decomposition

1) Protocol Decomposition

In protocol decomposition, a PSM ($\mathcal{P}$) is split into several sequential sub-protocols, such as bootstrapping and authentication, represented as $\mathcal{P} = \mathcal{P}_1 \cdot \mathcal{P}_2 \cdot \cdots \cdot \mathcal{P}_n$. To achieve this, protocol decomposition consists of two phases. First, we use Tarjan's Algorithm [32] to identify Strongly Connected Components (SCCs) within the protocol, forming the initial decomposition. Second, we check whether these SCCs have disjoint cryptographic primitives. If the SCCs have disjoint cryptographic primitives, each SCC is treated as a separate sub-protocol. Otherwise, SCCs that share cryptographic primitives are combined into a single sub-protocol to satisfy the disjointness requirement. If, after these two phases, the result is a single sub-protocol, it indicates that the protocol cannot be decomposed into multiple sub-protocols and does not meet WCDCAnalyzer's requirements. In the running example shown in Fig. 3, the protocol is decomposed into two sub-protocols, $\mathcal{P}_1 \cdot \mathcal{P}_2$. It is then verified that $CryptoPrims(\mathcal{P}_1) \cap CryptoPrims(\mathcal{P}_2) = \emptyset$ to ensure that the decomposition satisfies the *disjoint cryptographic primitives*. In detail, the

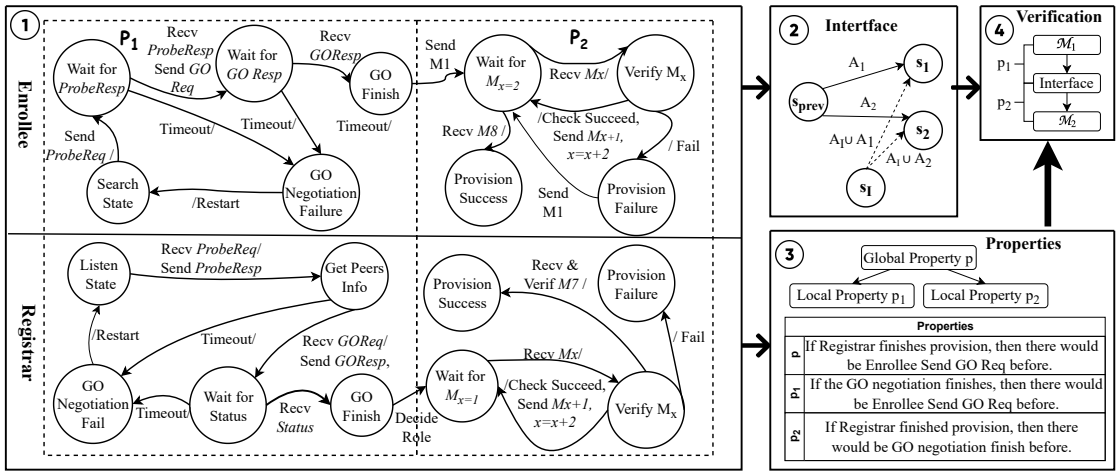

Fig. 3. Application of decomposition method integrated WCDCAnalyzer to a simplified WI-FI DIRECT protocol. *Step 1*: model construction and decomposition; *Step 2*: interface design and addition; *Step 3*: properties extraction and decomposition; *Step 4*: properties verification and results combination.

GOFinish state in $\mathcal{P}_1$ is the only state connected to $\mathcal{P}_2$, indicating that the previous sub-protocol has been executed successfully. The `GOFinish` state remains part of $\mathcal{P}_1$, while the transitions originating from `GOFinish` to states in $\mathcal{P}_2$ are included in $\mathcal{P}_2$. An interface is then added as the initial state at the beginning of $\mathcal{P}_2$.

2) Interface Building

We now focus on adding an interface to the subsequent sub-protocol to manage dependent terms. For example, in the case of two sequential sub-protocols, $\mathcal{P} = \mathcal{P}_1 \cdot \mathcal{P}_2$, $\mathcal{P}_2$ executes after $\mathcal{P}_1$ and may rely on shared terms from $\mathcal{P}_1$. To address this, we design an *interface* at the beginning of $\mathcal{P}_2$ to pass shared or dependent terms between the two sub-protocols; the definition of the interface is shown below in Def 6.

Definition 6 (Interface)**.** If a protocol is decomposed into two sub-protocols $\mathcal{P}_1$ and $\mathcal{P}_2$, we denote the state in $\mathcal{P}_1$ that connects to $\mathcal{P}_2$ as s_{prev}. And the interface is defined as the tuple $\{s_I, A_I\}$, where

- s_I is a new state we defined as *interface state*.
- A_I is an action set including the actions to transfer the dependent terms.

The details of the interface building algorithm are shown in Alg 1. Consider a protocol $\mathcal{P}$ decomposed into $\mathcal{P}_1 \cdot \mathcal{P}_2$. Firstly, the dependent terms are identified by calculating the intersection of the terms in $\mathcal{P}_1$ and $\mathcal{P}_2$, as shown in line 2 of Alg. 1. Next, the action set A_I is calculated. For each dependent term t between $\mathcal{P}_1$ and $\mathcal{P}_2$, a secrecy property for t in $\mathcal{P}_1$ is generated and verified. If t is secret during the execution of $\mathcal{P}_1$, the action $fresh(t)$ is added to A_I; otherwise, both $fresh(t)$ and $send(t)$ are included. This step corresponds to lines 8-15 and ensures the consistency of adversary knowledge when executing $\mathcal{P}_2$. Once A_I is computed, its actions are added to the original sets of actions of the transitions from s_{prev} to the states in $\mathcal{P}_2$, as described in lines 17-20. For instance, in step 2 of Fig. 3, s_{prev} belongs to $\mathcal{P}_1$, while s_1 and s_2 in $\mathcal{P}_2$ are directly connected to s_{prev}. The original action sets between s_{prev} and s_1 (A_1) as well as s_{prev} and s_2 (A_2) are updated by adding A_I, resulting in new action sets. Finally, s_I becomes the initial state of $\mathcal{P}_2$. Without this interface, formal analysis of $\mathcal{P}_2$ would fail due to missing dependent terms—such as *DeviceInfo* in WI-FI DIRECT. Moreover, if a term like *groupID* is not secret in $\mathcal{P}_1$ but the corresponding output action is missed, $\mathcal{P}_2$ may falsely verify its secrecy, resulting in unsound analysis.

Algorithm 1 Interface Building

Require: Protocol $\mathcal{P}$ is decomposed into sub-protocols $\mathcal{P}_1$ and $\mathcal{P}_2$. State s_{prev} in $\mathcal{P}_1$ is the only state transitioning to $\mathcal{P}_2$, where S_2 includes all states in $\mathcal{P}_2$ directly connected to s_{prev}.

1: /* $Terms(\mathcal{P}_1)$, $Terms(\mathcal{P}_2)$ are the terms sets of $\mathcal{P}_1$ and $\mathcal{P}_2$ */
2: $Terms_d \leftarrow Terms(\mathcal{P}_1) \cap Terms(\mathcal{P}_2)$
3: **Initiate** a new interface state s_I
4: **Initiate** interface action set $A_I = \{\}$.
5: /* Get the assumptions of all terms in $Terms_d$ */
6: **for** each term $t \in Terms_d$ **do**
7: /* The secrecy properties of $\mathcal{P}_1$ are in the same format and easy to generate. */
8: $p_t \leftarrow$ local secrecy property of $\mathcal{P}_1$
9: $r_t = Verify(\mathcal{P}_1, p_t)$
10: **if** r_t is false **then**
11: $A_I \leftarrow A_I \cup \{fresh(t), send(t)\}$
12: **end if**
13: **if** r_t is true **then**
14: $A_I \leftarrow A_I \cup \{fresh(t)\}$
15: **end if**
16: **end for**
17: **for** each state $s_2 \in S_2$ **do**
18: The original action set from s_{prev} to s_2 is A_2
19: /* Add the interface actions to the original action set */
20: $A_2 \leftarrow A_I \cup A_2$
21: **end for**
22: **return** (s_I, A_I)

3) Properties Decomposition

① **Executability:** The property $Executable(\mathcal{P})$ is decomposed into $Executable(\mathcal{P}_1)$ and $Executable(\mathcal{P}_2)$. Specifically, the actions in $Executable(\mathcal{P})$, which represent a trace of successful execution, are split into $Executable(\mathcal{P}_1)$ and $Executable(\mathcal{P}_2)$ according to their corresponding sub-protocols. For example, in the WI-FI DIRECT protocol, the group owner negotiation step is modeled in $\mathcal{P}_1$, while the provisioning steps are in $\mathcal{P}_2$. Each part must be completed

to ensure overall executability.

② **Secrecy:** The property $Secret(v, \mathcal{P})$ is decomposed into $Secret(v, \mathcal{P}_1)$ and $Secret(v, \mathcal{P}_2)$, indicating that the variable v must remain unknown to the attacker during the execution of $\mathcal{P}_1$ and $\mathcal{P}_2$. If v is not present in a particular sub-protocol, the corresponding decomposed property will be set to `None`. For example, in WI-FI DIRECT, the private key used in cryptographic key derivation appears only in $\mathcal{P}_2$ (provisioning), so $Secret(v, \mathcal{P}_1)$ is `None`, and secrecy is verified only in $\mathcal{P}_2$.

