Topology-Driven Symbolic Verification of Post-Quantum Migration Paths Using Tamarin Prover

Vishnu Ajith, Muhammed Ibrahim, and Muhammed Sihan Haroon

***Abstract*—The transition from classical public-key cryptography to post-quantum cryptography introduces protocol-level risks that are not fully addressed by configuration review, performance benchmarking, or endpoint reachability testing. Under the current abstraction, deployments may appear operationally correct while still permitting secrecy, authentication, or forward-secrecy violations at the protocol level. This paper presents a topology-driven symbolic verification workflow that translates distributed-system communication graphs into Tamarin models for analysis under the Dolev–Yao adversary model. The workflow derives protocol roles, communication constraints, and migration policies from a graph-based deployment representation, producing `.spthy` models and associated lemmas for executability, secrecy, authentication, and forward secrecy. A canonical topology representation is used to ensure deterministic model generation from semantically equivalent graph inputs. Experimental evaluation across five scenarios indicates that the framework produces discriminative symbolic outcomes rather than uniform failure reports. A registration-only control scenario verifies all reported lemmas, while the remaining scenarios exhibit two distinct falsification patterns: secrecy and forward-secrecy failures in three scenarios, and authentication failure in one scenario. These results indicate that symbolic verification provides a complementary assurance layer for post-quantum migration analysis and can reveal protocol-level risks that are not observable through operational testing alone.**

***Index Terms*—Post-Quantum Cryptography, Tamarin Prover, Symbolic Verification, Security Protocols, Distributed Systems, Forward Secrecy, Authentication, Graph-Based Modeling.**

I. Introduction

Post-quantum migration is no longer only an algorithm-selection problem; it is also a protocol-correctness problem. The standardization of ML-KEM, ML-DSA, and SLH-DSA by NIST has accelerated adoption efforts across distributed systems, service meshes, gateways, and application-layer security protocols [1]–[5]. However, replacing or augmenting classical primitives does not by itself guarantee that the surrounding protocol logic remains secure.

A recurring difficulty is that migration decisions are made at multiple layers simultaneously: key-exchange policy, certificate handling, fallback behavior, gateway mediation, and endpoint capability constraints. A system may therefore satisfy operational expectations—for example, endpoints are reachable and cryptographic primitives execute successfully—while still violating deeper security properties such as forward secrecy or peer authentication. This distinction is especially important in transitional hybrid deployments, where classical and post-quantum mechanisms may coexist.

Symbolic verification provides a rigorous way to reason about such protocol behaviors. The Tamarin Prover has been successfully applied to TLS 1.3, 5G authentication, the Noise framework, EMV payment protocols, and recent post-quantum-secure messaging designs [6]–[12]. Yet, in practice, symbolic analysis is often performed manually on standalone models detached from actual deployment structure. This creates a modeling gap between system topology and formal protocol reasoning.

This paper focuses specifically on topology-conditioned symbolic checking for post-quantum migration scenarios. The central idea is to represent a distributed deployment as a directed graph, derive a canonical symbolic model from that graph, and use Tamarin to check whether the migration path preserves key security properties under an active adversary. The aim is not to symbolically model the internal mathematics of ML-KEM or ML-DSA. Instead, the workflow checks protocol-level consequences of deployment policy, topology, and mixed migration states.

The contributions of this paper are as follows:

1) We define a topology-driven method for generating Tamarin models from distributed-system migration graphs.
2) We formalize a compact graph semantics for migration-aware protocol checking, including endpoint capability and channel policy constraints.
3) We introduce a canonical topology representation enabling deterministic symbolic model generation from semantically equivalent graph inputs.
4) We report per-scenario symbolic outcomes together with executability checks so that meaningful verification can be distinguished from aggregate verdicts.
5) We experimentally show that protocol-level vulnerabilities can be exposed even when deployment posture appears stronger under runtime or policy-level interpretation.

The remainder of this paper is structured as follows. Section II reviews related work. Section III presents the system model. Section IV describes the Tamarin-based methodology. Section V reports results. Section VI discusses limitations. Section VII concludes.

V. Ajith, M. Ibrahim, and M. S. Haroon are Lecturers in Computing at Ulster University (via QA Higher Education), UK. V. Ajith is also an R&D Engineer at Quentangle Quantum Systems, UK.

