Cryptographic Security through Kleene's Theorem and Automata Theory

Mike Wa Nkongolo[1]
mike.wankongolo@up.ac.za

University of Pretoria, Department of Informatics
South Africa

Abstract. This study addresses the challenge of strengthening cryptographic security measures in the face of evolving cyber threats. The aim is to apply Kleene's Theorem and automata theory to improve the modeling and analysis of cybersecurity scenarios, focusing on the CyberMoraba game. Representing the game's strategic moves as regular expressions and mapping them onto finite automata provides a solid framework for understanding the interactions between attackers and defenders. This approach helps in identifying optimal strategies and predicting potential outcomes, which contributes to the development of stronger cryptographic security protocols. The research advances the theoretical use of automata theory in cybersecurity while offering practical insights into enhancing defense mechanisms against complex cyber attacks. This work connects theoretical computer science with practical cybersecurity, demonstrating the importance of automata theory in cryptology.

Keywords: Cryptography · Automata Theory · Cybersecurity · Kleene's Theorem · CyberMoraba · Finite Automata · Cyber Attack Modeling · Theoretical Computer Science.

1 Introduction

In the face of evolving cyber threats, the challenge of strengthening cryptographic security measures is becoming increasingly critical [1]. Cryptographic systems, which rely on complex algorithms and protocols, must be robust against sophisticated attack vectors [2]. This requires not only stronger algorithms but also a mathematical understanding of how these threats interact with defensive mechanisms [1], [2]. To address this challenge, we turn to the rigorous frameworks offered by automata theory [3] and Kleene's Theorem [4]. Kleene's Theorem is a fundamental result in automata theory, which establishes a deep connection between regular expressions and finite automata [3], [4]. Formally, the theorem states that the set of languages recognized by finite automata is exactly the set of languages that can be described by regular expressions [4]. Specifically, let L be a language over an alphabet Σ. Then:

$$L \text{ is regular} \iff \exists \text{ a DFA } M \text{ such that } L = L(M),$$

where $L(M)$ denotes the language recognized by the deterministic finite automaton M. This theorem implies that any process, including sequences of actions in cybersecurity, can be represented both as a regular expression and as a finite automaton. In the context of cybersecurity, particularly in the strategic game CyberMoraba [5], this provides a structured way to model interactions between an attacker and a defender. The game is represented by a series of strategic moves between two players: the attacker, A, and the defender, D (see Table 1). Each player's move can be represented by an element of the set Σ, where Σ is the alphabet of possible actions (e.g., sending a malicious email, applying a security patch). The sequence of moves in a game forms a string over Σ, which can be analyzed using the tools of automata theory. By representing the game's strategic moves as regular expressions and mapping them onto finite automata, this approach provides a rigorous mathematical framework for understanding the dynamic interactions between attackers and defenders [5]. Moreover, it enables the identification of optimal strategies [6] and the prediction of potential outcomes, thereby contributing to the development of more robust cryptographic security protocols. In tandem with the above, this study proposes the methodology of applying Kleene's Theorem and automata theory to model and analyze cybersecurity scenarios within the context of CyberMoraba. This approach provides a mathematical framework to optimize cryptographic security measures. The study addresses the following research questions:

1. **RQ1**: How can Kleene's Theorem be applied to enhance the modeling of cybersecurity scenarios?
2. **RQ2**: What are the benefits of using finite automata to represent and analyze strategic interactions in CyberMoraba?
3. **RQ3**: How does the application of automata theory improve the prediction of outcomes and optimization of strategies in cybersecurity games?
4. **RQ4**: What practical insights can be gained from modeling CyberMoraba for strengthening cryptographic security protocols?

The study makes three major contributions in addressing these questions:

1. It demonstrates how Kleene's Theorem and automata theory can be applied to model complex cybersecurity scenarios.
2. It provides a framework for using finite automata to analyze and optimize strategic interactions in the CyberMoraba game.
3. It offers practical insights into improving cryptographic security protocols through theoretical modeling and analysis.

2 Background

Prokopev [7] provides a practical automated approach to testing cryptographic protocols using advanced state machine and declarative languages.