③ **Authentication:** Authentication properties represent *synchronization*. For $Auth(Role_1, Role_2, \mathcal{P})$, if $Role_1$ accepts $Role_2$'s identity, $Role_2$ must have performed corresponding actions. The decomposition:

- If $Role_1$ accepts $Role_2$'s identity" is in $\mathcal{P}_1$, $Role_2$'s actions are also in $\mathcal{P}_1$. The decomposed property for $\mathcal{P}_1$ remains $Auth(Role_1, Role_2, \mathcal{P})$, and $\mathcal{P}_2$ has `None`.
- If the acceptance action is in $\mathcal{P}_2$, $Role_2$'s actions span both sub-protocols since $\mathcal{P}_2$ executes after $\mathcal{P}_1$ completes:
  - $\mathcal{P}_1$: "If $Role_2$ successfully executes $\mathcal{P}_1$, it must have completed related actions in $\mathcal{P}_1$."
  - $\mathcal{P}_2$: "If $Role_1$ accepts $Role_2$'s identity, $Role_2$ must have executed actions in $\mathcal{P}_2$."

In WI-FI DIRECT, the registrar accepts the enrollee in provisioning $\mathcal{P}_2$, but the enrollee must exchange discovery/negotiation messages in $\mathcal{P}_1$ to authenticate its $\mathcal{P}_2$ actions.

④ **Privacy:** The property $Privacy(id, \mathcal{P})$ is decomposed into $Privacy(id, \mathcal{P}_1)$ and $Privacy(id, \mathcal{P}_2)$. Specifically, the trace-based definition of privacy in $\mathcal{P}$ ensures that the identity id remains unlinkable to the attacker, is split according to the sub-protocols. Each sub-protocol $\mathcal{P}_1$ and $\mathcal{P}_2$ must independently preserve the unlinkability of id to ensure that privacy holds for the entire protocol $\mathcal{P}$. For example, in WI-FI DIRECT, UUIDs are only announced in $\mathcal{P}_2$ during the provisioning phase. To ensure privacy, $\mathcal{P}_2$ must guarantee that the UUID is not observable, while $Privacy(id, \mathcal{P}_1)$ is set to `None`.

A global property p of protocol $\mathcal{P}$ can be decomposed into local properties p_1 and p_2. When both p_1 and p_2 are defined, this indicates that p spans multiple sub-protocols. However, if either p_1 or p_2 is `None`, then p pertains to only one sub-protocol.

C. Compositional Verification

To verify the global properties in sub-protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ for protocol $\mathcal{P}$, we design the property verification algorithm described in Alg 2. The verification process proceeds as follows. First, check whether p_1 or p_2 is `None`, which indicates that the security property p is local in a single sub-protocol. In this case, the remaining non-`None` property is verified to determine the verification result (Alg. 2, lines 1-6). If neither p_1 nor p_2 is `None`, the verification process then involves verifying p_1 in $\mathcal{P}_1$ to obtain r_1 and p_2 in $\mathcal{P}_2$ to obtain r_2. The global property p is considered verified only if both p_1 and p_2 are successfully verified (lines 7-11).

Algorithm 2 Property Verification

Require: Protocol $\mathcal{P}$ is decomposed to $\mathcal{P}_1$ and $\mathcal{P}_2$. A global property p has been decomposed to p_1 and p_2.

```
1: if p1 is none then
2:     r = Verify(P2, p2)
3:     return r
4: else if p2 is none then
5:     r = Verify(P1, p1)
6:     return r
7: else if p1 and p2 both exist then
8:     /*p is a global property*/
9:     r1 = Verify(P1, p1)
10:    r2 = Verify(P2, p2)
11:    return r1 ∧ r2
12: end if
```

D. Implementation and Theoretical Basis

Implementation. We implemented the algorithms for finding SCCs, disjoint cryptographic primitives checking, protocol decomposition, and interface building in Python 3.10, comprising approximately 350 lines of code. For verification, we used the Tamarin Prover, version 1.8.0. Overall, our Tamarin models have more than 1,500 lines of code in total.

Theoretical Basis. For the theoretical foundation of WCDCAnalyzer, we rely on the *protocol composition theorem* from [15]. This result states that if two sub-protocols are individually secure and use disjoint sets of cryptographic primitives, then their composition also preserves security. Formally, if $CryptoPrims(\mathcal{P}_1) \cap CryptoPrims(\mathcal{P}_2) = \emptyset$, and both $\mathcal{P}_1$ and $\mathcal{P}_2$ are secure, then the combined protocol $\mathcal{P}_1; \mathcal{P}_2$ is also secure. In our application, the WI-FI DIRECT protocol is decomposed into two sub-protocols: the first sub-protocol encompasses the device discovery and group owner negotiation processes. The second sub-protocol covers the provisioning process. The first sub-protocol involves exchanging device information, such as device name and group owner intent; therefore, it does not include cryptographic functions. In contrast, the second sub-protocol employs cryptographic primitives, including *symmetric encryption*, *Diffie-Hellman*, and *hash*. Since the first sub-protocol does not involve cryptographic primitives, the two sub-protocols satisfy the disjoint cryptographic primitives requirement specified in [15]. Thus, the soundness of WCDCAnalyzer is ensured under this requirement.

V. SECURITY ANALYSIS

To evaluate the effectiveness of WCDCAnalyzer in uncovering vulnerabilities in Wi-Fi Device Connectivity protocols, we pose the following research question: **RQ1:** How effective is WCDCAnalyzer in identifying and proving protocol issues in the three WCDC protocols under study? To address this question, we introduce 10 newly discovered vulnerabilities that are discovered using WCDCAnalyzer. A summary of these findings is presented in Table I.

A. Analysis Results for WI-FI DIRECT Protocol

WI-FI DIRECT has been thoroughly analyzed for verification using WCDCAnalyzer with 20 properties.

THREAT MODEL. We adopt the Dolev-Yao model [31] as our theoretical threat model. For the practical threat model, we assume that the attacker is positioned as a Man-in-the-Middle (MitM) [34]. This attacker can inject, modify, delete, and replay messages, aligning with our theoretical threat model. The MitM attacker is commonly used in practical scenarios

TABLE I
SUMMARY OF WCDCAnalyzer'S FINDINGS: 10 NEW PROTOCOL ISSUES IDENTIFIED AND FORMALLY VERIFIED.

Attack	Protocol	Vulnerability Description	Adversary Assumption	Impact
D1: Downgrade Attack	WI-FI DIRECT	The attacker sniffs the Device Name of the legal enrollee and modifies the Device Password ID to the PushButton configuration method. Then, the attacker sends the activation request to the victim registrar with a fake but legal device name. This leads the registrar to mistakenly identify the request as coming from a legitimate source.	The attacker is in the MitM location.	Auth Bypass
D2: UUID Privacy Leakage	WI-FI DIRECT	The M1 message in the provisioning process contains a UUID serving as the device's identity. The protocol does not require periodic UUID address changes.	The attacker can sniff the M1 message.	Privacy Leakage
D3: P2P Address Privacy Leakage	WI-FI DIRECT	The M1 message in the provisioning process contains the enrollee's MAC address, which serves as the device's identity. The protocol doesn't require periodic address changes.	The attacker can sniff the M1 message.	Privacy Leakage
D4: Authentication Value Causing DoS attack	WI-FI DIRECT	From message M2 to M8, an authentication part is at the end of the message. By randomly changing the authentication field's value randomly, the receiver will drop the current connection.	The attacker can sniff the message and inject the modified message.	DoS
D5: Message Code Causing DoS attack	WI-FI DIRECT	The message M1 doesn't have authentication protection. Its message code indicates which message it is currently. Modifying the message code will cause the current authentication to fail and disconnect.	The attacker can sniff the message and inject the modified message.	DoS
EC1: Bootstrapping Privacy Leakage Issue	WI-FI EASY-CONNECT	The attacker starts by sniffing the correct hash value of the responder device's bootstrapping key and sending an authentication request using this hash value. If the device confirms the hash and sends the authentication response, the attacker can confirm the device's identity.	The attacker can sniff and inject packets.	Privacy Leakage
EC2: MAC Address Leakage Issue	WI-FI EASY-CONNECT	The MAC address encoded in the QR code is static and corresponds to the device's global MAC address. If an attacker learns this MAC address, they can easily track the device's location by monitoring broadcast packets.	The attacker can sniff the packets from legal devices.	Privacy Leakage
EM1: Bootstrapping Privacy Leakage Issue	WI-FI EASYMESH	The attacker sniffs the responder's bootstrapping key hash, sends an authentication request using it, and confirms the responder's identity and location if a response is received.	The attacker can sniff and inject packets.	Privacy Leakage
EM2: MAC Address Leakage Issue	WI-FI EASYMESH	The QR code reveals the device's static global MAC address, allowing attackers to learn the MAC address and track its location.	The attacker can sniff the packets.	Privacy Leakage
EM3: Chirp Hash Value Privacy Leakage	WI-FI EASYMESH	Since the bootstrapping key can be reused and its chirp-hashed value is broadcast in the Announcement packet, an attacker can sniff this and use it to locate the device.	The attacker can sniff and inject packets.	Privacy Leakage

and real-world attacks, such as those described in [35], [36]. A common MitM attack technique is to trick the client into connecting to a rogue Access Point (AP) by making the client believe that it is connecting to a legitimate one. This technique is known as the evil twin attack [37]. Once the client connects to the malicious AP, the attacker assumes a MitM position, enabling it to view and manipulate the traffic the client sends and receives.