II. Related Work

A. Symbolic Verification of Security Protocols

The Tamarin Prover is a mature framework for the symbolic analysis of security protocols under an adversary with full network control [6], [7]. It has been used to verify major real-world protocols, including TLS 1.3 [8], [9], 5G authentication [10], the Noise framework [11], and EMV payment protocols [12]. These studies show that symbolic methods can reveal subtle attack traces that are invisible to conventional testing. Recent work has also examined the ProVerif tool as a complementary framework for automated symbolic analysis [16], and formal verification has been applied to post-quantum secure channel protocols [17].

B. Post-Quantum Protocol Assurance

Recent research has expanded from post-quantum algorithm design to deployment assurance. NIST standardized ML-KEM, ML-DSA, and SLH-DSA in 2024 [3]–[5], and prior work has benchmarked post-quantum key exchange and signatures in TLS and related protocols [13]. Machine-checked verification of Kyber-family implementations has further demonstrated the importance of high-assurance validation beyond benchmark-driven evaluation [14]. Hybrid key exchange combining classical and post-quantum primitives has been formally analyzed as a migration strategy [18]. However, most existing work either focuses on primitive correctness or protocol verification in isolation. Less attention has been given to automatically deriving symbolic protocol models from distributed deployment structure during post-quantum migration.

C. Graph-Based and Topology-Aware Security Analysis

Graph-based representations of distributed systems have been used in network security analysis and attack graph research [19], [20]. However, the use of deployment topology graphs as the direct source for generating symbolic protocol models remains largely unexplored, particularly in the PQC migration context. Our work bridges this gap by deriving Tamarin specifications directly from annotated communication graphs.

D. Gap Addressed in This Paper

The gap addressed here is not the design of a new cryptographic primitive, but the formal checking of migration behavior in a system topology. Our focus is a workflow that links deployment graphs to Tamarin models and evaluates whether migration posture, channel policy, and capability asymmetry introduce protocol-level risk.

III. System Model

We model a distributed deployment as a directed graph

$$G = (V, E), \tag{1}$$

where V is the set of nodes and $E \subseteq V \times V$ is the set of directed communication edges.

A. Node Semantics

Each node $v \in V$ is associated with an attribute tuple

$$\alpha(v) = (\rho_v,\, \eta_v,\, \kappa_v,\, \sigma_v), \tag{2}$$

where ρ_v denotes the functional role (e.g., client, server, gateway), η_v denotes the endpoint identifier, $\kappa_v \in \{0, 1\}$ indicates whether the node is configured to support post-quantum cryptography, and $\sigma_v \in \{0, 1\}$ indicates whether strict PQ-only operation is enforced.

B. Edge Policy Semantics

Each edge $e = (u, v) \in E$ carries a communication policy

$$\pi(e) \in \{\texttt{classical},\, \texttt{hybrid},\, \texttt{pqc_only}\}. \tag{3}$$

This policy constrains how the channel should be modeled and what forms of fallback are admissible.

C. Security Goals

For a generated symbolic model $M(G)$, the main property classes considered in this paper are executability, secrecy, authentication, and forward secrecy. At a conceptual level, these correspond to the following security intentions:

Secrecy: session data should not become derivable by the adversary.

Authentication: a completed receive event should correspond to a prior matching send event.

Forward Secrecy: later compromise should not reveal previously established protected data.

These expressions are schematic and are included only to summarize the intended security goals. The concrete Tamarin lemmas used in the implementation are generated from scenario structure and may also include executability sanity lemmas and auxiliary policy-consistency lemmas when supported by the scenario.

D. Canonical Representation

To ensure stable symbolic model generation, the input topology is transformed into a canonical representation:

$$C(G) = \mathrm{Norm}(V,\, E,\, \alpha,\, \pi,\, \omega), \tag{4}$$

where ω denotes relevant observed wire-evidence fields. Semantically equivalent topologies should therefore produce the same normalized representation even when incidental ordering differs in the source input.

IV. Topology-Driven Tamarin Methodology

A. Overview

The proposed workflow maps a deployment graph into a Tamarin model in three steps:

1) Canonicalize the graph and extract role and channel semantics.
2) Generate a symbolic protocol model and associated lemmas.
3) Execute Tamarin and classify the resulting per-lemma outcomes.

