Towards Formal Methods of IoT Application Layer Protocols

1st Katharina Hofer-Schmitz
DIGITAL
JOANNEUM RESEARCH Forschungsgesellschaft mbH
Steyrergasse 17, Graz, Austria
katharina.hofer-schmitz@joanneum.at

2nd Branka Stojanović
DIGITAL
JOANNEUM RESEARCH Forschungsgesellschaft mbH
Steyrergasse 17, Graz, Austria
branka.stojanovic@joanneum.at

***Abstract*—This paper provides an overview of the application of formal methods for two most commonly used application layer protocols in IoT domain, MQTT and CoAP. Formal methods give the possibility to improve security and are even able to provide security guarantees with respect to a given model. Our research shows, that we can distinguish three formal verification fields for the two protocols, namely qualitative and quantitative analysis, implementations and security properties. A formal verification review of selected protocols is provided in accordance with these fields, used tools and considered properties. Based on that research, a short summary and open challenges are given.**

***Index Terms*—MQTT, CoAP, Formal Verification, Model Checkers, Security, IoT**

I. Introduction

This paper deals with protocols in IoT application layer for machine-to-machine (M2M) communication (see [1] for an overview of these protocols). According to [2] the most popular protocols for machine-to-machine (M2M) technology are Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP). M2M technology based on these two protocols can be found in a lot of sectors including manufacturing, public administration, defence, building automation, transport and healthcare. Due to the flexibility of these two protocols, they are adopted in a wide variety of settings. However, there are several vulnerabilities and issues regarding design and implementations along with thousands of non-secure deployments found in a security analysis by [2]. Such a large number of security issues might originate from the optional security. Furthermore, most of the tutorials in the internet providing examples on the use of these protocols do not have security enabled. That entails the risk that the code in question is not just intended for test cases only, but also in the production system. Moreover, changing standards and the popular agile integration in IoT are quite challenging for developers and might run the risk of using different version of the protocols.

In general security requirements are taken into account by performing risk analysis, applying attack trees, creating threat models and performing penetration tests. Since systems are getting more and more complex in both functionality and connectivity, challenges for security experts have increased. Due to limited resources in capacity and storage in IoT devices, detecting open security issues with complex security mechanisms at runtime is not an option. A promising technique to detect weaknesses and possible vulnerabilities and ascertain the correctness of a design by using a diverse set of mathematical and logical methods is that of formal methods (see e.g. [3]). Thereupon, the functional correctness of implementations, programming bugs and the fulfilment of security properties can be proven or falsified. In case of a successful proof, security guarantees with respect to the verified model can be made.

The main goal of this paper is to give a review on approaches applying formal methods to IoT application layer protocols with a focus on MQTT and CoAP. Section II provides some basic information for both protocols. Section III considers security in application layer in general and additionally security issues for the two selected protocols. In section IV, a short introduction to formal verification is given. Moreover, an overview of publications using formal methods for MQTT and CoAP is given. Details, especially regarding tools and considered properties are provided.

II. Protocol's overview

In general, protocols (see [4]) can be seen as a set of rules transmitting information in a predefined format allowing the communication between two or more entities (e.g. physical media, different layers). These rules, their syntax and its semantics define the protocol, which can be implemented by hardware, software or a combination of both. Protocols are usually defined in technical standards. There are several reference models for protocol standardization, providing a standard architecture to define network communication with several layers. One of the most popular models is the TCP/IP model [5], [6]. However, relevant literature proposes a reference model that fits better to the IoT domain (see e.g. [7]). It consists of three layers, the application layer, the network layer and the perception layer, see Fig. 1. The perception layer is responsible for identifying objects and gathering information, the network layer for transmitting and processing information from the perception layer and the application layer is responsible for

This work was funded in part by **Federal Ministry Republic of Austria** Transport, Innovation and Technology and in part by European Union's Horizon 2020 research and innovation programme under grant agreement No 830892, project SPARTA

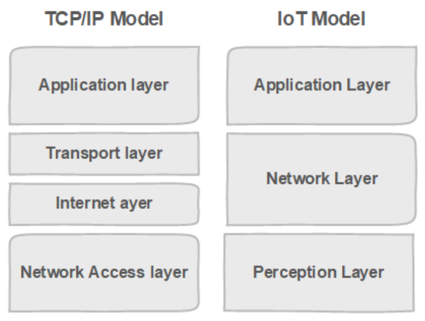

Fig. 1. TCP/IP model and IoT reference model

the representation of data. Moreover, it provides an interface for the user, giving them the possibility to monitor and control variables by sending commands over the network (see e.g. [8], [9]).

