Paradigms in Security Protocol Verification: A Multi-Tool Analysis

DOI: 10.54654/isj.v3i23.1059

Le Vinh Thinh*

***Abstract*— In the realm of internet security, ensuring the robustness and integrity of communication protocols is paramount. This paper offers a comprehensive analysis of four leading tools for security protocol verification: Scyther, ProVerif, CryptoVerif, and Tamarin. Each tool is evaluated for its unique strengths and applications, particularly in the contexts of cloud computing and IoT. The study begins with Scyther, highlighting its proficiency in automated falsification and multi-protocol verification. Next, ProVerif is examined for its capabilities in symbolic reasoning and its efficiency in handling complex security protocols. The paper then explores CryptoVerif's computational approach to protocol verification, focusing on how it models and verifies protocols under a variety of cryptographic assumptions. Finally, Tamarin's advanced features in symbolic analysis and its ability to manage intricate security properties are discussed, emphasizing its depth in formal protocol verification. This comparative analysis not only underscores the distinct contributions of each tool but also provides a broader perspective on their effectiveness in addressing both current and emerging security challenges. By dissecting the methodologies and limitations of these tools, the paper aims to offer valuable insights into the evolving landscape of security protocol verification and potential future directions in this critical area of cybersecurity research.**

***Tóm tắt*— Trong lĩnh vực bảo mật Internet, đảm bảo tính vững chắc và toàn vẹn của các giao thức truyền thông là điều tối quan trọng. Bài báo này cung cấp một phân tích toàn diện về bốn công cụ hàng đầu để xác minh giao thức bảo mật: Scyther, ProVerif, CryptoVerif và Tamarin. Mỗi công cụ được đánh giá về những điểm mạnh và ứng dụng độc đáo của nó, đặc biệt trong bối cảnh điện toán đám mây và IoT. Nghiên cứu bắt đầu với Scyther, nhấn mạnh khả năng làm giả tự động và xác minh đa giao thức của nó. Tiếp theo, ProVerif được xem xét về khả năng lập luận biểu tượng và hiệu quả trong việc xử lý các giao thức bảo mật phức tạp. Bài báo sau đó khám phá cách tiếp cận tính toán của CryptoVerif đối với việc xác minh giao thức, tập trung vào cách nó mô hình hóa và xác minh các giao thức dưới nhiều giả định mật mã khác nhau. Cuối cùng, các tính năng nâng cao của Tamarin trong phân tích biểu tượng và khả năng quản lý các thuộc tính bảo mật phức tạp được thảo luận, nhấn mạnh độ sâu của nó trong xác minh giao thức chính thức. Phân tích so sánh này không chỉ nhấn mạnh những đóng góp riêng biệt của mỗi công cụ mà còn cung cấp một góc nhìn rộng hơn về hiệu quả của chúng trong việc giải quyết cả những thách thức bảo mật hiện tại và mới nổi. Bằng cách phân tích các phương pháp và hạn chế của các công cụ này, bài báo nhằm cung cấp những hiểu biết có giá trị về bối cảnh đang phát triển của việc xác minh giao thức bảo mật và các hướng đi tiềm năng trong tương lai trong lĩnh vực nghiên cứu an ninh mạng quan trọng này.**

***Keywords*—Verification protocol; falsification protocol, analysis protocol, security, cryptography.**

***Từ khóa*— Giao thức xác minh, giao thức làm giả, giao thức phân tích, bảo mật, mật mã học.**

I. INTRODUCTION

In the realm of digital security, security protocols, particularly key agreement protocols, are fundamental to ensuring secure communications over potentially insecure networks [1, 2]. These protocols enable the establishment of shared secret keys, which are crucial for safeguarding subsequent exchanges of data. Given the critical role these protocols play, verifying their robustness is essential to prevent vulnerabilities such as man-in-the-middle, replay attacks, and side-channel

This manuscript was received on October 4, 2024. It was reviewed on November 11, 2024, revised on December 13, 2024 and accepted on December 18, 2024.

* Corresponding author

exploits. Recent studies have explored the analysis and verification of hierarchical identity-based authenticated key agreement (HIB-AKA) protocols, employing advanced tools like Scyther and Tamarin for a thorough evaluation [3]. Research has also investigated identity-based authenticated key agreement protocols within the Diffie-Hellman family, enabled by Weil or Tate pairings, to address intricate issues related to cryptographic security [4]. Moreover, the rise of quantum computing has led to a focus on lightweight lattice-based secure systems, especially those that offer efficient security in the post-quantum era [5]. Scyther is one of the leading tools in security protocol verification, recognized for its intuitive interface and capacity to automate the falsification of security protocols. Its ability to handle an unbounded number of protocol sessions makes it highly effective for analyzing key agreement protocols across various contexts, including cloud computing and IoT environments [6 - 8]. Similarly, Tamarin is widely regarded for its robust symbolic analysis capabilities, enabling it to handle complex security properties and a wide range of adversary models [9 - 11]. This tool represents a significant advancement in security protocol verification by formalizing and verifying protocols in large-scale, real-world scenarios [12]. Verification tools can be widely applied in various domains. For example, Pham et al. [13] demonstrated a robust approach for web attack detection using deep learning and natural language processing techniques. While their work focuses on attack detection, integrating verification tools like Scyther or Tamarin could further formalize the system's security properties, ensuring robustness against evolving threats .A notable development has been the integration of finite-state machine (FSM) model learning with Tamarin, which facilitates the detection of logical errors, thereby improving protocol verification processes. ProVerif also plays a pivotal role in the formal verification of security protocols, leveraging symbolic reasoning to verify properties such as secrecy and authentication. Using Horn clauses and a resolution algorithm, ProVerif is adept at analyzing various cryptographic protocols, including MQV-based key exchange protocols and identity-based schemes [14 - 16]. Moreover, ProVerif has been extended to verify protocols with stateful and algebraic properties, expanding its application to more complex scenarios [17, 18]. Operating in the computational model, CryptoVerif provides stronger security guarantees than symbolic models by verifying security properties such as secrecy, indistinguishability, and authentication under concrete cryptographic assumptions [19]. CryptoVerif is particularly suited for verifying modern protocols like TLS 1.3 and Signal, offering computational proofs that these protocols are robust against sophisticated attacks [20, 21]. It has also been employed to verify dynamic key compromise scenarios, demonstrating its utility in evaluating key exchange protocols under evolving threats, such as in TLS 1.3 and the WireGuard VPN protocol [22, 23]. Additionally, the introduction of CV2EC, a translation tool bridging CryptoVerif and EasyCrypt, has further expanded CryptoVerif’s capabilities by allowing for the verification of cryptographic primitives alongside protocol verification, as demonstrated in various case studies [24]. Beyond protocol verification, Tamarin has proven invaluable in large-scale applications such as multi-party cryptographic systems and dynamic environments like 5G networks. Its ability to formalize advanced adversary models and verify stateful protocols has made it an essential tool for evaluating complex security properties under dynamic threats. Together, these tools, Scyther, ProVerif, CryptoVerif, and Tamarin, provide a comprehensive suite for security protocol verification, each contributing distinct methodologies that address the evolving challenges in securing communication protocols across a wide range of applications.

This paper is organized into five sections, each providing an in-depth exploration of key aspects related to security protocol verification tools. Section 2 introduces the primary tools under consideration, including Scyther, ProVerif, CryptoVerif, and Tamarin, and highlights their relevance in today's security landscape. Section III focuses on a comparative analysis of these tools, examining their features, strengths, and limitations across various parameters. Section IV presents a discussion of the tools and case study, analyzing how they adapt to technological shifts, evolve with cryptographic advancements, enhance user experience, and address emerging security challenges. Finally, Section V concludes the paper by summarizing the main findings and outlining the broader implications for future research and tool development in security protocol verification.

II. Theoretical Comparison of Security Protocol Verification Tools

In this section, the paper turns the attention to the foundational aspects of our study. Previous work [25] provided only a basic and a simple comparison of two methods, without addressing their relevance and importance in the current context. This paper overcome these limitations by offering a more comprehensive and timely analysis .The session is organized into four subsections, each dedicated to a significant tool in the field of security protocol verification: Scyther, ProVerif, CryptoVerif, and Tamarin. The first subsection focuses on Scyther, a key tool for verifying and analyzing security protocols. We will explore its methodology, distinctive features, and its wide-ranging applications in both industrial and academic contexts. Additionally, we will assess its strengths and limitations to provide a comprehensive understanding of its impact in security protocol analysis. The following subsection will delve into ProVerif, examining its unique symbolic reasoning approach, capabilities, and diverse use cases in cryptographic protocol verification. This analysis will highlight how ProVerif advances the field by efficiently verifying key security properties such as secrecy and authentication. Next, we will explore CryptoVerif, focusing on its role in computational verification and how it models cryptographic assumptions to provide stronger security guarantees. Finally, we will present a detailed comparison of these tools, outlining their respective advantages and drawbacks, offering a holistic view of their contributions to security protocol verification.

A. The Scyther Tool

Detailed Analysis of Scyther's Methodology and Features

Scyther represents a major advancement in the analysis of security protocols. Designed with a focus on usability and efficiency, this tool uses a distinctive methodology that distinguishes it from other verification systems. At its core is a pattern refinement algorithm, which enables a concise representation of potentially infinite sets of execution traces. This algorithm plays a key role in examining classes of attacks, potential protocol behaviors, and validating the correctness of security protocols across an unlimited number of sessions.

Scyther operates on the well-established Dolev-Yao intruder model, a commonly used approach in the analysis of security protocols. This model assumes that an attacker has full control over the network but cannot break cryptographic primitives. Leveraging this model, Scyther simulates possible attacks and uncovers vulnerabilities in protocols under review. A standout feature of the tool is its capacity to handle unbounded verification with guaranteed termination, allowing it to provide definitive conclusions regarding the security of the analyzed protocols.

