Journal of Theoretical and Applied Information Technology
15th March 2025. Vol.103. No.5
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EVALUATING FORMAL METHODS FOR VERIFYING SECURITY PROTOCOLS: A CASE STUDY OF TAMARIN, AVISPA, AND PROVERIF

ADEL HASSAN[1], ISAM ISHAQ[2], JORGE MUNILLA[3]

[1]Faculty of Engineering and Information Technology, Arab American University, Jenin, Palestine
[2]Faculty of Engineering and Information Technology, Al-Quds University, Jerusalem, Palestine
[3]Faculty of Engineering and Information Technology, Telecommunication Engineering, University of Malaga, Spain
Email: [1]adel7377@gmail.com , [2]isam@alquds.edu , [3]munilla@ic.uma.es

ABSTRACT

Verifying security protocols using formal methods is crucial to ensure their robustness against cyber threats. Several verification tools, including Tamarin, AVISPA, and ProVerif, offer different methodologies for protocol analysis. However, a comprehensive comparative analysis of these tools under uniform conditions remains limited. This study systematically evaluates these three tools by assessing their verification mechanisms, supported programming languages, and usability. A standardized testing framework was employed to ensure a consistent comparison, focusing on two widely used security protocols: the Diffie-Hellman Key Exchange Protocol and the Needham-Schroeder Public Key Protocol. The findings highlight distinct strengths and weaknesses in each tool. Tamarin demonstrated superior capability in detecting active attacks such as Man-in-the-Middle (MitM) attacks, while ProVerif was more effective in identifying passive attacks like eavesdropping. AVISPA, on the other hand, provided a broader but less detailed security analysis. These insights help researchers and practitioners select the most appropriate tool based on protocol complexity and security requirements. Unlike prior research that focused on individual tools, this study offers a comprehensive empirical comparison, providing deeper insights into their practical effectiveness and limitations. The results contribute to enhancing security protocol verification methodologies and informing future improvements in formal verification tools.

Keywords: *Formal Methods, Security, Security Protocol, Protocol Modeling, Verification Processes, Testing Tools.*

1. INTRODUCTION

A network protocol is a set of instructions and rules that govern the exchange of information across a network. In terms of security protocols, these rules ensure that transmitted messages remain unaltered and protected using encryption mechanisms. However, despite these encryption techniques, security protocols may still be vulnerable to attacks, making security, privacy, and data confidentiality critical concerns for researchers. This underscores the need for robust security verification methods that can assess protocol integrity, privacy, and authenticity.

According to Yang et al. [2], security protocols are typically verified using two primary approaches:

1. **Proven security**: where a protocol's security is mathematically demonstrated by evaluating its maximum confidentiality level.
2. **Formal methods**: which use mathematical modeling techniques to verify protocol specifications and security guarantees.

Despite the availability of these approaches, several key challenges persist in security protocol verification, as outlined below. Current Challenges in Security Protocol Verification With the increasing complexity of security protocols and their reliance on advanced encryption mechanisms, verifying their security has become a significant challenge for researchers and developers. Although numerous formal verification tools exist, three main challenges remain unresolved:

1. **Protocol Complexity**: Modern security protocols involve multiple interactions between entities, making formal verification computationally intensive and time-consuming.
2. **Evolving Attacks**: The emergence of new attack types, such as Man-in-the-Middle (MitM) attacks and Replay Attacks, necessitates tools that can dynamically detect and mitigate diverse security threats.
3. **Limitations of Existing Tools**: Many formal verification tools struggle to handle complex protocols or fail to detect specific attack types. Additionally, some tools require advanced technical expertise in programming languages or mathematical models, limiting their usability among non-experts.

To address these challenges, this research systematically evaluates three widely used verification tools—Tamarin, AVISPA, and ProVerif—to determine their strengths, weaknesses, and applicability in different security contexts.

This study is structured as follows:

- **The first part** reviews key formal verification tools used in security protocol testing.
- **The second part** presents the results of empirical security tests conducted on two well-known protocols: Needham-Schroeder Public Key Protocol and Diffie-Hellman Key Exchange (DHKE) Protocol.
- **The third part** analyzes and compares the characteristics of these tools, leading to a set of practical recommendations for improving security protocol verification methods.

Research Hypothesis

Different formal verification tools exhibit varying effectiveness in detecting security vulnerabilities, with each tool excelling in specific attack scenarios and protocol structures. This study hypothesizes that no single tool is universally superior; rather, a combination of verification tools may provide optimal security assessment and a more comprehensive evaluation of security protocols.

2. IMPORTANCE OF THE RESEARCH AND ITS OBJECTIVES

2.1 Research Objective

This study aims to explore and evaluate the performance of three protocol testing tools—Tamarin, AVISPA, and ProVerif—by conducting comprehensive security tests on two widely used security protocols: the Needham-Schroeder Public Key Protocol and the Diffie-Hellman Key Exchange (DHKE) Protocol. The research investigates the unique advantages of each tool and assesses their effectiveness in identifying vulnerabilities. Additionally, it examines whether the success of a verification tool is influenced by the nature of the protocol itself.

2.2 Importance of the Research

Given the complexity of security protocols and the limitations of existing verification tools, a systematic comparative study is crucial. This research bridges the gap by analyzing the effectiveness of Tamarin, AVISPA, and ProVerif in identifying vulnerabilities within well-established protocols.

Although numerous studies have explored security protocol verification, no comprehensive comparative analysis has been conducted to highlight the strengths and weaknesses of each tool across different security scenarios. This study addresses this gap by evaluating the tools against two well-known security protocols, offering insights into their real-world applicability.

Through this comparative analysis, the study provides practical guidelines for researchers and developers in selecting the most suitable verification tool based on protocol characteristics and attack models. Additionally, it contributes to a deeper understanding of formal verification methodologies and proposes recommendations to improve the efficiency and flexibility of security verification tools in the future.

2.3 Research Problem Statement

With the growing reliance on security protocols in modern systems, ensuring their robustness has become a critical necessity. However, the increasing complexity of contemporary security protocols and the evolving nature of cyber threats pose significant verification challenges. While numerous formal verification tools and methodologies exist, their effectiveness and applicability remain constrained by specific limitations, necessitating further evaluation and comparative analysis.

For instance, widely used tools such as Tamarin, AVISPA, and ProVerif offer distinct advantages and drawbacks. Some tools excel at detecting specific attacks but struggle with highly complex protocols, while others provide user-friendly interfaces yet cannot identify advanced security threats. Additionally, many verification tools require specialized knowledge of programming languages or mathematical models, limiting their accessibility to non-expert users.

Despite extensive research on security protocol verification, a structured comparative evaluation of these tools under consistent testing conditions remains lacking. Prior studies have either assessed individual tools in isolation or provided limited comparative insights, leaving an open research gap in understanding how these tools perform across diverse protocol complexities and attack scenarios.

To bridge this gap, this study conducts an empirical evaluation of Tamarin, AVISPA, and ProVerif, systematically analyzing their effectiveness in detecting security vulnerabilities in two widely used protocols: the Needham-Schroeder Public Key Protocol and the Diffie-Hellman Key Exchange (DHKE) Protocol. Through this analysis, we aim to identify the strengths and weaknesses of each tool and provide practical guidelines for researchers and developers in selecting the most suitable verification tool based on protocol characteristics and security requirements.