Table 1. The Attacker and Defender Tokens

Token	**Image**	**Definition**	**Token**	**Image**
A1	Email	Malicious e-mail	D1	Denying
A2	Phone	Malicious phone call	D2	Network monitoring
A3	Chat	Malicious chat	D3	Avoid clicking
A4	Attachment	Malicious attachment	D4	Identification
A5	Donate	Malicious directory	D5	No trust
A6	Password	Malicious password	D6	Upload
A7	Connection	Malicious connection	D7	Trust
A8	Access	Malicious intrusion	D8	Provide
A9	Data	Malicious data	D9	Confidential
A10	Data loss	Data loss process	D10	Report
A11	Click	Malicious link	D11	Social media
A12	Sensitive data	Theft of data	D12	Connection
A13	Message	Malicious communication	D13	Backup

The study emphasizes automation, practical application, and handling complex requirements. In contrast, we focused on the theoretical application of finite automata and Kleene's Theorem for modeling and verifying cryptographic protocols using the CyberMoraba gameplay. While both approaches aim to improve protocol security, they differ in their focus, methodology, and practical applications. Kassa et al. [8] introduced significant advancements in hardware efficiency to secure data transmission at the nano-scale. The study provides practical insights into circuit design and performance using QCA technologies. In contrast, our study emphasize formal verification and mathematical proofs focusing on the theoretical application of finite automata and Kleene's Theorem to cryptographic protocols. While both studies aim to enhance security, they operate in different domains: one in hardware design and the other in theoretical analysis. Our study and the research on Parikh's Theorem proposed by Hague et al. [9] both engage with automata theory and verification. While our research focused on formal analysis and optimization of cryptographic protocols, the Parikh's Theorem research addresses symbolic representations and complexity handling for large or infinite alphabets [9]. Both studies delve into automata theory, though with different focuses:

– **Our study**. Applies finite automata and Kleene's Theorem to model and optimize cybersecurity protocols, focusing on theoretical analysis and formal verification.
– **Parikh's Theorem research**. Utilizes Parikh's Theorem to connect the letter-counting abstraction of languages recognized by finite automata to Linear Integer Arithmetic, with a focus on symbolic representations and handling large or infinite alphabets [9].

Both approaches are relevant to verification tasks. We target the formal verification of cryptographic protocols, enhancing protocol security through automata-based analysis.

Parikh's Theorem [9] applies to the verification of cryptographic protocols using Horn clauses and symbolic automata, assist in the verification process with large or complex data. Both studies address complexity and symbolic representation. We model cryptographic protocols with finite automata, using formal methods and regular expressions for analysis while [9] developed symbolic automata for efficient handling of large or infinite alphabets, providing polynomial-time algorithms for existential formulas and parametric symbolic grammars. Both works emphasize algorithmic approaches and formal methods, our study uses formal proofs and automata theory to analyze protocol behavior and properties, but [9] proposes new algorithms for applying Parikh's Theorem to symbolic automata, focusing on polynomial-time solutions and enhanced efficiency in symbolic representations. We used the CyberMoraba game introduced in [5] as a tool to model secure cryptographic protocols by applying concepts from finite automata to simulate and analyze various protocol states, transitions, and potential vulnerabilities.

3 Game Description

In the CyberMoraba game two players, an attacker and a defender, are assigned a set of tokens representing various strategic actions [5]. The attacker's tokens are labeled $A_1, A_2, \ldots, A_{13}$, and the defender's tokens are labeled $D_1, D_2, \ldots, D_{13}$. The game unfolds on a finite board B, initially empty, with the attacker making the first move. The board B can be mathematically represented as a finite set of states $\{s_0, s_1, \ldots, s_n\}$, where each state s_i corresponds to a possible configuration of token placements.

4 Mathematical Model

Let $\mathcal{T} = \{A_1, A_2, \ldots, A_{13}\} \cup \{D_1, D_2, \ldots, D_{13}\}$ denote the set of all tokens. A move by the attacker or defender is defined as a mapping $m : B \times \mathcal{T} \to B$ where the state of the board transitions from s_i to s_{i+1} under the application of a token. Each move is evaluated by a scoring function $\sigma : B \times \mathcal{T} \to \{0, 1\}$, where $\sigma(s_i, A_j) = 1$ represents an optimal move by the attacker, and $\sigma(s_i, D_k) = 1$ represents an optimal move by the defender. The scoring system is presented in Table 2.