Case Studies and Applications in Various Contexts

Scyther has demonstrated its versatility and effectiveness across a wide range of applications, both in academic and industrial settings. It has been employed to analyze and verify the security of various protocols. In the industrial domain, Scyther has been pivotal in the verification of complex security protocols, such as IKE (versions 1 and 2) and ISO/IEC 9798, providing critical insights into their security properties. Additionally, its application in academic environments has been valuable for teaching purposes, allowing students to explore the nuances of protocol analysis. Scyther's user-friendly interface, coupled with its graphical depiction of protocol interactions, makes it a useful educational tool, helping students and researchers grasp the intricacies of security protocols more effectively.

Strengths and Limitations

One of Scyther's key strengths lies in its ease of use, which lowers the barrier for those new to the field of security protocol analysis. Its automated analysis capabilities allow for quick and efficient verification, saving both time and resources. Furthermore, Scyther's ability to perform unbounded verification and deliver conclusive results adds to its reliability and effectiveness, making it a valuable tool for security professionals and researchers alike.

However, while it excels in analyzing certain security properties, such as authentication and secrecy, it may struggle with more complex properties like non-repudiation or handling certain denial-of-service (DoS) attacks. Moreover, its reliance on the Dolev-Yao model although advantageous in many scenarios can limit its applicability in real-world cases that involve physical attacks or side-channel attacks, as these aspects are not accounted for in the model.

Overall, Scyther is a powerful, user-friendly tool that significantly contributes to the field of security protocol analysis. Its methodology, applications, strengths, and limitations make it an essential resource for researchers, educators, and security professionals (Figure 1). The tool provides an automated, comprehensive approach to protocol verification, enhancing the understanding and fortification of security protocols across various digital communication environments. Like any tool, however, its effectiveness is optimized when users are aware of its scope and limitations, ensuring that it is applied appropriately within the context of broader security analysis strategies.

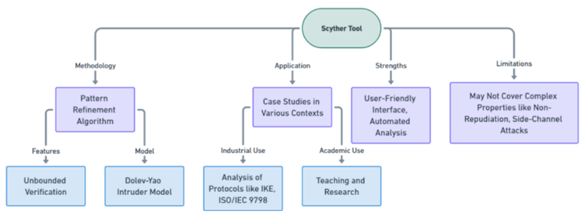

Scyther Tool
Methodology
Application
Strengths
Limitations
Pattern Refinement Algorithm
Case Studies in Various Contexts
User-Friendly Interface, Automated Analysis
May Not Cover Complex Properties like Non-Repudiation, Side-Channel Attacks
Features
Model
Industrial Use
Academic Use
Unbounded Verification
Dolev-Yao Intruder Model
Analysis of Protocols like IKE, ISO/IEC 9798
Teaching and Research

Figure 1. Overall key aspects of Scyther

Language

The Scyther tool uses a specialized language for specifying security protocols, known as the Scyther Specification Language (SSL). This language is designed to be simple yet expressive enough to define various aspects of security protocols, including roles, messages, and cryptographic operations.

Key Features of SSL:

1. Role-Based Specification: Protocols are defined in terms of roles (like initiator, responder) and their interactions.

2. Message Format: Messages are defined using a straightforward syntax that describes the composition of messages sent and received.

3. Cryptographic Constructs: Common cryptographic operations like encryption, decryption, and hashing are supported.

4. Variables and Constants: The language allows the use of variables (for dynamic values like nonces) and constants (for static values like keys).

To be clearer, considering an example is of the Needham-Schroeder Public Key Protocol using the Scyther SSL. This protocol is designed for establishing a secure communication channel between two parties, identified as I (Initiator) and R (Responder), by exchanging nonces (random numbers) to verify each other's identity and establish shared secrets. Let's break down the example:

Protocol Definition:

Protocol ns3(I, R) {...}: Defines a new protocol named ns3 with two roles, I (Initiator) and R (Responder).

Role I (Initiator):

1. fresh ni: Nonce; Initiator generates a fresh nonce ni.

2. var nr: Nonce; Declares a variable nr to store the nonce received from the Responder.

3. send_1(I, R, {ni, I}pk(R)); Initiator sends a message to Responder, containing the nonce ni

and its identity I, encrypted with Responder's public key pk(R).

4. recv_2(R, I, {ni, nr}pk(I)); Initiator receives a message from Responder, containing the original nonce ni and Responder's nonce nr, encrypted with Initiator's public key pk(I).

5. send_3(I, R, {nr}pk(R)); Initiator sends back the nonce nr received from Responder, encrypted with Responder's public key pk(R).

The Initiator then makes several claims about the protocol's security properties, such as secrecy of nonces, aliveness, agreement, commitment, and synchronization.

Role R (Responder):

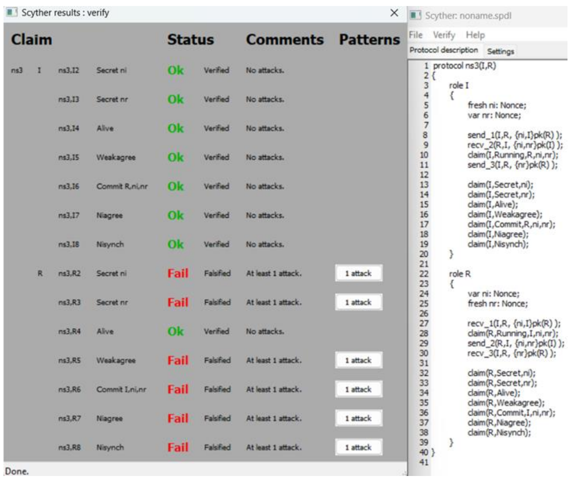
1. var ni: Nonce; Responder declares a variable ni to store the nonce received from the Initiator.
2. fresh nr: Nonce; Responder generates a fresh nonce nr.
3. recv_1(I, R, {ni, I}pk(R)); Responder receives the first message from Initiator, decrypts it to get Initiator's nonce ni and identity I.
4. send_2(R, I, {ni, nr}pk(I)); Responder sends a message back to Initiator, containing the received nonce ni and its own nonce nr, encrypted with Initiator's public key pk(I).
5. recv_3(I, R, {nr}pk(R)); Responder receives the final message from Initiator, containing the nonce nr.

The Responder also makes similar claims about the protocol's security properties.

Security Claims:

- claim(...); These statements are used to assert various security properties like secrecy (nonces are not known to others), aliveness (other party is active), agreement (both parties agree on the nonces), commitment (a party is committed to a session), and synchronization (nonces are synchronized between parties).

This protocol aims to securely establish a shared secret between the Initiator and Responder, ensuring that both parties are who they claim to be. The claims are used to verify the security properties of the protocol using the Scyther tool. The result is presented in Figure 2 that shows the number of attacks can affect the protocol.



Claim				Status		Comments	Patterns
ns3	I	ns3,I2	Secret ni	Ok	Verified	No attacks.	
		ns3,I3	Secret nr	Ok	Verified	No attacks.	
		ns3,I4	Alive	Ok	Verified	No attacks.	
		ns3,I5	Weakagree	Ok	Verified	No attacks.	
		ns3,I6	Commit R,ni,nr	Ok	Verified	No attacks.	
		ns3,I7	Niagree	Ok	Verified	No attacks.	
		ns3,I8	Nisynch	Ok	Verified	No attacks.	
	R	ns3,R2	Secret ni	Fail	Falsified	At least 1 attack.	1 attack
		ns3,R3	Secret nr	Fail	Falsified	At least 1 attack.	1 attack
		ns3,R4	Alive	Ok	Verified	No attacks.	
		ns3,R5	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		ns3,R6	Commit I,ni,nr	Fail	Falsified	At least 1 attack.	1 attack
		ns3,R7	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		ns3,R8	Nisynch	Fail	Falsified	At least 1 attack.	1 attack

```
protocol ns3(I,R)
{
	role I
	{
		fresh ni: Nonce;
		var nr: Nonce;

		send_1(I,R, {ni,I}pk(R) );
		recv_2(R,I, {ni,nr}pk(I) );
		claim(I,Running,R,ni,nr);
		send_3(I,R, {nr}pk(R) );

		claim(I,Secret,ni);
		claim(I,Secret,nr);
		claim(I,Alive);
		claim(I,Weakagree);
		claim(I,Commit,R,ni,nr);
		claim(I,Niagree);
		claim(I,Nisynch);
	}

	role R
	{
		var ni: Nonce;
		fresh nr: Nonce;

		recv_1(I,R, {ni,I}pk(R) );
		claim(R,Running,I,ni,nr);
		send_2(R,I, {ni,nr}pk(I) );
		recv_3(I,R, {nr}pk(R) );

		claim(R,Secret,ni);
		claim(R,Secret,nr);
		claim(R,Alive);
		claim(R,Weakagree);
		claim(R,Commit,I,ni,nr);
		claim(R,Niagree);
		claim(R,Nisynch);
	}
}
```

Figure 2. Scyther language and Input/Output

Furthermore, Scyther can trace attacks by presenting graphs, as demonstrated in Figure 3.

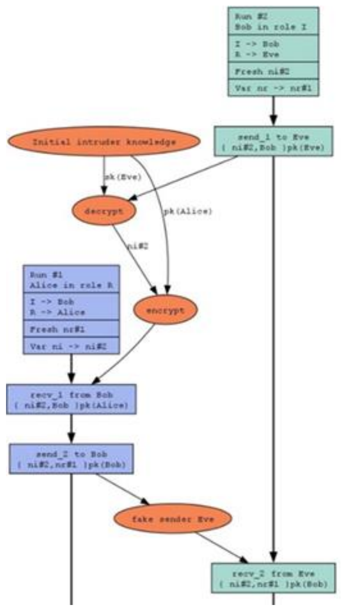


Figure 3. Attack claims in Scyther

B. *Proverif*

Detailed Analysis of ProVerif's Methodology and Features

ProVerif is a powerful tool widely used for the symbolic verification of security protocols. Unlike Scyther, which primarily handles unbounded verification of protocol sessions, ProVerif excels at verifying security properties such as secrecy, authentication, and integrity using symbolic reasoning. It is based on an abstraction known as Horn clauses, and it uses a resolution algorithm to automatically analyze cryptographic protocols. ProVerif operates in the Dolev-Yao model, where the adversary has complete control over the network but cannot break cryptographic primitives. One of ProVerif’s key features is its ability to handle infinite state spaces, meaning it can analyze protocols with an unbounded number of sessions or messages without explicitly enumerating all possible states. This capability makes it highly efficient for analyzing security properties in complex protocols. Additionally, ProVerif can verify a range of cryptographic operations, such as encryption, digital signatures, and Diffie-Hellman key exchanges, making it versatile for various protocol types.