Next, a short introduction to the selected application layer protocols is given. For further application layer protocols see e.g. [1].

A. MQTT

MQ Telemetry Transport (MQTT), a protocol for machine to machine communication, was invented in 1999. It is a publish/subscribe, very simple and lightweight messaging protocol providing one-to-many message distribution and decoupling of applications. It is designed for constrained devices and low-bandwidth, high-latency or unreliable networks. Furthermore, it ensures reliability and assurance of delivery. Therefor, it can be perfectly applied for M2M or in the Internet of Things, where bandwidth and battery power are usually limited.

MQTT $v5.0$ and $v3.1.1$ are OASIS standards, the $v3.1.1$ has also been ratified by ISO. An older name for that protocol is the SCADA protocol. In order to keep the protocol simple and lightweight, there is no built-in security. However, user name and password can be passed with an MQTT packet in $v3.1$. For encryption across the network, SSL can be used, independently of MQTT protocol. Additional security can be added by an application encrypting data that it sends and receives.

The protocol runs over TCP/IP or other protocols which provide ordered, lossless, bi-directional connections. There are three Quality of Service (QoS) for message delivery. QoS 0, called “at most once”, messages are delivered according to the best efforts of the operating environment. Therein message loss can occur. QoS 1, named “at least once”. There, messages arrive, but duplication can occur. In QoS 2, “exactly once”, it is guaranteed, that messages arrive exactly once, see [10].

B. CoAP

The Constrained Application Protocol (CoAP) is a server-client protocol developed by the Internet Engineering Task Force. It is a specialized web transfer protocol for usage with constrained nodes and constrained networks (e.g. low-power, lossy). CoAP is designed for M2M applications, especially such as smart energy and building automation. Moreover, it fulfils requirements as multicast support, very low overhead and simplicity for constrained environments.

The protocol provides a request/response interaction model between application endpoints. Furthermore it supports built-in discovery of services and resources. It also includes URIS and Internet media types. There are four security modes in CoAP named NoSec, PreSharedKey, RawPublishKey and Certificate. While in the first mode the system is secured just by keeping attackers from being able to send or receive data packets from network with CoAP nodes, in the other three modes Datagram TLS (DTLS) is used to secure CoAP. CoAP itself does not provide protocol security primitives for authentication or authorization, see [11].

III. Security in IoT application layer

In order to ensure security, critical properties of information need to be protected in an information system. These properties are usually described by the C.I.A. triangle. *Confidentiality* means that just authorized users have rights and privileges to access information. *Integrity* ensures, that changes of information don’t happen unnoticed. *Availability* states that information is accessible in a required format without interference or obstruction. Additionally there are the properties: *Accuracy*, ensuring that there are no mistakes or errors and *Authenticity*, stating that information is in the same state, when it was created, placed, stored or transferred (see [12]).

A. Attacks on application layer

Attacks harm one or more of these critical properties. For communication protocols, generally speaking, the most important security properties are confidentiality and authenticity.

In [9] a classification of attacks based on the IoT layers is given. Attacks on application layers in general affect the end user, e.g. by passing a non-intended information or by denial of service. According to [9] attacks targeting application layers include phishing attacks, malware (e.g. worms, viruses, spywares and trojans), software vulnerabilities, sniffing attacks, keylogger attacks. Moreover, possible harm can also be done by Denial of Service (on user or web service) and encryption attacks containing side channel attacks and cryptoanalysis attacks.

B. Security issues in MQTT and CoAP

In [2] the following threats in MQTT and CoAP are reported:

- *Target reconnaissance*: Numerous insecure deployments of endpoints all over the world have been found, leaking sensitive data such as technical details, names, phone numbers, credentials and network configuration details.
- *Industrial espionage*: They were able to find production data of critical industry sectors (e.g. manufacturing, healthcare, building automation) by using simple keyword-based searches leading e.g. to location and identifying information of assets and personnel. Moreover, the technology used by a given company and the business-to-business relations, including exchanged communication, were also compromised.