By establishing a unified comparative framework, this research contributes to the advancement of formal verification methodologies, assisting both academia and industry in enhancing the security of cryptographic protocols.

2.4 Previous Literature

Previous research has extensively explored formal verification of security protocols using advanced mathematical modeling techniques. Palombo et al. (2015) [3], highlighted that formal verification tools such as ProVerif face challenges when dealing with protocols with unbounded states, affecting the accuracy and effectiveness of verification.

Additionally, Palombo demonstrated that certain tools may fail to detect complex attacks due to inherent modeling limitations. For instance, some verification frameworks struggle with analyzing security properties under dynamic threat conditions, leading to potential blind spots in security assessments.

Furthermore, Palombo analyzed the application of techniques such as Horn clauses and pi-calculus in ProVerif, emphasizing the practical challenges associated with implementing these methods. Despite their mathematical robustness, these techniques often require deep expertise and may not generalize well to all protocol types.

Despite these studies, a comprehensive comparative evaluation of Tamarin, AVISPA, and ProVerif under uniform conditions and across multiple security protocols remains lacking. This research builds upon existing literature by conducting a structured, empirical comparison of these tools, analyzing their effectiveness in detecting vulnerabilities and identifying potential areas for improvement to better align with modern security requirements, Palombo et al. (2015) [3].

3. FORMAL METHODS VERIFICATION TOOLS

There are many accredited formal methods built on formal loading tools, three of which were selected and tested in this research, namely:

1. AVISPA.
2. ProVerif.
3. Tamarin.

3.1. AVISPA

AVISPA [4, 5, 6, 7] (Automated Validation of Internet Security Protocols and Applications) is a multi-party tool developed to analyze information security protocols that support the new generation of Internet applications. This tool is designed to be a comprehensive system for automatic verification of the security level of security protocols. This tool integrates different approaches to security analysis, starting from model inspection techniques for protocol forgery analysis to symbolic verification methods based on abstract verification. The main feature of this tool is the loading tools provided. It consists of four tools – Figure (1) illustrates the structure of AVISPA and its tools – where the protocol is encoded in HLPSL (High-Level Protocol Specification Language).

AVISPA consists of four main tools:

1. CL-Atse (Constraint-Logic-based Attack Searcher): It uses constraint logic, where it applies solvers for solution and simplification with redundancy elimination techniques.
2. OFMC (On-the-Fly Model-Checker): It uses encoding techniques to examine the performance of protocol penetration, in addition to bounded loading, by exploring the state space in a need-based manner.
3. Sat-MC (SAT-based Model-Checker): It builds proposed encoding equations for all the potential effects on the protocol and uses a SAT-type solution algorithm.
4. TA4SP (Tree Automata for Security Protocols): It relies on an automatic approximation method for loading to know the penetration, and it uses regular tree languages and rewrites the protocol to provide approximate difference values.

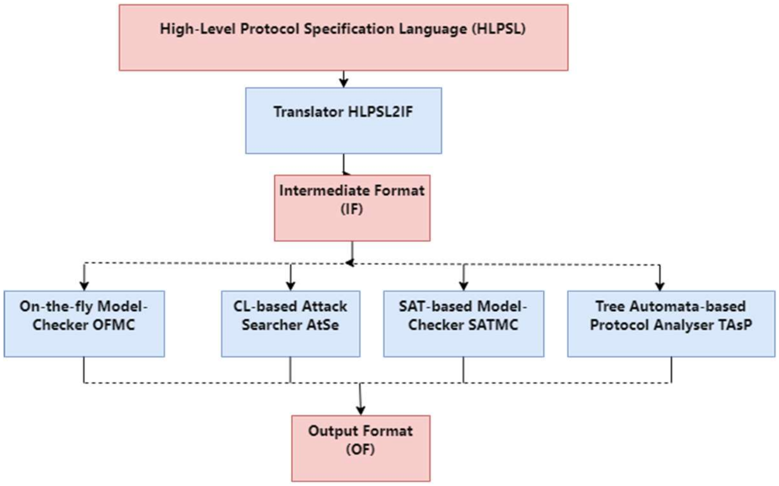

Figure 1: Structure of AVISPA and its tools [3].

3.1.1 AVISPA PROPERTIES:

1. Working method: machine tool.

2. Complexity: Somewhat difficult to use.

3. Prerequisites for using this tool:

   A. Deep knowledge of the analyzed protocols.

   B. Learn a New Programming Language (HLPSL).

4. Reliability: Validation or detection of defects.

5. Ease of use: This can be used to prove a malfunction in the protocol.

6. Analysis method: Analyze all messages that make up the protocol at the same time.

7. The tool is efficient in: Verifying that the protocol under test is strong against restart attacks and intermediary attacks.

3.2. ProVerif :

ProVerif is a tool for solving security protocols using pi calculus techniques and their extensions of equation and function theories, which can represent cryptographic operations. ProVerif is capable of handling an unlimited number of protocol sessions and unlimited messaging space Copet et al. (2024) [8].

The main steps in the verification process are:

1. An attacker sentence is added to each message.
2. The attacker then tries to infer the data through Horn sentences M. Arapinis et al. (2014) [9].

Horn sentences are a type of logistical sentence where one sentence is often positive, and all the other sentences are negative¹. These sentences bear the name of Alfred Horn who described them in the 1951 article. These sentences are mainly used in logistics programming and provide a basis for logical thinking and logistics programming Blanchet et al. (2022) [10].

In 1999, Weidenbach proposed using Horn sentences to model security protocols [11]. In this model, protocols are encoded as first-order Horn sentences. This allows protocols to be analyzed more complexly and increases the intruder's ability to identify. This approach can allow false attacks and does not guarantee overall termination.

If the tool is unable to prove a particular property, it tries to rebuild an attack, i.e., tracking the execution of the protocol that fails to achieve the desired property. Figure 2 shows the structure of ProVerif:

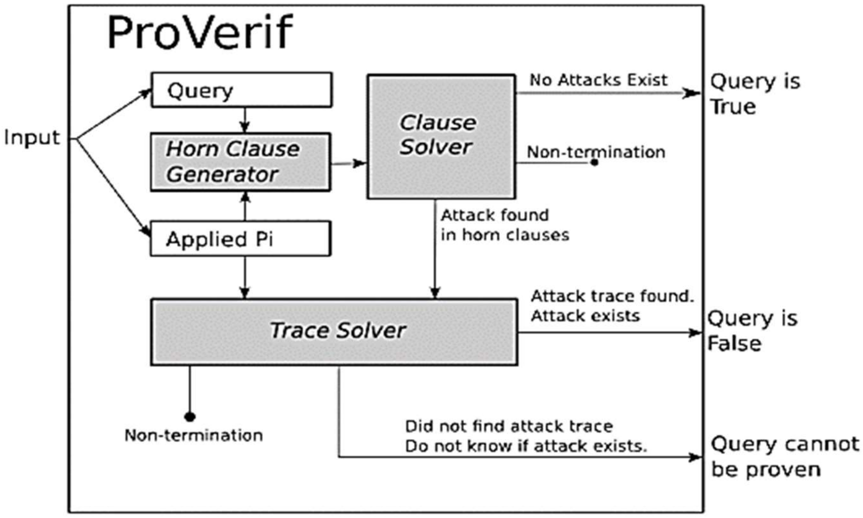

Figure 2: ProVerif architecture [10]

3.2.1 ProVerif properties

The protocol is designed using century phrases or pi-calculus.