5 Scoring and Strategic Analysis

The objective of each player is to maximize their cumulative score over a sequence of moves [5], [6]. The total score S_a for the attacker and S_d for the defender after n moves is defined as:

$$S_a = \sum_{i=1}^{n} \sigma(s_i, A_j), \quad S_d = \sum_{i=1}^{n} \sigma(s_i, D_k)$$

where $A_j \in \mathcal{T}$ are the tokens played by the attacker, and $D_k \in \mathcal{T}$ are the tokens played by the defender.

6 Strategic Optimization and Nash Equilibrium

We can extend the analysis by considering the game as a sequential decision-making process, where each player aims to optimize their strategy. The strategy space for the attacker and defender can be modeled as a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where:

– Q is a finite set of states representing the possible configurations of the board.
– $\Sigma = \{A_1, A_2, \ldots, A_{13}, D_1, D_2, \ldots, D_{13}\}$ is the input alphabet.
– $\delta : Q \times \Sigma \to Q$ is the transition function dictating the state evolution.
– q_0 is the initial state where the board is empty.
– $F \subseteq Q$ is the set of final states, where the game concludes.

The game's outcome can be analyzed by determining the Nash equilibrium, where neither player can unilaterally improve their score by changing their strategy [6]. Mathematically, this is found by solving the system of inequalities that results from the best-response functions of both players.

Table 2. Instance of the Scoring Scheme

Scenario	**Action**	**Score (Attacker)**	**Score (Defender)**
Email	Zero trust	0	1
Click	Denying	0	1
Chat	Identification	0	1
Phone call	Trust	1	0
Connection	Connection	0	1
Access	Identification	0	1
Data loss	Upload	1	0
Click	Provide	1	0
Total		**3**	**5**

This mathematical modeling of the game provides a rigorous framework for analyzing strategic interactions in cybersecurity scenarios. By representing the game as a finite automaton and applying game theory concepts, one can derive optimal strategies and predict outcomes, thus enhancing understanding and preparedness in real-world cybersecurity contexts.

6.1 Finite Automata Representation

We represent the CyberMoraba game as a universal set U, where U encompasses all possible states and actions within the game. The game's mechanics are modeled using a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where:

- Q is the set of states, corresponding to specific configurations of the cryptographic protocol (e.g., key exchange, authentication).
- Σ is the input alphabet, representing actions or events (e.g., sending a message, applying an encryption key).
- $\delta : Q \times \Sigma \to Q$ is the transition function that dictates state transitions based on inputs.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting (final) states, representing successful protocol completion.

6.2 Application of Kleene's Theorem

In this study, we explore the application of finite automata and Kleene's Theorem to model and analyze cybersecurity protocols. Kleene's Theorem asserts that for a given finite automaton $\mathcal{A}$, the language $L(\mathcal{A})$ recognized by $\mathcal{A}$ can be expressed as a regular expression. This allows us to formalize the behavior of a protocol through regular expressions and validate its properties.

7 Finite Automaton Modeling

Consider a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ used to model a cybersecurity protocol. Here, Q is a finite set of states, Σ is the input alphabet consisting of tokens, $\delta : Q \times \Sigma \to Q$ is the transition function, q_0 is the initial state, and $F \subseteq Q$ is the set of final states. The automaton $\mathcal{A}$ represents the sequence of valid actions in the protocol.

8 Application of Kleene's Theorem

The language $L(\mathcal{A})$ accepted by the automaton $\mathcal{A}$ can be expressed by a regular expression R. Formally,

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \text{transitions in } \mathcal{A} \text{ that leads to a final state}\}.$$

The regular expression R capturing $L(\mathcal{A})$ can be constructed using the transition function δ and the state set Q. To derive R, one can use the following algorithmic approach:

1. ****Construct the Transition Matrix****: Define a matrix T where each entry T_{ij} represents the regular expression for transitions from state q_i to state q_j. This matrix encapsulates the transitions and their regular expressions.

$$T_{ij} = \bigcup_k \text{Regex}_{ik}\text{Regex}_{kj}$$

where Regex_{ik} and Regex_{kj} are regular expressions corresponding to transitions between states q_i and q_k, and q_k and q_j, respectively.

