This is a repository copy of *A Survey of Practical Formal Methods for Security*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/194011/

Version: Published Version

Article:
Kulik, Tomas, Dongol, Brijesh, Larsen, Peter Gorm et al. (4 more authors) (2022) A Survey of Practical Formal Methods for Security. Formal Aspects of Computing. 3522582. ISSN: 1433-299X

https://doi.org/10.1145/3522582

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

A Survey of Practical Formal Methods for Security

TOMAS KULIK, Aarhus University, Denmark
BRIJESH DONGOL, University of Surrey, United Kingdom
PETER GORM LARSEN and HUGO DANIEL MACEDO, Aarhus University, Denmark
STEVE SCHNEIDER, University of Surrey, Denmark
PETER W. V. TRAN-JØRGENSEN, Bank Data, Denmark
JAMES WOODCOCK, University of York, Aarhus University, United Kingdom, Denmark

In today's world, critical infrastructure is often controlled by computing systems. This introduces new risks for cyber attacks, which can compromise the security and disrupt the functionality of these systems. It is therefore necessary to build such systems with strong guarantees of resiliency against cyber attacks. One way to achieve this level of assurance is using formal verification, which provides proofs of system compliance with desired cyber security properties. The use of Formal Methods (FM) in aspects of cyber security and safety-critical systems are reviewed in this article. We split FM into the three main classes: theorem proving, model checking, and lightweight FM. To allow the different uses of FM to be compared, we define a common set of terms. We further develop categories based on the type of computing system FM are applied in. Solutions in each class and category are presented, discussed, compared, and summarised. We describe historical highlights and developments and present a state-of-the-art review in the area of FM in cyber security. This review is presented from the point of view of FM practitioners and researchers, commenting on the trends in each of the classes and categories. This is achieved by considering all types of FM, several types of security and safety-critical systems, and by structuring the taxonomy accordingly. The article hence provides a comprehensive overview of FM and techniques available to system designers of security-critical systems, simplifying the process of choosing the right tool for the task. The article concludes by summarising the discussion of the review, focusing on best practices, challenges, general future trends, and directions of research within this field.

CCS Concepts: • **Security and privacy → Logic and verification**; *Trust frameworks*; • **General and reference → Surveys and overviews;**

Additional Key Words and Phrases: Formal Methods, model checking, theorem proving, cyber security

This work is supported by the Manufacturing Academy of Denmark; for more information, see www.made.dk. Brijesh Dongol is supported by grants "FaCT: Faithful Composition of Trust," EPSRC grants EP/R032556/1 and EP/V038915/1, and ARC Discovery Grant DP190102142. Steve Schneider is supported by EPSRC grants EP/P031811/1 and EP/R006938/1. Jim Woodcock is supported by the Poul Due Jensen Foundation and grants EP/M025756/1, EP/R025479/1, and IEC/NSFC/170319.
Authors' addresses: T. Kulik, P. G. Larsen, and H. D. Macedo, Aarhus University, Finlandsgade 22, Aarhus, Denmark, 8200; emails: {tomaskulik, pgl, hdm}@ece.au.dk; B. Dongol, University of Surrey, Surrey, United Kingdom; email: b.dongol@surrey.ac.uk; S. Schneider, University of Surrey, Surrey, United Kingdom; email: s.schneider@surrey.ac.uk; P. W. V. Tran-Jørgensen, Bank Data, Aarhus, Denmark; email: peter.w.v.jorgensen@gmail.com; J. Woodcock, University of York, Aarhus University, York, Aarhus, United Kingdom, Denmark; email: jim.woodcock@york.ac.uk.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
0934-5043/2022/07-ART5 $15.00
https://doi.org/10.1145/3522582

ACM Reference format:
Tomas Kulik, Brijesh Dongol, Peter Gorm Larsen, Hugo Daniel Macedo, Steve Schneider, Peter W. V. Tran-Jørgensen, and James Woodcock. 2022. A Survey of Practical Formal Methods for Security. *Form. Asp. Comput.* 34, 1, Article 5 (July 2022), 39 pages.
https://doi.org/10.1145/3522582

1 INTRODUCTION

Digital services are currently spreading to all aspects of society [207]. This in turn causes dependence of society on the cyber infrastructure needed to support these services. The heavy reliance on cyber infrastructure poses new challenges in the form of cyber attacks and potentially cyber terrorism [142], with threat actors encompassing the full range from interpersonal offenders, cyber criminals, and "hacktivists" through to well-resourced state actors [208]. Disturbances in financial, industrial, or day-to-day consumer services could lead to significant financial and societal costs. As digitisation spreads further, the potential attack surfaces only grow larger, increasing the challenge of protecting digital services [250, 260]. As systems grow larger and more complex, significant resources have to be spent to secure these system against known cyber attacks. Often the protection mechanisms are incorporated to close vulnerabilities uncovered after a successful cyber attack, and hence are of a reactive nature. This approach relegates cyber security from a primary challenge to be solved within the system to an afterthought [236].

Due to the wide spectrum of cyber attacks, it is difficult to directly quantify their impact on society [101], however, very often they involve significant financial costs as well as potential disruptions in quality of life. One example is a potential cyber attack against electricity infrastructure, including electricity marketplace, which could lead to destruction of generators and disclosure of confidential data [191]. Another example is attacks against manufacturing facilities causing delays or decrease in quality of production [47, 204]. These examples demonstrate that cyber threats should be considered as significant as physical threats against societal infrastructure.

The earlier the potential cyber security threats are discovered within new systems, the cheaper the mitigation for these threats will be [257]. **Formal Methods (FM)** provide an opportunity for discovery and mitigation of cyber threats at all stages of the lifecycle of a system. Using FM brings mathematical rigour to the field of cyber security assurance. This is possible, since FM are techniques that use model-based approaches, where the models are rigorously specified [261] that allow for development of precise statements about what systems under investigation should do without putting constraints on how to do it [264]. These models represent the software, hardware, or a combination of the two for the system in question. The primary benefit of using FM stems from the mathematical proof of the internal consistency of the system design [115]. This proof provides strong assurances, since it considers the entire system behaviour, and once proven true it remains true, whereas in traditional testing it is only possible to cover specific scenarios. FM can be seen as a tool well suited for providing assurances of cyber security for digital society [262]. Beyond the assurance of behavioural correctness of a system, the adoption of a fully fledged formal approach is known to reduce the number of implementation errors, which are the building blocks of exploits.

It is important to note that there exists a variety of FM. The main categories we consider are:

- *Theorem Proving*, analysing a formal description for important properties based on computer-based proofs.
- *Model Checking*, checking whether a finite-state model of a system meets a given specification in an exhaustive manner.