ProVerif uses a query-based approach, where users pose specific security questions (queries) about the protocol, and ProVerif determines whether those properties hold or if there are attacks that violate them. For example, a typical query might ask if a shared secret remains confidential during communication. ProVerif’s internal algorithms attempt to prove or refute such queries, providing a detailed analysis of potential vulnerabilities.

Case Studies and Applications in Various Contexts

ProVerif has been applied in numerous real-world contexts, underscoring its practical utility. It has been particularly effective in the verification of cryptographic protocols such as MQV-based key exchange protocols, where it can detect subtle attacks like the Unknown Key Share (UKS) attack. Moreover, ProVerif has been used to verify stateful protocols, where the state of a system changes over time, such as in TLS (Transport Layer Security) and Single Sign-On (SSO) authentication systems. In industry, ProVerif has been instrumental in analyzing the security of software-defined networking (SDN) protocols, where its ability to handle both symbolic and algebraic properties is critical. Academic research also widely employs ProVerif to model complex systems, from cloud authentication mechanisms to IoT communication protocols. In these contexts, ProVerif provides crucial insights into protocol weaknesses, helping to ensure that they meet the desired security standards.

Strengths and Limitations

One of ProVerif’s major strengths is its automation and ability to handle infinite state spaces, making it extremely efficient for protocols that involve an unbounded number of participants or messages (Figure 04). Its query-based approach also allows for flexible and specific analysis, meaning users can tailor their security questions based on the requirements of the protocol they are investigating. Additionally, ProVerif's support for various cryptographic primitives, including symmetric encryption, public-key encryption, and hash functions, makes it applicable to a wide variety of modern protocols. However, ProVerif is not without limitations. It operates under the symbolic Dolev-Yao model, which abstracts cryptographic primitives, potentially oversimplifying certain real-world behaviors of cryptographic systems. As a result, it might miss attacks that exploit the actual implementation of cryptographic algorithms. Furthermore, while ProVerif excels in analyzing secrecy and authentication, it may struggle with other properties like non-repudiation or attacks that rely on side-channel information, as these are not fully modeled in the symbolic approach. Additionally, ProVerif's learning curve can be steep for beginners due to the need for users to understand both formal methods and protocol-specific features.

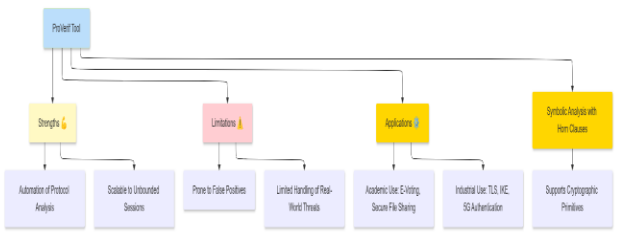
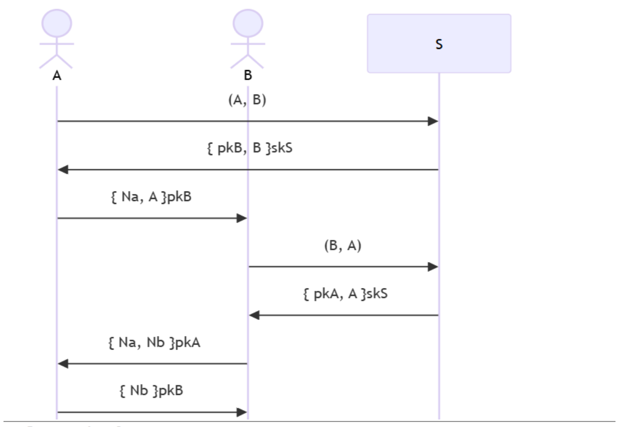

ProVerif Tool
Strengths 💪
Limitations ⚠️
Applications ⚙️
Symbolic Analysis with Horn Clauses
Automation of Protocol Analysis
Scalable to Unbounded Sessions
Prone to False Positives
Limited Handling of Real-World Threats
Academic Use: E-Voting, Secure File Sharing
Industrial Use: TLS, IKE, 5G Authentication
Supports Cryptographic Primitives

Figure 4. Overall key aspects of ProVerif

Language

The ProVerif language, used to model security protocols, is designed to be flexible and expressive. Protocols are represented using a process calculus, where communication is modeled using channels and cryptographic operations like encryption and decryption are built-in functions. ProVerif provides users with the ability to specify protocol actions, such as sending or receiving messages, generating fresh keys, or performing cryptographic operations, all within the formal framework of the process calculus. Figure 05 demonstrates the Needham-Schroeder Public Key Protocol implement in Proverif and its output.



```
-- Process 1-- Query
   not attacker_bitstring(secretANa[])
   not attacker_bitstring(secretANb[])
   not attacker_bitstring(secretBNa[])
   not attacker_bitstring(secretBNb[])
 in process 1
Translating the process into Horn clauses...
Completing...
200 rules inserted. Base: 188 rules (87 with conclusion selected). Queue: 28 rules.
ok, secrecy assumption verified: fact unreachable attacker_skey(skA[])
ok, secrecy assumption verified: fact unreachable attacker_skey(skB[])
ok, secrecy assumption verified: fact unreachable attacker_sskey(skS[])
```

Figure 5. The Needham-Schroeder Public Key Protocol implement in Proverif and its output.

This language supports the specification of security queries, allowing users to express properties such as confidentiality, authentication, and integrity. ProVerif's language also supports equivalences, allowing users to check if two different executions of a protocol are indistinguishable to an adversary, which is useful for analyzing anonymity or privacy properties. Despite its power, the language can be complex to master, requiring users to familiarize themselves with process calculi and formal security properties.

C. CryptoVerif

Detailed Analysis of CryptoVerif's Methodology and Features

CryptoVerif is a computational protocol verifier designed to provide cryptographic security proofs under real-world cryptographic assumptions. Unlike symbolic verification tools like ProVerif, which abstract cryptographic primitives using the Dolev-Yao model, CryptoVerif operates within the computational model, offering stronger and more realistic security guarantees. It aims to prove that protocols are secure against probabilistic polynomial-time (PPT) adversaries, which aligns more closely with how cryptography is implemented in practice. CryptoVerif's methodology is based on game-based cryptographic proofs, a common approach used in modern cryptography. The tool models protocols as probabilistic processes, defining a sequence of games (cryptographic reductions) where each game is progressively transformed to get closer to a proof of the desired security property. This approach is particularly suited for proving secrecy, indistinguishability, and authentication properties under computational assumptions such as the hardness of discrete logarithm or Diffie-Hellman problems. One of the key features of CryptoVerif is its ability to generate proofs that protocols are secure in the presence of active adversaries, who can observe, intercept, and modify communications. CryptoVerif constructs formal proofs by showing that the success probability of any adversary is negligible. The tool also supports various cryptographic primitives, including symmetric encryption, public-key encryption, message authentication codes, digital signatures, and key exchange mechanisms, making it highly versatile. Unlike symbolic tools, CryptoVerif does not attempt to generate counterexamples or attacks. Instead, it focuses on proving the absence of vulnerabilities by analyzing protocol behavior under standard computational assumptions (Figure 6).

Case Studies and Applications in Various Contexts

CryptoVerif has been applied to several high-profile cryptographic protocols, providing formal proofs of security for some of the most widely used systems. One prominent case study involves the verification of the TLS 1.3 protocol, a critical protocol for secure internet communication. By analyzing TLS 1.3, CryptoVerif has been able to demonstrate its resistance to various attacks, ensuring that it meets strict security standards under computational assumptions. Additionally, CryptoVerif has been employed in verifying the security of the Signal protocol, a popular end-to-end encryption protocol used in secure messaging applications. This analysis has helped validate Signal's claims regarding secrecy and authentication. Another notable application of CryptoVerif is in the evaluation of VPN protocols, such as WireGuard. In this context, the tool has been used to confirm that WireGuard's cryptographic components (e.g., its key exchange and encryption mechanisms) are secure under the computational model. These real-world case studies highlight CryptoVerif's utility in validating the security properties of widely deployed protocols in modern communication systems. CryptoVerif is also applicable in post-quantum cryptography, a rapidly growing area of research in response to the potential threats posed by quantum computing. By verifying lattice-based and other post-quantum cryptographic schemes, CryptoVerif contributes to ensuring that future cryptographic systems remain secure even in a post-quantum world.