2. ****Apply Kleene's Theorem****: Use Kleene's algorithm to eliminate states and simplify the transition matrix to obtain the regular expression R. This involves iteratively removing states and updating the transition expressions.

3. ****Obtain the Regular Expression****: The resulting regular expression R represents the language $L(\mathcal{A})$, capturing all valid sequences of actions in the protocol.

$$\exists w \in \Sigma^* \text{ such that } \delta^*(q_0, w) \in F$$

where:

- $\delta^*(q_0, w)$ denotes the state reached from the initial state q_0 after processing the string w through the transition function δ.
- F is the set of final states.

This notation signifies that there exists a string w in the set of all possible strings Σ^* such that the state reached by processing w starting from q_0 is a member of the final states F.

9 Formal Verification

To ensure that the protocol modeled by $\mathcal{A}$ meets its security requirements, we must verify that the regular expression R accurately represents all acceptable sequences. This involves checking that:

$$R \text{ is a regular expression such that } R = L(\mathcal{A}) \cap C$$

where:

- $L(\mathcal{A})$ denotes the language recognized by the finite automaton $\mathcal{A}$, representing all valid sequences of actions in the protocol.
- C denotes the set of sequences that satisfy the specified security constraints.
- R is the regular expression that captures all sequences corresponding to valid protocol actions and adheres to these constraints.

This notation signifies that R encompasses all sequences that are both valid in terms of protocol actions and compliant with the specified security constraints. By comparing R against expected patterns (e.g., handshake, data exchange, termination), we can confirm that the automaton correctly models the protocol and meets its security goals. By expressing the language recognized by the automaton as a regular expression, we can rigorously validate the protocol's behavior and ensure its adherence to security requirements. This approach enhances the formal verification process and contributes to the development of secure cryptographic protocols [10], [11], [12].

9.1 Formal Verification of Security Properties

We define a security property P as a subset of Σ^* (the set of all strings over the alphabet Σ). The protocol is secure if:

$$L(\mathcal{A}) \subseteq P \tag{1}$$

This means that every sequence w in the language $L(\mathcal{A})$ must satisfy the security conditions defined by P. If the regular expression R derived from the automaton $\mathcal{A}$ conforms to the security property P, then the CyberMoraba game, as modeled by the finite automaton $\mathcal{A}$, effectively models and optimizes the secure cryptographic protocol. Thus, the game can be used as a tool to validate the security of cryptographic protocols.

9.2 Final Automaton

The final automaton for the game, as derived from the scoring Table 2 models the interactions between the attacker and defender. This automaton captures the scenarios in which the attacker and defender make moves, and it transitions between states based on these moves. Each state represents a particular scenario in the game, and transitions are defined by the actions taken by the players. The automaton can be described as follows:

- **States**: Each state represents a specific game scenario based on Table 1 and Table 2.
- **Alphabet**: Actions (e.g., $A1$, $D5$).
- **Transitions**: Determined by the moves and scoring.
- **Start State**: The initial state before any actions.
- **Accept States**: States where the game outcome meets the winning criteria.

Fig. 1 is the diagram of the final automaton, illustrating the transitions based on the scoring Table 2. The regular expressions corresponding to different game scenarios are derived as follows (Table 1 and Table 2):

- **Email**: $A1D5$ where the attacker uses an email (A1) and the defender applies zero trust (D5).
- **Click**: $A11D4$ where the attacker uses a malicious link (A11) and the defender uses identification (D4).
- **Chat**: $A3D4$ where the attacker uses a malicious chat (A3) and the defender uses identification (D4).
- **Connection**: $A7D12$ where the attacker uses a malicious connection (A7) and the defender suggests secure connection (D12).
- **Access**: $A8D4$ where the attacker uses malicious access (A8) and the defender identifies the malicious access (D4).