- *Lightweight FM*, using formal techniques to analyse a system either statically or dynamically (this concept was coined in Reference [132], but we have extracted the model checking from their characterisation into a category of its own).

For a detailed coverage of the technical foundations of formal approaches to security, we refer the reader to Reference [32]. That chapter focuses on the technical foundations of the area, whereas our survey concentrates on specific application areas.

In all cases, the methods are applied to determine if a system behaves in a correct way, and many approaches have received significant tool support for automation of the verification and validation process [10]. In this survey, we consider all of the approaches and their application to specific areas of digital society. We further consider FM as applied to the specific level of abstraction of system behaviour ranging from the application level to the hardware level. By considering the state-of-the-art research in formal verification across these dimensions, we provide a non-exhaustive overview of application of FM in specific disciplines. The aim of this survey is to allow practitioners to identify a proven method applicable to a system in their domain, hence hopefully increasing the adoption of FM in the field of cyber security. We believe that similar surveys in different areas of use of FM could be a catalyst to increased adoption, as it has been determined that education and experience as well as finding positive examples of use of FM could lead to professionals adopting FM towards their area of expertise [104].

1.1 Methodology

The amount of research publications within the area of applying FM towards cyber security challenges is significant. Therefore, several constraints have been placed on the choice of research publications to be considered within this survey. The first important constraint is the recency of the research reported, considering the landscape of *the past decade*, limiting the publication date to be no earlier than 2012. Furthermore, all of the research work needs to be *published in scientific venues* such as journals, conferences, or workshops. The next constraint is focus on computer-based tool supported FM, i.e., only FM with tools that can provide *computer-based analysis* and often guide users on performing this analysis are considered. This consideration is to focus more on the FM that could be potentially applied outside of academia, bringing the benefits of the *formal security analysis to industry*. This goes hand-in-hand with our focus on the applied FM, searching for research publications, where a tool-supported FM is utilised to deal with *a concrete cyber security problem*. Hence, this survey does not focus on theoretical advances of FM in security or proposed processes that briefly mention use of FM, such as theoretical approaches to model checking algorithms, specification of hyperproperties, and similar. Furthermore, our survey does not cover the approach to security commonly referred to as *provable security*. This refers to a mathematical approach to analysing the security of cryptographic mechanisms or systems. The approach considers the system in the context of an attacker model and expresses the security requirements within that model as a limitation on what the attacker should be able to achieve. A proof consists of establishing that the attacker would need to break a known hard problem (such as the Quadratic Residuosity Problem [105]) to break the security of the system. Thus, the security of the system is reduced to the difficulty of the underpinning hard problem. This approach is typically used within the field of cryptography rather than secure systems, and so falls outside the scope of our survey. We point the reader to Reference [28] providing the report within the area of FM in cryptography. Finally, we constrain our search to research that considers aspects of security explicitly, and not as a by-product of safety or correctness. The search for the research publications was carried out as a cross database search using Google Scholar, while focusing on research papers, excluding research abstracts or extended abstracts.

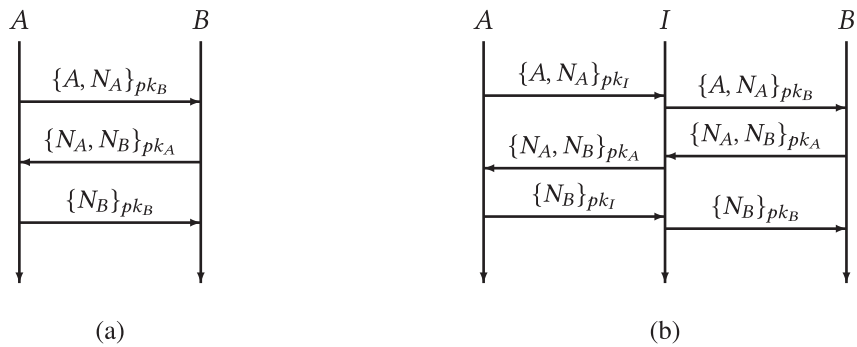

Fig. 1. (a) Needham-Schroeder authentication protocol and (b) attack.

As this survey provides readers with a quick overview of the research conducted, we have further decided to categorise different research publications by the industry (domain) on which they focus as well as the level of abstraction on which FM is utilised. In this way, researchers and also potentially industrial users can quickly find the area of their interest within this survey. Furthermore, we classify the research based on the cyber security problem classification as elicited from the discovered research papers and inspired by existing literature [208].

1.2 History

This section presents a history of impactful research works within FM in security over the past 40 years. We choose four case studies where formal methods have been applied to secure systems:

(1) The Needham-Schroeder Public-Key Protocol. Lowe used a refinement model checker to find a triangular attack on the protocol. This was a new attack on a protocol that had previously been proven correct by Burrows et al. [52].
(2) The Mondex smartcard. This was the first commercial product to be certified to ITSEC Level E6. There was considerable discussion at the time as to whether this was even possible.
(3) The Tokeneer ID Station. There were similar questions about the feasibility of using FM to achieve the level of rigour required by the higher assurance levels of the Common Criteria. Tokeneer settled this matter.
(4) The seL4 Microkernel. This system has the reputation of being the world's most assured microkernel. Significantly, it demonstrates that security and the use of formal methods do not lead to poor performance.

The Needham-Schroeder Public-key Protocol. The Needham-Schroeder Public-Key Protocol is a transport-level protocol for communication between network devices [190], providing mutual authentication between two parties in a network. The protocol is visualised in Figure 1(a). Simple and well known, it has become a popular benchmark for testing security protocol verification technology. We discuss it here because it is an important security protocol that nevertheless contained a significant error. This error was found through formal modelling and analysis.

Lowe [165] showed that, contrary to its intention, the protocol fails to ensure authentication. In particular, he demonstrated that an intruder can impersonate an agent A during a run of the protocol. The impersonator tricks another agent B into thinking that they are talking to A.

The protocol uses public key cryptography. Each agent A possesses a public key, which any other agent can get from a server. A also possesses a secret key that is the inverse of its public key.

Any agent can encrypt a message using A's public key, but only A can decrypt it, ensuring secrecy. The protocol also uses *nonces*: random numbers coined for single runs of the protocol.

Lowe encoded the protocol in CSP [123] and analysed it by use of CSP's model checker, FDR [103]. Lowe did not direct FDR where to look in the protocol for a vulnerability—he simply modelled the intruder capabilities, but the exhaustive search carried out by the model checker found an attack in spite of this.

Suppose that I (an intruder) is a network user who can take part in network sessions. I can also intercept messages and inject new ones, but is not able to decrypt messages without the key. I can produce a new message in two circumstances: if I invents the nonce or if I already understands the message's contents. This intruder can also replay complete encrypted messages, even without understanding the contents [213]. This approach is commonly known as the Dolev-Yao model [85].