- *Lateral movement*: By abusing a specific functionality of the M2M technology maintaining persistent access to a target via software upgrades or performing lateral movement is possible.

The authors in [2] claim, that the current security status of M2M technology facilitates an attacker with the above threats in mind due to design-related and technical issues with that technology and due to a low security awareness. These results even show a great basis for formidable Advanced Persistent Threats (APTs).

Beside [2], also [13] and [14] consider possible security issues for MQTT and CoAP. Summarizing these results for MQTT shows, that there are some corner cases providing opportunities for misunderstandings and consequent vulnerabilities, which might be introduced in implementations. Moreover, although MQTT version 5.0 improves security, there is the drawback that it is not entirely compatible with previous versions, which might introduce security issues and most likely, could delay its future adoption. For CoAP, there is a security risk consideration already highlighted in protocol's specification in [11]. However, possible security issues might arise due to the missing handshake phase. Reported attacks include amplification attacks, where an attacker can use the end device to convert small packages into larger packets, the parsing attack, where a remote node can be crashed by executing an arbitrary code on the node, caching attacks, where proxys with the ability to cache can gain control. The latter especially poses a threat for clients exchanging data with the proxy.

Most of the afore mentioned risks cannot be prevented by following general advice and using recommended security measures (e.g. data access only for authorized users, encryption of data, regular audits). However, some of them could be detected at an early stage at design stage by using formal methods - helping to avoid unclear specifications and provide security checks for implementations of these protocols in order to further avoid a widespread usage of vulnerable applications.

IV. Application of Formal Methods on IoT application layer protocols

Formal methods are a useful tool for providing quantitative statements about safety and security properties of digital systems (see e.g. [3]). These methods can be used to formally verify a model. Therefore, two kind of tools are distinguished: model checkers and theorem provers. While the first type exhaustively and automatically verifies a given system's model in it's model space, the second one often requires human expertise for proof guidance by providing design and specification characteristics as theorems or algebraic constraints.

One of the most popular and often cited examples emphasising the importance of such verification is the Needham-Schroeder public key protocol [15], which appeared 17 years before a possible attack – the Lowe's attack [16] (allow an attacker to impersonate another agent) – was detected by applying the Failures Divergence Refinement Checker (FDR) [17]. According to [18], two things should be noted: the attack is nonintuitive and violates a security assumption (agents do not reveal secrets) Needham and Schroeder explicitly made in their paper. However, losing this assumption has very surprising consequences. Furthermore, it has to be stated that the attack does not depend on any flaws present in cryptographic primitives. It just requires a simple adversary model, where the adversary only uses operations such as concatenation, splitting of messages and encryption/decryption.

Besides detecting vulnerabilities in a protocol through its design presented in [16], other applications fields for formal verification include checking the implementations of a protocol. There, an implementation is verified with respect to its formal specification. In that case especially programming bugs and programming vulnerabilities (e.g. buffer overflows, arithmetic overflow and underflow, format-string vulnerabilities) come into play and should be checked in detail. Furthermore, the functionality of a protocol or implementation can be considered, since this can also lead to possible attacks. An example is an intended operation such as a strong authentication method, which is absent (see e.g. [3]).

In the following subsections, an overview about formal method approaches on MQTT and CoAP are given. This paper tends to provide a review on most important methods and obtained results in this field, and literature was selected in accordance with that. Based on literature, three different application fields can be distinguished: Formal methods for qualitative and quantitative analysis (see subsection IV-A), implementations (see subsection IV-B) and verification of security properties (see subsection IV-C).

A. Qualitative and quantitative analysis

Aziz presents in his work in [19], [20] a formal model of the MQTT protocol version, including a formulation of a specification of the MQTT protocol and an analysis of its semantics. The formal model is based on a process algebra called `TPi` and originally inspired by [21] and further developed by [22]. TPi is a synchronous message-passing calculus capable of expressing timed inputs. The author uses TPi to approximate the behaviour of processes by performing an abstract static analysis, i.e. the number of copies of input variables and new names, which can be captured in the analysis during communication. There, different *quality of service scenarios* are considered, where the focus is on the initial published message from the server to the subscribers. The analysis is then applied manually to the protocol for these scenarios. The result shows, that the first two QoS modes of operation in the protocol are clearly specified. Nevertheless, the last case of QoS 2 delivery semantics has potential vulnerabilities. This is due to the fact, that a *simple attacker model can cause undetermined semantics*. [20] considers the updated MQTT standard in version 3.1.1, whereas [19] considers version 3.1.

