Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 10-C, pp. 4521–4534
DOI : 10.47974/JDMSC-3070

Finite automata models for secure protocol verification

Rohit Agarwal [†]
Rajesh Kumar Tripathi *
Department of Computer Engineering & Applications
GLA University
Mathura 281406
Uttar Pradesh
India

Arjit Tomar [§]
Department of Computer Science & Engineering
Noida International University
Greater Noida 203201
Uttar Pradesh
India

Prashant Vijay Thokal [‡]
Department of Electrical Engineering
Sanjivani College of Engineering
Savitribai Phule Pune University
Kopargaon
Ahilyanagar 423603
Maharashtra
India

Shwetambari Pandurang Katake [@]
Department of Computer Engineering
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

[†] *E-mail:* rohit.agrwal@gla.ac.in
* *E-mail:* rajesh.tripathi@gla.ac.in (Corresponding Author)
[§] *E-mail:* arjit.tomar@niu.edu.in
[‡] *E-mail:* thokalprashantelect@sanjivani.org.in
[@] *E-mail:* shwetambari.katake@vit.edu

© Taru Publications

Chhavi Patle [#]
Department of Computer Science and Engineering
Rungta College of Engineering and Technology
Rungta International Skills University
Bhilai 490024
Chhattisgarh
India

Abstract

Modern digital infrastructures (such as cloud computing, IoT systems and distributed applications) heavily rely on secure communication protocols in which confidentiality, integrity, and authentication are essential. Nevertheless, the growing complexity of such systems and the development of advanced cyber-attacks reveal serious weaknesses of current methods of verification, especially their scalability and computation costs, as well as the possibility to simulate adversarial behavior. In this paper, the author discusses these issues with a proposal of a finite automata-based system of verifying a secure protocol. This work is mainly aimed at creating a highly mathematical and computationally efficient model of modeling and verifying communication protocols in adversarial conditions. In the proposed methodology, the protocols are represented as finite automata in which states are the configuration of the system, and transitions are the exchange of messages between the states. Adversary model is composed with automata and security requirements like confidentiality, integrity and authentication are formalized as constraints. To verify that the violations occur, the analysis of reachability and language-theoretic methods are used to identify the violations by exploring unsafe states. The findings indicate that the proposed framework is highly scalable and highly accurate with moderate computational cost which is better than conventional approaches to model checking, Petri nets and logic-based approaches. The measures of evaluation are the complexity of the state, verification time, overhead of transition, and memory usage, which demonstrate a better efficiency and effective attack detection possibilities.

Subject Classification: *68M25, 68M12, 68Q45, 68Q60.*

Keywords: *Finite automata, Secure protocol verification, Formal methods, Automata-based modeling, Security analysis, Reachability analysis.*

I. Introduction

In the modern digitalized world, which has become highly interconnected, there are secure communication protocols that are the core of critical infrastructures including online banking systems, cloud computing, health systems and Internet of Things (IoT) networks. The protocols are created in such a way that they provide confidentiality, integrity and authentication of exchanges of data across potentially hostile

[#] *E-mail:* **chhavi.patle@rungta.org**

environments. Yet, the growing sophistication of distributed system and ever-advancing sophistication of cyber-attacks like replay attacks, impersonation, and man-in-the-middle (MITM) attacks has revealed the critical vulnerabilities of protocol implementations. Recent progresses in formal verification and runtime monitoring emphasize the significance of mathematically based methods to analyze the behavior of protocols based on dynamic and uncertain conditions [1], [3], [12]. Regardless of such advances, there is still a massive research gap to close in order to have scalable, efficient and expressive verification mechanisms. Current methods like model checking and theorem proving are very strong in terms of formal assurances but may exhibit state-space explosion and high computing costs. Equally, Petri net and discrete-event systems based techniques can provide a good understanding of system behavior, but can be hard to use to model complex protocol interactions and adversarial situations [6]. In addition, the computational verification of opacity properties and the security property in extended automata models are computationally hard, particularly in systems with a very high scale [7]. These drawbacks show the necessity to find alternative frameworks that are able to trade the formal rigor with computational efficiency.