The attack involves two simultaneous runs of the protocol, as shown in Figure 1(b). A establishes a valid session with I. At the same time, I impersonates A to establish a fake session with B. The flawed run of the protocol could be explained as follows: A sends a message with nonce N_A to I, who decrypts the message with I's secret key. I relays the message to B, pretending to B that A is communicating. B sends N_B in response, encrypted for A, and so I relays this encrypted nonce to A. A decrypts N_B and confirms it to I, who learns it. I re-encrypts N_B and returns it to B, which convinces B that A is the other party. At the end of the attack, B falsely believes that A is the communication partner and that only A and B know N_A and N_B. This shows that the protocol is insecure. Protocol analysts call this a *man-in-the-middle attack*. Here, it has been discovered automatically.

Mondex. The Mondex application consists of smart cards with electronic purses (wallets) for use in the electronic commerce [235]. Customers use Mondex smart cards for low-value, cash-like transactions that need no third-party involvement. The Bank of England (the financial regulator in this instance) considered the requirements for Mondex to be security-critical: Mondex must have no implementation or design bugs that could allow electronic counterfeiting. So the developers certified Mondex to the highest standard available at the time. This was ITSEC Level E6 [129], equal to Common Criteria Evaluation Assurance Level 7 [57].[1] Mondex was the first commercial product to achieve ITSEC Level E6 (EAL7).

Reference [235] further describes the development of the Mondex application, with its abstract and concrete models. The abstract model describes the world of electronic purses: Atomic transactions exchange value and the abstract model expresses their required security properties. The concrete model is the purse design and the message protocol for value exchange.

The design team used the Z notation [233, 263] to specify both models. They proved that the concrete model is formally a refinement of the abstract one. This means that the concrete model respects all the abstract security requirements. The abstract model and its security properties is often easier to understand than the concrete model. Developers wrote manual proofs, believing that no efficient automated tools existed for such a large task. Instead, proof steps were type-checked using the *f*UZZ[2] and Formaliser tools [97]. Proofs were also checked by independent external evaluators.

[1]The levels of the Common Criteria are:

EAL1:	Functional testing.	EAL5:	Semi-formal design and testing.
EAL2:	Structural testing.	EAL6:	Semi-formally verified design and testing.
EAL3:	Methodical testing and checking.	EAL7:	Formally verified design and testing.
EAL4:	Methodical design, testing, and reviewing.		

[2]See spivey.oriel.ox.ac.uk/corner/Fuzz_typechecker_for_Z.

There were four principal security properties:

- The system and its users may not create value.
- The system must account for all value.
- Purses must have enough value for their intended transaction.
- All transfers must be between authentic purses.

The design team changed a secondary protocol after the proof revealed a bug. A detailed account of the project is given in Reference [265]. Mondex has proved to be a dependably secure system, guaranteed by its formal development.

Tokeneer. The Tokeneer system was designed and developed by the **US National Security Agency (NSA)** [30].[3] It provides secure access to an enclave of workstations with controlled physical entry. Access control requires biometric checks and security tokens. These tokens describe a user's permitted actions within a particular visit to the enclave.

Developers needed to assure the security properties. They did this by conformance with the Common Criteria Evaluation Assurance Level 5 [57]. They also needed to show they could do this in a cost-effective way. NSA invited bids to use FM to develop a component of the Tokeneer system and then monitored this experiment to measure the effort and skills needed to perform the development.

Praxis (a UK company) won the contract and wrote a formal specification in Z [233, 263], formally refining the specification to a SPARK program. SPARK is a subset of Ada with an accompanying tool-set [29]. They proved key system properties and the absence of runtime errors, using traditional methods to develop extra software. These extra Ada programs provided interfaces with peripherals.

The project required 260 person-days, three people part-time, and nine months' elapsed time. It produced about 10K lines of SPARK code with about 16.5K contracts. About 200 loc were written on average per day during the implementation phase, with about 40 loc through the entire project. A further 3.5K lines of standard Ada code were produced, with about 200 loc per day in the implementation phase or 90 loc throughout the project. System testing took about 4% of the project effort, much smaller than usual.

Two defects were found in Tokeneer. One was found using formal analysis, another was found by code inspection.[4] The testing team discovered two in-scope failures: missing items in the user manual.

The task set by NSA was to conform to Common Criteria EAL5. The Tokeneer development actually exceeded EAL5 requirements in several areas: configuration control, fault management, and testing. Although the main body of the core development work was carried out to EAL5, the development areas covering specification, design, implementation, correspondence were accomplished to EAL6 and EAL7. Why? Because it was cheaper!

The seL4 Microkernel. The third-generation microkernel seL4 provides abstractions for virtual address spaces, threads, and inter-process communication. It provides an explicit memory management model and capabilities for authorisation. There is a guarantee that the binary code of

[3]For comprehensive information on Tokeneer, see the AdaCore webpages www.adacore.com/tokeneer, where the entire project archive can be downloaded. AdaCore distributes the material generated by Altran under contract to the NSA under the terms of the Technology Transfer Agreement agreed between Praxis and the NSA. This material consists of all the core and support software for the Tokeneer ID Station, project documents, test cases derived from the system test specification, test tokens, and biometric data.

[4]Diomidis Spinelli: www.spinellis.gr/blog/20081018/.

the ARM version of the seL4 microkernel is a correct implementation.[5] seL4 meets its abstract specification and *does nothing else*. In particular, the seL4 ARM binary meets the classic security properties of integrity and confidentiality.

The seL4 micro-kernel has a formal proof of its C code against its abstract specification [144]. This proof is machine-checked in Isabelle/HOL [194]. This assumes correctness of boot code, cache management, hardware, and hand-written assembly code.

The developers claim seL4 to be the only verified general-purpose **operating system (OS)** kernel. An operational model of the system forms as an abstract specification. A Haskell program prototypes the kernel. This prototype provides an automatic translation into Isabelle/HOL. The Isabelle code is then an executable, design-level specification of the kernel. This is hand-coded in C to form a high-performance C implementation of seL4. Refinement proofs link the specifications and the C code. Developers proved that attackers cannot subvert the kernel, not even if they use buggy encodings, spurious calls, or buffer overflow attacks.

1.3 Definitions/background

Here, we provide a set of common terms and definitions used throughout this article. This is particularly important, since the fields of formal verification and security developed independently for many years, and hence, some terms are overloaded and have slightly different meanings, depending on the context in which they are used. For example, in security (particularly cryptography), a ***certificate*** refers to a document that is used to bind an entity to a cryptographic key. However, within FM, ***certificate*** is used as a proof of correctness of a system or protocol.