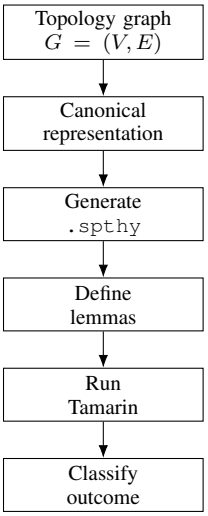


Fig. 1: Topology-driven symbolic checking workflow. A deployment graph is transformed into a canonical representation, translated into a Tamarin model and lemma set, and checked under symbolic adversary assumptions.

B. Graph-to-Model Translation

Each graph edge induces a communication relation between protocol roles. Nodes identified as gateways or policy-enforcement points introduce mediation rules into the symbolic model. Strict PQ-only nodes prohibit classical fallback transitions, while hybrid nodes admit controlled coexistence of classical and post-quantum behavior.

The generated `.spthy` model includes:

- protocol rules representing initiator and responder message flow,
- facts encoding node capabilities, edge configuration, and observed wire posture,
- restrictions corresponding to permitted or forbidden fallback behavior,
- sanity lemmas for executability,
- primary security lemmas for secrecy, authentication, and forward secrecy,
- auxiliary policy-consistency lemmas when the scenario structure supports them.

C. Adversary Model

We adopt the standard Dolev–Yao adversary model [15]. The adversary has complete control over the communication network and may intercept, replay, modify, block, and inject messages arbitrarily. Cryptographic primitives are assumed perfect at the symbolic level; attacks therefore arise from protocol structure, trust assumptions, and migration policy logic rather than computational cryptanalysis.

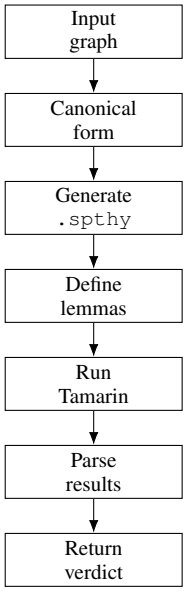


Fig. 2: Execution pipeline for topology-driven symbolic verification. The deployment graph is normalized, translated into a symbolic model, checked in Tamarin, and parsed into per-lemma outcomes.

D. Generated Model Example

To make the generation process concrete, Listing 1 shows an excerpt from a generated PQC-only channel model. The excerpt preserves the actual identifiers emitted by the generator for edge configuration, observed wire posture, message transmission, delivery, and authenticated-flow executability. Ellipsis markers (`...`) in the Receive rule denote fields abbreviated for space; the full rule is emitted without truncation by the generator.

The example demonstrates that the graph-to-model translation does not stop at node metadata: it also materializes channel policy, observed wire posture, symbolic send/receive rules, and executability conditions.

E. Why This Matters for PQ Migration

Post-quantum migration is typically incremental rather than instantaneous. As a result, systems often move through intermediate hybrid states that are vulnerable to policy inconsistency, fallback ambiguity, and hidden attack surfaces. The proposed method enables these transitional states to be checked formally at the topology level rather than only at the primitive or endpoint level.

Listing 1: Excerpt from a generated PQC-only channel model (fields abbreviated with ... are emitted in full by the generator).

```
rule Channel_client_service_0_Policy:
  [ ]
  --[ EdgeConfigured('client__service_1',
        'client','service','pqc_only'),
      EdgeRolePath('client__service_1',
        'server','server'),
      ObservedWireState('client__service_1',
        'PQC_ONLY') ]->
  [ !EdgeReady('client__service_1') ]

rule Channel_client_service_0_Send:
  let payload_0 = <~tok_0, ~data_0>
      ct_0  = aenc(payload_0, sessPk_0)
      sig_0 = sign(<'client__service_1',
                    'pqc_only',ct_0,
                    'client','service',~tok_0>,
                   ltkS_0)
  in
  [ !EdgeReady('client__service_1'),
    !Ltk('client',ltkS_0),
    !SessionPk('client__service_1',sessPk_0),
    Fr(~data_0), Fr(~tok_0) ]
  --[ PQCTransmit('client','service',~data_0),
      Secret(~data_0),
      TokenUsed(~tok_0),
      Running('client','service',~data_0) ]->
  [ PendingToken('client__service_1',~tok_0),
    Out(<'pqc_only','client__service_1',
         ct_0,sig_0,~tok_0>) ]

rule Channel_client_service_0_Receive:
  [ !SessionSk('client__service_1',sessSk_0),
    PendingToken('client__service_1',tok_0),
    In(<'pqc_only','client__service_1',
        aenc(<tok_0,data_0>,pk(sessSk_0)),
        sign(<'client__service_1','pqc_only',
              aenc(<tok_0,data_0>,pk(sessSk_0)),
              'client','service',tok_0>,
             ltkS_0),
        tok_0>) ]
  --[ Delivered('client__service_1',
               'client','service',data_0),
      AcceptedToken('client__service_1',tok_0),
      Commit('service','client',data_0) ]->
  [ !EdgeReady('client__service_1') ]

lemma sanity_authenticated_flow_executable:
  exists-trace
  "Ex a b data #i #j.
     Running(a,b,data) @ #i &
     Commit(b,a,data) @ #j & #i < #j"
```