Rodriguez et al. in [23] provide a formal `Colored Petri Net` model for MQTT version 3.1.1. Colored Petri Nets (CPN) is a graphical language for modelling and validating concurrent and distributed systems [24]. The goal of the authors in [23] is to simulate and perform a state-space exploration to *verify an extensive list of behavioural properties* and thereby validate the correctness of the model. The authors state, that they encountered several parts that are vaguely defined, which could lead developers to different implementations, e.g. a gap in the specification with the lossless property wherein the protocol MQTT protocol is described to run over TCP/IP (or another transport protocol providing ordered, lossless and bidirectional connections). Furthermore, the authors detected several issues in the specification of QoS levels 1 and 2.

Houimli et al. in [25] focus on the formal modelling of MQTT 3.1.1. As a first step a semi-formal model performed by UML diagrams is proposed. As a second step, formal modelling by using Probabilistic Timed Automata, a modelling formalism for systems with real-time and non-determinism as the main characteristics governing their behaviour, is done. As tool for formal modelling and later on, verification, the authors use `UPPAAL`. UPPAAL (see e.g. [26]) is a toolbox for verification of real-time systems and is especially useful for systems that can be modelled as a collection of non-deterministic processes with finite control structures and real-valued clocks, communicating through channels or shared values. It consists of three main parts, a description language for modelling, a simulator for validation and a model checker based on the theory of timed automata. For the latter, the SMC model checker of UPPAAL is used to perform a *qualitative and quantitative analysis* of the MQTT. The authors check several essential properties such as *reachability*, *safety* and *vivacity*. Moreover, some important metrics are analysed such as the number of active and inactive nodes, the success rate of transfer as well as the reception of message.

Hcine and Hafaiedh in [27] provide a model for MQTT by using the formalism of timed automata to describe different *quality of service levels*. For the overall architecture, they considered three parts, namely *Broker, Publisher* and *Subscriber*. Their purpose of the verification is to check whether the functionalities for MQTT can be accomplished through the proposed model. By using the `UPPAAL` model checker the authors proved several properties: *Deadlock-Freedom*, *Invariance*, and *Store* Property. The last one shows, that it is important for the QoS levels 1 and 2, that the publisher stores the topic before delivering it to the broker. Moreover, for QoS level 1 and 2 the following properties are checked:

- Each topic is received once by the broker.
- Each topic is received once by the subscriber.
- A subscriber gets the topics only if they were subscribed to the topics.

While the first two properties are not fulfilled in QoS level 1, all the other properties are satisfied. Since in QoS level 1 the delivery of one topic more than once can appear, the functionality of MQTT can be verified. Furthermore, the authors conduct several simulations by using the UPPAAL simulator. They considered for each QoS level several aspects in different possible configurations such as: the percentage of messages sent by the publishers, messages received by the subscribers and lost messages.

Diwan and D'Souza propose in [28] a unified approach to verify communication protocols through a framework in `Event-B`. Event-B is a formal method for *system modelling and analysis*, see [29]. Its key features are the use of set theory of modelling notation, the use of refinement to represent systems at different levels and the usage of mathematical proof for verifying consistency between refinement levels. The authors of [28] created an abstract model of a protocol consisting of commonalities among various application layer protocols like communication modes, connection establishment procedures, message layers, time tracking and attacker modules. The abstract model is then fit in the individual protocols, namely MQTT, MQTT-SN and CoAP by using refinement and decomposition techniques of Event-B. Simulation has been done using Rodin [30], a suite of tools aiding in the design and analysis of Event-B models. There they have formally verified properties related to *QoS*, *persistent session*, *will*, *retrain messages*, *resource discovery*, *two layered request-response architecture*, *caching*, *proxying* and *message deduplication*. Their simulation shows, that the protocols work as intended in an uninterrupted network as well as in the presence of an intruder, which consumes messages in the network. Moreover, the protocols are also able to reduce overhead by providing features like persistent connections, retain messages, caching and proxying. These features are essential for IoT systems.