Throughout this survey, on the ***security*** side, we use terms such as ***authentication*** to refer to the process of *identifying* and *validating* whether a user (an entity or individual) accessing a system is who the user claims to be. This is in contrast with ***authorisation***, which is the process of *allowing* a user access to a system based on their identity. We furthermore often find security protocols designed to provide specific properties such as ***isolation***, which is a design principle in which processes are separated and given privileged access to shared resources, e.g., shared memory (typically using techniques such as containerisation or virtualisation) or ***non-repudiation***, ensuring that it is not possible to deny a validity of a statement or a message, especially in terms of its authorship. We would also like to point out the difference between ***anonymity*** where an identity of a user or a process shall not be disclosed and ***confidentiality*** where the content encoded as a block of data shall not be disclosed.

On the ***formal methods*** side it could be said that systems under consideration typically comprise a set of coordinated ***processes***, which are program instances defining a set of instructions that are executed by one or more threads. We think of processes as being active entities in a system, as opposed to programs that are passive entities. Formal frameworks to describe the behaviour of processes include CSP, CCS, ACP, π-calculus, and so on. Processes typically implement ***protocols***, i.e., a set of rules for transmission of data, and may synchronise over ***shared memory*** or communicate over a ***channel***, which is an abstraction of a physical communication network. Shared memory implementations are increasingly complex due to the use of intermediate processor caches and may implement many different consistency models [6]. Similarly, one may place many different assumptions on a channel, e.g., FIFO ordering of messages; whether the channel guarantees integrity, availability, and confidentiality; whether the channel is error-free; whether message types can be distinguished, and so on.

[5]On the ARM platform, there is a further proof that the binary code that executes on the hardware is a correct translation of the C code for sel4. This means that the compiler does not have to be trusted, and extends the functional correctness property to the binary. See docs.sel4.systems/FrequentlyAskedQuestions.html.

In general, verification is with respect to a ***specification***, which is an abstract (formal or informal) description of the allowable behaviours of an entity, e.g., hardware, a system, computer program, data structure, and so on. Formal verification often proceeds with respect to a ***model*** of a system, which provides a precise formal description of an entity, capturing the key characteristics of the entity being modelled. One must ensure that every feature described by a model is an actual feature of the entity. Different models of the same entity may be developed, depending on the properties that are of interest; a computer program for example may be modelled by relations between pre/post state; traces of states; functions between inputs and outputs, and so on. A model may describe behavioural functionality, protocols, and so on. In FM, one typically develops models at several levels of abstraction, with precise descriptions of the relationship between these levels. FM for security also requires a model of an attacker, e.g., the Dolev-Yao model, which is used in the context of communicating systems.

Within hardware verification, the term ***co-verification*** is used to prove that system software executes correctly on a *representation* of the underlying hardware design. It enables integration of software with hardware, before any physical devices (e.g., chips or boards) are available. The aim of verification is to ensure that it meets its ***implementation*** (of a specification), i.e., the physical manifestation of an entity. In some instances, one may refer to an implementation as a model that provides enough detail about an entity for the corresponding physical entity to be readily obtained.

To finalise the definitions it is important to revisit our classifications of FM. The first category we classify is ***theorem proving***, where a proof of correctness of a system is provided in symbolic logic. This survey focuses on automated theorem proving, where proof assistants are utilised to generate the proofs. This method utilises a system of logic trying to determine if a statement ϕ follows from a set of statements $\Gamma = \{\psi_1, \ldots, \psi_n\}$. The second class we consider is ***model checking*** where a finite state model of a system is utilised to systematically search the entire state space of this model (i.e., all possible system states contained in the model) for detection of counter-examples to statements about the system expressed as properties, for example in linear temporal logic as $\Box\Diamond p$ stating that a property p holds in infinitely many states. As with theorem proving, we focus on automated model checking. Finally, we define a class of ***lightweight formal methods*** where we place methods that do not provide exhaustive analysis. A simple example could be static code analysis, where the code is analysed while not running to determine if it breaks predefined rules. Another example could be VDM [38], where the system is modelled in a formal modelling language and properties are expressed as contracts, i.e., pre, post conditions and system invariants, for example ***inv*** $t == t.issueTime < t.expirationTime$ stating that universally t must be issued before it can expire. To perform the check, the model is animated and only specific scenarios are set (often those considered critical), however, VDM can also utilise combinatorial testing combining input paths to generate large amount of tests, hence significantly increasing the test coverage.

2 SURVEY

2.1 Categorisation and Overview

Since FM in security are applied across many domains, we structure the scope of the survey by presenting a categorisation based on a domain and a level of abstraction. This is done to provide a systematic overview of the wide field of FM in security. The labels for the four domains we have selected are:

Financial (Section 2.2): Aggregates the works applying FM in the area of finance/money as payment systems, home banking, financial markets, crypto-currencies. Examples are mobile banking apps, ATM infrastructures, the FIX stock exchange protocol, smart-cards/hardware wallets.

Industrial (Section 2.3): This label agglomerates works dealing with computing systems applied in the production of goods or services, manufacturing, and industrial control. Examples are a Water Treatment Management Panels, PLC control networks, Modbus/TCP, motor controllers.

Consumer (Section 2.4): It categorises works focusing on the security of end-users'/individuals' personal computation devices and applications such as a command-line shell, a home operating system, a Voice over IP protocol, and an exercise smart appliance.

Enterprise (Section 2.5): This is the dual of the Consumer category, as it is used to group the works focusing on the security of corporate systems providing computing services satisfying the needs of organisations instead of individuals. Examples are email services, e-government systems, the sn2 protocol, data server warehouses.

As presenting the four domains would only separate the FM in security research by the field of application, we further present five levels of abstraction at which the formal verification is carried out. These levels of abstraction are:

Application: Used for works that apply FM for security at the application or purpose of computation level.

System: Used for works that apply FM for security within the architectural level, often encompassing multiple subsystems.

Protocols: It is used to apply FM to assert properties or analyse communication protocols between system components level.

Implementation: This is a cross-cutting category encompassing all the works that focus on application/usage of FM directly on the resulting system (e.g., runtime monitoring) instead of emphasis on designs and specifications.

Hardware: Used to classify works applying FM in the process of hardware development.

This categorisation allows us to systematically review the state-of-the-art-research and provide an overview based on this. To provide a clear overview of FM in security, we further apply a third dimension, defining the type of the FM used, i.e., model checking, theorem proving, and lightweight FM. This provides a quantitative overview of different research works within Figure 2.