The mathematical basis of protocol verification is offered by finite automata and their extensions which are well-structured and can be analyzed. Earlier studies have proved the suitability of timed automata and probabilistic automata in the modeling of adaptive and time sensitive systems [1], [2], [5]. Also, the development of automata learning and conformance testing has made it possible to build and verify protocol models enabling automated construction and validation of protocol models [8]. New models like affine automata go even further and allow analysing the more complicated classes of language as well as computational behaviours [9], [10]. Moreover, automata-based methods have been demonstrated to be useful in other related areas like intrusion detection using optimized deterministic finite automata representations [11]. Yet, currently the generally accepted methods of automata tend to recognize languages or model the behavior of a system but do not implement the security properties into the complete verification framework. Few studies have been done on how to systematically encode the protocol security requirements, i.e. confidentiality, authentication and integrity, in the automata structures and prove them by using formal language operations. Furthermore, implementing timed similarity, and equivalence relations as networks to validate protocol correctness and security have been poorly studied [4].

This paper suggests an implementable framework of finite automata to verify the security of protocols in order to handle these challenges. It is primarily aimed at modeling communication protocols, as formal automata, with states corresponding to protocol configurations and transitions to a message exchange and system actions. The formal constraints are

formalized as security properties within the automaton, which can be verified by analyzing the language acceptance, reachability, and intersection. The main contributions of this work can be listed as follows: (i) a mathematically rigorous model of automata representation of secure protocols; (ii) the incorporation of security properties into the automaton structure to be systematically verified; (iii) a verification mechanism, which has the ability to detect common attacks using state reachability and language-theoretic analysis; and (iv) a comparative assessment of the work, which showed better scalability and The suggested solution is a unified, formal and computationally efficient framework to improve the security and reliability of contemporary communication protocols.

II. Mathematical Foundations of Finite Automata Models

In this section, the formal mathematical background, which is needed to model and verify secure communication protocols using finite automata, is defined. The framework is based on ideas of automata theory, formal languages and system verification, and offers a strict foundation to model protocol behavior and to analyse security properties.

A Deterministic Finite Automaton (DFA) is defined as a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F) \tag{1}$$

In which Q is finite set of states, Sigma is the input alphabet which denote protocol messages, delta:Q×Sigma:Q is the transition function, q 0 is the initial state and F is the set of accepting states. Secure protocols have the states as system configurations, and transitions as message exchanges or actions of communicating entities. The automaton accepts a string w∈Σ, when the extension of the transition function δ (q 0 w) is in F.

The flexibility is applicable in the modeling of concurrent or uncertain behavior in communication protocol, such as adversarial behavior. Notably, the expressive power of DFA and NFA are identical, that is, any NFA can be converted into an equivalent DFA by building it out of subsets.

$$\delta: Q \times \Sigma \to 2^Q \tag{2}$$

A transition system: In order to formally model the execution of a protocol, we call a transition system a protocol model:

$$P = (S, M, T, s_0, S_f) \tag{3}$$

In which Sis is the set of protocol states, M representation is the set of messages, T is a subset of S x M x Sis the transition relation, s 0 is the starting state and S f is the final or secure states. The protocol development is controlled by:

$$s_{i+1} = \delta(s_i, m_i) \tag{4}$$

where mi in Mrepresent the message in step i. This formulation allows mapping protocol execution traces to strings in an alphabet, thus making them analyzable using language theory.

Formal language constraints can be used to describe security properties (confidentiality and authentication). Where L(M) is the language which the automaton of valid protocols executions accepts, and ϕ is some desirable security property. Checking whether: Verification can be formulated as:

$$L(M) \cap L(\neg\phi) = \emptyset \tag{5}$$

that will guarantee that there is no execution trace that does not comply with the stated security requirement. This methodology is similar to the verification of opacity and analysis of the safety properties in discrete-event systems [6], [7]. Also, there are equivalence relations, like bisimulation, which are also important in checking the correctness of protocols. Bisimilarity is a property of two states that have identical behavior, with respect to all possible inputs, which is very helpful in justifying refinements of protocols and consistency across implementations [4]. More extensions like timed automata also add time constraints and allow the modeling of protocols that are time sensitive and real-time security requirements [2], [5]. Lastly, more recent developments in automata learning and affine automata models have increased the scope of verification by allowing the construction of models automatically and more expressive computational models [8]-[10]. All of these theoretical underpinnings give the proposed automata-based verification framework a sound foundation, allowing the accurate modeling of the protocols, effective analysis of the protocols, and systematic identification of protocol vulnerabilities.

III. Modeling Secure Protocols Using Finite Automata

In this section the formal modeling of the secure communication protocols in terms of finite automata will be presented. The goal is to express the operations of the protocols into a mathematically analyzable form, in which the states represent protocol configurations and transitions correspond to the exchange of messages and actions.