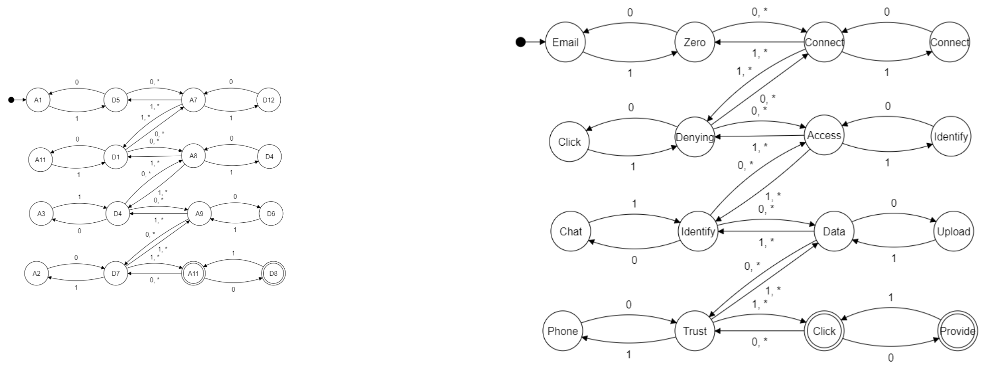


Fig. 1. The automaton diagrams showing the final state machine (FSM) models. The automaton on the left and right illustrate the FSMs for different scenarios in the scoring table.

9.3 Definitions and Notations

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton where:

- Q is the set of states: $Q = \{\text{Email}, \text{Click}, \text{Chat}, \text{Connection}, \text{Access}, \text{Final State}\}$.
- Σ is the set of input symbols representing attacker and defender actions.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- q_0 is the start state: $q_0 = \text{Email}$.
- F is the set of accepting states: $F = \{\text{Final State}\}$.

The finite automaton $\mathcal{A}$ transitions between states based on the input sequences of attacker and defender actions. The transition function δ is defined as follows:

$$\begin{aligned}
\delta(\text{Email}, A1D5) &= \text{Click},\\
\delta(\text{Click}, A11D4) &= \text{Chat},\\
\delta(\text{Chat}, A3D4) &= \text{Connection},\\
\delta(\text{Connection}, A7D12) &= \text{Access},\\
\delta(\text{Access}, A8D4) &= \text{Final State}.
\end{aligned}$$

To demonstrate that $\mathcal{A}$ correctly models the given scenarios, we will show that each transition accurately represents the interactions described in the game scenarios.

– **Email**: For the input $A1D5$ starting from the state Email:

$$\delta(\text{Email}, A1D5) = \text{Click}$$

This transition represents the attacker using email (A1) and the defender applying zero trust (D5).

– **Click**: For the input $A11D4$ starting from the state Click:

$$\delta(\text{Click}, A11D4) = \text{Chat}$$

This transition represents the attacker using a malicious link (A11) and the defender using identification (D4).

– **Chat**: For the input $A3D4$ starting from the state Chat:

$$\delta(\text{Chat}, A3D4) = \text{Connection}$$

This transition represents the attacker using a malicious chat (A3) and the defender using identification (D4).

– **Connection**: For the input $A7D12$ starting from the state Connection:

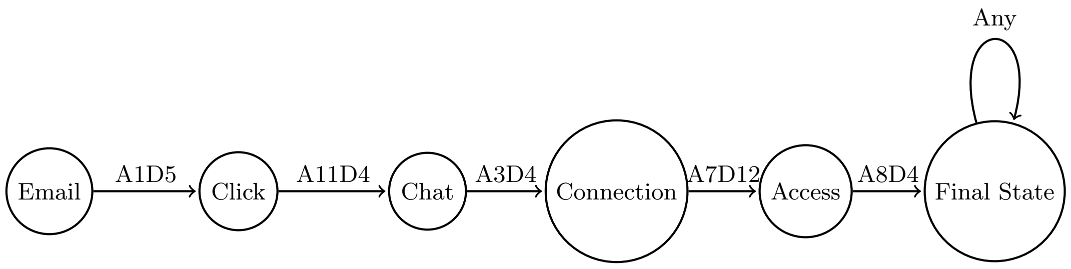
$$\delta(\text{Connection}, A7D12) = \text{Access}$$

This transition represents the attacker using a malicious connection (A7) and the defender ensuring a secured connection (D12).