D1: Downgrade Authentication Bypass. In this attack, an adversary can downgrade the authentication procedure to *Push-Button* configuration and impersonate a legitimate enrollee, thereby bypassing authentication.

Detection. We use WCDCAnalyzer to verify the authentication property of the WI-FI DIRECT protocol under both *PIN-based* and *PushButton* configurations. Since these configurations occur during the provisioning phase, the analysis is localized to the second sub-protocol, $\mathcal{P}_2$. The authentication property is stated as follows: *"If Device2 completes the provisioning process, then there must exist a Device1 that previously executed the provisioning process."* Our verification shows that this property holds under the *PIN-based* configuration but is violated under the *PushButton* configuration. This discrepancy indicates that the *PushButton* configuration lacks proper authentication guarantees, which motivated further investigation into a possible downgrade attack. To understand how the attacker might exploit this, we examined the integrity of the *Device Password ID* ($DevPwdID$), which signals the chosen configuration method. In WI-FI DIRECT, $DevPwdID$ appears in both the group owner negotiation messages and the provisioning messages (M1 and M2). As such, ensuring the integrity of $DevPwdID$ is crucial to preventing configuration tampering. We specify an integrity property over $DevPwdID$: *"If device2 receives a message of type $\mathcal{T}$ from device1 containing $DevPwdID$, then device1 must have previously sent a message of type $\mathcal{T}$ with the same $DevPwdID$ value."* As this is a global property spanning both the discovery and provisioning phases, we decompose it into two local properties: In $\mathcal{P}_1$: *"If device2 receives a GO negotiation request from device1 with $DevPwdID$, then device1 must have sent such a message with the same value."* In $\mathcal{P}_2$: *"If device2 receives M1 from device1 with $DevPwdID$, then device1 must have previously sent M1 with that value."* Both local properties are falsified by WCDCAnalyzer, indicating that the attacker can arbitrarily modify $DevPwdID$—for example, to force the use of the insecure *PushButton* configuration. By combining the results of authentication and integrity property violations, we derive the downgrade authentication bypass attack illustrated in Fig. 4, which would be difficult to identify and confirm without the rigorous formal analysis conducted by WCDCAnalyzer.

Attack Steps. When a legitimate device (enrollee) intends to connect to a registrar, it initiates the process by sending a *Probe Request*, which includes its device name in plaintext. The registrar replies with a *Probe Response*, enabling device discovery. At this stage, the attacker passively sniffs the network to obtain the legitimate enrollee's device name. Using this information, the attacker impersonates the legitimate enrollee and sends a *PushButton Activation Request* to the registrar. Since the registrar's interface only displays the device name. The user is misled into believing that the request comes from the legitimate device and presses the button to

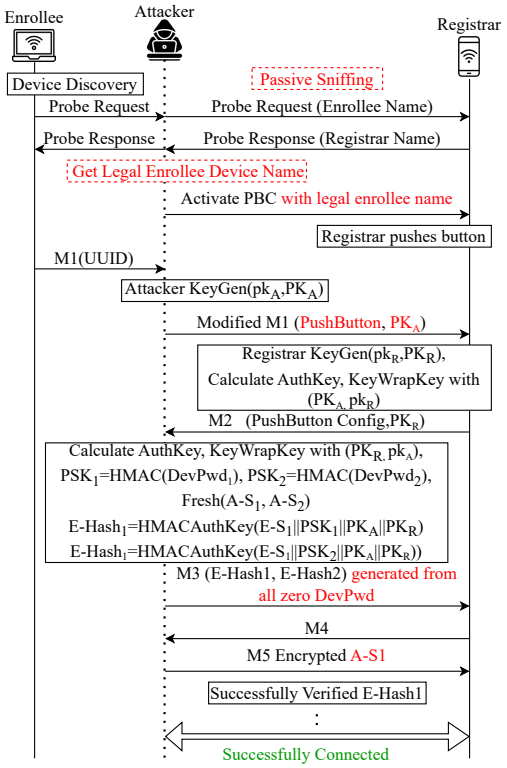

Fig. 4. The **D1** downgrade attack workflow in the WI-FI DIRECT protocol. The attacker sniffs the legitimate device name and then impersonates a legitimate device to the victim. Due to the prompt issue, the victim cannot distinguish whether the attacker is a legitimate device.

confirm. Upon pressing the button, the registrar begins the provisioning process. The attacker forges the M1 message with a modified configuration setting `DevPwdID` to indicate *PushButton*, including its own generated public key PK_A. The registrar, having activated PushButton mode, responds with M2, including its public key PK_R and matching configuration. Next, both parties compute shared keys *AuthKey*, *KeyWrapKey* based on their public keys. And because it's a *PushButton* configuration, the attacker can easily calculate the *PSK* value using an all-zero device password.

$$\text{PSK1} = \text{HMAC}(DevPwd_1), \quad \text{PSK2} = \text{HMAC}(DevPwd_2)$$

The attacker proceeds to craft M3 using E-Hash values and freshly generated secret values $A\text{-}S1$ and $A\text{-}S2$:

$$\text{E-Hash1} = \text{HMAC}_{AuthKey}(A\text{-}S1 \| PSK1 \| PK_A \| PK_R)$$
$$\text{E-Hash2} = \text{HMAC}_{AuthKey}(A\text{-}S2 \| PSK2 \| PK_A \| PK_R)$$

By using an all-zero DevPwd, the attacker ensures the registrar can successfully validate these hash values, accepting the forged session as legitimate. Finally, the session continues with encrypted values (e.g., A-S1) and completes the provisioning phase, allowing the attacker to connect without ever possessing valid credentials. This attack demonstrates how an adversary can exploit weak configuration enforcement and the lack of identity verification in PushButton mode to bypass authentication entirely.

Impact. As for the impact, users will click the button in the diagram showing "Invitation from a legal device name". As shown in Fig. 6 in the appendix, the victim device, a *Xiaomi 11 Lite*, receives a prompt indicating an invitation to connect from a *OnePlus Nord N30*. However, this prompt is sent from our attacker device, a *Dell XPS 13 Plus*. This shows that the victim device cannot distinguish whether the invitation is from a legitimate device. The attacker can then arbitrarily connect to the registrar without identity authentication, which will result in an authentication bypass.

D2: UUID Privacy Exposure. We observe that an attacker may learn the $UUID$ used during provisioning and potentially use it to identify or track devices.

Detection. We model the $UUID$ as an identifier that should not be passively exposed to an adversary. According to Table 7 of the specification [38], the UUID is a device-specific identifier that uniquely distinguishes an operational device. It appears in the M1 and M2 messages during the WI-FI DIRECT provisioning phase and can potentially be used to associate with a specific device. To evaluate this, we define a passive secrecy property over $UUID$: *"If a $UUID$ is generated, then the attacker should not be able to derive or observe its value."* Since the UUID is generated and sent during provisioning, this property is modeled as a local property within sub-protocol $\mathcal{P}_2$. We use WCDCAnalyzer to verify this property. The verification result indicates that the property is violated, meaning that $UUID$ is accessible to the adversary in some traces. This violation highlights an observable behavior in the protocol where $UUID$ is transmitted in plaintext.

Attack and impact. Since the $UUID$ is transmitted in plaintext, an attacker within range can passively capture it. Moreover, the M1 and M2 messages that carry the UUID do not require authentication to be triggered. An attacker can therefore collect UUIDs from nearby devices without needing to establish a legitimate session. Previous work on Wi-Fi privacy has focused on identifiers such as SSIDs and MAC addresses [39]–[41]. Our analysis highlights that similar tracking risks may arise from the UUID used in the provisioning phase.

Other Issues and Summary. Due to space limitations, the details of attacks **D3**, **D4**, and **D5** are provided in Appendix B-A. These include a denial-of-service (DoS) scenario, an all-zero private key issue in WI-FI DIRECT, and an implementation vulnerability caused by Android's default configuration. In conclusion, our formal analysis of the WI-FI DIRECT protocol using WCDCAnalyzer reveals both the strengths and weaknesses of the protocol design. On the one hand, we formally verify that the protocol maintains authentication under the *PIN-based* configuration, providing assurance that this mode adheres to its intended security guarantees. On the other hand, WCDCAnalyzer identifies several critical issues—including a downgrade authentication bypass and risks related to privacy exposure—that highlight the importance of rigorous, formal reasoning in discovering flaws.

B. Analysis Results for WI-FI EASYCONNECT

We analyze the overall security of the WI-FI EASYCONNECT protocol using WCDCAnalyzer with 13 properties.