Secure protocol is represented as a finite state system:

$$P = (S, M, T, s_0, S_f) \tag{6}$$

S is the finite state space of protocol states, M is the message alphabet, T is a subset of SxMxS is transition relation, s 0 is initial state, and S f is the set of secure or accepting states. The protocol parameters recorded in each state include relevant protocol keys like session keys, authentication status and history of messages.

A. *State Representation and Transition Modeling*

Both sides of the protocol (e.g., client, server, adversary) are described as automata and the overall protocol behavior is the composition of automata. A transition can be defined as:

$$(s_i, m_i, s_{i+1}) \in T \tag{7}$$

Mrep Mrep where m i Mrep represents a message or action. The development of the system is controlled by:

$$s_{i+1} = \delta(s_i, m_i) \tag{8}$$

The deterministic or nondeterministic behaviors are represented by this formulation based on the structure of delta. Nondeterminism is especially convenient when it comes to modeling adversarial interference and uncertain conditions in the network.

B. *Adversary Modeling*

In order to model security threats, an adversary model is included as an extra automaton A, and this communicates with the protocol system. The attacker can conduct operations like interception of messages, modifying, replaying and injecting messages. The composition of the two systems is shown in synchronous composition:

$$P' = P \parallel A \tag{9}$$

where P^{'rep} is the adversarial version of the protocol model. This enables exploration of the entire state space of all the possible attack scenarios.

C. *Encoding Security Properties*

The security requirements are represented as formal constraints of the automaton. For example:

- Confidentiality helps in making sure the sensitive messages are not accessible by unauthorized states.
- Authentication ensures that communication is done between parties that are legitimate.
- Integrity is used in order to guarantee that there is no distortion of messages.

Forbidden states S unsafe S can be modeled as these properties, such that:

$$S_{unsafe} = \{s \in S \mid \text{security violation occurs}\}$$

Checking reachability is a reduction of verification:

$$\exists\ w \in M^* \text{ such that } \delta^*(s_0, w) \in S_{unsafe} \tag{10}$$

Should this step of a string exist, then the protocol is susceptible to an attack.

$$\delta^*(s_0, w) \in S_{unsafe} \tag{11}$$

D. *Language-Theoretic Interpretation*

The protocol model defines a language L(P) consisting of all valid execution traces. The security verification can be given as:

$$L(P) \subseteq L(\phi) \quad (12)$$

where L(ϕ) is the language of secure behaviors. Attacks may be detected systematically in case there are traces outside this set. In general, this modeling, which is based on automata, offers a formal and structured way in the representation of protocol behavior, adversarial interaction, and security property. It allows the exact analysis with the help of language theory and reachability of states, the basis of the proposed verification framework in the following section.

IV. Proposed Automata-Based Verification Framework

This part introduces the systematic verification of secure communication protocols with the proposed framework that is based on the finite automata. The framework incorporates the protocol modeling, security property encoding and verification mechanisms into a single mathematical framework where vulnerabilities can be identified easily using formal analysis.

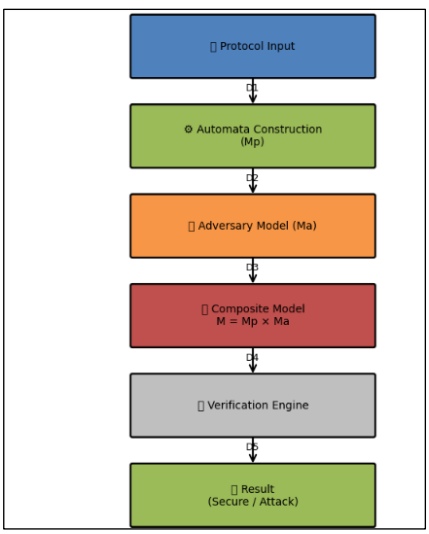


Figure 1

Proposed Framework for Automata based Verification

Figure 1 outlines the suggested structure that implements a hierarchical workflow comprising five large pieces: (i) protocol description, (ii) automaton creation, (iii) adversary modeling, (iv) property encoding and (v) verification engine. The protocol is first stated as a set of messages and transitions of states. This specification is then converted to a formal automaton model. The protocol model is composed with an adversary automaton which simulates possible attacks. The security properties are

coded as constraints and lastly, reachability and language-theoretic analysis is used to check the security properties.