V. Results

This section evaluates the effectiveness of the topology-driven symbolic verification workflow in detecting protocol-level vulnerabilities during post-quantum migration. The evaluation focuses on five scenarios: one sanity-control case and four migration-path cases with distinct topology and policy characteristics.

A. Experimental Setup

A prototype implementation was used to generate Tamarin models from distributed communication topologies and execute symbolic verification under the Dolev–Yao adversary model. Experiments were conducted through a scenario harness that exports machine-readable results, per-scenario summaries, and trace metadata where available. Each scenario was executed five times; median wall-clock latency is reported.

TABLE I: Scenario Roles in the Evaluation

Scenario	Role in Evaluation
S0	Registration-only control for model generation and executability; no security trace exported
S1	Classical direct path used to examine baseline symbolic failure modes
S2	Mixed-capability strict-PQC path used to test asymmetry under stricter policy assumptions
S3	Gateway-mediated path used to test secrecy and forward-secrecy behavior under mediation
S4	Stronger PQC multi-hop path used to test whether stronger posture necessarily preserves authentication

TABLE II: Per-Scenario Symbolic Lemma Outcomes

Scen.	Name	Exec.	Sec.	Auth.	FS	Overall
S0	Reg.-only control	✓	✓	✓	✓	VERIFIED
S1	Classical direct	✓	×	✓	×	FALSIFIED
S2	Asym. strict PQC	✓	×	✓	×	FALSIFIED
S3	Classical via PQC GW	✓	×	✓	×	FALSIFIED
S4	PQC multi-hop	✓	✓	×	✓	FALSIFIED

✓ = verified; × = falsified

The evaluated scenarios are:

1) **S0 – Registration-Only Positive Control:** a control scenario intended to confirm model generation and basic executability.
2) **S1 – Classical Direct Path:** a direct classical communication path used to examine baseline symbolic failure modes.
3) **S2 – Asymmetric Strict PQC:** a mixed-capability configuration with stricter post-quantum enforcement assumptions.
4) **S3 – Classical via PQC Gateway:** classical endpoints communicating through an intermediate PQC-capable gateway.
5) **S4 – PQC Multi-Hop Path:** a stronger multi-hop PQC path used to test whether stronger posture preserves authentication.

Table I summarizes the role of each scenario.

B. Per-Scenario Lemma Matrix

The most important result is that the framework does not behave as a uniform failure detector. Table II reports the per-scenario, per-lemma outcomes. Executability is verified in all scenarios, while the security outcomes separate into two distinct patterns.

As shown in Table II, the results split into two distinct falsification patterns. In S1–S3, secrecy and forward secrecy fail while authentication remains verified. In S4, secrecy-related properties remain verified while authentication fails. The practical implication is that stronger deployment posture

TABLE III: Runtime Assessment Versus Symbolic Verification

Scenario	Runtime (Status)	Symbolic Outcome
S0	PASS	All lemmas verified
S1	PASS	Secrecy and FS falsified
S2	PASS	Secrecy and FS falsified
S3	PASS	Secrecy and FS falsified
S4	PASS	Authentication falsified

TABLE IV: Median Verification Latency Across Five Repeated Runs

Scenario	Median Latency (ms)
S1	174.069
S2	169.519
S3	317.826
S4	1328.424

does not monotonically translate into stronger symbolic guarantees.