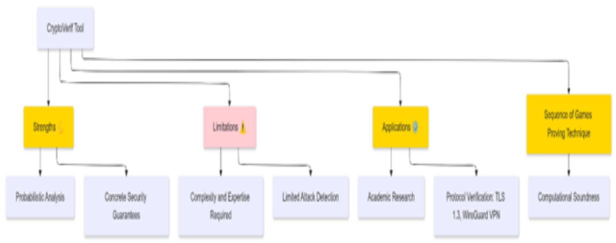

CryptoVerif Tool
Strengths
Limitations
Applications
Sequence of Games Proving Technique
Probabilistic Analysis
Concrete Security Guarantees
Complexity and Expertise Required
Limited Attack Detection
Academic Research
Protocol Verification: TLS 1.3, WireGuard VPN
Computational Soundness

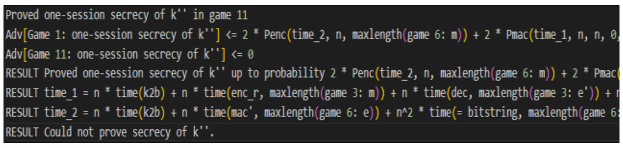
Figure 6. Overall key aspects of CryptoVerif

Strengths and Limitations

CryptoVerif's primary strength lies in its ability to generate computational security proofs. Because it operates in the computational model, its results are more aligned with practical cryptography and real-world implementations. This makes CryptoVerif especially valuable when analyzing protocols that require strong security guarantees, such as those used in banking systems, secure messaging, or national security applications. Furthermore, CryptoVerif supports a wide range of cryptographic primitives and can handle complex protocol behaviors. The tool's game-based approach allows for detailed cryptographic reductions, enabling researchers to develop strong security proofs. However, CryptoVerif also has some limitations. Its reliance on the computational model means that it often requires more detailed protocol specifications compared to symbolic tools like ProVerif. This makes CryptoVerif less automated, requiring substantial expertise in cryptographic concepts to accurately model protocols and interpret results. Additionally, generating computational proofs is computationally intensive, so the tool may require significant resources and time to analyze complex protocols. Another limitation is that CryptoVerif does not produce attack counterexamples like symbolic tools. Instead, it provides proofs of security or reports that the security proof could not be completed. This means it is primarily used for verifying protocols that are believed to be secure, rather than identifying vulnerabilities in protocols.

Language

The CryptoVerif language is designed for specifying cryptographic protocols at a detailed level. It is more expressive and complex than the symbolic languages used by tools like ProVerif because it must account for probabilistic processes and computational assumptions. The language allows users to specify the actions and cryptographic operations of the protocol, as well as the adversary's capabilities. In CryptoVerif, users can define secrecy properties, authentication goals, and indistinguishability properties using cryptographic reductions. These are the types of security properties that are proven or disproven based on the probabilistic behavior of the protocol under analysis. While the language is powerful, it is also challenging to learn and requires a solid understanding of both cryptography and formal verification techniques. Figure 7 presents an example of language use in CryptoVerif. Moreover, the language is closely tied to the game-based proof methodology. Users must define cryptographic games that represent the protocol's behavior in the presence of adversaries. CryptoVerif then applies reductions to simplify these games, ultimately proving the security properties or showing that they hold a negligible advantage for the adversary.



```
Proved one-session secrecy of k'' in game 11
Adv[Game 1: one-session secrecy of k''] <= 2 * Penc(time_2, n, maxlength(game 6: m)) + 2 * Pmac(time_1, n, n, 0,
Adv[Game 11: one-session secrecy of k''] <= 0
RESULT Proved one-session secrecy of k'' up to probability 2 * Penc(time_2, n, maxlength(game 6: m)) + 2 * Pmac(
RESULT time_1 = n * time(k2b) + n * time(enc_r, maxlength(game 3: m)) + n * time(dec, maxlength(game 3: e')) + n
RESULT time_2 = n * time(k2b) + n * time(mac', maxlength(game 6: e)) + n^2 * time(= bitstring, maxlength(game 6:
RESULT Could not prove secrecy of k''.
```

Figure 7. Language in CrytoVerif

D. The Tamarin Prover

Capabilities and Design

The Tamarin Prover is a highly advanced tool for the verification of security protocols, offering both falsification and unbounded verification within the symbolic model. It utilizes multiset rewriting systems to specify protocols, which define a labeled transition system. This system's state includes a symbolic representation of the adversary's knowledge, network messages, freshly generated values, and the protocol's internal state. Tamarin also supports the equational specification of cryptographic operators, such as Diffie-Hellman exponentiation and bilinear pairings, making it well-suited to model complex cryptographic scenarios [10].

Case Studies and Applications

Tamarin's flexibility and power are evident through its applications across a broad range of use cases. It has been successfully employed to model and analyze web applications, with its repository featuring numerous examples from academic papers, serving as a rich resource for modeling other protocols. Its ability to conduct symbolic analysis of security protocols further highlights its practical value in real-world settings.

Strengths and Limitations

Tamarin shines when it comes to handling protocols that involve non-monotonic mutable global state and intricate control flows, such as loops. The prover supports a form of induction and efficiently parallelizes proof searches, making it an effective tool for complex protocol analysis. However, its reliance on the symbolic model means it is limited to capturing attacks that fit within the model and the defined equational theory. Moreover, due to the undecidable nature of the underlying verification problems, Tamarin does not guarantee termination during verification.

Notable Applications and Use Cases

Tamarin has played a crucial role in identifying new vulnerabilities, significantly influencing the security of several real-world systems. Its formal verification of 5G protocols and symbolic analysis of web single-sign-on (SSO) protocols are key examples of its impact on critical, modern technologies. Several academic papers and surveys have explored Tamarin's capabilities and contributions. For example, one study [12] provides a detailed overview of Tamarin, highlighting the key findings and achievements made using the tool, while another paper [10] delves deeper into its design and broad range of applications.

In summary, the Tamarin Prover is a robust and flexible tool in the realm of security protocol verification. Its sophisticated capabilities and wide range of applications, from web protocols to 5G technologies, underscore its significant role in securing digital communication. While Tamarin's strengths in managing complex protocols and cryptographic operators are clear, it is important to recognize its limitations within the symbolic model and its potential for non-termination. The extensive body of research and case studies surrounding Tamarin not only emphasizes its importance but also provides a valuable resource for ongoing exploration and development in security protocol analysis. The following diagram (Figure 8) summarizes the key aspects of the Tamarin Prover.

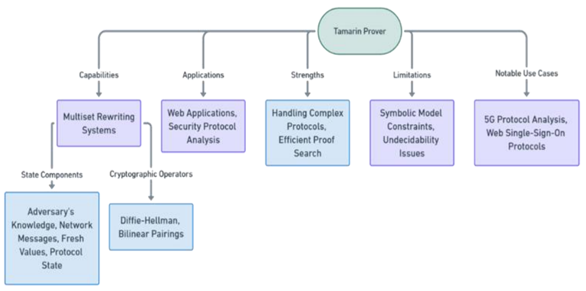

Tamarin Prover
Capabilities
Applications
Strengths
Limitations
Notable Use Cases
Multiset Rewriting Systems
Web Applications, Security Protocol Analysis
Handling Complex Protocols, Efficient Proof Search
Symbolic Model Constraints, Undecidability Issues
5G Protocol Analysis, Web Single-Sign-On Protocols
State Components
Cryptographic Operators
Adversary's Knowledge, Network Messages, Fresh Values, Protocol State
Diffie-Hellman, Bilinear Pairings

Figure 8. Overall key aspects of Tamarin

Language

The Tamarin Prover uses a specialized language based on multiset rewriting rules for specifying and analyzing security protocols. This language is designed to express the behavior of protocols in terms of actions (like sending and receiving messages) and states. It is quite expressive and allows for the specification of complex properties and behaviors of security protocols.

Key Features of Tamarin's Language:

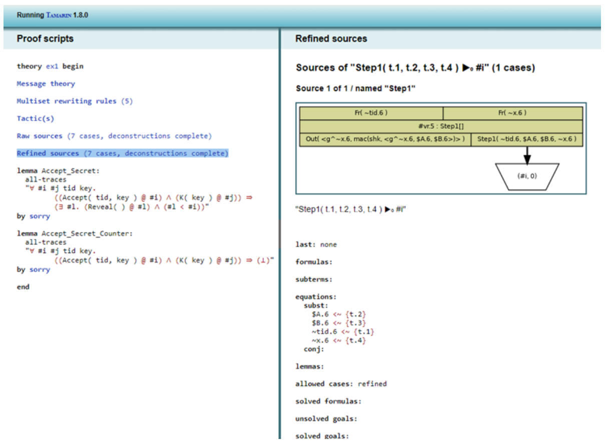
1. Multiset Rewriting Rules: These rules define how the state of the protocol evolves with each action.

2. State Representation: The state includes the knowledge of the adversary, the messages on the network, and the internal state of the protocol participants.

3. Fact-Based Syntax: The language uses facts to represent the state of the system and the actions taken by the protocol participants.

4. Support for Cryptographic Primitives: Tamarin can model various cryptographic operations and equational theories.

Let's consider a simple example of a protocol with two roles: an initiator and a responder. The protocol involves the initiator sending a nonce to the responder, who then responds with the nonce encrypted with a shared key (Figure 9).



```
theory ex1 begin

Message theory

Multiset rewriting rules (5)

Tactic(s)

Raw sources (7 cases, deconstructions complete)

Refined sources (7 cases, deconstructions complete)

lemma Accept_Secret:
  all-traces
  "∀ #i #j tid key.
        ((Accept( tid, key ) @ #i) ∧ (K( key ) @ #j)) ⇒
        (∃ #l. (Reveal( ) @ #l) ∧ (#l < #i))"
by sorry

lemma Accept_Secret_Counter:
  all-traces
  "∀ #i #j tid key.
        ((Accept( tid, key ) @ #i) ∧ (K( key ) @ #j)) ⇒ (⊥)"
by sorry

end
```

Sources of "Step1(t.1, t.2, t.3, t.4) ▶₀ #i" (1 cases)

Source 1 of 1 / named "Step1"

"Step1(t.1, t.2, t.3, t.4) ▶₀ #i"

```
last: none

formulas:

subterms:

equations:
  subst:
    $A.6 <~ {t.2}
    $B.6 <~ {t.3}
    ~tid.6 <~ {t.1}
    ~x.6 <~ {t.4}
  conj:

lemmas:

allowed cases: refined

solved formulas:

unsolved goals:

solved goals:
```

Figure 9. Tamarin Proper

In this Tamarin representation:

- Each rule represents an action in the protocol (sending or receiving a message).
- The [...] brackets represent the multiset before and after the action.
- !Initiator(I) and !Responder(R) denote the roles of the initiator and responder.
- Fr(~ni) denotes the generation of a fresh nonce ni.
- In(...) and Out(...) represent incoming and outgoing messages.
- St_I and St_R are states in the protocol.
- ---[...] → denotes the transition from one state to another, with labels representing events in the protocol.