THREAT MODEL. In the threat model, the attacker is at the MitM location, where the attacker can sniff and inject packets into legal devices.

EC1: Bootstrapping Key Information Exposure. The *Bootstrapping key* is an initial ECC public key used before mutual authentication. Under certain conditions, an attacker may use the hash of a device's bootstrapping public key to infer its presence and track it over time.

Detection with WCDCAnalyzer. We formalize an *active privacy* property over the bootstrapping key as follows: *"If an adversary sends an authentication request containing the hash of the bootstrapping public key to a responder, if the responder replies, the response can help the adversary confirm the device's presence."* This property is modeled and found to be violated by WCDCAnalyzer. The violation arises when a static bootstrapping key is reused across sessions. While this behavior may seem intuitive, formal analysis pinpoints the exact conditions under which such exposure becomes exploitable, helping validate the risk systematically. According to the specification, bootstrapping keys may remain static (e.g., when printed on a device's label or encoded into QR codes), which enables the attacker to trigger authentication requests using pre-collected $SHA(B_R)$ values. This attack is illustrated in Fig. 7 in the appendix.

Attack and Impact. During an earlier authentication process, the attacker begins by monitoring and obtaining the hash of the responder's public bootstrapping key, denoted as $SHA(B_R)$. Using $SHA(B_R)$, the attacker can get the responder's location by initiating an authentication request. As illustrated in Fig. 7, the attacker sends $SHA(B_R)$ and their own public key to the responder in the authentication request. If the responder validates $SHA(B_R)$ against its own public bootstrapping key and finds a match, it will reply with an authentication response. This confirmation allows the attacker to locate the responder device successfully.

Other attack and Summary. Due to space limitation, the details of attack **EC2** is provided in Appendix B-B. Through formal analysis with WCDCAnalyzer, we confirm that the DPP protocol satisfies key security properties under the modeled assumptions. At the same time, WCDCAnalyzer helps uncover potential deployment-related risks—such as information exposure through bootstrapping keys and MAC addresses—that, while not protocol-level specification violations, may pose privacy challenges in real-world scenarios.

C. Analysis Results for WI-FI EASYMESH

We evaluate the WI-FI EASYMESH protocol using WCDCAnalyzer with 16 properties.

THREAT MODEL. We assume a man-in-the-middle (MitM) attacker capable of passively monitoring and actively injecting messages between legitimate devices.

Summary. Due to space limitations, the detailed descriptions of the identified issues **EM1**, **EM2** and **EM3** are provided in Appendix B-C. Our formal analysis confirms that EASYMESH satisfies critical security properties, including mutual authentication and session key secrecy. However, we identify several privacy-related risks stemming from reused identifiers, such as bootstrapping keys and MAC addresses. While not specification violations, these issues may introduce linkability in certain deployment scenarios. We recommend using ephemeral bootstrapping keys and MAC address randomization to reduce such risks.

D. The defenses for these attacks.

Overall, the defense proposals for these Wi-Fi Device Connectivity protocols include abandoning insecure authentication mechanisms, periodically changing hard identifiers, and regularly refreshing public keys. Here, we propose some defense methods to secure these protocols: *(i)* WI-FI DIRECT. The most severe security issue regarding WI-FI DIRECT is the authentication bypass caused by a downgrade attack. Another critical problem is the widespread usage of unsecured authentication methods, such as Push-Button Configuration, which can lead to authentication bypass attacks and impersonation attacks. To defend against authentication bypass, we recommend abandoning the Push-Button Configuration in real-world applications and choosing the PIN-based configuration method. Regarding the privacy leakage problems caused by unchanged *P2P Interface addresses* and *UUIDs* in the Direct protocol, we recommend periodically changing the *P2P Interface address* and *UUIDs* to avoid privacy leakage caused by an unchanged, unique identity. *(ii)*WI-FI EASYCONNECT *and* WI-FI EASYMESH. Since the basic authentication methods of WI-FI EASYCONNECT and WI-FI EASYMESH are both DPP, their issues have the same root cause. The most severe problems in these protocols are the privacy leakages caused by the reused *bootstrapping public key* and the usage of a global *MAC address*. To mitigate these issues, we recommend regenerating a new *bootstrapping public key* each time and avoiding the use of a global *MAC address* or the same *MAC address* repeatedly.

VI. EVALUATION

We now discuss the evaluation results of WCDCAnalyzer and the testbed verification of the Wi-Fi Device Connectivity protocols. We aim to answer the following research question with our evaluation: **RQ2.** How do the found issues affect commercial devices? (Section VI-A) **RQ3.** How does WCDCAnalyzer perform with respect to computational scalability? (Section VI-B) **RQ4.** How does WCDCAnalyzer compare with approaches proposed in prior work? (Section VI-C)

A. Testbed Validation

We test 19 commercial devices, including phones, network cards, and development boards, to validate the 10 newly discovered issues. Our testing devices range from Wi-Fi 4 to Wi-Fi 6E and are from all the popular vendors. Table IV (in appendix) lists the devices we use. Note that the implementations of the three protocols are quite different, which means that not all devices implement all the protocols. Therefore, we

discuss the testbed validation of the three protocols in detail separately. The details of how we set up the testbed are shown in Appendix C. Because the vendors are still implementing the EASYMESH protocol, we cannot validate the attacks **EM1**, **EM2** and **EM3** on commercial devices.

Validation of WI-FI DIRECT. Our results show that the identified issues in the DIRECT protocol impact most commercial devices (shown in Table V). Each WI-FI DIRECT device is affected by at least four issues, and nine out of eleven devices are affected by all of these issues. The Xiaomi 12T is not vulnerable to **D3** due to its additional implementation of built-in MAC address randomization, preventing privacy leakage. The standards do not specify anything about MAC address randomization, but vendors like Xiaomi 12T proactively deploy these defensive measures. Similarly, Alfa AWUS036ACM is not vulnerable to **D1** because it operates on Linux, where the WI-FI DIRECT implementation requires explicit command input for connection rather than pushing a button, making it difficult for an attacker to impersonate a legitimate device. As the implementations do not support the *PushButton* action, users need to input a command; therefore, the downgrade authentication bypass attack is impossible.

Validation of WI-FI EASYCONNECT. Our results demonstrate that the issues in the EASYCONNECT protocol affect most commercial devices, with each device affected by at least one issue (shown in Table VI). ESP32 is not vulnerable to **EC1** because the implementation allows the generation of new keys if static keys are not defined.

B. Resource Consumption

We evaluate WCDCAnalyzer's scalability improvements by comparing the memory and time consumption with and without the decomposition method.

Experimental setup. The experiments are carried out on a server equipped with an AMD Ryzen Threadripper PRO 5965WX processor with 24 cores, 48 threads, and 256 GB of memory. The WCDC protocols are chosen to test memory and time consumption. We compare the memory and time consumption of directly verifying the three WCDC protocols $\mathcal{P}$ with Tamarin against our *decomposition and compositional verification* method in WCDCAnalyzer, which decomposes $\mathcal{P}$ into sub-protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ for separate verification.

Memory&Time Consumption. Table II presents the memory and time consumption for verifying different properties of WI-FI DIRECT protocol. We chose different types of properties to represent the range of typical verification goals. Some of these properties span both $\mathcal{P}_1$ and $\mathcal{P}_2$, whereas others apply to only one sub-protocol. The ✗ in the table indicates that the verification was terminated due to running out of memory. We chose the executability property p to illustrate how memory usage evolves over time. The executability property p of protocol $\mathcal{P}$ is a global property that spans all sub-protocols. Therefore, our decomposition method decomposes it into two local properties p_1 and p_2.

Tables VIII and VII show that, on these smaller protocols EASYCONNECT and EASYMESH, decomposition keeps memory and time usage below the whole protocol for every property. In both cases, the simple sum P_1+P_2 stays comfortably under the consumption of P, reinforcing that the savings observed on WI-FI DIRECT extend to other protocols. In conclusion, these results demonstrate that the WCDCAnalyzer decomposition methodology is crucial to reducing memory and time consumption and greatly enhancing scalability.

Property	$\mathcal{P}$		$\mathcal{P}_1$		$\mathcal{P}_2$		$\mathcal{P}_1 + \mathcal{P}_2$	
	Mem	Time	Mem	Time	Mem	Time	Mem	Time
Executability	191 G✗	80 m	195.3 M	3.7 s	156 G	494 m	156.2 G	494 m
PrivKey Secrecy	211.2G✗	110 m	*None*	*None*	336.7 M	19.9 s	336.7 M	19.9 s
Addr Privacy	189.5G✗	90 m	178 M	3.5 s	269.5 M	19.6 s	447.5 M	23.1 s
Provision Authentication	189.8 G✗	350 m	*None*	*None*	4.59 G	774 s	4.59 G	774 s
DevPwdID Integrity	190 G✗	105 m	198.7 M	3.4 s	336.8 M	17.6 s	535.5 M	20.9 s

TABLE II
MEMORY (MEM) AND TIME (TIME) CONSUMPTION FOR THE WHOLE WI-FI DIRECT PROTOCOL ($\mathcal{P}$) AND ITS SUB-PROTOCOLS ($\mathcal{P}_1$, $\mathcal{P}_2$). *None* INDICATES THAT THE PROPERTY HAS NO *actions* IN THE CORRESPONDING SUB-PROTOCOL. ✗ INDICATES THE RUN WAS KILLED BY OUT-OF-MEMORY.