- The tool must be run through the command line interface.
- It generates the following possible outputs:
  - the property is true,
  - the property is false and the attack effect is generated, and the property cannot be proven when a false attack is found,
  - the tool may not end.
- A step-by-step tracking is created to explain the operation and attack.
- Trace is generated only for the inspected property.
- Connected parties need to be process modeled.
- Equality can be verified using ‘if…then’ or ‘let…in’.
- It only checks for those attacks for which the "query" is specified in the code.
- ProVerif does not require any such specifications.
- No special code is required for a ProVerif Novelty attack.
- Communication channels must be identified.
- ProVerif is a powerful tool for verifying protocols in formal models. It works for an unbounded number of sessions and an unbounded message space.

3.3 Tamarin:

Tamarin Prover [12, 13] is a powerful tool for symbolic modeling and analysis of security protocols. It takes as input a security protocol model, specifying the actions performed by agents running the protocol in different roles (for example, protocol initiator, reply, trusted key server), along with identifying the adversary and specifying the desired protocol properties. Tamarin can be used to create automatic proof that the protocol, even when an infinite number of instances of protocol roles overlap in parallel, along with adversary actions, meets its specified characteristics.

3.3.1 Characteristics of the Tamarin Tool

1. Protocol modeling: The Tamarin tool models protocols using Maude, a symbolic rewrite language. This allows Tamarin to model complex protocols with different cryptographic operations and state transitions.
2. Tool execution: Tamarin is executed through a command-line interface, making it easy to integrate into an automated security analysis workflow.
3. Output generation: Tamarin generates the following possible outputs:

- Valid property: When the protocol meets the specified security property.
- Property violation detected: When the protocol violates the specified security property, an attack trace is generated.
- Property cannot be proven: If the tool fails to prove or refute the specified security feature.
- Potential non-termination issue: It refers to a situation where the program can encounter an infinite loop, meaning the program continues to execute certain operations or commands repeatedly and endlessly. This usually happens due to complex states in the program or due to a design or programming error. In this case, the program does not reach a specific or final result, and therefore it continues to operate continuously without stopping or ending. This phenomenon is also known as ‘infinite loop’ or ‘infinite repetition’.
- Attack Tracking: Tamarin provides detailed step-by-step tracking of the attack, explaining how a security violation occurred. Table -1 shows a comparison of the characteristics of protocol security analysis tools.

4. Property identification: Tamarin allows security properties to be defined using temporal logic, enabling the definition of a variety of security requirements, such as confidentiality, authenticity, and non-rebuttal capabilities.
5. Attack detection: Tamarin detects both active and passive attacks, including replay attacks, man-in-the-middle attacks, and impersonation attacks.
6. Message sequence analysis: Tamarin effectively analyzes message sequences, identifying potential vulnerabilities and discrepancies in the flow of protocol communications.

7. Complexity Processing: Tamarin can handle protocols of medium to high complexity, making it suitable for analyzing protocols used in real-world applications.
8. Ease of use: Tamarin is relatively easy to use, even for users with a limited background in formal methods.
9. Learning resources: Tamarin provides comprehensive documentation and learning resources to help users use the tool effectively.

Tamarin provides a balance between ease of use, powerful attack detection capabilities, and support for complex protocols, making it a valuable tool for analyzing the security of cryptographic protocols.

Table 1 - Comparison of the characteristics of protocol security analysis tools

Feature	Tamarin	AVISPA	ProVerif
Protocol Modeling	Maude language	HLPSL language	Horn clauses or pi-calculus
Tool Execution	Command-line interface	Semi-automatic requires user interaction	Command-line interface
Output Generation	Property is true, Property is false, Property cannot be proven, Tool might not terminate.	Property is true, Property is false, Attack trace is generated, Property cannot be proven.	Property is true, Property is false, Attack trace is generated, Property cannot be proven.
Attack Trace	Step-by-step trace explaining the run and attack	Detailed explanation of the attack and the path to the attack	Step-by-step trace explaining the run and attack
Property Specification	Temporal logic	Temporal logic, CTL* (computation tree logic*),	Temporal logic

		and LTL (linear temporal logic)	
Attack Detection	Active and passive attacks	Active and passive attacks	Active and passive attacks
Message Sequence Analysis	Effective analysis of message sequences	Analyzing all messages simultaneously	Analyzing message sequences in isolation
Complexity Handling	Moderate to high complexity	High complexity	Moderate to high complexity
Ease of Use	Relatively easy to use for users with basic formal methods knowledge	Difficult to use for beginners, requires deep HLPSL knowledge	Relatively easy to use for users with basic formal methods knowledge
Learning Resources	Comprehensive documentation and learning resources	Limited documentation and learning resources	Comprehensive documentation and learning resources

4. TESTED PROTOCOLS:

Two protocols have been chosen:

1- First Protocol: Needham-Schroeder Public Key Protocol

2- Protocol II: Diffie-Hellman Key Exchange (DHKE)

4.1 Needham-Schroeder Public Key Protocol:

The Needham-Schroeder Public Key Protocol is a two-party mutual authentication protocol using public key cryptography. The protocol was proposed by Roger Needham and Michael Schroeder in 1978.

The Needham-Schroeder Public Key Protocol is secure against retransmission attacks, but vulnerable to man-in-the-middle attacks [14, 15, 16, 17].

The Needham-Schroeder Public Key protocol relies on the use of a public-key encryption algorithm. In this context, both Alice (A) and Bob (B) collaborate with a trusted server (S) to distribute public keys upon request. These keys include:

K_{PA}: The public key of A
K_{PB}: The public key of B
K_{PS}: The public key of server S
K_{SS}: The private key of server S

The protocol operates as follows:

$$A \rightarrow S: \{A, B\}$$

Here, A requests the public key of B from S.

$$S \rightarrow A: \{B, KPb\}_{K_{SS}}$$

S responds with the public key K_{PB} along with the identity of B, signed by the server for authentication purposes.

$$A \rightarrow B: \{N_A, A\}_{K_{PB}}$$

A chooses a random number Na and sends it to B.

$$B \rightarrow S: \{B, A\}$$

Now, B knows that A wants to communicate, so B requests the public keys of A.

$$S \rightarrow B: \{K_{PA}, A\}_{K_{SS}}$$

The server responds.

$$B \rightarrow A: \{N_A, N_B\}_{K_{PA}}$$

B chooses a random number N_B and sends it to A along with N_A to prove the ability to decrypt using K_{SB}.

$$A \rightarrow B: \{N_B\}_{K_{PB}}$$

A confirms N_B to B, to prove the ability to decrypt using K_{SA}.

At the end of the protocol, both A and B know each other's identities, and both know N_A and N_B. These nonces are not known to eavesdroppers.