C. Interpretation of the Verified Control

S0 verifies all reported lemmas and therefore serves as a positive control for model generation and basic symbolic executability. At the same time, its security interpretation should remain cautious because it is structurally simpler than the message-carrying scenarios. Its main value is to show that the framework does not simply falsify every input.

D. Runtime Posture Versus Symbolic Outcome

A central result of the evaluation is that deployment posture does not reliably predict symbolic security outcome. Table III summarizes this divergence across all five scenarios.

The three scenarios S1–S3 all falsify secrecy and forward secrecy, despite differing in communication assumptions and migration posture. By contrast, S4 preserves secrecy and forward secrecy while falsifying authentication. This shows that symbolic analysis is not merely rephrasing deployment labels. Instead, it exposes topology-dependent failure modes that operational posture alone cannot distinguish.

E. Verification Latency

The scenario harness executed five repeated prover runs per scenario and recorded wall-clock timing data for the four migration scenarios S1–S4. Median wall-clock latency across those repeated runs is reported in Table IV. For completeness, harness-native first-run timings were S1: 279 ms, S2: 179 ms, S3: 375 ms, and S4: 1613 ms.

The results show a clear separation between the simpler scenarios and the more expensive multi-hop case. The large gap between S4 and the other scenarios is consistent with gateway-mediated topologies inducing more complex symbolic search behavior than two- and three-node cases.

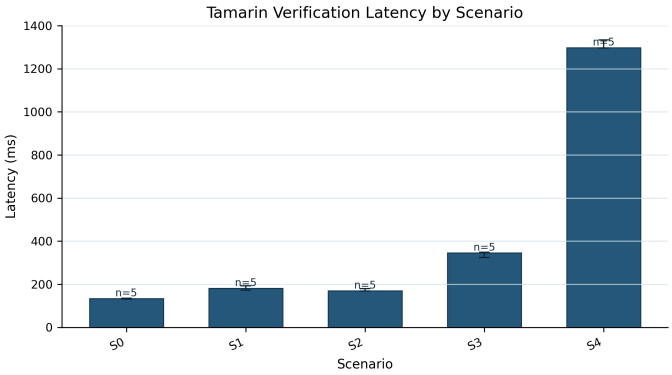


Fig. 3: Verification latency across the primary migration scenarios (five runs each). The S4 case is materially more expensive than the simpler scenarios.

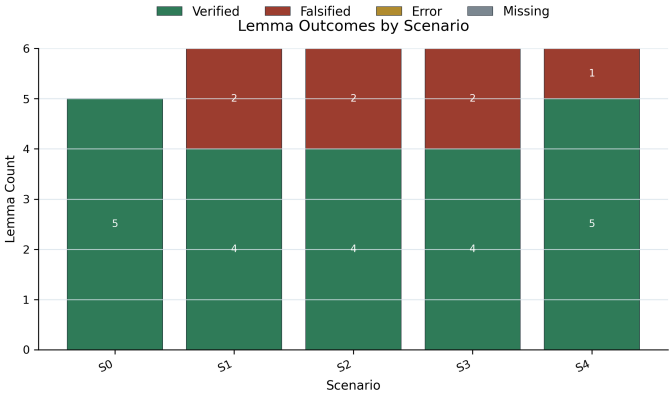


Fig. 4: Verified and falsified lemma outcomes by scenario. The framework produces differentiated results rather than uniform acceptance or rejection.

F. Lemma Outcomes by Scenario

Figure 4 visualizes the distribution of verified and falsified outcomes across scenarios, complementing Table II. Only these two outcome categories were observed across all evaluated scenarios.

G. Partial Trace Support

The current implementation exports more than a binary trace flag. For falsified lemmas, the per-lemma records include `traceAvailable` together with raw trace excerpts, action labels, involved entities, heuristic hints, and parser-confidence notes where extraction succeeds. This remains partial rather than full Tamarin counterexample reconstruction, so the paper treats trace support conservatively. In particular, a *Trace Available* indicator in Table II would denote that at least one falsified lemma in that scenario produced exported trace metadata; it should not be interpreted as full structured attack-trace recovery.

H. Summary of Findings

The results support four conclusions:

1) The framework produces discriminative symbolic outcomes rather than uniform failure reports.