Decomposition overhead. The SCC-based decomposition is very light-weight compared with formal analysis, it finishes in <10s and uses <20MB of memory. Because the decomposition performs only graph analysis, no constraint solving or state exploration is involved, making its cost negligible compared with formal analysis.

C. Comparison with prior works

We now answer the question *how does* WCDCAnalyzer *compare with the approaches proposed in previous work?* We compare WCDCAnalyzer with the three most closely related verification frameworks, including BluetoothVerif [13], Segment Wi-Fi Verification [12], and 5GCVerif [42], from two complementary angles. In BluetoothVerif [13], a *modular* design is used to construct various configurations of the Bluetooth [43] protocols. Similarly, [12] employs a *segment-based* design to verify the Power Save Mode in Wi-Fi protocols. Despite their contributions, these approaches focus on modular design for model development rather than for the verification process. 5GCVerif constructs the abstract model and addresses the state explosion problem in model checking. However, this work does not provide a systematic methodology or algorithm for using the abstraction method to reduce resource consumption.

Method	PrivKey Secrecy	Addr Privacy	Provision Auth.	PwdID Integrity
BluetoothVerif [13]	✓	—	✓	—
Segment Wi-Fi Verification [12]	✓	—	✓	—
5GCVerif [42]	✗	✓	✗	✓
WCDCAnalyzer	✓	✓	✓	✓

TABLE III
PROPERTY-LEVEL VERIFICATION OUTCOME (✓=VERIFIED, ✗= UNABLE DISCOVER VULNERABILITY, —=CANNOT FORMALLY VERIFY).

Table III presents a *property-level* evaluation on five representative security goals, highlighting whether each approach

can successfully verify a global property that spans multiple protocol components. BluetoothVerif and Segment Wi-Fi can verify only *local* properties; because they offer no reasoning mechanism across component boundaries, global properties remain out of reach. 5GCVerif can sometimes establish global goals, but its success depends on a hand-tuned abstraction level without a systematic algorithm. Although 5GCVerif can verify properties, it overlooks some critical vulnerabilities. If we lower the abstraction level to preserve those flows, the analysis runs out of memory, illustrating why a principled decomposition is needed. Keeping the full flow, on the other hand, triggers a memory blowup, further motivating the need for decomposition. In contrast, WCDCAnalyzer decomposes the protocol while preserving global reasoning power and therefore verifies all five properties.

VII. Related Work

Related works can be classified into the following three broad categories:

Attacks on WCDC and Wi-Fi. Blanco et al. [44] found vulnerabilities in the implementation of Wi-Fi Direct, including a hard-coded PIN. Altaweel et al.'s *EvilDirect* attack [45] demonstrates that a rogue Group Owner can hijack client traffic, and proposes a Received Signal Strength (RSS)-based countermeasure; Yoon et al. [46] further analyze Wi-Fi Direct for key-cracking and wireless DoS vulnerabilities. To our knowledge, no prior work systematically studies the security and privacy of the WCDC protocols' *specification* itself. In terms of the original WiFi protocol, there have been very well-known attacks against the Wi-Fi key exchange handshake [35], [47], encrypted traffic [48], and framing queue components [49]. Although WPA3 was then introduced, it has not addressed all the vulnerabilities of WPA2 [50].

Formal Analysis. Several approaches have been proposed that use formal methods to analyze the security of network protocols. These range from 5G [21], [22], [51], BLE [52], EDHOC [53], and WPA2 four-way handshake, the group-key handshake, and the WNM sleep mode [5]. Our work differs from previous work by being the first to focus on the WCDC protocols. Additionally, we developed a systematic decomposition methodology and applied it to Tamarin to address the verification challenges of Wi-Fi Direct.

Security Protocol Composition and Decomposition. The last related works are about security protocol composition. In other words, if two secure protocols are combined, what about the security of the combined protocol. Although security protocol composition and WCDCAnalyzer decomposition may appear to be two reverse processes, they address the same fundamental problem: whether the combined protocol is secure when both sub-protocols are individually secure. To clarify the definition, we generalize isolated protocols as *sub-protocol* and composed protocols as *protocol*. Ciobaca et al. [15] proved that the protocol is secure if sub-protocols use disjoint cryptographic primitives. Other works [54]–[56] demonstrate that tagged cryptographic primitives for various sub-protocols ensure secure composition under different scenarios. These works provide the theoretical foundation for our decomposition methodology. WCDCAnalyzer leverages this theoretical basis and applies it to verifying the Wi-Fi Direct protocol.

VIII. Discussion

Generality of Decomposition Method. Our developed decomposition methodology requires that the protocols be time-ordered and the procedures designed so that they can be divided using the concept of SCCs into sequential sub-protocols. Furthermore, the sub-protocols should use disjoint cryptographic primitives. Although we develop the approach for the three Wi-Fi companion protocols analyzed in this paper, other wireless communication protocols (e.g., 5G [57], Bluetooth [43], and LTE [58]) share similar properties. These protocols share common processes such as bootstrapping, key exchange, and authentication, which are strictly time-ordered. This temporal structuring makes them suitable candidates for applying our decomposition methodology. For future work, we aim to apply this methodology to formally analyze other larger protocols.

Limitations. A key limitation of our approach stems from its premise: *both* the protocol and the verification property are decomposed. Consequently, the validity of a global security goal relies on the soundness of *every* sub-property proved on the corresponding sub-protocol. If even one local proof fails, the global property collapses, because the contract-composition lemmas require all local properties to hold. In such a scenario, the tool can expose only the *local* counterexample trace; reconstructing the complete attack graph for the entire protocol still requires manual reasoning to construct the traces from multiple subprotocols. This human post-processing step limits full automation, particularly for protocols whose sub-components interact in subtle ways that generate cross-component attacks.

IX. Conclusion & Future Work

In this paper, we present WCDCAnalyzer, which formally analyzes the security and privacy of Wi-Fi Certified Device Connectivity (WCDC) protocols. We design a *decomposition and compositional verification* method to address the scalability problem for the verification with Tamarin of the large Wi-Fi Direct protocol. To systematize the *decomposition* methodology, we detail the *protocol decomposition*, *interface design* and *compositional verification*. Our experiments show that the decomposition can effectively reduce time and memory consumption. From the verification findings, we identify and present 10 new specification issues, along with a new implementation issue discovered during testing.

Future Work. We plan to formally verify practical defenses for the identified vulnerabilities and deploy them on real-world devices to validate their effectiveness.

Ethics considerations

All the experiments in this paper followed the ethics consideration policies. The experiments were done in a lab setup where both the attack and victim devices were ours. Furthermore, it was ensured that no other adjacent devices

were present and affected by the attacks. The discussed attacks have been responsibly disclosed to Wi-Fi Alliance and to all the other affected vendors. Most vendors have acknowledged and said they will modify the implementation following the guidelines of Wi-Fi Alliance.

Acknowledgment

The authors would like to express sincere gratitude to Adrian Li, Mir Imtiaz Mostafiz, and Yiwei Zhang for their valuable feedback and suggestions that greatly improved this work. Their thoughtful comments and careful review significantly enhanced the quality and clarity of this paper.

References

[1] 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. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[2] Wi-Fi Alliance. *Wi-Fi Direct (P2P) Technical Specification*, 2010. https://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

[3] DemandSage. Android statistics 2024: Market share, users, and revenue, 2024. Accessed: 2024-09-03.

[4] Wi-Fi Alliance. Internet of things. https://www.wi-fi.org/discover-wi-fi/internet-things.

[5] Cas Cremers, Benjamin Kiesl, and Niklas Medinger. A formal analysis of IEEE 802.11's WPA2: Countering the kracks caused by cracking the counters. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1–17. USENIX Association, August 2020.

[6] Wi-Fi Alliance. *Wi-Fi EasyConnect Technical Specification*, 2018. https://www.wi-fi.org/discover-wi-fi/wi-fi-easyconnect.

[7] Wi-Fi Alliance. *Wi-Fi EasyMesh Technical Specification*, 2017. https://www.wi-fi.org/discover-wi-fi/wi-fi-easymesh.

[8] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*, pages 696–701. Springer, 2013.

[9] Edmund M Clarke, Thomas A Henzinger, Helmut Veith, Roderick Bloem, et al. *Handbook of model checking*, volume 10. Springer, 2018.

[10] Edmund M Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. Model checking and the state explosion problem. In *LASER Summer School on Software Engineering*, pages 1–30. Springer, 2011.

[11] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–1542, 1994.

[12] Zilin Shen, Imtiaz Karim, and Elisa Bertino. Segment-based formal verification of wifi fragmentation and power save mode. *arXiv preprint arXiv:2312.07877*, 2023.

[13] Jianliang Wu, Ruoyu Wu, Dongyan Xu, Dave Jing Tian, and Antonio Bianchi. Formal model-driven discovery of bluetooth protocol design vulnerabilities. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2285–2303, 2022.