This protocol establishes a secure communication channel between parties A and B, allowing them to exchange messages confidentially. The server plays an essential role in facilitating key exchange and ensuring the authenticity of the parties involved, as shown in Figure 3.

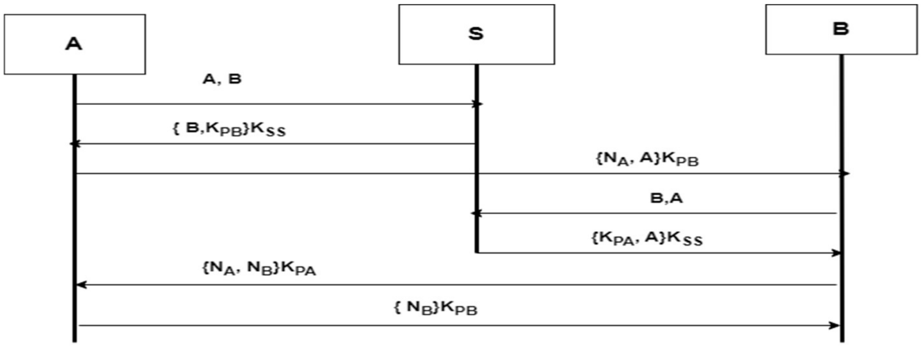

Figure 3: Needham-Schroeder Public Key Protocol Scheme and Code

In a man-in-center attack, the attacker eavesdrops on communications between parties *A* and *B*, then intervenes and sends fake messages to both parties.

In the case of the Needham-Schroeder Public Key Protocol, an attacker could do the following:

$$A \rightarrow I: \{N_A, A\}_{K_{PI}}$$

A sends N_A to *I*, who decrypts the message with K_{SI}

$$I \rightarrow B: \{N_A, A\}_{K_{PB}}$$

I relays the message to *B*, pretending that *A* is communicating

$$B \rightarrow I: \{N_A, N_B\}_{K_{PA}}$$

B sends N_B

$$I \rightarrow A: \{N_A, N_B\}_{K_{PA}}$$

I relays it to *A*

$$A \rightarrow I: \{N_B\}_{K_{PI}}$$

A decrypts N_B and confirm it to *I*, who learn it

$$I \rightarrow B: \{N_B\}_{K_{PB}}$$

I re-encrypts N_B, and convinces *B* that she's decrypted it

At the end of the attack, B falsely believes that *A* is communicating with him and that N_A and N_B are known only to *A* and *B*.

Thus, the attacker has managed to grab the session key for both parties, allowing him to read all messages that are sent between the two parties.

Figure 4 shows how a man-in-the-middle attack occurs on the Needham-Schroeder Public Key Protocol.

A man-in-the-middle attack can be avoided by using other key exchange techniques, such as the Diffie-Hellman protocol.

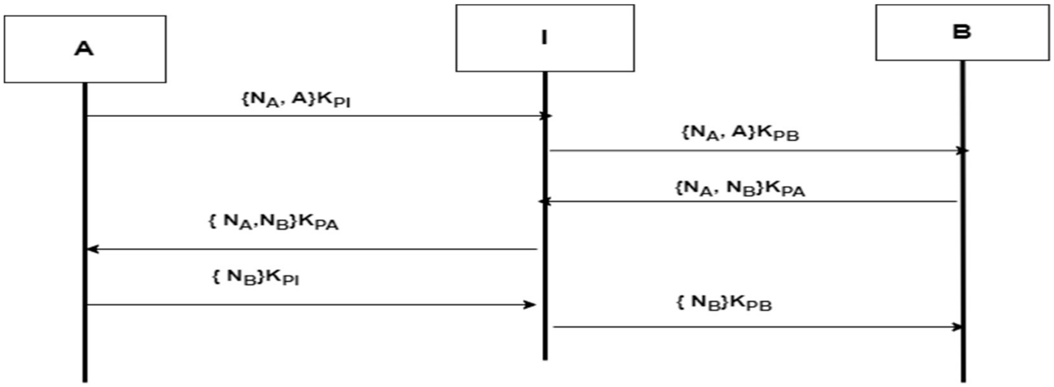

Figure 4: Attack on Needham-Schroeder Public Key Protocol

4.1.1 Needham-Schroeder Public Key Protocol Test Using Proverif Technology:

The test result is as follows:

```
Verification summary:
Query inj-event(endBparam(x)) ==> inj-event(beginBparam(x)) is false.
Query inj-event(endAparam(x)) ==> inj-event(beginAparam(x)) is true.
Query not attacker(secretANa[]) is true.
Query not attacker(secretANb[]) is true.
Query not attacker(secretBNa[]) is false.
Query not attacker(secretBNb[]) is false.
```

These results reflect the security characteristics of the protocol:

1. Query inj-event(endBparam(x)) ==> inj-event(beginBparam(x)) is false.: This means that client B does not necessarily start a session before it ends. This could indicate a vulnerability in the protocol where an attacker could send fake messages.
2. Query inj-event(endAparam(x)) ==> inj-event(beginAparam(x)) is true.: This means that client A must start a session before it ends. This is the expected behavior.
3. Query not attacker(secretANa[]) is true. and Query not attacker(secretANb[]) is true.: This means that the attacker cannot access the secrets secretANa and secretANb. This is good for protocol security.
4. Query not attacker(secretBNa[]) is false. and Query not attacker(secretBNb[]) is false.: This means that the attacker has access to the secrets secretBNa and secretBNb. This indicates a security vulnerability in the protocol.

Based on the results of the Proverif test, it can be said that the Needham-Schroeder Public Key protocol is not completely secure. The reasons are:

1. Client B does not have to start a session before it ends, which means that an attacker can send fake messages.
2. The attacker has access to the secrets secretBNa and secretBNb.

So, obviously, there are some security vulnerabilities in the protocol that need to be addressed.

4.1.2 Using AVISPA Technology:

The test result is as follows:

```
% OFMC
% Version of 2006/02/13
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  /home/span/span/testsuite/results/Needham.if
GOAL
  secrecy_of_nb
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.01s
  visitedNodes: 8 nodes
  depth: 2 plies
ATTACK TRACE
i -> (a,6): start
(a,6) -> i: {Na(1).a}_ki
i -> (b,3): {Na(1).a}_kb
(b,3) -> i: {Na(1). Nb(2)}_ka
and -> (a,6): {Na(1). Nb(2)}_ka
(a,6) -> i: {Nb.) _ki
i-> (i,17): Nb(2)
i-> (i,17): Nb(2)

% Reached State:
%
% secret(Nb(2),nb,set_70)
% witness(b,a,alice_bob_na,Na(1))
% contains(a,set_70)
% contains(b,set_70)
% secret(Na(1),na set_74)
% contains(a,set_74)
% contains(i,set_74)
% state_alice(a,i,ka,ki,4,Na(1),Nb(2),set_74,6)
% state_bob(b,a,ka,kb,3,Na(1),Nb(2),set_70,3)
% state_alice(a,b,ka,kb,0,dummy_nonce,dummy_nonce,set_62,3)
% witness(a,i,bob_alice_nb,Nb(2))
% request(a,i,alice_bob_na,Na(1),6)
```

The result indicates that the protocol is not secure. An attack was found to violate the confidentiality of the protocol. The attack is carried out by attacking messages that are shared between the parties.