In real-world protocols would typically be more complex, involving multiple steps and various cryptographic operations. The Tamarin Prover allows for the analysis of these protocols by checking them against various security properties like secrecy and authentication.

III. COMPARISON

In this section, we provide a detailed comparison of the pros and cons of various security protocol verification tools, as summarized in Table 1. The table offers an in-depth and comprehensive comparison of four widely used tools in the field: Scyther, ProVerif, CryptoVerif, and Tamarin. This comparison evaluates the tools based on several key criteria, including their main purpose, protocol verification capabilities, methods for protocol specification, and a range of additional features offered by each tool.

This analysis goes beyond simply listing the features of the tools; it provides a critical evaluation of their strengths and weaknesses in various contexts. Some tools, like Scyther, are better suited for handling standard protocol analyses with high efficiency, while others, such as Tamarin, are specifically designed to address more complex cryptographic challenges, including dynamic adversary models and mutable states. Similarly, ProVerif excels in symbolic verification of cryptographic protocols, providing an automated yet flexible approach, whereas CryptoVerif operates in the computational model, offering more realistic security guarantees at the cost of requiring more cryptographic expertise.

By comparing the tools in this manner, we aim to assist researchers and practitioners in the field of cybersecurity in making informed decisions about which tool best suits their specific needs. Whether the focus is on education, academic research, or real-world applications like cloud computing, IoT, or 5G networks, understanding the unique capabilities of each tool is critical. Scyther's accessibility and efficiency make it ideal for educational purposes and simple protocol analysis, whereas Tamarin's advanced features make it better suited for researchers dealing with complex scenarios. ProVerif strikes a balance between automation and flexibility, making it widely applicable in research settings, while CryptoVerif provides the computational rigor required for high-assurance security proofs.

Additionally, this comparison serves as a valuable guide for understanding not just how each tool operates but also how these tools can be adapted or extended to meet the challenges of evolving technologies. By identifying the unique strengths of each tool - whether in terms of user-friendliness, protocol specification methods, or advanced cryptographic support - we provide a clear roadmap for selecting the most appropriate tool for various security challenges. Furthermore, by highlighting the limitations of these tools, we suggest areas where future research and development could focus, aiming to improve the overall effectiveness and applicability of security protocol verification tools considering new technologies and security threats.

TABLE 1. COMPARATIVE ANALYSIS OF SECURITY PROTOCOL VERIFICATION TOOLS

Aspect	Scyther (Pros)	Scyther (Cons)	Tamarin (Pros)	Tamarin (Cons)	ProVerif (Pros)	ProVerif (Cons)	CryptoVerif (Pros)	CryptoVerif (Cons)
Main Purpose	Ideal for automatic verification of security protocols, especially standard analyses.	May not offer the advanced features needed for complex protocol analysis scenarios.	Designed for advanced, cutting-edge features in security protocol verification.	Might be overly complex for basic or standard protocol analysis needs.	Well-suited for symbolic verification of cryptographic protocols and properties.	Symbolic model may not capture real-world cryptographic behaviors.	Ideal for proving cryptographic security properties under computational assumptions.	Requires more expertise in cryptography for effective use.
Protocol Verification	Excellent for scenarios with an unbounded number of sessions and nonces.	Limited in handling more complex protocol structures compared to Tamarin.	Superior in handling dynamic corruption and user-specified adversaries.	Requires more in-depth understanding of protocol complexities.	Can handle infinite state spaces, highly efficient for verifying secrecy and authentication.	Struggles with advanced cryptographic properties like non-repudiation or side-channel attacks.	Supports verification against probabilistic polynomial-time adversaries, more realistic in real-world cryptographic scenarios.	Less automated than symbolic verification tools, requiring detailed specifications.
Protocol Specification	Linear role scripts make it user-friendly and easier to understand.	Less flexible in protocol specification compared to Tamarin's Multiset Rewriting.	Multiset Rewriting allows for more detailed and complex protocol specification.	Higher learning curve due to the complexity of Multiset Rewriting.	Process calculus-based specification allows flexibility in defining protocols.	Can be complex to model certain types of protocols due to symbolic abstraction.	Can model detailed cryptographic operations and assumptions, aligning more closely with practical cryptography.	Complex to model protocols compared to symbolic approaches like ProVerif.
Analysis Features	Efficient characterizations of protocols, with a finite representation of behaviors.	May lack advanced analysis features like proof visualization.	Advanced features like attack finding, visualization, and API support for global state.	Could be overkill for simpler analysis requirements.	Supports verification of advanced cryptographic operations like encryption and digital signatures.	Lacks computational model support, limiting its real-world applicability.	Supports game-based proofs for advanced cryptographic analysis, more comprehensive in cryptographic assumptions.	Computational proofs may be resource-intensive and time-consuming.

Cryptographic Message Model	Simpler, more straightforward free term algebra.	Less extensive than Tamarin's model, lacking advanced cryptographic supports.	Extensive model including Diffie-Hellman, bilinear pairing, and advanced cryptographic terms.	The complexity of the cryptographic message model can be a barrier for less experienced users.	Operates in the symbolic model, efficiently analyzing properties under idealized conditions.	Less suitable for real-world cryptographic assumptions compared to computational tools.	Works within the computational model, providing stronger security guarantees.	Not beginner-friendly due to the complexity of the computational model.
User-friendliness	More accessible for beginners and for educational purposes.	Might not cater to needs of advanced users seeking sophisticated analysis tools.	Interactive proof construction appeals to advanced users.	Potentially intimidating for beginners or those seeking straightforward tools.	Automated tool, accessible for researchers familiar with process calculi.	Challenging for beginners unfamiliar with process calculi or formal verification.	Flexible for expert users in cryptographic verification.	Intimidating for those unfamiliar with formal cryptographic proofs.
Efficiency	Highly efficient for standard protocol analyses.	May not perform as well in complex or advanced analysis scenarios.	Ideal for complex protocol verifications due to its advanced capabilities.	May require more resources and time for analysis compared to Scyther.	Efficient in handling large-scale protocols with symbolic analysis.	Not designed for proving computational security properties.	Can handle complex and real-world cryptographic systems, effective in resource-intensive analysis.	Resource-intensive, especially for large and complex cryptographic systems.
Educational Use	Excellent resources for learning and teaching about security protocols.	May not cover advanced topics in depth.	Provides detailed tutorial materials and manual for advanced learning.	Might be challenging for those new to security protocol analysis.	Widely used in research for teaching formal verification methods.	May require significant effort to fully understand for teaching advanced topics.	Used in advanced cryptography teaching and research, provides deep insights into cryptographic protocol verification.	Requires a deep understanding of computational cryptography.
Development and Community Support	Active development and available on GitHub; good community support.	Limited to specific use cases and might not evolve rapidly.	Continuous development with cutting-edge updates; strong community involvement.	The tool's complexity could limit its accessibility and user base.	Strong community support and ongoing development.	Focused on research and academic use; may not evolve as quickly as other tools.	Actively developed and supported in the cryptographic community, strong tool for real-world cryptographic protocol analysis.	High complexity limits accessibility for non-expert users.

IV. Discussion

In this section, we discuss the four tools - Scyther, Tamarin, ProVerif, and CryptoVerif - through the lens of their ability to adapt to technological shifts, evolve alongside cryptographic advancements, improve user experience, address new security challenges and a specific case study.

Adapting to Technological Shifts

As technology evolves, so do the demands placed on security protocol verification tools. With the rise of new technologies like cloud computing, 5G, and blockchain, the need for tools that can handle increasingly complex and distributed systems has become paramount. Scyther, though highly efficient for traditional security protocol analysis, may fall short in addressing the intricacies of these evolving systems, particularly due to its simplified model. On the other hand, Tamarin excels in handling more advanced and dynamic environments, such as protocols in 5G networks and blockchain-based systems, thanks to its sophisticated handling of mutable global states and advanced adversary models. Both ProVerif and CryptoVerif are adapting to these technological changes by being used to verify modern protocols like TLS 1.3 and Signal, which are critical for securing communications in the current landscape.

Evolving with Cryptographic Developments

The field of cryptography is constantly advancing, with new encryption algorithms and cryptographic primitives being developed to counteract emerging threats, such as those posed by quantum computing. CryptoVerif stands out in this respect, as it is specifically designed to prove cryptographic security under computational assumptions, which align more closely with practical cryptography and real-world implementations. This makes CryptoVerif well-positioned to handle future cryptographic developments, including post-quantum cryptography. ProVerif also continues to evolve by extending its support for more sophisticated cryptographic operations, such as algebraic properties and stateful protocols, while Tamarin's support for Diffie-Hellman exponentiation and bilinear pairings ensures that it can handle increasingly complex cryptographic scenarios. Scyther, while effective, remains more suited for standard cryptographic verification tasks and may not be as flexible in handling emerging cryptographic developments.

Enhancing User Experience and Accessibility

User experience and accessibility are crucial factors for the wider adoption of security verification tools, especially as they are increasingly used by non-experts, such as engineers or students, in addition to researchers. Scyther is often lauded for its user-friendly interface and intuitive design, making it particularly well-suited for beginners and educational purposes. However, more advanced users may find it lacking in features required for complex analyses. Tamarin, while more powerful in terms of capabilities, has a steeper learning curve, which could deter new users, but its interactive proof construction appeals to experienced users who require precise control over the verification process. ProVerif strikes a balance by offering automated verification while maintaining flexibility for those familiar with formal methods. CryptoVerif, on the other hand, is highly flexible but can be daunting for beginners due to its focus on computational proofs and the need for cryptographic expertise.

Leveraging Cloud and Distributed Computing

With the shift toward cloud computing and distributed systems, the need for tools that can verify security protocols across distributed environments is increasing. Tamarin excels in this area, with its support for modeling complex distributed systems, such as multi-party protocols and stateful systems that are prevalent in cloud infrastructures. ProVerif and CryptoVerif also show promise in adapting to cloud environments. For instance, ProVerif's ability to handle infinite state spaces makes it useful for protocols in large, distributed networks, while CryptoVerif's focus on computational security allows it to verify the resilience of cryptographic systems in the cloud. Scyther, while efficient in traditional setups, may need more refinement to handle the growing complexity of cloud-based environments.