Vattakunnel et al. in [31] propose a verification model for application layer protocols considering its interaction with routing layer illustrated for CoAP. As a tool, the `SPIN` model checker [32], which targets the efficient verification of multi-threaded software, is used. With SPIN, the logical consistency of a specification can be checked. Moreover, that tool reports on deadlocks, race conditions, different types of incompleteness and unwarranted assumptions about the relative speed of processes. As input language, the tool supports the high level language PROMELA for specifying system descriptions, which is also used by the authors in [31] for building their model. The model is used to consider *reliable message exchanges among various entities*, i.e. its interaction with the routing layer for constrained devices communicating through multiple hops. In order to analyze the external behavior of the model, a message sequence chart depicting the message flow between nodes is generated. Subsequently, the model is verified by using the SPIN model checker covering *safety*, *liveness* and *correctness* properties. Moreover, they analysed the verification results in terms of memory usage, number of state transitions and maximum search depth.

B. Implementations

Mladenov in [33] used formal verification to check if *MQTT implementations adhere to the standard* by using the `Test and Test Control Notation version 3` (TTCN-3). That tool enables the description of the behaviour of a given implementation. Moreover, it can be verified if the System Under Test adheres to the given definitions. For testing three different open source implementations, namely

MOSQUITTO version 1.4.0, Emqttd version $2.2 - rc.1$, Rabbit MQ version 3.6.10 are chosen. They tested 13 normative statements, that were selected to cover many different aspects of the MQTT message exchange defined in sections 2 and 3 of the standard. It was expected for the implementations to pass the test, however, for the normative requirement $2.3.1 - 1$ all three of them fail. That requirement states, that the Packet ID should be greater than zero in the case of an exchange of certain types of messages with QoS lever greater than 0. Moreover, for several other normative requirements two of their implementations fail for the same requirement.

Hernandes et al. in [34] propose a framework based on `fuzzing technique` to test and verify the security of applications implementing MQTT. The fuzzing technique is a testing approach in order to detect vulnerabilities in software applications by sending unexpected input data to target systems and then monitor the results. By using their fuzzer - only suited for MQTT and no other protocols - they found several unknown vulnerabilities (like *Denial of Service, Communication resets of Brokers*) in widely used applications implementing the MQTT protocol, like MOQUETTE or MOSQUITTO.

Tromp in [35] considers formal methods to successfully *identify faulty behaviour* in (implemented) communication protocols by using `online automata learning`. This technique helps to infer formal models of protocol implementations by inferring an automaton. Therefore, they used RALibisan, an extension of LearnLib, a modular framework for online automata learning. They considered three open source libraries of CoAP, namely *Californium, CoAPthon* and *txThings*. The test of these implementations lead to 11 specification violations. Furthermore, the authors states that only parts of the protocol have been studied, therefor the approach might not find every fault in the implementations.

C. Security properties

Kim et al. in [36] formally analyse MQTT and CoAP by applying the `Tamarin tool` (see e.g. [37], [38]). This tool is developed for the symbolic modeling and analysis of security protocols, where the protocol specification is based on multiset rewriting rules. It can handle protocols with non-monotonic global state and complex control flows such as loops. The authors used the Tamarin prover under the traditional Dolev-Yao (D-Y) [39], the extended Canetti-Krawczyk model (eCK) [40] and also considered Perfect Forward Secrecy (PFS) [41]. The D-Y model considers an insecure wireless channel, where all messages can be intercepted by an adversary, but cryptographic operators do not leak information, e.g. the only way for the adversary to decrpyt an encrypted message is to get the encryption key. In the eCK model the adversary may compromise a limited number of long-term and session keys with the possibility of corrupting random number generators. PFS protects past sessions against future compromises of secret keys. For MQTT the authors in [36] applied Tamarin with respect to two attacker models, the Dolev-Yao and the extended Canetti-Krawczyk. For both attacker models, they successfully verified authentication, key establishment and message secrecy. In case of CoAP two different standard version security primitives such as Public Key Cryptography and Pre-Shared Key are taken into account. They authors consider the same properties for MQTT and managed to find an attack on message secrecy for CoAP using a Pre-Shared Key and an attack on key establishment in the case of using Public Key Cryptography.