Offensive tracking shows the steps an aggressor follows to gain access to confidential information. In this case, the aggressor has access to the value Nb (2), which must be confidential.

Statistical analysis shows that the tool has visited 8 nodes and 2 layers deep to find this attack within 0.01 seconds.

Target status at the end of the trace shows the final state of the protocol after the attack. It can be seen that the aggressor received the value Nb (2), which confirms that the attack was successful.

Please note that this attack is based on the aggressor's ability to listen to and manipulate messages shared between parties. If this capability is not available, the protocol may be secure. Therefore, security should always be assessed in the context of the surrounding environment.

4.1.3 Using Tamarin Technology:

The test result is as follows showing a summary of summaries:

```
analyzed: Needham.spthy
  types (all-traces): verified (33 steps)
  nonce_secrecy (all-traces): verified (54 steps)
  injective_agree (all-traces): verified (92 steps)
  session_key_setup_possible (exists-trace):
verified (5 steps).
```

4.1.4 Explanation of the test result:

The Needham.spthy protocol has been analyzed and the protocol has been validated in all possible cases. The following characteristics have been verified:

1. types (all-traces): Species-related properties validated.
2. nonce_secrecy (all-traces): Validated characteristics related to the confidentiality of random numbers.
3. injective_agree (all-traces): Validated characteristics related to the real agreement.
4. session_key_setup_possible (exists-trace): Validated properties related to the ability to set up the session key.

All properties have been successfully verified. The Needham-Schroeder Public Key Protocol can be considered safe due to the test result. The protocol was validated in all possible cases and the characteristics related to types, the confidentiality of random numbers, the real agreement and the possibility of setting up the session key were verified. All properties have been successfully verified.

4.2 Protocol II: Diffie-Hellman Key Exchange (DHKE)

The Diffie-Hillman key exchange protocol is one of the most important advances in public key cryptography and is still frequently implemented in a variety of modern security protocols. It allows two parties who have never met before to create a key that they can use to secure their communications [18, 19, 20].

In the Diffie-Hillman key exchange protocol, each party generates a public/private key pair and distributes the public key. After obtaining an original copy of each other's public keys, Alice and Bob can calculate a shared secret without an internet connection. A shared secret can be used, for example, as a key for symmetric encryption.

The basic steps of the Diffie-Hillman key exchange protocol are as follows:

1. A sends the following information to B:

- n: common large exponential prime number.
- g: exponential root of variable unit n.
- g^x mod n: a synthetic result calculated by user A using a secret number x and the global numbers n and g. This value is converted to a specific formula.

2. B receives the information from A and sends the following information to A:

- g^y mod n: synthetic result calculated by user B using a secret number y and the global numbers n and g. This value is converted to a specific formula.

After receiving both steps, both user A and user B use the global numbers they received to calculate the shared key.

To calculate the shared key, user A calculates g^{yx} mod n and uses it as a shared key, while user B calculates g^{xy} mod n and also uses it as a shared key.

Now, A and B have the same common key that can be used to encrypt and decrypt messages by the symmetric encryption algorithm. Figure (5) shows the mechanism and the Diffie-Hellman Key Exchange code:

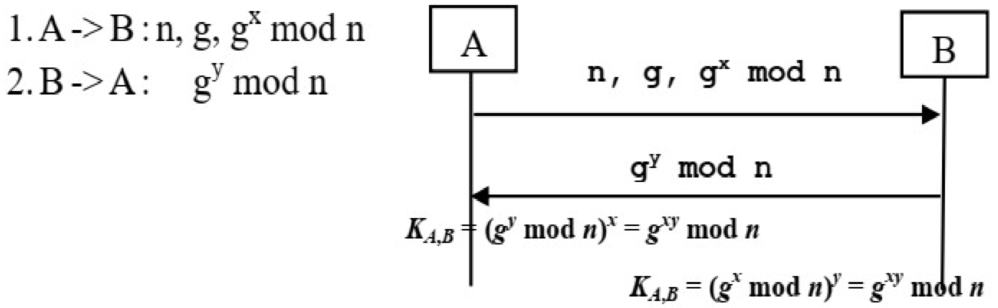

Figure 5: Diffie-Hellman Key Exchange Mechanism and Code [21]

4.2.1 Testing the protocol with ProVerif:

Test result:

```
C:\ProVerif>check Diffie_Hellman.pv

Linear part:

exp(exp(g,x),y) = exp(exp(g,y),x)

Completing equations...

Completed equations:

exp(exp(g,x),y) = exp(exp(g,y),x)

Convergent part: No equation.

Biprocess 0 (that is, the initial process):

{1}new a: exponent;

{2}new b: exponent;

{3}new c: exponent;

{4}out(d,
(exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c
)]))

-- Observational equivalence in biprocess 0.

Translating the process into Horn clauses...

Termination warning: v ≠ v_1 &&
attacker2(v_2,v) && attacker2(v_2,v_1) -> bad

Selecting 0

Termination warning: v ≠ v_1 &&
attacker2(v,v_2) && attacker2(v_1,v_2) -> bad

Selecting 0

Completing...

Termination warning: v ≠ v_1 &&
attacker2(v_2,v) && attacker2(v_2,v_1) -> bad

Selecting 0

Termination warning: v ≠ v_1 &&
attacker2(v,v_2) && attacker2(v_1,v_2) -> bad

Selecting 0

RESULT Observational equivalence is true.

------------------------------------------------------------
---

Verification summary:

Observational equivalence is true.
```

Explaining the result from ProVerif in detail:

1. Linear part: exp(exp(g,x),y) = exp(exp(g,y),x): This refers to the linear part of the protocol, and it expresses the equation that must be true at all times. In this case, it expresses the basic property of the Diffie-Hillman protocol: $(g^{ab}) = (g^{ba})$.
2. Completing equations... Completed equations: exp(exp(g,x),y) =

exp(exp(g,y),x): This indicates that ProVerif has completed the equations necessary to verify the protocol.

3. Convergent part: No equation.: This indicates that there are no equations that need to be solved in the converging part of the protocol.
4. Biprocess 0 (that is, the initial process): {1}new a: exponent; {2}new b: exponent; {3}new C: exponent; {4}out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a), b),exp(g,c)])): This refers to the initial process analyzed. In this case, three secret numbers (a, b, and c) are generated and a message containing g^a, g^b, and a choice is sent between $(g^a)^b$ and g^c.
5. -- Observational equivalence in biprocess 0. : This indicates that the protocol has passed the observed equivalence test.
6. Translating the process into Horn clauses...: This indicates that ProVerif translates the protocol into a set of Horn statements to verify security features.
7. Termination warning: v ≠ v_1 & attacker2(v_2,v) & attacker2(v_2,v_1) -> bad: These are termination warnings. They indicate that ProVerif has not been able to prove that the protocol always expires. This doesn't necessarily mean there's a problem, but it does indicate that ProVerif couldn't verify this aspect of the protocol.
8. : RESULT Observational equivalence is true.: This indicates that the protocol has passed the observed equivalence test. This means that an attacker cannot distinguish between the two different protocols.
9. Verification summary: Observational equivalence is true.: This is a summary of the results and confirms that the protocol has passed the observed equivalency test.