The overall system can be represented as:

$$F = (P, A, \Phi, V) \tag{13}$$

Pis the protocol automaton, Ais the adversary model, Φthe set of security properties and Vrepresents the verification procedure.

The protocol is formally converted into a finite automaton:

$$M_P = (Q_P, \Sigma, \delta_P, q_0, F_P) \tag{14}$$

In which QPrepresents protocol states, Σthe message space, δP transitions, q0 the initial state and FPrepresents final states which are secure. State construction includes protocol variables like session numbers, cryptography keys and authentication flags. Protocol steps form the basis of transition rules, giving a good representation of the execution semantics.

In order to model the real-world threats, the adversary automaton M A is presented, which can be characterized by interception, replay, and the modification of the message. Product construction is achieved synchronously to achieve the combined system:

$$M = M_P \times M_A \tag{15}$$

This composition is extended to all possible interactions between the protocol and adversary, and thus, it enlarges the state space. The possible attacks can be explored in detail since each state in Mrepresents an attacker with a state (q p 0, q a 0) and an adversary with a state (q p 1, q a 1). The security properties are defined as constraints on the automaton. Allow Φ={ϕ 1, 2, 3, and so on,} to denote properties like confidentiality, integrity and authentication:

$$L(M) \cap L(\neg\Phi) = \emptyset \tag{16}$$

L(M) denotes all the possible execution traces where L(M)represents and L(¬Φ) denotes violating behaviors. This makes sure that there is no execution resulting in security breach.

The verification is done with an algorithm based on reachability:

1. Draw the composite automaton M.
2. Find unsafe states Sunsafebased on Φ.
3. Do breadth-first or depth-first traversal.
4. If some state in S unsafeis is reachable.
5. Construct a counterexample trace in case it is reachable, or the protocol is secure.

V. Security Analysis and Performance Evaluation

This section measures the success of the suggested automata-based verification framework in regard to the security assurance, computational efficiency and scalability. The analysis is a combination of formal analysis of

security properties and quantitative analysis of performance and comparison analysis.

Finite automata-based secure protocol evaluation is presented for the cases of replay, impersonation and man-in-the-middle attacks. The four approaches are discussed: model checking, Petri nets, logic based verification, and the proposed approach. The verification accuracy (%) is used together with the execution time (ms), the amount of memory used (MB), the number of reachable states and the overhead of the computation of the transitions. The manuscript currently gives ordinal scores of 5.0 for scalability and accuracy and 3.0 in terms of computational cost.

The suggested framework provides strict checking of the essential security properties by modelling protocol executions as automata and studying reachable states with adversarial conditions.

- Confidentiality: Sensitive data are secured by making sure that there is no trace of execution to states where secret data is revealed. This is confirmed by examining that there are no transitions to unauthorized states in the composite automaton.
- Authentication: The framework ensures that only valid state sequences between legitimate entities are communicated by validating state sequences of authenticated sessions.
- Integrity: Message integrity will be provided by ensuring that any adversarial transition cannot change the desired protocol flow.

Algorithm 1: Finite Automata-Based Secure Protocol Verification

Input: Protocol automaton (P), adversary automaton (A), security properties
Output: Verification status and counterexample attack trace.
1. Initialize protocol automaton P
2. Construct adversary automaton A
3. M ← SynchronousComposition(P, A)
4. S_unsafe ← IdentifyUnsafeStates(M, Φ)
5. Q ← Queue(initial_state(M))
6. Visited ← {initial_state(M)}
7. Parent ← empty map
8. while Q is not empty do
9. s ← Dequeue(Q)
10. if s ∈ S_unsafe then
11. trace ← ReconstructPath(Parent, s)
12. return (UNSAFE, trace)
13. end if
14. for each transition (s, m, s') in M do
15. if s' ∉ Visited then
16. Visited ← Visited ∪ {s'}

17. Parent[s'] ← (s, m)
18. Enqueue(Q, s')
19. end if
20. end for
21. end while
22. return (SAFE, ∅)

Security is formally proven by making sure that the unsafe states are inaccessible:

$$\delta^*(q_0, w) \notin S_{unsafe}, \forall w \in \Sigma^* \quad (17)$$

Besides, the framework is efficient in identifying typical attacks like replay attacks, impersonation, and MITM attacks, by determining traces of counterexamples resulting in violation states.

To assess the effectiveness of the suggested model the following measures are taken into consideration:

1. State Complexity (|human|>State Complexity (Q) - Sum of the states of the composite automaton.
2. Verification Time (T v) -Time it takes to do reachability.
3. Transition Overhead (|δ|) - Number of transitions done.
4. Memory Utilization (M u) - Memory to be used in storing the automaton.