[14] Sergey Berezin, Sérgio Campos, and Edmund M Clarke. Compositional reasoning in model checking. In *International Symposium on Compositionality*, pages 81–102. Springer, 1997.

[15] Stefan Ciobâca and Véronique Cortier. Protocol composition for arbitrary primitives. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 322–336. IEEE, 2010.

[16] Bruno Blanchet. Automatic verification of security protocols in the symbolic model: The verifier proverif. In *International School on Foundations of Security Analysis and Design*, pages 54–87. Springer, 2012.

[17] Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.

[18] Wi-Fi Alliance. *Wi-Fi Simple Config (WSC) Technical Specification*, 2007. https://www.wi-fi.org/discover-wi-fi/wi-fi-protected-setup.

[19] DeepSec Prover Development Team. Deepsec prover: Security protocol verifier, 2024. Accessed: 2024-08-26.

[20] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of tls 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1773–1788, New York, NY, USA, 2017. Association for Computing Machinery.

[21] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1383–1396, New York, NY, USA, 2018. Association for Computing Machinery.

[22] Cas Cremers and Martin Dehnel-Wild. Component-based formal analysis of 5g-aka: Channel assumptions and session confusion. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[23] Aleksi Peltonen, Ralf Sasse, and David Basin. A comprehensive formal analysis of 5g handover. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 1–12, New York, NY, USA, 2021. Association for Computing Machinery.

[24] David Basin, Ralf Sasse, and Jorge Toro-Pozo. The emv standard: Break, fix, verify. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1766–1781. IEEE, 2021.

[25] Cas Cremers, Alexander Dax, and Aurora Naska. Formal analysis of SPDM: Security protocol and data model version 1.2. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6611–6628, Anaheim, CA, August 2023. USENIX Association.

[26] Qinying Wang, Shouling Ji, Yuan Tian, Xuhong Zhang, Binbin Zhao, Yuhong Kan, Zhaowei Lin, Changting Lin, Shuiguang Deng, Alex X. Liu, and Raheem Beyah. MPInspector: A systematic and automatic approach for evaluating the security of IoT messaging protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4205–4222. USENIX Association, August 2021.

[27] Cas Cremers, Charlie Jacomme, and Aurora Naska. Formal analysis of Session-Handling in secure messaging: Lifting security from sessions to conversations. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1235–1252, Anaheim, CA, August 2023. USENIX Association.

[28] Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, and David Basin. A spectral analysis of noise: A comprehensive, automated, formal analysis of Diffie-Hellman protocols. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1857–1874. USENIX Association, August 2020.

[29] Vincent Cheval, Charlie Jacomme, Steve Kremer, and Robert Künnemann. SAPIC+: protocol verifiers of the world, unite! In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3935–3952, Boston, MA, August 2022. USENIX Association.

[30] Jon Barwise. An introduction to first-order logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier, 1977.

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

[32] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[33] Jianliang Wu, Patrick Traynor, Dongyan Xu, Dave Jing Tian, and Antonio Bianchi. Finding traceability attacks in the bluetooth low energy specification and its implementations.

[34] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, 18(3):2027–2051, 2016.

[35] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in wpa2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1313–1328, New York, NY, USA, 2017. Association for Computing Machinery.

[36] Mathy Vanhoef. Fragment and forge: Breaking Wi-Fi through frame aggregation and fragmentation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 161–178. USENIX Association, August 2021.

[37] Harold Gonzales, Kevin Bauer, Janne Lindqvist, Damon McCoy, and Douglas Sicker. Practical defenses for evil twin attacks in 802.11. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–6. IEEE, 2010.

[38] Wi-Fi Alliance. Wi-Fi Protected Setup. https://www.wi-fi.org/discover-wi-fi/wi-fi-protected-setup. Accessed: 2025-04-23.

[39] Adriano Di Luzio, Alessandro Mei, and Julinda Stefa. Mind your probes: De-anonymization of large crowds through smartphone wifi probe requests. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.
[40] Maxim Chernyshev, Craig Valli, and Peter Hannay. On 802.11 access point locatability and named entity recognition in service set identifiers. *IEEE Transactions on Information Forensics and Security*, 11(3):584–593, 2015.
[41] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. A study of mac address randomization in mobile devices and when it fails. *arXiv preprint arXiv:1703.02874*, 2017.
[42] Mujtahid Akon, Tianchang Yang, Yilu Dong, and Syed Rafiul Hussain. Formal analysis of access control mechanism of 5g core network. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 666–680, New York, NY, USA, 2023. Association for Computing Machinery.
[43] J.C. Haartsen. The bluetooth radio system. *IEEE Personal Communications*, 7(1):28–36, 2000.
[44] Andrés Blanco. Wi-fi direct to hell.
[45] Ala Altaweel, Radu Stoleru, and Guofei Gu. Evildirect: A new wi-fi direct hijacking attack and countermeasures. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–11. IEEE, 2017.
[46] Seokung Yoon, SoonTai Park, Haeryoung Park, and Hyeong Seon Yoo. Security analysis of vulnerable wi-fi direct. In *2012 8th International Conference on Computing and Networking Technology (INC, ICCIS and ICMIC)*, pages 340–343. IEEE, 2012.
[47] Mathy Vanhoef and Frank Piessens. Release the kraken: New kracks in the 802.11 standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 299–314, New York, NY, USA, 2018. Association for Computing Machinery.
[48] Kr00k: A serious vulnerability deep inside wi-fi encryption, 2020.
[49] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. Framing frames: Bypassing wi-fi encryption by manipulating transmit queues. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.
[50] Christopher P Kohlios and Thaier Hayajneh. A comprehensive attack flow model and security analysis for wi-fi and wpa3. *Electronics*, 7(11):284, 2018.
[51] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 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*, pages 669–684, 2019.
[52] Min Shi, Jing Chen, Kun He, Haoran Zhao, Meng Jia, and Ruiying Du. Formal analysis and patching of BLE-SC pairing. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 37–52, Anaheim, CA, August 2023. USENIX Association.
[53] Charlie Jacomme, Elise Klein, Steve Kremer, and Maïwenn Racouchot. A comprehensive, formal and automated analysis of the edhoc protocol. In *USENIX Security'23-32nd USENIX Security Symposium*, 2023.
[54] Vincent Cheval, Véronique Cortier, and Bogdan Warinschi. Secure composition of pkis with public key protocols. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 144–158. IEEE, 2017.
[55] Céline Chevalier, Stéphanie Delaune, Steve Kremer, and Mark D Ryan. Composition of password-based protocols. *Formal Methods in System Design*, 43(3):369–413, 2013.
[56] Véronique Cortier and Stéphanie Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34:1–36, 2009.
[57] A. Gupta and R. K. Jha. A survey of 5g network: Architecture and emerging technologies. *IEEE Access*, 3:1206–1232, 2015.
[58] Amitava Ghosh, Rapeepat Ratasuk, Bishwarup Mondal, Nitin Mangalvedhe, and Tim Thomas. Lte-advanced: next-generation wireless broadband technology. *IEEE wireless communications*, 17(3):10–22, 2010.
[59] Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings 10th computer security foundations workshop*, pages 31–43. IEEE, 1997.
[60] Philippe Biondi et al. Scapy. https://scapy.net/.
[61] Tcpdump Group . TCPDUMP, 1999-2023.
[62] Android Developers. Android Debug Bridge (adb), 2023.
[63] Jouni Malinen et al. wpa_supplicant: Wi-fi protected access supplicant. https://w1.fi/wpa$_$supplicant/.
[64] Intel Corporation. iwd-inet wireless daemon. https://iwd.wiki.kernel.org/.

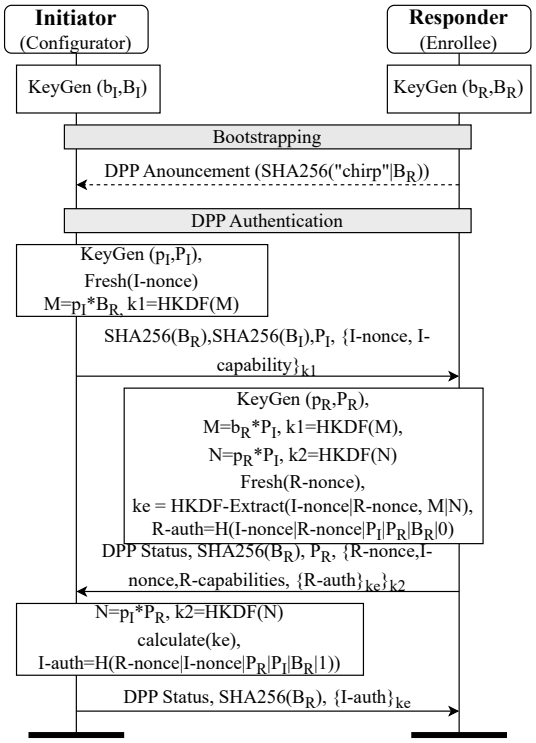

Fig. 5. The flow includes DPP bootstrapping and authentication processes. Messages shown with dotted lines are not transmitted over Wi-Fi channels; instead, they might be sent using other methods like QR code scanning. Messages depicted with solid lines are sent via Wi-Fi channels. B_R and B_I represent the public bootstrapping keys of responder and initiator, respectively, and P_R and P_I represent their public protocol keys.