Specialization for IoT and Emerging Networks

The rise of Internet of Things (IoT) devices and emerging networks like 5G requires tools that can verify protocols designed for low-power, resource-constrained devices. In this context, Tamarin stands out, as it has been successfully applied to protocols in 5G and similar networks, where security needs are paramount, and the network's complexity is vast. Scyther can efficiently verify simpler IoT protocols but lacks the advanced features required for comprehensive IoT network analysis. ProVerif and CryptoVerif also provide useful insights, particularly for ensuring confidentiality and authentication in IoT communication protocols. As IoT networks grow in scale and importance, these tools will need to continue evolving to address the unique challenges they present, such as lightweight cryptography and low-latency communication.

Interoperability and Collaboration

Interoperability between different verification tools is becoming more important as the complexity of protocols increases, and multiple verification paradigms are needed for comprehensive security analysis. While Scyther and Tamarin offer standalone solutions, they don't yet have strong capabilities for integration with other tools. ProVerif, on the other hand, has been adapted into other verification environments and frameworks, making it highly interoperable. CryptoVerif has also taken steps toward integration with EasyCrypt via the CV2EC translation tool, enabling users to verify cryptographic primitives in EasyCrypt after verifying protocols in CryptoVerif. Collaborative environments are essential for tackling modern, multifaceted security challenges, as no single tool may suffice for complex, real-world scenarios.

Addressing New Security Threats

The security landscape is continuously evolving, with new threats such as side-channel attacks, zero-day exploits, and the rise of post-quantum cryptographic attacks. CryptoVerif is particularly well-suited to address these new challenges due to its grounding in the computational model, which allows it to verify cryptographic protocols under real-world conditions. ProVerif continues to adapt by adding support for new types of security properties, though its reliance on the symbolic model means it may miss certain real-world vulnerabilities. Tamarin's ability to model complex adversary behaviors and dynamic corruption makes it particularly effective at discovering novel attacks, especially in cutting-edge technologies like 5G and blockchain. Scyther, while highly efficient for standard security analysis, is more limited in addressing these emerging threats.

Fostering Community and Open-Source Development

Open-source development and community support are critical for the continued improvement and adoption of security verification tools. Scyther, Tamarin, ProVerif, and CryptoVerif all benefit from strong community involvement and active development. Tamarin has seen significant contributions from the research community, which has helped it stay on the cutting edge of security protocol verification. ProVerif and CryptoVerif are both widely used in academic research, and their open-source nature ensures that they continue to evolve with the latest advancements in cryptographic theory and protocol design. Scyther's GitHub presence and active development make it a popular choice for educational purposes, though its development may not be as rapid as that of more complex tools.

Educational Initiatives

Educational initiatives play a vital role in fostering the next generation of security professionals. Scyther stands out for its ease of use and intuitive interface, making it an excellent teaching tool for students who are new to the field of security protocol analysis. Its straightforward setup allows students to focus on learning the core concepts without being overwhelmed by complex features. Tamarin and ProVerif, while more advanced, offer extensive tutorials and resources that cater to more experienced users or graduate-level students who are looking to deepen their understanding of formal verification methods. CryptoVerif, though challenging for beginners, is often included in advanced cryptography courses, where students can gain experience with computational proofs and cryptographic protocol verification.

Case study

In the context of Mobile Cloud Computing, ensuring confidentiality and integrity between the Mobile Device, Trusted Third Party (TTP), and Verifier is crucial. These tools aid researchers in protocol verification when proposing a new protocol. For instance, in papers [26], to develop a protocol suitable for the context, we start with basic information and then use tools to analyze and identify weaknesses, errors, or unreasonable points. This process continues until the desired value is achieved.

Scyther can be effectively used to assess the Message Exchange Protocol, identifying limitations or vulnerabilities that could lead to potential attacks or failures. This understanding enables the development of targeted solutions to address these issues. The following example outlines the message exchange process of a simple protocol involving three parties.

Message Exchange Steps (Figure 10):

1. The Mobile Device sends an initial attestation request to the Verifier, encrypted with the Verifier's public key and signed using the Mobile Device's private key.

2. The TTP forwards a nonce to the Verifier to ensure freshness, encrypted with the Verifier's public key.

3. The Mobile Device sends verification data back to the TTP, including hashes of nonces and keys, encrypted and signed.

4. The Mobile Device sends attestation information directly to the Verifier.

5. The TTP sends the attestation results to the Verifier to confirm the results.

6. The Verifier confirms the verification results back to the Mobile Device.

Figure 11 below illustrates how Scyther presents the message exchange process, revealing the outcomes shown in Figure 12. Due to the basic design, the protocol appears susceptible to potential vulnerabilities. This figure clearly highlights the protocol's weaknesses, allowing developers to identify and address each error at various steps. Through this process, we can pinpoint current weaknesses in the protocol and propose solutions that align with the security requirements. This enables us to propose appropriate solutions to upgrade and refine the protocol.

For example, the main drawback of Step 1 in this message exchange protocol lies in its lack of strong integrity verification and replay attack prevention measures. Specifically, in Step 1, the Mobile Device sends an initial attestation request to the Verifier, which is encrypted using the Verifier's public key (pk(Verifier)) and signed with the Mobile Device's private key (sk(MobileDevice)). Although signing the message provides authenticity of the source (proving that the message came from the Mobile Device) and encryption provides confidentiality, the protocol lacks additional context-binding information to ensure that this request cannot be used out of its intended context. An attacker who intercepts this message could potentially replay it to disrupt the protocol or attempt to forge communications with the Verifier later. This weakness could lead to the Mobile Device unknowingly being part of a replay attack. Furthermore, there is no mention of timestamps, sequence numbers, or session identifiers, which are critical for ensuring that the messages cannot be reused by a malicious party at a different time. This makes the protocol vulnerable to replay attacks and context misuse. We addressed this drawback by incorporating an ephemeral key (EphemeralKeyM) alongside the signed NonceM. This key serves as an additional freshness factor and binding to the current session, reducing the effectiveness of replay attacks by making it more difficult for an attacker to reuse the captured message in a meaningful way. Additionally, having the Mobile Device sign the NonceM and other session-related information further strengthens the integrity of the request, ensuring that the Verifier can validate the origin and guarantee that the request has not been modified in transit. These measures significantly enhance the security of Step 1 against replay and impersonation attacks. (Figure 13). This enhancement provides protection against replay and impersonation attacks by confirming the authenticity of the sender and the integrity of the message. Figure 14 presents the result of proposed solution, which can improve the protocol to avoid several attacks. Finally, the Figure 15 demonstrates the alternative message exchange based on the proposed solution.

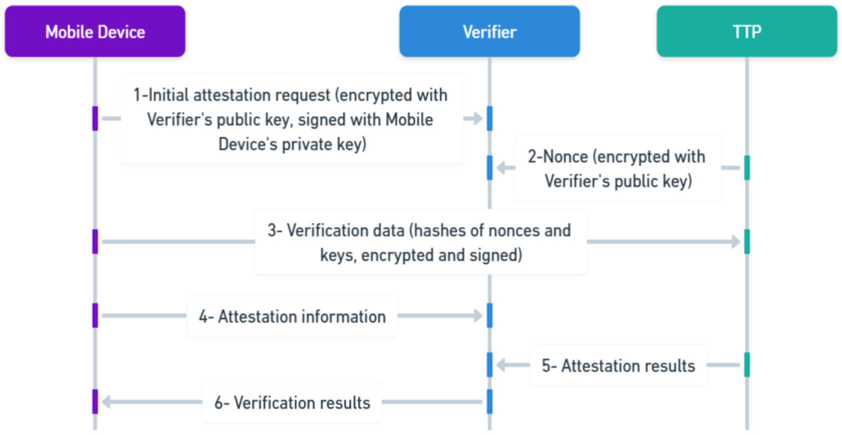
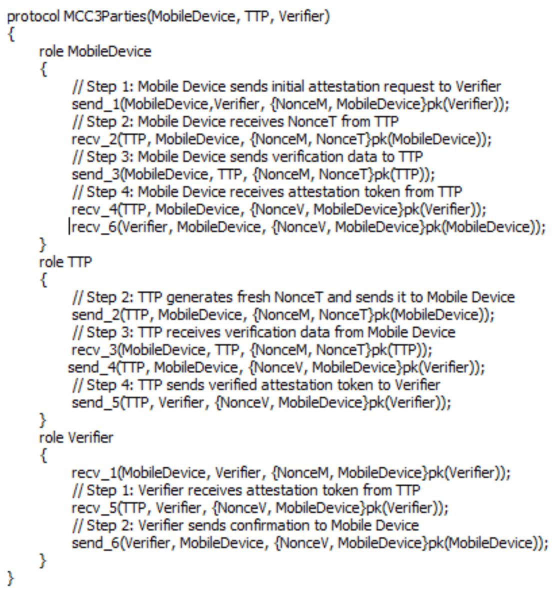
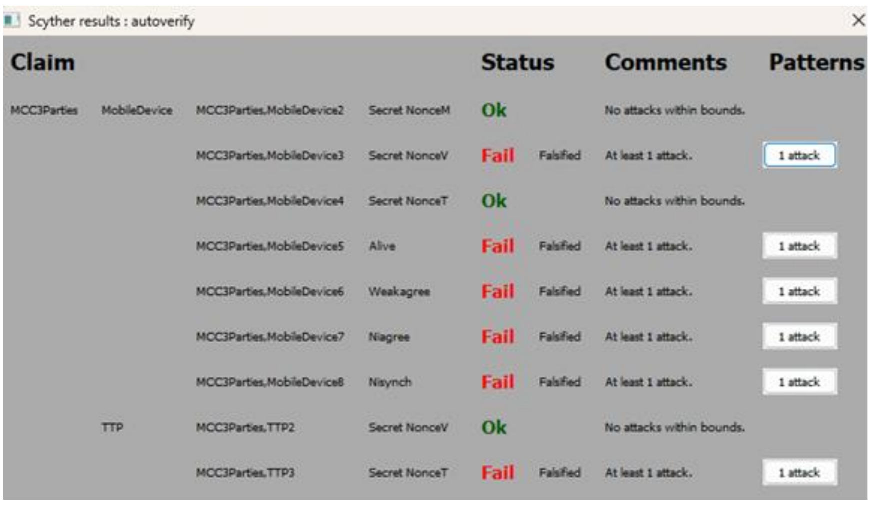