– **Access**: For the input $A8D4$ starting from the state Access:

$$\delta(\text{Access}, A8D4) = \text{Final State}$$

This transition represents the attacker using malicious access (A8) and the defender identifying the malicious access (D4). The finite automaton diagram is presented in Fig. 2.



$$\begin{tikzcd}
\text{Email} \arrow[r, "\text{A1D5}"] & \text{Click} \arrow[r, "\text{A11D4}"] & \text{Chat} \arrow[r, "\text{A3D4}"] & \text{Connection} \arrow[r, "\text{A7D12}"] & \text{Access} \arrow[r, "\text{A8D4}"] & \text{Final State} \arrow[loop above, "\text{Any}"]
\end{tikzcd}$$

Fig. 2. Finite Automaton Modeling the Cybersecurity Game Scenarios

To apply Kleene's Theorem to the finite automaton $\mathcal{A}$, we need to derive a regular expression that represents the language accepted by the automaton. The automaton transitions between states based on specific input sequences, and the goal is to express this language with a regular expression. To construct the regular expression, we concatenate the input sequences for each transition:

- **From Email to Click**: $A1D5$
- **From Click to Chat**: $A11D4$
- **From Chat to Connection**: $A3D4$
- **From Connection to Access**: $A7D12$
- **From Access to Final State**: $A8D4$

Combining these transitions, the regular expression R representing the language accepted by the automaton is:

$$R = A1D5\,A11D4\,A3D4\,A7D12\,A8D4$$

The regular expression $R = A1D5\,A11D4\,A3D4\,A7D12\,A8D4$ accurately represents the language accepted by the finite automaton $\mathcal{A}$. According to Kleene's Theorem, this confirms that the language accepted by the automaton is regular [16].

9.4 Mapping Protocol States to Automaton States

Let P be a cryptographic protocol with a finite set of states $S = \{s_0, s_1, \ldots, s_n\}$ and transitions representing protocol steps. We construct a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ corresponding to P as follows:

- $Q = S$: Each protocol state is a state in the automaton.
- Σ: The input alphabet consists of all possible actions or messages in the protocol.
- δ: The transition function maps protocol steps to state transitions.
- q_0: The initial protocol state.
- F: The set of accepting states corresponds to successful protocol completion.

9.5 CyberMoraba as Automaton Simulation

In CyberMoraba, moves and configurations can simulate the DFA $\mathcal{A}$:

- **States**: Game configurations correspond to protocol states.
- **Moves**: Player actions represent inputs Σ causing state transitions.
- **Transitions**: Legal moves in the game simulate δ.

10 Formal Verification Using Automata Theory

10.1 Proving Protocol Correctness

Theorem 1 (Protocol Correctness). *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be the DFA modeling a cryptographic protocol P. If for all $w \in \Sigma^*$, $\delta(q_0, w) \in F$ only if w adheres to the protocol's security specifications, then P is correct.*

Proof. Assuming $\mathcal{A}$ accurately models P, any accepted string w (i.e., leading to a state in F) represents a successful protocol execution. If w violates security specifications but is still accepted, this indicates a flaw in P or its modeling. Thus, ensuring that only security-compliant w are accepted by $\mathcal{A}$ confirms protocol correctness.

10.2 Ensuring Safety Properties

Definition 1 (Safety Property). *A property is* safe *if it stipulates that "something bad" never happens during protocol execution.*

Theorem 2 (Safety Verification). *If all reachable states in $\mathcal{A}$ from q_0 avoid unsafe configurations (e.g., exposure of secret keys), then the protocol maintains safety.*

Proof. By performing a reachability analysis on $\mathcal{A}$, we examine all states accessible from q_0. If none of these states correspond to unsafe configurations, the protocol upholds safety properties.

10.3 Verifying Liveness Properties

Definition 2 (Liveness Property). *A property is* live *if it ensures that "something good" eventually happens, such as protocol completion.*

Theorem 3 (Liveness Verification). *If for every state $q \in Q$, there exists a sequence $w \in \Sigma^*$ such that $\delta(q, w) \in F$, then the protocol satisfies liveness properties.*

Proof. This condition guarantees that from any state q, the protocol can proceed to successful completion. If such a sequence w exists for all q, the protocol does not encounter deadlocks or infinite loops, satisfying liveness.