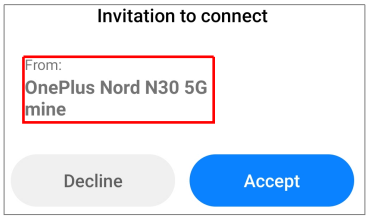

Fig. 6. Downgrade attack example running on *Xiaomi 11 Lite*, the prompt is not sent by a legal device *OnePlus Nord N30 5G*, it's sent by attacker *Dell XPS13 plus*.

APPENDIX A
TAMARIN MODEL AND PROPERTIES DESIGN

A. How to write Tamarin model and Properties

Tamarin uses multiset rewriting rules to write the protocol model and write properties in guarded first logic. To formalize

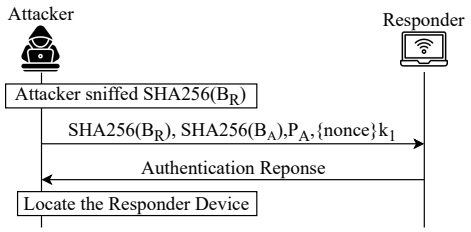

Fig. 7. The message flow of **EC1: bootstrapping privacy leakage issue**. The sniffed hash value of the responder's public bootstrapping key $SHA(B_R)$ is used to locate the device.

the security protocol with Tamarin, Tamarin represents protocol as a collection of multiset rewriting rules., writing as $[l] - [a] \rightarrow [r]$, where l, a and r are multisets of facts. For example, the following rule specifies the transmission of the hash of a received message:

$$[In(m)] - [SendHash(a, m)] -> [State1(A, m), Out(h(m))]$$

This rule states that if a term m is received, the state is changed to $Satet1(A, m)$ and term $h(m)$ is sent to the channel. The transition has action $SendHash(A, m)$, meaning that A sent the hash of m. Security properties are defined as first-order logic formulas over the traces in the protocol. These properties may encompass confidentiality, authentication, integrity and privacy, serving as metrics for verifying the security of the protocol. For example, consider the following first-order logic formula:

```
not(
  Ex #i #j privKey. Dev1GenerateKey(privKey)  @ #i
  & K(privEnrolKey) @ #j )
```

The formula specifies that, for all the protocol traces, once the device generates a private key. Then, there should not exist a time j at which the attacker knows the private key. This property helps in verifying the secrecy of the private key.

B. Our Designed Properties

We introduce how we design and implement the security properties in Tamarin here. Our designed properties include several security aspects, including privacy, authentication and secrecy.

Privacy property relates to device traceability [33], where the tracer acts as the attacker. We consider two traceability types: *passive tracking* (monitoring transmitted packets without interaction) and *active tracking* (directly sending requests to the target device). The following example shows a passive privacy property in DPP: The attacker cannot locate the *Responder* by passive tracking.

Property 1 (Passive Privacy)**.** *The responder in* DPP *protocol P satisfies passive privacy if for every trace* $\alpha \in traces(P)$*:*

```
not(Ex #i #j iniID respAddress bootPubRespKey.
Bootstrap(iniID, respAddress, bootPubRespKey) @ #i
& K(respAddress) @ #j )
```

Here, `respAddress` is the responder's MAC address (typically global), and `Bootstrap` indicates completed bootstrapping. Passive non-traceability means the attacker should not know `respAddress` in any trace. The next example shows active privacy property:

Property 2 (Active Privacy)**.** *The responder in* DPP *protocol P satisfies active privacy if for every trace* $\alpha \in traces(P)$*:*

```
not(
Ex #i #j hashBootPubRespKey protoPubTracKey
    respAddress.
TracerSendAuthReq(hashBootPubRespKey,
    protoPubTracKey) @ #i
& RespSendResponse(respAddress,
    hashBootPubRespKey, protoPubTracKey) @ #j &
    j>i)
```

Our model includes a tracer entity for active scenarios. `TracerSendAuthReq` indicates the tracer sent an authentication request containing `hashBootPubRespKey` (SHA256 hash of responder's public bootstrapping key) and `protoPubTracKey` (tracer's public protocol key). The responder must not reply to preserve active privacy.

Our next property corresponds to *injective agreement* [59]. We give an authentication example of WI-FI DIRECT.

Property 3 (Authentication Property)**.** *The initiator in pairing protocol P satisfies authentication to responder if for every trace* $\alpha \in traces(P)$*:*

```
All #i deviceID1 N1 N2 dev1Pwd1 dev1Pwd2.
Dev1FiniProvisioning(deviceID1, N1, N2, dev1Pwd1,
    dev1Pwd2) @ #i
==>
(Ex #j deviceID2. Dev2RunningProvision(deviceID2,
    N1,N2,dev1Pwd1,dev1Pwd2)
@ #j & #j<#i
& not ( Ex deviceID1_new #i2.
  Dev1FiniProvisioning(deviceID1_new, N1, N2,
      dev1Pwd1, dev1Pwd2) @i2
  & not(i2=i) )
)
```

`Dev1FiniProvisioning` indicates *device1* (`deviceID1`) completed provisioning using identifiers `N1`, `N2` and passwords `dev1Pwd1`, `dev1Pwd2`. `Dev2RunningProvision` states *device2* (`deviceID2`) is engaged in provisioning with the same passwords.

The next property is secrecy; we use private key secrecy in WI-FI DIRECT as an example.

Property 4 (Secrecy)**.** *The* WI-FI DIRECT *protocol satisfies secrecy if for every trace* $\alpha \in traces(P)$*:*

```
not( Ex #i #j deviceID1 N1 privEnrolKey.
Dev1GenerateKey(deviceID1, N1, privEnrolKey)  @ #i
& K(privEnrolKey) @ #j )
```

`Dev1GenerateKey` means the enrollee generates private key `privEnrolKey`, and `K(privEnrolKey)` means the attacker knows this key. This property ensures the private key remains secret from the attacker.

We also consider *executability*, which assesses whether protocol execution reaches successful device connection.

Property 5 (Executability)**.** *A protocol P is executable if there exists trace* $\alpha \in traces(P)$ *such that:*

```
Ex #i #j #k initiatorID responderID
    password hashK.
BothSetCred(initiatorID, responderID,
    password) @ #i &
```

```
5 RespFiniSetup(responderID, initiatorID, password,
      hashK) @ #j
6 & IniFiniSetup(initiatorID, responderID, password,
      hashK) @ #k
```

`BothSetCred` means both devices share the same pre-shared secret `password`, while `RespFiniSetup` and `IniFiniSetup` indicate both devices completed setup with generated credential `hashK`.

APPENDIX B
OTHER ATTACKS AND ISSUES

A. Other attacks and issues of WI-FI DIRECT

D3: P2P Interface Address Privacy Leakage. Another value that can identify the device in the WI-FI DIRECT protocol apart from UUIDs is *P2P Interface Address*. We then design and verify the passive privacy property of *P2P Interface Address*, which is falsified. Additionally, there is no protection for the *P2P Interface Address* from privacy leakage in the protocol. Because the address does not change, if the attacker can sniff the messages sent by the victim's device, then the attacker can obtain private information about the victim's device, which will cause a privacy leakage issue.

D4: Modify Authentication Value & D5: Modify Message Code. During provisioning, messages M1-M8 authenticate the new device during provisioning. Messages M2-M8 contain an *authentication value* for verification. Failed verification causes connection drop, allowing attackers to modify this value for DoS attacks (**D4**). Messages M1-M8 use *message code* to identify message types in lock-step fashion. However, the integrity property of M1's message code is falsified: "*If device2 receives M1 with message code x, device1 should have sent M1 with message code x*". Attackers can modify M1's *message code*, causing legal devices to drop connections upon receiving modified M1, resulting in DoS attack **D5**.

All-Zero Private Key. The Diffie-Hellman private key is randomly generated during provisioning. An attacker could potentially manipulate the public key corresponding to an all-zero private key, resulting in g^0 - a critical security flaw. However, this risk is not exploitable due to integrity checks on public keys. This issue remains noteworthy for researchers investigating cryptographic vulnerabilities.

An Implementation Issue. Android built-in applications and popular third-party apps (with millions of downloads) typically use the PushButton configuration method for WI-FI DIRECT. However, PushButton is less secure and improper implementations can cause serious authentication bypass issues.

B. Other issues of EasyConnect

EC2: MAC Address Information Exposure. Inspired by prior work on MAC address privacy [39], we examine the passive exposure of MAC addresses in the DPP protocol.

Detection with WCDCAnalyzer. We model a *passive privacy* property for MAC addresses: *"If a MAC address is used during protocol execution, it should not be inferable by the attacker."* This property is violated in our verification, as the MAC address is transmitted in cleartext and not obfuscated across sessions. Moreover, the DPP specification does not mandate MAC randomization or periodic updates, leaving implementations free to use global, static MAC addresses. Our analysis highlights a potential privacy risk—MAC addresses may enable long-term device identification. While this does not directly violate a security guarantee, it suggests that deployments using static MACs should consider additional protections, such as MAC randomization or ephemeral identifiers.