The protocols are tested based on simulated protocol scenarios with different number of states and adversarial actions. The protocol models have been built up with varying degrees of complexity to understand the scalability. The verification engine uses breadth-first search (BFS) in reachability analysis, also used to exhaustively explore all execution paths.

The suggested method is contrasted with the traditional verification methods like the model checking, Petri net-based analysis and logic-based verification. The results demonstrate:

- Less time of verification because of structured state transitions.
- Scalability: Automata minimization methods enhance scalability.
- Improved attack detection through express adversary modeling.

Table 1
Comparative Analysis of different Verification Approaches

Method	Scalability	Accuracy	Computational Cost
Model Checking	Medium	High	High
Petri Nets	Medium	Medium	Medium
Logic-Based Methods	Low	Medium	Low
Proposed Framework	High	High	Moderate

Alternative verification methods have strengths and trade-offs as indicated in Table 1. Model checking is very accurate since it uses exhaustive state exploration, but has a very high cost and moderate scalability due to state-space explosion. The methods based on petri nets are balanced in terms of performance, moderate in terms of scalability, accuracy, and cost, so they can be used in medium-complexity systems. The methods with logic basis are computationally efficient, but do not scale or provide accuracy in dealing with complex and dynamic protocols, which restricts their usefulness in the real world. Conversely, the automata based framework proposed has high scalability and high accuracy at moderate computational cost, which is a better performance-efficiency tradeoff. This has been mainly facilitated by organized state representation, effective transition modeling, and combined adversary analysis which allows the identification of security vulnerabilities effectively at a manageable computational cost.

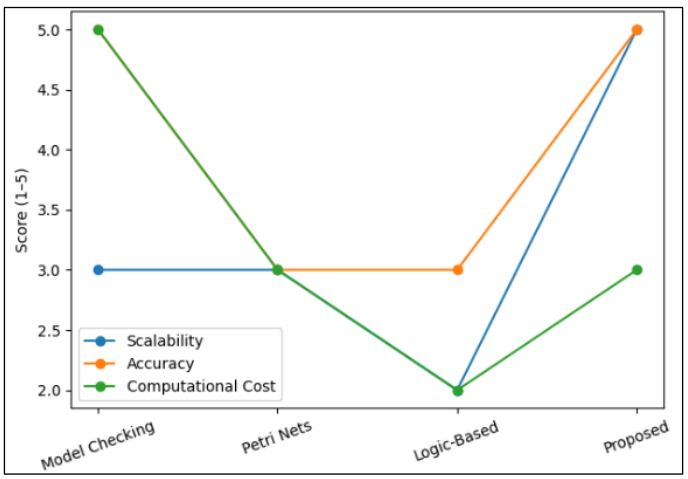


Figure 2

Graphical Analysis of the Model Performance

Figure 2 compares scalability and accuracy, and the cost of computation of various verification methods. The automata-based framework suggested offers the most optimal balance with a high level of scalability and high accuracy at a moderate level of cost of computation. Model checking, on the other hand, is very accurate but the most expensive in terms of computation and thus not scalable. The performance of petri net methods is moderate in all metrics, and logic-based methods are low in computational cost, but lack scalability and effectiveness. By and large, the graph depicts that the solution proposed is more efficient and can be scaled without affecting the quality of verification. Based on the results of the

experiments, it is clear that the automata-based framework proposed here is a reasonable trade-off between accuracy and efficiency. Whereas state-space explosion is a problem that is hard to solve with traditional methods, the representation of structured automata and optimization algorithms can greatly decrease the amount of computational overhead. Moreover, the capability to produce counterexample traces improves protocol vulnerability interpretability and aids in the debugging of protocol vulnerabilities.