Using Kleene's Theorem, we can derive a regular expression R corresponding to $L(\mathcal{A})$, the language recognized by $\mathcal{A}$. This expression encapsulates all valid sequences w leading from q_0 to F. The regular expression R allows for:

- **Pattern Recognition**: Identifying allowed and disallowed sequences.
- **Simplification**: Optimizing the protocol by eliminating redundant steps.
- **Validation**: Ensuring that all sequences in R comply with security requirements.

11 Case Study: Modeling a Simple Authentication Protocol

11.1 Protocol Description

Consider a basic authentication protocol with the following steps:

1. Client sends login request.
2. Server sends challenge.
3. Client responds with credentials.
4. Server verifies and grants access.

11.2 Automaton Construction

We model this protocol as DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$, representing each protocol step.
- $\Sigma = \{\text{login\_req}, \text{challenge}, \text{credentials}, \text{access\_granted}\}$.
- δ defined as:

$$\begin{aligned} \delta(q_0, \text{login\_req}) &= q_1 \\ \delta(q_1, \text{challenge}) &= q_2 \\ \delta(q_2, \text{credentials}) &= q_3 \\ \delta(q_3, \text{access\_granted}) &= q_4 \end{aligned}$$

- $F = \{q_4\}$.

Applying the Kleene's theorem yield to:

- **Correctness**. Since only the sequence login_req $\rightarrow$ challenge $\rightarrow$ credentials $\rightarrow$ access_granted leads to q_4, and this sequence adheres to security specifications, the protocol is correct.
- **Safety**. There are no transitions leading to states where credentials are exposed without verification.
- **Liveness**. From any state q_i where $i < 4$, there exists a path to q_4, ensuring liveness.

12 CyberMoraba Simulation

12.1 Game Mapping

In CyberMoraba, each protocol step corresponds to a move:

- **Move 1**: Placing a piece representing login_req.
- **Move 2**: Responding with a piece for challenge.
- **Move 3**: Placing a piece for credentials.
- **Move 4**: Final piece for access_granted.

By simulating these moves in CyberMoraba, players can:

- Visualize protocol progression.
- Identify potential vulnerabilities (e.g., intercepting credentials).
- Explore alternative sequences and their outcomes.

Through rigorous mathematical modeling and application of automata theory, CyberMoraba proves to be a valuable tool for the formal verification of cryptographic protocols. By constructing corresponding finite automata and deriving regular expressions, we can analyze and ensure protocol properties such as correctness, safety, and liveness, ultimately enhancing cybersecurity measures. The application of finite automata and regular expressions to cryptographic security protocols provides robust methods for formal verification, optimization, and analysis. By modeling a protocol as a finite automaton, one can rigorously verify its correctness, ensuring it only accepts valid sequences and adheres to intended security properties such as authentication and confidentiality. Regular expressions derived from these automata facilitate the detection of protocol flaws, enabling the identification of security vulnerabilities. Additionally, these expressions aid in optimizing protocols through minimization, reducing computational overhead, and improving performance [17]. In pattern matching, regular expressions can define attack signatures and detect anomalies, enhancing security monitoring [18]. Moreover, finite automata and regular expressions support precise specification and design of protocols, ensuring they meet formal security standards and are resilient to various attacks [17], [18], [19].

13 Conclusion

The application of Kleene's Theorem enhances the modeling of cybersecurity scenarios by providing a formal mechanism to represent and analyze sequences of actions within these scenarios. The use of finite automata in representing strategic interactions in CyberMoraba offers a structured and precise method for analyzing complex decision-making processes, allowing for the identification of optimal strategies. Automata theory, when applied to cybersecurity games, improves the prediction of outcomes by enabling a systematic exploration of all possible action sequences and their consequences, leading to better strategy optimization. Moreover, modeling CyberMoraba provides practical insights into the formal verification of cryptographic security protocols, highlighting the potential to strengthen protocol security through automata-based analysis and ensuring that all protocol actions conform to the expected secure patterns.