The sections within this survey follow a logical organisation, where the research works are grouped together by the type of the application, system, protocol, implementation, or hardware that they are applied to. Within this grouping, the research works are further organised into paragraphs following logical categorisation. As an example, the first paragraph could consider works related to manufacturing, while the next paragraph considers works related to industrial control. In both cases the research is aimed at industrial domain but with a different scope. Each section within the survey represents a single domain, where we present a systematic summary of research works belonging to different levels of abstraction. As this survey attempts to categorise large amount of research works, each of them is only presented briefly focusing on the analysed problem and utilised FM technology. For more details on individual research works, we point the reader to a technical report that this survey is based on [149].

2.2 Financial

Financial computing, including banking systems, independent budgeting applications, and mobile payment applications, is a rapidly developing field. This section provides an overview of how FM have been used to analyse the security of banking mobile applications, alternative currencies, such as cryptocurrencies, smart contracts, banking backend systems, electronic trading systems,

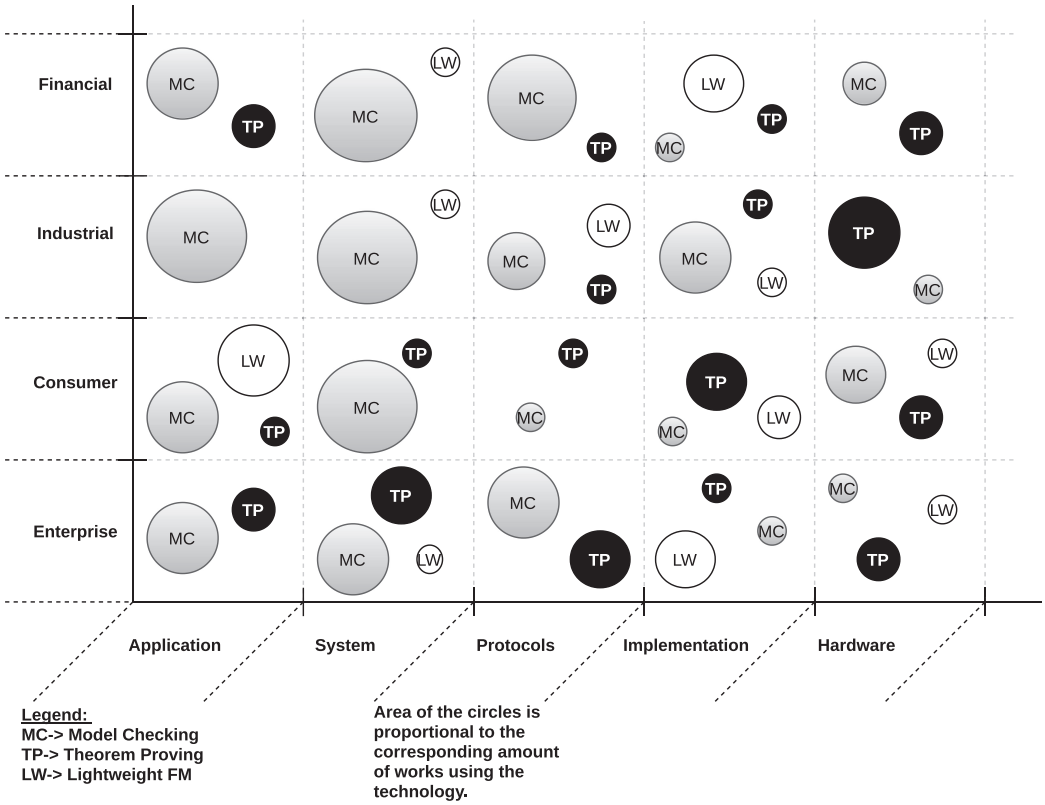

Fig. 2. Classification of formal methods in security.

payment protocols, cryptocurrency hardware, and wallets. On a different level of abstraction, it could be seen within this section that the *system* level consists of most research works. This section also mentions the legal challenges when applying FM to financial systems. These legal challenges arise from limited access to these systems and a certain level of avoidance of publication of potential vulnerabilities by vendors of these systems, i.e., often making it difficult for researchers to get deep insight into these systems.

The method most used to analyse security within the financial domain is model checking, and authors apply different model checkers to specific problems. This could be attributed to the fact that the different entities whose security is being analysed lend themselves well to be modelled in a state transition representation and also that their state space is sufficiently limited to be analysed without issues such as the state space explosion. The cyber security topics present within the financial section are shown in Figure 3.

Application. Nowadays, banks not only provide mobile applications, but whole alternative currencies are being developed. This rapid growth provides many opportunities for use of FM on an application level.

An application of FM to banking apps could be found in Reference [63], where the authors analyze security of apps from 15 leading UK banks discovering several vulnerabilities. The authors proposed a correction to one of the flaws, which was formally verified using ProVerif.

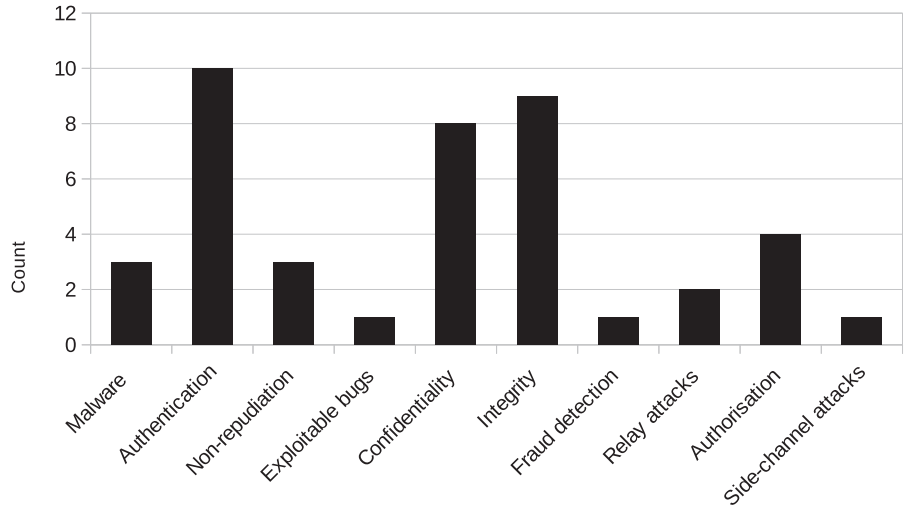

Fig. 3. Cyber security topics in the financial section.

Another widespread security threat in banking is malware. In this area, the authors of Reference [152] analysed Android banking applications using Krakatau byte-code tool[6] to generate the Java byte-code of the application, further translating it to **Calculus of Communicating Systems (CCS)** [181]. The authors then dispatched the CCS model to the **Concurrency Workbench of New Century (CWB-NC)** model checker [69] searching for malware properties. The authors have accomplished 98% malware detection rate. Similar approach was utilised by the authors of Reference [126] focusing on the banking SMS messages.