In general, this result means that the protocol you provided works as expected and is secure according to the characteristics verified by ProVerif.

4.2.2 Protocol testing with AVISPA:

Test result:

```
result Diffie_Hellman.hlpsl
% OFMC
% Version of 2006/02/13
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  /home/span/span/testsuite/results/DH3.if
GOAL
  secrecy_of_nb
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.01s
  visitedNodes: 12 nodes
  depth: 2 plies
ATTACK TRACE
i -> (a,6): start
(a,6) -> i: {Na(1).a}_ki
i -> (b,3): {Na(1) XOR i XOR b.a}_kb
(b,3) -> and: {Nb(2). Na(1) XOR i}_ka
and -> (a,6): {Nb(2). Na(1) XOR i}_ka
(a,6) -> i: {Nb.) _ki
i-> (i,17): Nb(2)
i-> (i,17): Nb(2)
% Reached State:
%
% secret(Nb(2),nb,set_66)
% contains(a,set_66)
% contains(b,set_66)
% witness(a,i,bob_alice_na,Na(1))
% secret (Na(1),na,set_70)
% contains(a,set_70)
% contains(i,set_70)
% state_alice(a,i,ka,ki,2,Na(1),Nb(2),set_70,6)
%state_bob(b,a,kb,ka,1,Na(1) XOR
b,Nb(2),set_66.3)
%
state_alice(a,b,ka,kb,0,dummy_msg,dummy_non
ce,set_57,3)
% witness(b,a,alice_bob_nb,Nb(2))
% wrequest(a,i,alice_bob_nb,Nb(2),i)
```

Explanation of the result:
The result of testing the protocol with AVISPA indicates the presence of an attack (ATTACK_FOUND) and the non-fulfillment of the confidentiality property of the element Nb (secrecy_of_nb). Let's interpret the results in detail:

- The tested protocol is in the Diffie_Hellman.hlpsl file.
- The objective required for verification is the confidentiality of the element Nb (secrecy_of_nb).
- The OFMC algorithm was used as an algorithm for analysis.
- An attack (ATTACK_FOUND) was found, which means that there is an attack that can be carried out in the protocol.
- Detailed information was presented about the context of the attack, which is a series of messages and interactions between the participants of the protocol.
- The situation reached during the attack is clarified, and shows the existence of confidential information NB and Na.

In short, there is a vulnerability in the Diffie-Hellman protocol that allows an attacker to expose and manipulate the value of Nb. This means that the protocol is not secure and does not achieve the confidentiality of important elements. The protocol should be reviewed and improved to correct this vulnerability and ensure its integrity.

The results indicate that the protocol is not secure. An attack was found to be a secret breach.

The attack is carried out as follows:

1. The hacker sends a "start" message to Alice.
2. Alice responds with a message encrypted with the hacker's public key.
3. The hacker sends a message to Bob encrypted with Bob's public key.
4. Bob responds with a message encrypted with Alice's public key.
5. The intruder sends the message he received from Bob to Alice.
6. Alice responds with a message encrypted with the hacker's public key.
7. The hacker can now decrypt the message and obtain the secret key.

This means that the hacker can interfere with the communication between Alice and Bob and obtain the secret key. Therefore, the protocol must be modified to prevent this type of attack.

4.2.3 Test result with Tamarin-prover as follows:

Summary of summaries:

analyzed: Diffie_Hellman.spthy

can_be_run (exists-trace): verified (11 steps)

man_in_the_middle (all-traces): falsified - found trace (11 steps)

Tamarin-prover is a symbolic analysis tool for security protocols[1]. It can validate protocols and look for potential vulnerabilities. The result you received indicates that the tested protocol (Diffie_Hellman protocol) has been parsed.

Other details in the result include:

1. can_be_run (exists-trace): verified (11 steps): This indicates that the protocol can run, and this has been verified. The statement "exists-trace" indicates that the protocol can run if there is at least one path that protocol 2 can follow. The number "11" indicates the number of steps taken to verify this[2].
2. man_in_the_middle (all-traces): falsified - found trace (11 steps): This indicates that the protocol is not safe against "man-in-the-middle" attacks. The phrase "all-traces" indicates that the protocol should be safe against "man-in-the-middle" attacks in all possible paths. However, a path was found that the attacker could use to carry out a "man-in-the-middle" attack, thus confirming that the protocol was not secure.

The results of the test using the three techniques are shown in Table -2-:

Table 2 - Test results using the three techniques:

Protocol	ProVerif	AVISPA	Tamarin
Needham-Schroeder Public Key	unsafe	unsafe	safe
Diffie-Hellman Key Exchange	safe	unsafe	unsafe

4.2.4 Evaluation of the Formal Methods Used:

In this research, a comparison was drawn between the advantages of each of the tools, and a table was developed to single out the advantages of each of these tools as in Table (3):

Table 3 - Comparison of protocol security analysis tools

Comparative Point	Tamarin Tool	AVISPA Tool	Proverif Tool
Method of Work	Semi-automatic	Automatic	Automatic
Complexity	Relatively easy to use	Rather difficult to use	Relatively easy to use
Prerequisites to Use	Familiarity with Maude's language	Deep knowledge of the analyzed protocols and HLPSL programming language	Familiarity with Horn clauses or pi-calculus
Reliability	Detects from both active and passive attacks	Detects from both active and passive attacks	Detects passive attacks
Usability	Provides detailed explanations of attacks	Can be used to prove a protocol flawed	Provides step-by-step traces of attacks
Method of Analysis	Analyzes message sequences in isolation	Analyzes all messages simultaneously	Analyzes message sequences in isolation
Efficiency	Suitable for protocols with moderate complexity	Efficient for complex protocols	Suitable for protocols with moderate complexity
Strengths	Ease of use, detailed explanations of attacks	Ability to detect a wider range of attacks	Efficient analysis of message sequences
Weaknesses	May miss subtle attacks	Requires more expertise in HLPSL	Less effective for complex protocols
Recommended Use	Suitable for protocols with moderate complexity and specific attack detection	Ideal for complex protocols that require comprehensive security analysis	Suitable for protocols with moderate complexity and specific attack detection

With Table 3- Tamarin provides ease of use and detailed explanation of attacks, AVISPA excels at analyzing complex protocols, and Proverif excels at analyzing message sequences efficiently.

5. LIMITATIONS OF PROTOCOL SECURITY ANALYSIS TOOLS:

Each protocol security analysis tool has a set of limitations that can affect its ability to properly test the protocol. Here are some of these limitations:

1. Language limitations: Some tools require the use of a specific sample language to describe the protocol. For example, ProVerif requires the use of Horn clauses or pi-calculus.
2. Model limitations: Some tools impose limitations on the protocol model that can be described. For example, Tamarin restricts the number of protocol participants and the number of messages that can be sent.
3. Attack limitations: Some tools focus on detecting only certain types of attacks. For example, AVISPA focuses on detecting active attacks, while ProVerif focuses on detecting passive attacks.
4. Efficiency limitations: Some tools may be slow or ineffective in detecting attacks.