2) Executability is verified in all evaluated scenarios, indicating that the generated models are symbolically runnable.
3) Secrecy and forward secrecy fail together in S1–S3, whereas S4 exhibits a different failure mode in which authentication is falsified while secrecy-related properties remain verified.
4) Deployment posture alone does not determine symbolic success; different migration topologies fail in different ways.

VI. Limitations

The method has several limitations. First, symbolic guarantees depend on the fidelity of the generated model. If an important deployment behavior is omitted during graph-to-model translation, the resulting outcome may not reflect the true system.

Second, the current workflow checks protocol-level symbolic properties rather than computational soundness or implementation-level side channels. Thus, a successful lemma outcome should not be interpreted as a complete implementation-security guarantee.

Third, the present abstraction is policy-level rather than cryptographic-primitive-level. The implementation does not symbolically model the internal mathematics of ML-KEM, ML-DSA, or SLH-DSA. Instead, it checks how migration posture, channel policy, and topology affect protocol properties.

Fourth, although the implementation now exports partial structured trace metadata, it does not yet provide full attack-trace reconstruction suitable for detailed counterexample narration.

Finally, the current evaluation remains compact. Broader scenario families, stronger negative controls, and richer timing studies would further improve empirical coverage.

VII. Conclusion

This paper presented a topology-driven workflow for symbolic verification of post-quantum migration paths using the Tamarin Prover. The approach translates distributed communication graphs into Tamarin models, checks executability and security properties under an active adversary, and evaluates the results across a verified control and four representative migration scenarios. The current implementation supports deterministic model generation, per-scenario outcome reporting, and partial trace export, enabling a more evidence-based characterization of symbolic outcomes than aggregate-only reporting. The experimental findings show that deployment posture alone does not determine symbolic success and that meaningful positive and negative cases can be distinguished under the current abstraction. These results support the use of symbolic checking as a complementary assurance layer for post-quantum migration analysis.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, pp. 188–194, 2017.

[3] NIST, "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Aug. 2024.

[4] NIST, "FIPS 204: Module-Lattice-Based Digital Signature Standard," National Institute of Standards and Technology, Aug. 2024.

[5] NIST, "FIPS 205: Stateless Hash-Based Digital Signature Standard," National Institute of Standards and Technology, Aug. 2024.

[6] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *Proc. CAV*, 2013, pp. 696–701.

[7] D. Basin, C. Cremers, J. Dreier, and R. Sasse, *Modeling and Analyzing Security Protocols with Tamarin*, Draft, 2024.

[8] C. Cremers, M. Horvat, S. Scott, and T. van der Merwe, "Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication," in *Proc. IEEE S&P*, 2016.

[9] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A Comprehensive Symbolic Analysis of TLS 1.3," in *Proc. ACM CCS*, 2017.

[10] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A Formal Analysis of 5G Authentication," in *Proc. ACM CCS*, 2018, pp. 1383–1396.

[11] N. Kobeissi, G. Nicolas, and K. Bhargavan, "Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols," in *Proc. IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019, pp. 356–370.

[12] D. Basin *et al.*, "EMV: Why Payment Systems Fail," *arXiv preprint arXiv:2006.08249*, 2020.

[13] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking Post-Quantum Cryptography in TLS," in *Proc. PQCrypto*, 2020, pp. 72–91.

[14] J. B. Almeida *et al.*, "Formally Verifying Kyber: Machine-Checked IND-CCA Security," in *Proc. CRYPTO 2024*, 2024.

[15] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[16] B. Blanchet, "Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif," *Foundations and Trends in Privacy and Security*, vol. 1, no. 1–2, pp. 1–135, 2016.

[17] C. Cremers, A. Dax, and N. Medinger, "Keeping Up with the KEMs: Stronger Security Notions for KEMs and Automated Analysis of KEM-based Protocols," in *Proc. ACM CCS*, 2023, pp. 2976–2990.

[18] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid Key Exchange in TLS 1.3," *IETF Internet-Draft draft-ietf-tls-hybrid-design*, 2023.

[19] S. Jha, O. Sheyner, and J. Wing, "Two Formal Analyses of Attack Graphs," in *Proc. IEEE Computer Security Foundations Workshop (CSFW)*, 2002, pp. 49–63.

[20] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: A Logic-Based Network Security Analyzer," in *Proc. USENIX Security Symposium*, 2005.