In the alternative currency area, the authors of Reference [249] analyse the Electrum Bitcoin wallet by creating a model in ASLan++ [253] of the two-factor authentication utilised by the wallet. The authors have uncovered potential vulnerabilities using the Cl-Atse protocol analyser [248].

Smart or programmatic contracts are also an important aspect of modern financial landscape. The automated contract enforcement requires an implementation that shall be free of vulnerabilities. This led to use of FM for creation of certified contract languages [17] and certified virtual machine byte-code [199].

A specific survey has been carried out in this area, providing more detail on utilisation of FM [180].

System. The financial systems of today could be categorised as classic systems such as SWIFT inter-bank network system and new systems often introduced by smaller players or by regulatory pressures [259] that tend to push the sector to move faster.

Most of the FM works related to classic systems are decades old with few works such as use of BAN logic for analysis of mobile payment system [7] or use of SPIN tool for analysis of ATM systems [197] or internet payment systems [272]. In Reference [216] the authors show legal barriers faced by proponents of FM in the domain.

Within the area of **Electronic Trading Systems (ETSs)** there is a trend of application of FM to decentralised systems such as blockchain-based cryptocurrencies. These works range from verifying algorithms in a cryptocurrency platform [268], analysis of Ethereum smart contracts [121], to verification of the blockchain system as a whole [87]. Other works have considered building

[6] github.com/Storyyeller/Krakatau.

models of Europay Master Visa standard by use of automata learning techniques [2] or use of lightweight formal specification in fraud detection [209].

Protocols. Protecting financial transactions is a work well suited for FM. Specifically in the case of **Near Field Communication (NFC)**, a short-range radio communication technology, utilised in contactless payments. This method has several vulnerabilities [175], where the authors of Reference [168] propose and verify an NFC protocol suing the Scyther tool [73] addressing several vulnerabilities. In similar fashion, in Reference [5] the authors propose a protocol for securing of NFC payments and analyse it using the FDR model checker. Authors of Reference [11] have focused on analysis of security of NFC enabled forgery protection also utilising the FDR model checker, discovering several potential attacks and providing mitigating measures. In the field of authentication protocols the authors of Reference [266] verified the mutual authentication properties of a secure electronic protocol using SPIN tool, discovering several vulnerabilities. Similarly in Reference [118] the authors have analysed a biometric transaction authentication protocol using ProVerif [41] and proposed fixes to discovered vulnerabilities.

Finally, the authors of Reference [44] have proposed a secure SMS based protocol for mobile payments and analysed it against several security properties using AVISPA [21].

Implementation. Since vulnerabilities in financial software could lead to financial losses, use of FM can provide a substantial benefit in this area. Smartphone applications are often used as a gateway to financial services. The authors of Reference [127] utilised static analysis tools, discovering that financial applications from developed countries contain less vulnerabilities than those from developing countries. Similarly, the authors of Reference [241] have statically analysed over 10,000 Android applications to compare the security of financial applications with the rest, which led to a discovery of a worrisome trend where the analysed applications have gained more vulnerabilities within a span of two years.

The authors of Reference [98] have modelled 80% of EMV2, a successor to EMV, in VDM to provide a formal model for the implementation and analyse security attributes of EMV2. The authors have further attempted to code generate parts of EMV2 to Java directly from the VDM model.

Another aspect of financial software is use of open APIs. The authors of Reference [94] have modelled a financial grade OpenID API as a set of theorems, discovering several vulnerabilities and proposing fixes to these. Finally, the authors of Reference [16] have analysed bitcoin contracts using UPPAAL, determining the secure time to live within the contract protocol. Similarly, other types of smart contracts are being utilised [35]. The authors of Reference [199] have verified Ethereum smart contracts such as the ERC20 token contract [90] using K-framework's reachability logic theorem prover [234], discovering that the token implementation that diverges form the ERC20 specification contains vulnerabilities.

Hardware. In the area of contactless payments, the NFC hardware can pose security challenges. To address one of these challenges, the authors of Reference [62] have introduced a scheme to prevent relay attacks based on a distance bounding protocol [46] and verify this scheme using ProVerif.

Cryptocurrency is often associated with a specific hardware, where in several cases FM were used for security improvement. To this end, the authors of Reference [22] have proposed a device for approval of security critical operations. The authors have verified a property of deterministic start of the device using an SMT solver. Similarly, the authors of Reference [171] have utilised theorem proving to check an unforgeability property of a hardware wallet to answer a question: *what if the manufacturer of the wallet cannot be trusted?* Finally, the authors of Reference [19] have

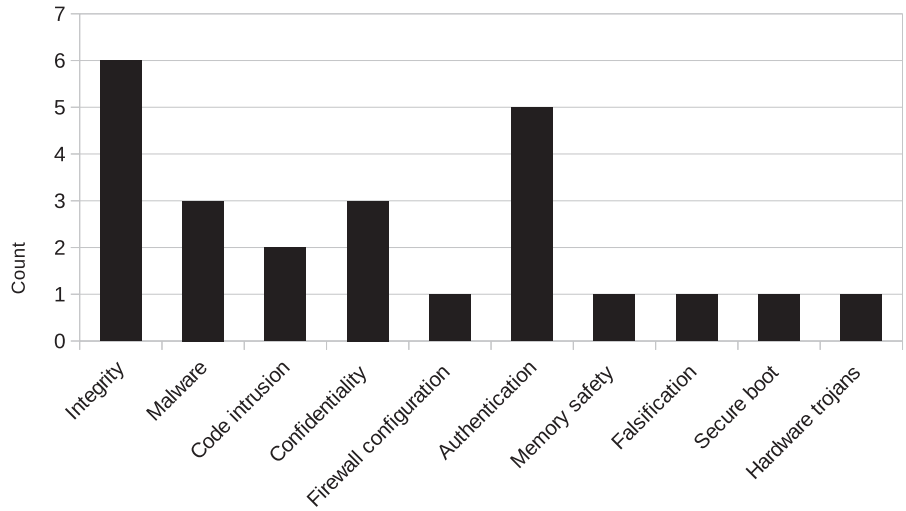

Fig. 4. Cyber security topics in the industrial section.

also attempted to prove security properties of several hardware wallets using theorem proving, showing that the wallets were secure only under specific assumptions.

2.3 Industrial

Industrial processes are a backbone of a modern society, as they provide control not only for production of necessary goods, but also utilities such as electricity and water treatment. This section provides an overview of how FM have been utilised in security analysis of automotive control applications, robotic applications, PLC software, industrial communication protocols such as Modbus and OPC UA, SCADA systems, and hardware devices underpinning industrial computing. An interesting note is that the research works are distributed uniformly across the different levels of abstraction, demonstrating that all aspects of industrial computing have been scrutinised using FM to provide either security analysis or security assurances. In the industrial application of FM the problem is often considered domain-specific, i.e., cyber security properties are based on whether the considered industrial system is, for example, an automotive controller or a water treatment plant.