5.1 How to avoid restrictions:

Some limitations can be avoided by choosing the right tool for the type of protocol being tested. For example, if the protocol has a large number of participants, a tool such as Tamarin that restricts the number of participants cannot be used.

In some cases, restrictions can also be avoided by modifying the protocol, which means modifying it to improve security, not just for testing purposes. In some cases, it may be necessary to modify the protocol to avoid certain types of attacks that the tool cannot handle or cannot provide advice to avoid. This does not mean that the protocol is modified just for testing, but it is modified to improve the overall security of the protocol.

5.2 Choosing the right tool:

When choosing a tool for protocol security analysis, consider the following factors:

1. Protocol type: The tool that supports a typical language must be chosen suitable for the type of protocol being tested. When choosing a tool for protocol security analysis, the language supported by this tool should be suitable for the type of protocol being tested. This means that there is a compatibility between the language of the tool and the type of protocol designed according to it. For example, if the protocol relies on a specific language or follows a specific pattern in part communication, the tool you choose should be able to understand and analyze this type of language or pattern effectively.

2. Model limitations: You must choose a tool that does not impose unnecessary restrictions on the protocol model.
3. Types of attacks: You must choose the tool that supports detecting the types of attacks you are interested in.
4. Tool efficiency: You should choose the tool that provides the right efficiency for your needs.

5.3 How can I choose one of the three tools to test a protocol?

When choosing one of the three tools to test a protocol, consider the following factors:

1. Protocol complexity: Each tool has its capabilities in dealing with protocols of different complexities. For example, Tamarin is suitable for medium-complexity protocols, while AVISPA is suitable for high-complexity protocols.
2. Ease of use: Protocol security analysis tools differ in their ease of use. For example, ProVerif is one of the easiest to use, while AVISPA is more difficult to use.
3. Required features: The required features of the protocol security analysis tool must be selected before selecting. For example, if you need a tool that can detect active and passive attacks, you should choose Tamarin or AVISPA.

6. RECOMMENDATIONS:

Based on the findings of this study, we propose the following recommendations to enhance the verification process for security protocols and improve the overall security of modern systems:

1. Selecting the Appropriate Verification Tool

- **For highly complex protocols**: We recommend using Tamarin, as it excels in handling intricate security protocols and detecting active attacks, such as Man-in-the-Middle (MitM) attacks. However, Tamarin requires proficiency in the Maude modeling language, which may limit its accessibility to non-experts.
- **For simple to moderately complex protocols**: ProVerif is a suitable choice due to its ease of use and effectiveness in detecting passive attacks, such as eavesdropping attacks. However, it may be less effective in analyzing highly complex protocols.
- **For comprehensive verification**: AVISPA is recommended when a global analysis of all protocol messages is required, especially when detecting multiple attack types. Nonetheless, its reliance on HLPSL makes it challenging for users unfamiliar with formal specification languages.

2. Improving Verification Methodologies

- **Tool Integration**: A hybrid verification approach that combines the strengths of multiple tools could enhance security assessments. For instance, Tamarin could be used for detecting active attacks, while ProVerif could focus on passive attack detection in parallel.
- **Simplifying the Modeling Process**: Developing user-friendly interfaces for these tools can reduce dependency on specialized knowledge in mathematical modeling, making formal verification more accessible to non-expert users.

3. Developing Next-Generation Verification Tools

- **Leveraging Artificial Intelligence**: AI-powered machine learning algorithms can be integrated into security verification tools to automatically detect attack patterns and predict potential future threats.
- **Enhancing Computational Efficiency**: Optimizing existing tools or developing new verification **frameworks** that reduce computational complexity can allow for the efficient analysis of highly intricate security protocols.

4. Expanding Protocol Testing Scope

- **Broader Protocol Coverage**: Future studies should extend the evaluation to include modern security protocols, such as those used in Internet of Things (IoT) applications and blockchain-based systems.
- **Real-World Environment Testing**: Conducting real-world simulations of protocol behavior under network constraints (e.g., bandwidth limitations, latency issues) would provide a more practical assessment of their security robustness.

5. Training Developers and Researchers

- **Workshops and Training Programs**: Organizing formal training sessions on security verification tools can help bridge the gap between theoretical knowledge and practical application.

- **Educational Resources**: Creating comprehensive learning materials, such as user guides, interactive tutorials, and video demonstrations, can improve accessibility and facilitate wider adoption of formal verification methodologies.

7. COMPARISON WITH PRIOR WORK:

Although numerous previous studies have examined security protocol verification using tools such as Tamarin, AVISPA, and ProVerif, this study introduces several new contributions and improvements compared to prior research. Below, we discuss the key differences between our work and existing studies, highlighting the advantages and limitations of our methodology.

1. Comprehensive Comparison of Tools

Most prior studies focused on evaluating a single tool or comparing only two tools. For example, Yang et al. (2022) [2] focused on evaluating ProVerif's performance in verifying cryptographic protocols, while Arapinis et al. (2014) [9] compared Tamarin and AVISPA in detecting active attacks. In contrast, our study provides a comprehensive comparison of three major verification tools (Tamarin, AVISPA, and ProVerif) with a detailed analysis of each tool’s performance in verifying two well-known security protocols: Needham-Schroeder Public Key Protocol and Diffie-Hellman Key Exchange (DHKE) Protocol. This broader approach enables a more precise identification of the strengths and weaknesses of each tool.

2. Analysis of Different Attack Types

Many previous studies focused on detecting only one type of attack, such as active or passive attacks. For instance, Denning et al. (2024) [1] focused on detecting replay attacks using AVISPA, while Just et al. (2005) [20] explored passive attacks using ProVerif. In our study, we analyze the tools’ ability to detect various attack types, including active attacks (e.g., Man-in-the-Middle (MitM) attacks) and passive attacks (e.g., eavesdropping attacks). This holistic analysis provides a broader understanding of each tool's effectiveness in addressing diverse security threats.

3. Improved Verification Methodology

Some previous studies relied on limited verification methodologies, such as using simple mathematical models or evaluating only basic protocols. For example, Bresson et al. (2002) [19] focused on key exchange protocols using simple mathematical models, while Lowe (1995) [18] analyzed security protocols with limited complexity. In our study, we enhance the verification methodology by employing advanced mathematical models and evaluating complex protocols such as Needham-Schroeder and Diffie-Hellman. Additionally, we apply a unified approach to presenting results, ensuring a more accurate comparison between the tools.

4. Advantages and Limitations of Our Methodology

Advantages:

- **Comprehensive Analysis** – Our study covers a wide range of attacks and protocols, providing a thorough evaluation of tool performance.
- **Standardized Results** – The use of a unified approach for presenting results allows for more precise tool comparisons.
- **Practical Recommendations** – The study provides clear guidelines for researchers and developers on how to select the most suitable verification tool based on protocol characteristics and attack types.

Limitations:

- **Modeling Complexity** – The use of advanced mathematical models may make the study more complex for non-specialist users.
- **Time and Computation Costs** – Verifying complex protocols requires significant time and computational resources, which may limit the study’s applicability on a larger scale.