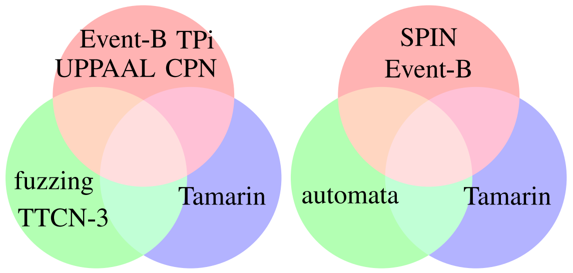

Fig. 2. Formal methods used for MQTT (left) and CoAP (right) with respect to application fields (top: Qualitiative and quantitative analysis, bottom right: Security properties, bottom left: Implementations.

TABLE I
FORMAL METHODS FOR EACH PROTOCOL AND FIELD

Protocol	**MQTT**	**CoAP**
Functional checks	[19], [20], [23], [25], [27], [28]	[28], [31]
Implementations	[33], [34]	[35]
Security properties	[36]	[36]

V. Conclusion

This paper concentrates on formal methods on IoT application layer protocol's, MQTT and CoAP, for improving security and detecting security issues at an early stage. Besides providing information on security issues on application layers in general, and several reported issues, the main contribution of this paper is an overview of application of formal methods for the selected protocols, including details for each approach. That summary reviews existing work, enabling a good starting point for further research.

Existing publications can be distinguished in three fields (see table I). It shows, that the functional checks of protocols - as qualitative and quantitative analysis - is much more common than considering implementations or verifying security properties. It has to be noted, that Quality of Service level is the most often considered term in publications, nearly all publications in subsection IV-A are referencing it. Moreover, there are many more publications dealing with MQTT than with CoAP. Implementations checked with formal methods cover MOSQUITTO, Emqttd and RabbitMQ for MQTT and Californium, CoAPthon and txThings. An overview of methods used for each protocol can be found in Fig. 2, which shows, that nearly for each paper a different method is used (except for UPPAAL used twice). It also shows, that none of the tools is used for different fields of application, e.g. for verifying security properties and performing a qualitative and quantitative analysis.

Open challenges include the formal verification of MQTT v5.0 released in April 2019, since all the publications cover older versions. There, formal verifications for security properties and making sure that encryption and authentication can be covered, might be the most urgent issues to address. Besides checking more implementations for MQTT and CoAP, the influence of different versions of a protocol in the environment of the widespread tools implementing MQTT and CoAP needs to be further investigated.

Another open challenge might be bringing formal verification into play at early stages, when a protocol is still in its standardization process, in order to avoid situations as it happened with MQTT, where the specification contained some gaps possibly leading to misunderstandings.

References

[1] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[2] D. Q. Federico Maggi, Rainer Vosseler, "The fragility of industrial iot's data backbone. security and privacy issues in mqtt and coap protocols," 2018, accessed at: 2019-08-29.

[3] K. Keerthi, I. Roy, A. Hazra, and C. Rebeiro, "Formal verification for security in iot devices," in *Security and Fault Tolerance in Internet of Things*. Springer, 2019, pp. 179–200.

[4] P. Merlin, "A methodology for the design and implementation of communication protocols," *IEEE Transactions on communications*, vol. 24, no. 6, pp. 614–621, 1976.

[5] R. Braden, "Requirements for internet hosts-communication layers," 1989.

[6] ——, "Requirements for internet hosts-application and support," 1989.

[7] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of internet of things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5. IEEE, 2010, pp. V5–484.

[8] S. Marksteiner, V. J. E. Jimenez, H. Valiant, and H. Zeiner, "An overview of wireless iot protocol security in the smart home domain," in *2017 Internet of Things Business Models, Users, and Networks*, Nov 2017, pp. 1–8.

[9] S. Shah, S. S. A. Simnani, and M. T. Banday, "A study of security attacks on internet of things and its possible solutions," in *2018 International Conference on Automation and Computational Engineering (ICACE)*. IEEE, 2018, pp. 203–209.