Mobile Device
Verifier
TTP
1-Initial attestation request (encrypted with Verifier's public key, signed with Mobile Device's private key)
2-Nonce (encrypted with Verifier's public key)
3- Verification data (hashes of nonces and keys, encrypted and signed)
4- Attestation information
5- Attestation results
6- Verification results

Figure 10. Message exchange



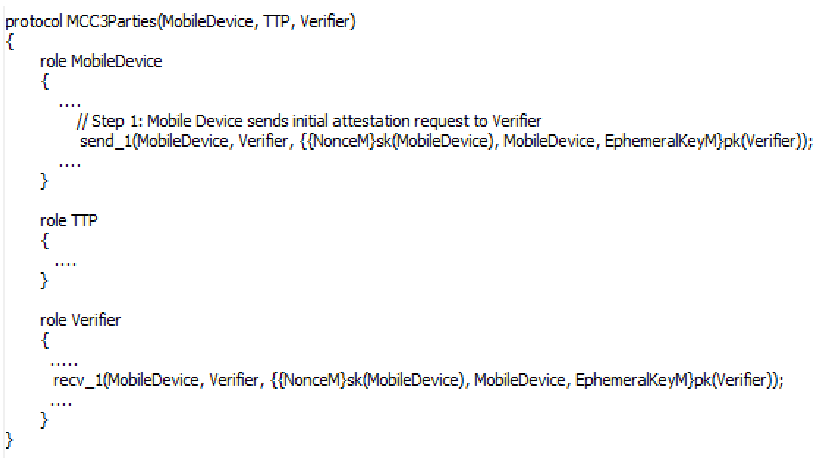
```
protocol MCC3Parties(MobileDevice, TTP, Verifier)
{
    role MobileDevice
    {
        // Step 1: Mobile Device sends initial attestation request to Verifier
        send_1(MobileDevice,Verifier, {NonceM, MobileDevice}pk(Verifier));
        // Step 2: Mobile Device receives NonceT from TTP
        recv_2(TTP, MobileDevice, {NonceM, NonceT}pk(MobileDevice));
        // Step 3: Mobile Device sends verification data to TTP
        send_3(MobileDevice, TTP, {NonceM, NonceT}pk(TTP));
        // Step 4: Mobile Device receives attestation token from TTP
        recv_4(TTP, MobileDevice, {NonceV, MobileDevice}pk(Verifier));
        recv_6(Verifier, MobileDevice, {NonceV, MobileDevice}pk(MobileDevice));
    }
    role TTP
    {
        // Step 2: TTP generates fresh NonceT and sends it to Mobile Device
        send_2(TTP, MobileDevice, {NonceM, NonceT}pk(MobileDevice));
        // Step 3: TTP receives verification data from Mobile Device
        recv_3(MobileDevice, TTP, {NonceM, NonceT}pk(TTP));
       send_4(TTP, MobileDevice, {NonceV, MobileDevice}pk(Verifier));
        // Step 4: TTP sends verified attestation token to Verifier
        send_5(TTP, Verifier, {NonceV, MobileDevice}pk(Verifier));
    }
    role Verifier
    {
        recv_1(MobileDevice, Verifier, {NonceM, MobileDevice}pk(Verifier));
        // Step 1: Verifier receives attestation token from TTP
        recv_5(TTP, Verifier, {NonceV, MobileDevice}pk(Verifier));
        // Step 2: Verifier sends confirmation to Mobile Device
        send_6(Verifier, MobileDevice, {NonceV, MobileDevice}pk(MobileDevice));
    }
}
```

Figure 11. Scyther for ensuring confidentiality and integrity between Mobile Device, TTP, and Verifie



Claim				Status		Comments	Patterns
MCC3Parties	MobileDevice	MCC3Parties,MobileDevice2	Secret NonceM	Ok		No attacks within bounds.	
		MCC3Parties,MobileDevice3	Secret NonceV	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,MobileDevice4	Secret NonceT	Ok		No attacks within bounds.	
		MCC3Parties,MobileDevice5	Alive	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,MobileDevice6	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,MobileDevice7	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,MobileDevice8	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	TTP	MCC3Parties,TTP2	Secret NonceV	Ok		No attacks within bounds.	
		MCC3Parties,TTP3	Secret NonceT	Fail	Falsified	At least 1 attack.	1 attack

Figure 12. Result of verification

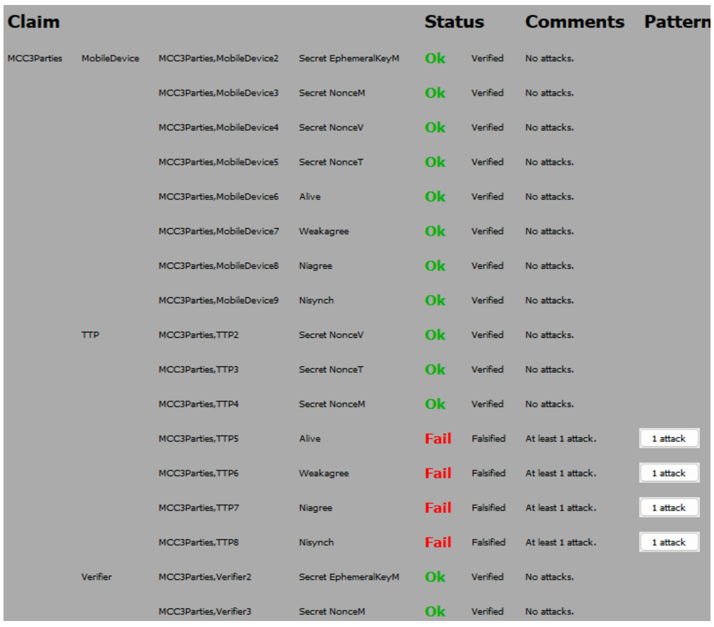


```
protocol MCC3Parties(MobileDevice, TTP, Verifier)
{
    role MobileDevice
    {
      ....
        // Step 1: Mobile Device sends initial attestation request to Verifier
         send_1(MobileDevice, Verifier, {{NonceM}sk(MobileDevice), MobileDevice, EphemeralKeyM}pk(Verifier));
      ....
    }

    role TTP
    {
      ....
    }

    role Verifier
    {
     .....
     recv_1(MobileDevice, Verifier, {{NonceM}sk(MobileDevice), MobileDevice, EphemeralKeyM}pk(Verifier));
     ....
    }
}
```

Figure 13. The revised Step 01 in Scyther



Claim				Status		Comments	Pattern
MCC3Parties	MobileDevice	MCC3Parties,MobileDevice2	Secret EphemeralKeyM	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice3	Secret NonceM	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice4	Secret NonceV	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice5	Secret NonceT	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice6	Alive	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice7	Weakagree	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice8	Niagree	Ok	Verified	No attacks.	
		MCC3Parties,MobileDevice9	Nisynch	Ok	Verified	No attacks.	
	TTP	MCC3Parties,TTP2	Secret NonceV	Ok	Verified	No attacks.	
		MCC3Parties,TTP3	Secret NonceT	Ok	Verified	No attacks.	
		MCC3Parties,TTP4	Secret NonceM	Ok	Verified	No attacks.	
		MCC3Parties,TTP5	Alive	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,TTP6	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,TTP7	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		MCC3Parties,TTP8	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	Verifier	MCC3Parties,Verifier2	Secret EphemeralKeyM	Ok	Verified	No attacks.	
		MCC3Parties,Verifier3	Secret NonceM	Ok	Verified	No attacks.	

Figure 14. The result of proposed solution

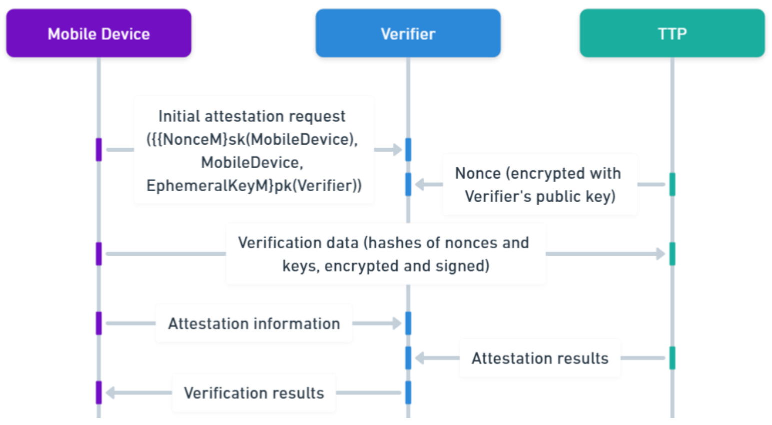

Mobile Device
Verifier
TTP
Initial attestation request ({{NonceM}sk(MobileDevice), MobileDevice, EphemeralKeyM}pk(Verifier))
Nonce (encrypted with Verifier's public key)
Verification data (hashes of nonces and keys, encrypted and signed)
Attestation information
Attestation results
Verification results

Figure 15. The revised message exchange of proposed solution

Although no protocol is flawless, Scyther significantly contributes to protocol improvement by pinpointing weaknesses it detects, thereby enhancing the protocol's robustness.