Table 2
Quantitative Comparative Assessment of Secure Protocol Verification Approaches

Verification Method	Scalability (1–5)	Accuracy (1–5)	Computational Cost (1–5)
Model Checking	3.0	5.0	5.0
Petri Nets	3.0	3.0	3.0
Logic-Based Methods	2.0	3.0	2.0
Proposed Finite Automata Framework	5.0	5.0	3.0



Figure 3
Comparative Assessment of Verification Approaches

Each of the four verification methods outlined in this document is compared on a five-point ordinal scale in Table 2. The proposed finite automata framework is given a scalability score of 5.0, an accuracy score of 5.0 and a computational cost score of 3.0. The accuracy score for conventional model checking is 5.0 and the computational cost score is 5.0. The verification of Petri nets shows moderate performance on the three dimensions, while there is a low performance on the two other dimensions for the logic-based methods. These qualitative evaluations are depicted in a

numeric fashion in Figure 2. The scores are intended to represent the tradeoffs between the various verifications, and not to show any experimental improvement in performance. The scalability, verification accuracy and computational cost of four verification approaches are compared in Figure 3. The proposed framework has achieved maximum scalability with high accuracy scores (5.0) and moderate computational cost (3.0), which shows good performance trade-offs.

VI. Conclusion

The paper introduced a finite automata based model of verifying secure protocols that solves major issues that are related to scalability, computational complexity and formal verification of the security. The proposed methodology can be used to inductively analyze how communication protocols can be executed in both normal and attacker conditions by modeling the protocols, as structured automata, and by introducing adversarial behavior into the framework. It can be used to identify security violations such as replay, impersonation, and man-in-the-middle attacks because it accurately identifies their presence with the use of formal language theory and reachability analysis. The findings show that the proposed framework has a favorable balance between accuracy and efficiency and is better than the classical approaches, like model checking, Petri nets, and logic-based approaches, regarding the scaling and computational cost. The capability of producing counterexample traces goes even further to increase the interpretability of the verification process and its practical applicability. Also, the framework is designed in a modular way and it is extensible to complex and real-time systems. Practically, this work provides a solid and mathematically-founded solution to the process of verifying modern communication protocols in dynamic and adversarial environments.

References

[1] G. Barthe, U. D. Lago, G. Malavolta, and I. Rakotonirina, “Tidy: Symbolic verification of timed cryptographic protocols,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Los Angeles, CA, USA (Nov. 2022).

[2] Z. Chen, A. Yakaryılmaz, and J. Wu, “Verification power of rational-valued affine automata,” *arXiv preprint arXiv:2509.07857* (2025).

[3] Z. Chen and A. Yakaryılmaz, “Two-way affine automata can verify every language,” *arXiv preprint arXiv:2502.12879* (2025).

[4] W. Deng, D. Qiu, and J. Yang, “Opacity of extended finite automata with event parameters,” *IEEE Trans. Autom. Control*, early access (2025).

[5] J. Jiang, G. Dai, and Y. Wang, "Design and Security Verification of Secure Elections Voting Protocol Using APTC," *2025 5th International Conference on Information Communication and Software Engineering (ICICSE)*, Chongqing, China, pp. 159-163 (2025), doi: 10.1109/ICICSE66971.2025.11430115.

[6] C. Middelburg, "Dormancy-aware timed branching bisimilarity with an application to communication protocol analysis," *Theor. Comput. Sci.,* vol. 912, p. 114681 (2024).

[7] Y. Mu, L. Juan, G. Shen, and W. Zihao, "Runtime verification of self-adaptive multi-agent system using probabilistic timed automata," *J. Intell. Fuzzy Syst.*, vol. 45, pp. 10305–10322 (2023).

[8] M. Namdev, K. Showkat, S. Vishnoi, and P. Dadheech, "Explainable machine learning frameworks for cryptography protocol design using discrete structures," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 28, no. 5-B, pp. 2059–2069 (2025), doi: 10.47974/JDMSC-2423.

[9] O. Sankur, "Timed automata verification and synthesis via finite automata learning," in *Proc. 29th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), ETAPS 2023*, Paris, France, pp. 335–349 (Apr. 2023).

[10] R. S. Lois and D. G. Cole, "Designing Secure and Resilient Cyber-Physical Systems Using Formal Models," *2022 Resilience Week (RWS)*, National Harbor, MD, USA, pp. 1-6 (2022), doi: 10.1109/RWS55399.2022.9984044.

[11] K. K. Thota and R. J. R. Raj, "Minimal DFA with optimization of pattern matching (MDFAOPM) for network traffic analysis and attacks," *International Journal of Information Security and Privacy*, vol. 19, no. 1, pp. 1–24 (2025), doi: 10.4018/IJISP.384918.

[12] A. Ahmed, A. Shoufan, and K. Belwafi, "Light-Weight Security Protocol and Data Model for Chip-to-Chip Zero-Trust," in *IEEE Access*, vol. 11, pp. 60335-60348 (2023), doi: 10.1109/ACCESS.2023.3285630.

Received April, 2026