[10] "Mqtt," accessed at: 2019-08-01. [Online]. Available: http://mqtt.org/faq

[11] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," 2014.

[12] M. E. Whitman and H. J. Mattord, *Principles of information security*. Cengage Learning, 2011.

[13] L. Nastase, "Security in the internet of things: A survey on application layer protocols," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2017, pp. 659–666.

[14] S. Arvind and V. A. Narayanan, "An overview of security in coap: Attack and analysis," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 655–660.

[15] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[16] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using fdr," in *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 1996, pp. 147–166.

[17] F. U. Manual, "Failures-divergence refinement," 2010.

[18] D. Basin, C. Cremers, and C. Meadows, "Model checking security protocols," in *Handbook of Model Checking*. Springer, 2018, pp. 727–762.

[19] B. Aziz, "A formal model and analysis of the mq telemetry transport protocol," in *2014 Ninth International Conference on Availability, Reliability and Security*, Sep. 2014, pp. 59–68.

[20] B. Aziz, "A formal model and analysis of an iot protocol," *Ad Hoc Networks*, vol. 36, pp. 49 – 57, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870515001183

[21] M. Berger and K. Honda, "The two-phase commitment protocol in an extended π-calculus," *Electronic Notes in Theoretical Computer Science*, vol. 39, no. 1, pp. 21–46, 2003.

[22] B. Aziz and G. Hamilton, "Detecting man-in-the-middle attacks by precise timing," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2009, pp. 81–86.

[23] A. Rodriguez, L. M. Kristensen, and A. Rutle, "On modelling and validation of the mqtt iot protocol for m2m communication," *CEUR Workshop Proceedings*, 2018.

[24] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*. Springer Science & Business Media, 2013, vol. 1.

[25] M. Houimli, L. Kahloul, and S. Benaoun, "Formal specification, verification and evaluation of the mqtt protocol in the internet of things," in *2017 International Conference on Mathematics and Information Technology (ICMIT)*, Dec 2017, pp. 214–221.

[26] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, Aug 2015. [Online]. Available: https://doi.org/10.1007/s10009-014-0361-y

[27] J. Hcine and I. B. Hafaiedh, "Formal-based modeling and analysis of a network communication protocol for iot: Mqtt protocol," in *International conference on the Sciences of Electronics, Technologies of Information and Telecommunications*. Springer, 2018, pp. 350–360.

[28] M. Diwan and M. D'Souza, "A framework for modeling and verifying iot communication protocols," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer, 2017, pp. 266–280.

[29] J.-R. Abrial, *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.

[30] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in event-b," *International journal on software tools for technology transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[31] A. J. Vattakunnel, N. S. Kumar, and G. S. Kumar, "Modelling and verification of coap over routing layer using spin model checker," *Procedia Computer Science*, vol. 93, pp. 299 – 308, 2016, proceedings of the 6th International Conference on Advances in Computing and Communications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916314557

[32] G. J. Holzmann, *The SPIN model checker: Primer and reference manual*. Addison-Wesley Reading, 2004, vol. 1003.

[33] K. Mladenov, S. van Winsen, C. Mavrakis, and K. Cyber, "Formal verification of the implementation of the mqtt protocol in iot devices," *SNE Master Research Projects 2016-2017*, 2017.

[34] S. Hernández Ramos, M. T. Villalba, and R. Lacuesta, "Mqtt security: A novel fuzzing approach," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[35] W. Tromp, "Identifying specification violations in coap libraries. an automata learning approach," Master's thesis, Open Universiteit Nederland, 2017.

[36] J. Y. Kim, R. Holz, W. Hu, and S. Jha, "Automated analysis of secure internet of things protocols," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 238–249.

[37] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 696–701.

[38] D. Basin, C. Cremers, J. Dreier, and R. Sasse, "Symbolically analyzing security protocols using tamarin," *ACM SIGLOG News*, 2017.

[39] 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.

[40] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *International conference on provable security*. Springer, 2007, pp. 1–16.

[41] J. Y. Kim, W. Hu, D. Sarkar, and S. Jha, "Esiot: enabling secure management of the internet of things," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2017, pp. 219–229.