V. Conclusion

In conclusion, this paper provides a detailed comparative analysis of four leading security protocol verification tools, Scyther, Tamarin, ProVerif, and CryptoVerif, and highlights their strengths, limitations, and unique contributions to the field. This work contributes to a deeper understanding of how these tools can be leveraged for different security challenges in modern technological environments, such as cloud computing, IoT, and 5G networks. A key contribution of this paper is the comprehensive comparison across critical dimensions like protocol verification capabilities, cryptographic support, user accessibility, and adaptability to technological shifts. This analysis helps researchers and practitioners make informed decisions on which tool to use based on their specific requirements, such as standard protocol analysis, advanced cryptographic proofs, or dynamic adversary modeling. Looking forward, this paper outlines potential areas for future work, such as the need for better interoperability between verification tools, improved support for post-quantum cryptography, and enhanced user accessibility to make these powerful tools more approachable for non-experts. The rapid growth in distributed systems and IoT also suggests a growing need for specialized tools capable of handling the complex security demands of these environments. This work contributes to the growing body of knowledge in security protocol verification, providing valuable insights that can guide future research and tool development. By addressing the gaps in current tools and suggesting directions for their evolution, this paper fosters a more secure and robust approach to the verification of security protocols in an ever-changing digital landscape.

Acknowledgement

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology and Education for this study.

References

[1] M. A. Fikri, K. Ramli, and D. Sudiana, “Formal Verification of the Authentication and Voice Communication Protocol Security on Device X Using Scyther Tool,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1077, no. 1, p. 012057, Feb. 2021. doi: 10.1088/1757-899X/1077/1/012057.

[2] N. Dalal, J. Shah, K. Hisaria, and D. Jinwala, “A Comparative Analysis of Tools for Verification of Security Protocols,” *Int. J. Commun. Netw. Syst. Sci.*, vol. 03, no. 10, pp. 779–787, 2010. doi: 10.4236/ijcns.2010.310104.

[3] H. A. Elbaz, M. H. Abd-elaziz, and T. M. Nazmy, “Analysis and Verification of a Key Agreement Protocol over Cloud Computing Using Scyther Tool,” vol. 2, no. 2, 2014.

[4] L. Chen and C. Kudla, “Identity based authenticated key agreement protocols from pairings,” in *16th IEEE Computer Security Foundations Workshop, 2003. Proceedings.*, Pacific Grove, CA, USA: IEEE Comput. Soc, 2003, pp. 219–233. doi: 10.1109/CSFW.2003.1212715.

[5] Y. Yang, H. Yuan, L. Yan, and Y. Ruan, “Post-quantum identity-based authenticated multiple key agreement protocol,” *ETRI J.*, vol. 45, no. 6, pp. 1090–1102, Dec. 2023, doi: 10.4218/etrij.2022-0320.

[6] C. J. Cremers, “The Scyther Tool: Verification, falsification, and analysis of security protocols,” in *International Conference on Computer Aided Verification*, Springer, 2008, pp. 414–418. Accessed: Nov. 04, 2016. Available: http://link.springer.com/chapter/10.1007/978-3-540-70545-1_38.

[7] C. J. F. Cremers, “Unbounded verification, falsification, and characterization of security protocols by pattern refinement,” in *Proceedings of the 15th ACM conference on Computer and communications security*, Alexandria Virginia USA: ACM, Oct. 2008, pp. 119–128. doi: 10.1145/1455770.1455787.

[8] C. Xi and L. Siqi, “Research on semantics and algorithm of formal analysis tool Scyther,” in *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, Dali, China: IEEE, Oct. 2022, pp. 1058–1074.doi: 10.1109/ICCASIT55263.2022.9987170.

[9] X. Zhang, Y. Zhu, C. Gu, and X. Miao, “A Formal Verification Method for Security Protocol Implementations Based on Model Learning and Tamarin,” *J. Phys. Conf. Ser.*, vol. 1871, no. 1, pp. 012102, Apr. 2021, doi: 10.1088/1742-6596/1871/1/012102.

[10] S. Meier, B. Schmidt, C. Cremers, and D. Basin, “The TAMARIN Prover for the Symbolic Analysis of Security Protocols,” in *Computer Aided Verification*, vol. 8044, N. Sharygina and H. Veith, Eds., in Lecture Notes in Computer Science, vol. 8044, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 696–701. doi: 10.1007/978-3-642-39799-8_48.

[11] Y. Xiong, C. Su, W. Huang, F. Miao, W. Wang, and H. Ouyang, “Verifying Security Protocols using Dynamic Strategies,” Aug. 25, 2019, *arXiv*: arXiv:1807.00669. A Jan. 18, 2024. Available: http://arxiv.org/abs/1807.00669.

[12] D. Basin, C. Cremers, J. Dreier, and R. Sasse, “Tamarin: Verification of Large-Scale, Real-World, Cryptographic Protocols,” *IEEE Secur. Priv.*, vol. 20, no. 3, pp. 24–32, May 2022, doi: 10.1109/MSEC.2022.3154689.

[13] P.V. Hau and D. T. T. Hien, “Enhancing Web Application Security: A Deep Learning and NLP-based Approach for Accurate Attack Detection” in *Journal of Science and Technology on Information Security*, vol. 3, no. 20, pp. 77-87, December 2023. doi: https://doi.org/10.54654/isj.v3i20.1008.

[14] B. Blanchet, “The Security Protocol Verifier ProVerif and its Horn Clause Resolution Algorithm,” Electron. Proc. Theor. Comput. Sci., vol. 373, pp. 14–22, Nov. 2022, doi: 10.4204/EPTCS.373.2.

[15] J. Yao, C. Xu, D. Li, S. Lin, and X. Cao, “Formal Verification of Security Protocols: ProVerif and Extensions,” in *Artificial Intelligence and Security*, vol. 13339, X. Sun, X. Zhang, Z. Xia, and E. Bertino, Eds., in Lecture Notes in Computer Science, vol. 13339, pp. 500–512, Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-031-06788-4_42.

[16] E.-Y. Yap, J.-J. Chin, and A. Goh, “Verifying MQV-Based Protocols Using ProVerif,” in *IT Convergence and Security*, vol. 782, H. Kim and K. J. Kim, Eds., in Lecture Notes in Electrical Engineering, vol. 782, pp. 55–63, Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-16-4118-3_6.

[17] M. Abadi, B. Blanchet, and C. Fournet, “The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication,” 2016, *arXiv*. doi: 10.48550/ARXIV.1609.03003.

[18] B. Blanchet and B. Smyth, “Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers,” in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, Lisbon: IEEE, pp. 310–324, Jun. 2016. doi: 10.1109/CSF.2016.29.

[19] B. Blanchet, “CryptoVerif: a Computationally-Sound Security Protocol Verifier (Initial Version with Communications on Channels),” 2023, *arXiv*. doi: 10.48550/ARXIV.2310.14658.

[20] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, “A Formal Security Analysis of the Signal Messaging Protocol,” *J. Cryptol.*, vol. 33, no. 4, pp. 1914–1983, Oct. 2020. doi: 10.1007/s00145-020-09360-1.

[21] K. Bhargavan, B. Blanchet, and N. Kobeissi, “Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate,” in *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA: IEEE, pp. 483–502, May 2017, doi: 10.1109/SP.2017.26.

[22] B. Blanchet, "Dealing with Dynamic Key Compromise in Crypto Verif," in *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, Enschede, Netherlands, pp. 495-510 2024. doi:10.1109/CSF61375.2024.00015.

[23] B. Lipp, B. Blanchet, and K. Bhargavan, “A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden: IEEE, Jun. 2019, pp. 231–246. doi: 10.1109/EuroSP.2019.00026.

[24] B. Blanchet, P. Boutry, C. Doczkal, B. Grégoire and P. -Y. Strub, "CV2EC: Getting the Best of Both Worlds," 2024 IEEE 37th Computer Security Foundations Symposium (CSF), Enschede, Netherlands, pp. 279-294, 2024, doi: 10.1109/CSF61375.2024.00019.

[25] T. M. C. Le, X. T. Pham, and V. T. Le, “Advancing Security Protocol Verification: A

Comparative Study of Scyther, Tamarin" in *Journal of Technical Education Science*, vol. 19, no. 1, pp. 43–53, Feb. 2024. doi: 10.54644/jte.2024.1523.

[26] T. L. Vinh, H. Cagnon, S. Bouzefrane, and S. Banerjee, "Property-based token attestation in mobile computing: Property-based token attestation in mobile computing," in *Concurr. Comput. Pract. Exp.*, pp. e4350, Oct. 2017. doi: 10.1002/cpe.4350.

ABOUT THE AUTHOR

Le Vinh Thinh

Workplace: Ho Chi Minh City University of Technology and Education (HCMUTE)

Email: thinhlv@hcmute.edu.vn

Education: After completing a B.Sc. at the University of Natural Sciences, Le Vinh Thinh obtained a Master's degree in IT at the U.P. Technology University in India in 2006 and completed a Ph.D. in Computer Science at the Conservatoire National des Arts et Métiers (CNAM), Paris, France, in 2017. Currently, he is a faculty member in the Department of Information Technology at Ho Chi Minh City University of Technology and Education, Vietnam. He is the author and co-author of over 20 scientific articles.

Recent research direction: Trust and Reputation Systems, Security, Mobile Cloud Computing, and the Internet of Things (IoT) based AI.

Lê Vĩnh Thịnh

Nơi làm việc: Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh (HCMUTE)

Email: thinhlv@hcmute.edu.vn

Học vấn: Sau khi hoàn thành bằng Cử nhân tại Trường Đại học Khoa học Tự nhiên, Lê Vĩnh Thịnh đã nhận bằng Thạc sĩ CNTT tại Đại học Công nghệ U.P ở Ấn Độ vào năm 2006 và hoàn thành Tiến sĩ Khoa học Máy tính tại Conservatoire National des Arts et Métiers (CNAM), Paris, Pháp, vào năm 2017. Hiện tại, đang là giảng viên tại Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh, Việt Nam. Là tác giả và đồng tác giả của hơn 20 bài báo khoa học.

Hướng nghiên cứu gần đây: Hệ thống tin cậy và danh tiếng, bảo mật, điện toán đám mây di động và AI dựa trên Internet vạn vật.