As within the *financial* section, the most-used FM to analyse the security properties is model checking. This could once again be attributed to the nature of the problem where, for example, PLC programs and industrial processes lend themselves to be easily modelled using state transition systems. As some of the industrial computing is complex, the problems are often modelled more abstractly to avoid the state space explosion problem. Within the hardware level of abstraction, however, in industrial computing, theorem proving is often the FM of choice, as it allows description of the hardware in more detail. The cyber security topics within the industrial section are shown in Figure 4.

Application. Industrial applications are often used to control critical processes, hence FM could provide strong assurance of security. The authors of Reference [120] utilised model checking based on automated translation of automotive ECU applications to CSP [122] and subsequently dispatching the model to the FDR model checker. Discovered counter examples were then provided to the implementation team.

Industry of the future utilises interconnectivity and robotics. In Reference [193] the authors have analysed a security of an application controlling a cap attaching robot by creation of a model using Maude [179] and two attacker models, discovering possible attack vectors. Similarly, the authors of Reference [254] analysed security of applications based on the Robotic Operating System [205], expressing the security properties in CTL and utilising the UPPAAL model checker [159]. The authors then automatically generate the implementation C++ code.

The authors of Reference [177] have focused on security analysis of a water treatment SCADA system by use of the system logs. The application behaviour was modelled as timed automata, while the security properties have been expressed in timed temporal logic [12], providing 100% attack detection rate based on log data. The authors of Reference [273] have successfully utilised the Z3SMT solver [80] for PLC malware detection, while the authors of Reference [146] used the NuSMV model checker [65] for automated detection of intrusion code, demonstrating the usefulness of FM in this area.

System. Cyber attacks against industrial systems could have severe consequences [258]. To mitigate this, the authors of Reference [211] have created a formal model in ASLan++ of a real-world industrial control system and carried out attacks utilising a Cyber Physical Dolev-Yao attacker [210]. The analysis with help of CL-Atse analyser has discovered seven out of eight possible attacks. Similarly, the authors of Reference [82] mitigate the cyber attacks by proposing a formally verified security framework for industrial control systems. The verification was carried out using ProVerif and proved the security aspects of the system utilising the framework. Furthermore, the authors of Reference [114] verified a PLC program using timed automata and UPPAAL ensuring that the program was not compromised. In a similar fashion the authors of Reference [255] have modelled a water-level control system in timed automata utilising the PAT model checker [237] to successfully verify security recovery mechanisms of the system.

Within the area of connectivity the authors of Reference [214] have demonstrated formalisation and analysis of firewall rules using the Z3 SMT solver, lowering the errors in firewall configurations. In Reference [150] the authors have utilised TLA+ [158] to ensure effectiveness of mitigations strategies against several cyber attacks, while the authors of Reference [246] have used combinatorial testing within VDMJ [161] to generate 145 million tests for a formal model expressed in VDM-SL [160] of an industrial control system, providing assurance for several security properties.

Protocols. Industrial communication protocols carry critical data, requiring a high level of security. The authors of Reference [224] have formally analysed security of the Modbus/TCP using Coloured Petri Nets [138] combined with Formal Component Analysis [201] discovering a possible attack. Similarly, the authors of Reference [187] modelled the Modbus protocol as a Dynamic State Machine [186], automatically translating it to Promela for verification using the SPIN model checker. The authors have uncovered a possible man-in-the-middle attack. Another protocol, OPC UA, has been analysed by the authors of Reference [203] using ProVerif finding vulnerabilities in the authentication sub-protocol. In similar fashion, the authors of Reference [86] analysed several Modbus and OPC UA authenticity and integrity properties using TAMARIN theorem prover [176], discovering the necessity for secure channels. In Reference [15] the authors have formally analysed the authentication properties of DNP3 protocol utilising the CPN state space analysis tool [137], discovering a potential for replay attack. Finally, the authors of Reference [49] have analysed an authenticated CAN protocol using ProVerif, discovering that limited use of cryptography allows for a replay attack with partially modified data.

Implementation. Since industrial software often consists of a large codebase it could be difficult to be formally analysed in its entirety. Over the years, several approaches to code analysis have

been presented; for example, the authors of Reference [18] use a dialect of UML, SysML-Sec to model of a large industrial codebase, iteratively refining the model and automatically translating it to π-calculus using the TTool [88] and analysing it using ProVerif. The authors propose for this approach to be integrated to the software development process.

In the aeronautics industry the authors of Reference [70] have created a formally verified implementation of unmanned aerial vehicle using several tools; for example, the jKind model checker [99], satisfying correctness of components and Isabelle/HOL theorem prover [194] to assure that system execution semantics matches the model. This approach has successfully prevented cyber attacks against the vehicle.

As PLCs are the backbone of industrial automation, a lot of focus have been put into ensuring that the PLC programs satisfy security properties. For example, in Reference [223] the authors use state transition diagrams of PLC programs as a basis for formal model dispatched to NuSMV model checker, while expressing the security properties in LTL. Similarly, the authors of Reference [247] utilised Petri Nets to develop software falsification detection by translating Petri Nets to Promela and dispatching it to the SPIN model checker, while modelling the falsification properties in LTL.

Hardware. FM for hardware verification is a well established field with specification languages such as Verilog [243] and VHDL [188]. The authors of Reference [124] focus on co-verification of **Intellectual Property Blocks (IPs)** for use within **System on a Chip (SoC)** architectures, considering technologies such as secure boot and concurrency in a time-of-check-to-time-of-use considerations [48]. The authors utilised semi-automatic co-verification methodology using a toolchain comprised of Boogie [31] as intermediate verification language, through Corral software verifier [157] and SMACK [206] for bit-precise checking with an ultimate goal of producing secure SoCs. The authors of Reference [162] consider that all software layers could be compromised and have developed an application-specific hardware monitor based on a formally analysed C code and a junction box validated in a hardware description language with a goal to monitor the hardware controller for malicious activity. The authors model their hardware monitor in Frama-C [74] with Jessie plugin, allowing for automatic deductive verification using Why [95].

Within the area of integrated circuits, the authors of Reference [164] consider the trustworthiness of hardware using ***Proof Carrying Code* (PCC)** [189], utilising Coq to derive theorems for the hardware descriptions annotated with PCC. This has been later extended in Reference [110] to a notion of a ***Proof Carrying Hardware* (PCH)**, utilised to verify security of IPs supplied by untrusted vendors by extending the VHDL and utilising Coq to carry out the verification of security theorems. Finally, the authors of Reference [3] consider the possibility of hardware trojans being injected during manufacturing and utilise the nuXmv model checker [56], while specifying the hardware properties in LTL to detect these trojans.