8. CONCLUSION:

This study addressed the key challenges in security protocol verification, focusing on the complexities of modern protocols and the diverse range of attacks they may encounter. Through a comprehensive comparative analysis of three major formal verification tools (Tamarin, AVISPA, and ProVerif), we assessed their effectiveness in verifying two widely used security protocols: the Needham-Schroeder Public Key Protocol and the Diffie-Hellman Key Exchange (DHKE) Protocol.

Our findings highlight the need for improvements in existing verification tools to enhance their effectiveness in detecting various attack types, particularly in complex security protocols. Specifically, our results indicate that:

- Tamarin excels in detecting active attacks such as man-in-the-middle (MitM) attacks.
- ProVerif demonstrates superior capability in identifying passive attacks such as eavesdropping attacks.
- AVISPA provides a broad, high-level analysis, making it suitable for general security assessments but less effective for highly intricate protocols.

However, each tool has inherent limitations. Tamarin's computational complexity makes it resource-intensive, AVISPA requires expertise in HLPSL modeling, and ProVerif may struggle with highly complex protocol structures.

Implications and Future Work

These findings have important practical implications for security researchers and protocol developers:

1. Tool Selection – This study provides clear guidelines on choosing the appropriate verification tool based on protocol complexity and attack type.
2. Enhancing Verification Tools – There is a growing need to develop more flexible and user-friendly tools that can handle complex security protocols while minimizing reliance on specialized mathematical knowledge.

For future research, we recommend:

- Expanding protocol testing – Future studies should evaluate additional security protocols, including those used in IoT and blockchain applications.
- Improving verification methodologies – Enhancing efficiency and accuracy through hybrid verification models that integrate multiple tools.
- Exploring AI-driven approaches – Leveraging machine learning to automate attack detection and improve verification scalability.

Final Remarks

Ultimately, our findings confirm the research hypothesis:

No single verification tool is universally superior. While Tamarin is highly effective in detecting active attacks, ProVerif excels in passive attack identification, and AVISPA offers a broad-spectrum security analysis. These findings underscore the need for a hybrid approach in security protocol verification, where multiple tools are leveraged to achieve comprehensive and reliable security assessments.

9. FUTURE WORK:

- It is proposed to test more protocols in the future to increase confidence in tools and protocols.
- It is proposed to test protocols in the future in new ways, based on artificial intelligence.

Abbreviations

AKE	Authenticated Key Exchange
AKA	Authentication and Key Agreement
AVISPA	Automated Validation of Internet Security Protocols and Applications
EPS	Evolution Packet System
HLPSL	High-Level Protocol Specification Language
HN	Home network
NRL	NRL protocol analyzer
NSPK	NSPK Protocol
OFMC	On-the-fly Model Checker
PFS	Perfect forward secrecy
PCS	Post Compromise Secrecy
SATMC	SAT-based Model Checker
SN	Serving Network
STP	signaling transport points
UE	user equipment

Compliance with Ethical Standards: The accuracy of the information in the manuscript rests with the writers. All ethical guidelines about scientific research were adhered to in the conduct of this investigation. The relevant ethical review committee gave its approval, and each study participant gave their consent. There was compliance with all relevant laws and regulations.

Competing Interests: There are no pertinent financial or non-financial interests that the authors need to disclose. Regarding the content of this article, the writers state that they have no competing interests. By signing this form, each author attests that they have no relevant connections to, or engagement with, any organization or institution that may have a financial or non-financial interest in the topics or materials covered in this book. The writers don't own any proprietary or financial stake in any of the content covered in this post.

REFERENCES:

[1] D. E. Denning, "Timestamps in Key Distribution Protocols," Communications of the ACM, Retrieved from Academia.edu, 2024.

[2] D. Feng and K. Yang, "Concretely Efficient Secure Multi-Party Computation Protocols: Survey and More," Security and Safety, vol. 1, 2022, DOI:10.1051/sands/2021001.

[3] M. Palombo, "Formal Methods for Security Protocol Analysis: Challenges and Limitations of Automated Tools," University of South Florida, Retrieved from digitalcommons.usf.edu, 2015.

[4] T. M. Chau Le, X. T. Pham and V. Thinh Le , "Advancing Security Protocol Verification: A Comparative Study of Scyther, Tamarin," Journal of Technical Education Science, vol. 19, no. 1, 2024, pp. 43-53, DOI:10.54644/jte.2024.1523.

[5] “Automated Validation of Internet Security Protocols and Applications”, AVISPA Project Page,2023.

[6] A. Hassan, I. Ishaq and J. Minilla " Automated verification tools for cryptographic protocols”, International Conference on Promising Electronic Technologies (ICPET), 2021.

[7] A. H. Shinde and A. J. Umbarkar, "Analysis of Cryptographic Protocols AKI, ARPKI and OPT using ProVerif and AVISPA," International Journal of Computer Network and Information Security, vol. 8, no. 3, 2016, p. 34.

[8] P. B. Copet and R. Sisto, "Automated Formal Verification of Application-Specific Security Properties," Engineering Secure Software and Systems Conference Paper, Retrieved from SpringerLink, 2024.

[9] M. Arapinis, "Stateful Applied Pi Calculus," in Principles of Security and Trust, edited by M. Abadi and S. Kremer, vol. 8414, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, ISBN: 978-3-642-54791-1.

[10] B. Blanchet, "The Security Protocol Verifier ProVerif and its Horn Clause Resolution Algorithm," HCVS/VPT@ETAPS, 2022, pp. 14-22.

[11]"Lecture 8a Reasoning with Horn Clauses - Stanford University," 24th Euro micro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), Retrieved from Stanford University, 2024.

[12] "Decidable First-Order Fragments of Linear Rational Arithmetic with Uninterpreted Predicates," Journal of Automated Reasoning, vol. 65, 2021, pp. 357-423.

[13] Tamarin Prover Project Page, "The Tamarin Prover Project Page provides general information about the project and the tool, including what it was developed for and how to use it," 2023.

[14] "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," Journal of Technical Education Science, vol. 19, no. 1, 2024, pp. 43-53, DOI:10.54644/jte.2024.15232.

[15] "Using Encryption for Authentication in Large Networks of Computers," Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), vol. 7, no. 2, 2024, pp. 3-193.

[16] "Timestamps in Key Distribution Protocols - ACM Digital Library," Communications of the ACM, vol. 63, no. 8, 2020.

[17] "Authentication Revisited: A Structured Approach to Design and Evaluation," Journal of Technical Education Science, vol. 19, no. 1, 2024, pp. 43-53, DOI:10.54644/jte.2024.15232.

[18] G. Lowe, "An Attack on the Needham-Schroeder Public Key Authentication Protocol," Information Processing Letters, vol. 56, no. 3, Nov. 1995, pp. 131-136, DOI:10.1016/0020-0190(95)00144-2.

[19] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions," in Advances in Cryptology - EUROCRYPT 2002, pp. 321-336.

[20] M. Just, "Diffie–Hellman Key Agreement," in Encyclopedia of Cryptography and Security, edited by H.C.A. van Tilborg, Springer, 2005, , pp. 321-336.

[21] W. Jirakitpuwapat and P. Kumam, "The Generalized Diffie-Hellman Key Exchange Protocol on Groups," in Econometrics for Financial Applications, 2018, pp. 115-119.