References

1. Mutombo, E.N. and Nkongolo, M.W., 2024. Blockchain security for ransomware detection. arXiv preprint arXiv:2407.16862.
2. Denning, D.E., 1982. Encryption algorithms. Cryptography and Data Security," Addison Wesley Publishing Company Inc., USA, pp.59-125.
3. Shallit, J., 2024. Rarefied Thue-Morse sums via automata theory and logic. Journal of Number Theory, 257, pp.98-111.
4. Da Ré, B., Szmuc, D. and Corbalán, M.I., 2024. Non-Reflexive Nonsense: Proof Theory of Paracomplete Weak Kleene Logic. Studia Logica, pp.1-17.
5. Nkongolo, M.W., 2024, March. Infusing Morabaraba game design to develop a cybersecurity awareness game (CyberMoraba). In International Conference on Cyber Warfare and Security (Vol. 19, No. 1, pp. 240-250).
6. Nkongolo, M., 2023, January. Game theory based artificial player for morabaraba game. In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 1210-1218). IEEE.

7. Prokopev, S., 2024. Cryptographic protocol conformance testing based on domain-specific state machine. Journal of Computer Virology and Hacking Techniques, 20(2), pp.249-259.
8. Kassa, S., Das, J.C., Lamba, V., De, D., Debnath, B., Mallik, S. and Shah, M.A., 2024. Novel design of cryptographic architecture of nanorouter using quantum-dot cellular automata nanotechnology. Scientific Reports, 14(1), p.10532.
9. Hague, M., Jeż, A. and Lin, A.W., 2024. Parikh's Theorem Made Symbolic. Proceedings of the ACM on Programming Languages, 8(POPL), pp.1945-1977.
10. Aranda, V., Martins, M. and Manzano, M., 2024. Propositional Type Theory of Indeterminacy. Studia Logica, pp.1-30.
11. Hu, J., Zeng, F., Zhao, Y., Zhang, Z., Zhang, L., Zhao, J., Chang, R. and Ren, K., 2024. ProveriT: A Parameterized, Composable, and Verified Model of TEE Protection Profile. IEEE Transactions on Dependable and Secure Computing.
12. Yuan, Y., Wang, Y. and Cheng, G., 2024, July. ProfistMAC: A Protocol Finite State Machine Classifier via Graph Representation. In Australasian Conference on Information Security and Privacy (pp. 350-369). Singapore: Springer Nature Singapore.
13. Mahe, E., Bannour, B., Gaston, C., Lapitre, A. and Le Gall, P., 2024, April. Finite Automata synthesis from interactions. In Proceedings of the 2024 IEEE/ACM 12th International Conference on Formal Methods in Software Engineering (FormaliSE) (pp. 12-22).
14. Nkongolo Wa Nkongolo, M., 2024. RFSA: A Ransomware Feature Selection Algorithm for Multivariate Analysis of Malware Behavior in Cryptocurrency. International Journal of Computing and Digital Systems, 15(1), pp.893-927.
15. Nkongolo, M.N.W., 2023. Zero-day vulnerability prevention with recursive feature elimination and ensemble learning. Cryptology ePrint Archive.
16. Tokmak, M. and Nkongolo, M., 2023. Stacking an autoencoder for feature selection of zero-day threats. arXiv preprint arXiv:2311.00304.
17. Jajan, K.I.K. and Zeebaree, S.R., 2024. Optimizing Performance in Distributed Cloud Architectures: A Review of Optimization Techniques and Tools. The Indonesian Journal of Computer Science, 13(2).
18. Abuabid, A. and Aldeij, A., 2024. Cyber Security Incident Response: The Effectiveness of Open-Source Detection Tools in DLL Injection Detection. Journal of Information Security and Cybercrimes Research, 7(1), pp.29-50.
19. Tehranipoor, M., Zamiri Azar, K., Asadizanjani, N., Rahman, F., Mardani Kamali, H. and Farahmandi, F., 2024. Advances in Logic Locking. In Hardware Security: A Look into the Future (pp. 53-142). Cham: Springer Nature Switzerland.