C. Issues of EasyMesh

EM1: Bootstrapping Key Reuse and Privacy Exposure. WI-FI EASYMESH uses DPP for authentication, with bootstrapping public keys often reused across sessions (e.g., printed on devices or QR codes). We model an *active privacy* property: *"If the adversary sends the hash of the bootstrapping key to a responder, the device should not reply to avoid confirming its identity."* WCDCAnalyzer finds this property violated with static bootstrapping keys, enabling attackers to confirm device presence through crafted authentication requests and introducing linkability risks.

EM2: MAC Address Static Exposure. We check a *passive privacy* property ensuring MAC addresses should not be inferable through passive observation. This property is violated as WI-FI EASYMESH lacks MAC address randomization. The same MAC address broadcasts across multiple sessions, allowing attackers to correlate traffic to specific devices. While not a security violation, this poses privacy risks in deployments requiring MAC address protection.

EM3: Privacy Exposure via Presence Announcements. Unprovisioned EasyMesh devices periodically broadcast DPP *Presence Announcement* messages containing their bootstrapping public key hash to trigger nearby configurator bootstrapping. We model a *passive privacy* property that the bootstrapping key hash should not enable long-term device tracking. This property is falsified when bootstrapping keys are reused, creating repeated hash values linkable across time and space by passive observers. Static bootstrapping keys thus create privacy exposure through standard protocol behavior.

Verified Security Properties. In parallel with identifying privacy risks, we also use WCDCAnalyzer to formally verify that EasyMesh satisfies essential security guarantees. These results confirm that EasyMesh is secure in terms of core authentication and cryptographic correctness, and that formal analysis can also expose auxiliary privacy risks that are not directly addressed in the specification.

APPENDIX C
TESTBED SETUP

A. WI-FI DIRECT *Testbed Setup*

WI-FI DIRECT has two roles, *Registrar* and *Enrollee*. We develop the testbed with Scapy [60] as *Enrollee* to evaluate if the attacker *Enrollee* can bypass the authentication and connect to the legal *Registrar*. Fig. 8 shows our testbed setup logic. Apart from the *Enrollee* role, the attacker can also inject packets arbitrarily as a MitM [34] attacker.

ID	Device Name	Chipset Vendor	Wi-Fi Version	OS Version
P1	OnePlus Nord N30	Qualcomm	Wi-Fi 5	Android 13
P2	Xiaomi 11 Lite	Qualcomm	Wi-Fi 6	Android 11
P3	OnePlus 8T+	Qualcomm	Wi-Fi 6	Android 13
P4	OnePlus 9 Pro	Qualcomm	Wi-Fi 6	Android 13
P5	OnePlus 7T	Qualcomm	Wi-Fi 5	Android 10
P6	REVVL 4+	Qualcomm	Wi-Fi 5	Android 11
P7	Oneplus Nord	Qualcomm	Wi-Fi 5	Android 10
P8	Motorola Edge30 Pro	Qualcomm	Wi-Fi 6	Android 13
P9	Xiaomi 12T	MediaTek	Wi-Fi 6	Android 12
P10	Honor 8X	Huawei	Wi-Fi 5	Android 8
N1	Alfa AWUS036ACM	MediaTek	Wi-Fi 5	Ubuntu 22
N2	TL-WN722N	Realtek	Wi-Fi 4	Ubuntu 22
N3	BrosTrend AC3L	Realtek	Wi-Fi 5	Ubuntu 22
N4	Netgear A8000	MediaTek	Wi-Fi 6E	Ubuntu 22
N5	EDUP EP-AX1672	MediaTek	Wi-Fi 6	Ubuntu 22
N6	Alfa AWUS036ACU	Realtek	Wi-Fi 5	Ubuntu 22
N7	Realtek rtl8812bu	Realtek	Wi-Fi 5	Ubuntu 22
N8	Intel AC 8265	Intel	Wi-Fi 5	Ubuntu 22
B1	ESP32	Espressif	Wi-Fi 4	FreeRTOS

TABLE IV
THE LIST OF THE TESTING DEVICES. *P* MEANS PHONE, *N* MEANS NETWORK CARD, *B* MEANS DEVELOPMENT BOARD.

Device	D1	D2	D3	D4	D5
OnePlus Nord N30	●	●	●	●	●
Xiaomi 11 Lite	●	●	●	●	●
OnePlus 8T+	●	●	●	●	●
OnePlus 9 Pro	●	●	●	●	●
OnePlus 7T	●	●	●	●	●
REVVL 4+	●	●	●	●	●
OnePlus Nord	●	●	●	●	●
Motorola Edge 30	●	●	●	●	●
LG Velvet	●	●	●	●	●
Xiaomi 12T	●	●	○	●	●
Alfa AWUS036ACM	○	●	●	●	●

TABLE V
THE TESTBED VALIDATION RESULTS OF THE WI-FI DIRECT ISSUES, ●MEANS THE ISSUE IS VALIDATED, ○MEANS THE ISSUE IS NOT VALIDATED.

In the attack **D1**, the attacker tries to impersonate a legal *Enrollee*, and the victim device acts as a *Registrar*. As for the attacker *Enrollee*, we employ the *Alfa AWUS036ACM* network card on an XPS 13 Plus laptop with an Intel i7-1260P CPU, 16GB DDR3 RAM, and Ubuntu 22.04 system. We use this attacker *Enrollee* to test the legal device. To access the legal *Registrars*, we use *tcpdump* [61], and Android Debug Bridge (ADB) [62].

As for the privacy leakage issues **D2** and **D3**, we use the methods discussed in previous works [33], [43]. For testing whether a device has a privacy leakage problem, we first decide on the unified identity that can help locate the device. In our case, the identities can be *UUID* or *P2P Interface Address*. Then, the testing device is connected to two devices at intervals of more than 12 hours. If the identities of the two times' connection are the same, then the privacy leakage problem is validated. As for the DoS issues **D4** and **D5**, we also implement the attacker on the *Alfa AWUS036ACM* network card on an XPS 13 Plus laptop.

B. WI-FI EASYCONNECT *Testbed Setup*

WI-FI EASYCONNECT has two roles, *Initiator* and *Responder*. We test the privacy leakage issues **EC1** and **EC2**. In the setting, the testing devices are *Responder* because only *Responder* will show the QR code. Our attacks all focus on the devices that can be a *Responder* in the implementation. This means that not all devices in our device list can be tested.

Device	EC1	EC2
ESP32	○	●
Alfa AWUS036ACM	●	●
Intel AC 8265	●	●
TL-WN722N	●	●
Netgear A8000	●	●
BrosTrend AC3L	●	●
EDUP EP-AX1672	●	●
ALFA AWUS036ACU	●	●
Realtek rtl8812bu	●	●

TABLE VI
THE TESTBED VALIDATION RESULTS OF THE WI-FI EASYCONNECT ISSUES, ●MEANS THE ISSUE IS VALIDATED, ○MEANS THE ISSUE IS NOT VALIDATED

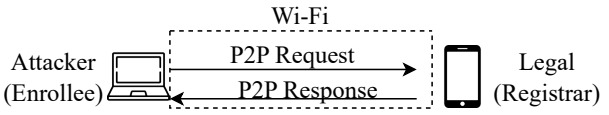

Fig. 8. The WI-FI DIRECT testbed setup logic. The attacker plays the role of the enrollee, and the victim device plays the role of the registrar.

C. WI-FI EASYMESH *Testbed Setup*

The WI-FI EASYMESH protocol is still in the development phase [7], and the vendors are still implementing the EasyMesh protocol. Furthermore, the two Wi-Fi software supports *wpa-supplicant* [63] and *iwd* [64] do not implement the EasyMesh protocol. So we do not validate the attacks in EasyMesh in commercial devices, which includes **EM1**, **EM2** and **EM3**.

TABLE VII
EASYMESH— MEMORY (MB) AND TIME (S) OF THE WHOLE PROTOCOL (P) AND ITS SUB-PROTOCOLS (P_1,P_2).

Property	P		P_1		P_2		P_1+P_2	
	Mem	Time	Mem	Time	Mem	Time	Mem	Time
executability	1007	609	10	0.4	891	517	901	518
PassivePrivAddr	970	546	10	0.4	894	534	904	535
cAuthIntegrity	980	557	–	–	919	533	919	533
keSecrecy	1192	616	–	–	916	584	916	584

TABLE VIII
EASYCONNECT — MEMORY (MB) AND TIME (S) OF THE WHOLE PROTOCOL (P) AND ITS SUB-PROTOCOLS (P_1,P_2).

Property	P		P_1		P_2		P_1+P_2	
	Mem	Time	Mem	Time	Mem	Time	Mem	Time
executability	429	54	5	0.6	403	40	408	41
macAddrPrivacy	299	39	5	0.6	298	38	303	39
keSecrecy	438	44	–	–	392	42	392	42
RespAuthIni	399	41	–	–	366	40	366	40