2.4 Consumer

Consumer computing such as use of personal computer, smartphones, and underlying connected services is an integral part of modern life. Consumer computing has often been characterised as of less critical nature than for example industrial systems, however, this view is changing as society introduces more digital technologies to everyday life. This section provides an overview of use of FM in analysis of cyber security of consumer computing, ranging from consumer electronics for fitness equipment, mobile operating systems, web browsers, consumer **Internet of Things (IoT)** devices to commodity hardware for devices such as personal electricity meters. An interesting fact within the consumer domain is significant use of so called lightweight FM, utilised often not only on the application level of abstraction, but also considering implementation and hardware. One challenge in formal analysis of consumer systems is a rapid nature of evolution of these systems, where the competition in consumer markets often forces fast adoption of new technologies.

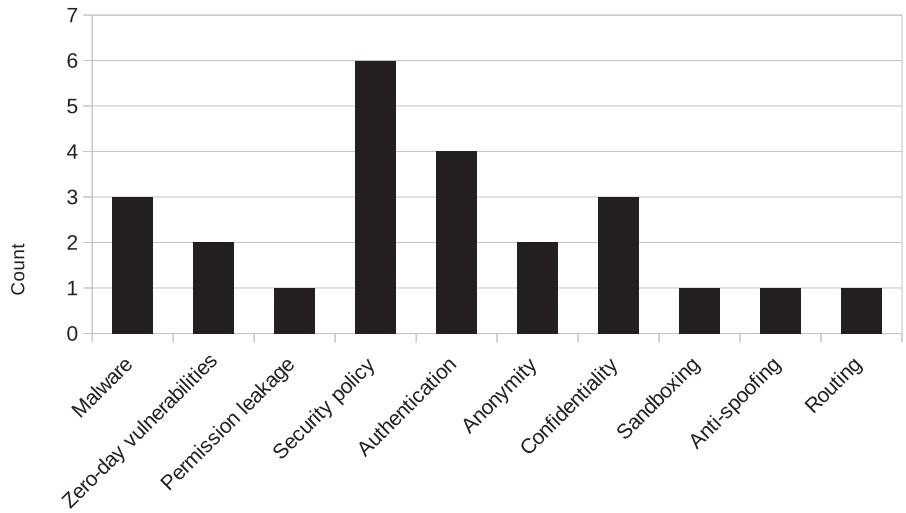

Fig. 5. Cyber security topics in the consumer section.

Once again the most utilised FM is model checking. It could be argued that this is due to the significant model checking experience gained in other domains. Many consumer computing entities are, however, complex, interconnected chains of services, which could explain wider utilisation of lightweight FM, especially as some of these are used directly to build chains used to construct the consumer computing entities. It should also, however, be stated that theorem proving has also been utilised on all the different levels of abstraction within the consumer computing domain. The cyber security topics present within the consumer section are shown in Figure 5.

Application. FM thrive in checking consumer applications for malware. A practical definition of malware (one that can be used to classify executable files) intersects the perspective shift FM advocate: focus on "what" is computed instead of the "how." Quoting Reference [147]: "any (formal) definition of the concept of malware depends on the definition of the concept of software system correctness." Also, a majority of malware is the product of tools generating variants of known vulnerabilities/attacks or known malware. The authors of Reference [66] show variants are easy to hide syntactically, but not semantically.

Model checking-based approaches provide malicious behaviour semantic signatures by providing counterexamples. Recent approaches as in References [229, 231] extract push-down automata as models. A promising area is the application of the techniques to the realm of the Android operating system [230]. Although successful, the FM techniques provide no panacea to consumer malware protection. The malware game advances with discovery of zero-day (latent) vulnerabilities. FM have been argued to avoid these vulnerabilities in the first place [174], but practically, new malicious behaviours are expected to appear, thus the problem becomes to learn malicious behaviours. There are several proofs of concept where FM leverage the signature learning either in terms of or using push-down automata reachability in the process [75, 76, 167]. There are a few works where theorem proving is applied in malware [227], but the number of publications is small, and it is difficult to ascertain if there is an effective gain from it. Perhaps model checking is more appropriate to the domain due to its non-interactive nature, since malware is inherently a game between attackers and an algorithm.

System. Recently, we have seen the adoption of the software marketplace paradigm to prevent attacks and malware to reach consumer systems. For instance, Android enforces permissions at the

application level, which could, however, lead to privilege escalation [78]. The authors of Reference [26] propose a tool-based approach, called COVERT, for compositional analysis of Android inter-app permission leakage. COVERT assesses the security of a system as a whole by generating Alloy specifications [131] and analysed more than 500 real-world applications, confirming the findings previously found within References [23, 91], showing that many Android applications are over-privileged. The authors of Reference [25] have moved towards analysing the permission protocol itself, identifying design flaws where two applications can apply the same custom permission, resulting in first installed application being able to access the resources of the second.

Operating systems usually contain an underlying security model amenable to FM checking. In Reference [83] the authors have verified a proposed access and integrity control for a Linux-like OS, using Alloy and Event-B [4]. While the authors have experienced scalability issues with Alloy, the analysis has uncovered bugs that could become more serious if discovered in the implementation phase. Another example is Reference [170], where the authors verify security policies in the form of invariants annotating the code of ExpressOS, a secure OS alternative to Android. The authors utilise automated theorem provers and report the verification overhead (added annotations) was roughly 2.8% source code.

Consumer IoT systems such as smart home devices pose security vulnerabilities and have been widely verified using FM. The authors of Reference [153] use model checking tools within AVISPA and by BAN logic to verify a framework ensuring anonymity, authentication, and integrity in smart home environments. In Reference [182] the authors have developed the *IoTRiskAnalyzer* tool used to help engineers apply the most fitting security policies. This has been achieved using a Markov Decision Process [202], formalisation of risk properties as probabilistic CTL formulas, and verification using the PRISM model checker [156]. Car manufacturers are also taking advantage of connected devices, especially smartphones. For instance, the authors of Reference [53] have developed a smartphone-based immobiliser with formally verified protocol using ProVerif against a Dolev-Yao attacker model to ensure strong guarantees of security requirements.

Protocols. The area of consumer communication protocols covers text and multimedia communication lending itself to formal verification of security.

In Reference [71] the authors have created a formal model of the Signal protocol in terms of predicates and theorems and have applied theorem proving, resulting in improvements in the use of the protocol's random generator.

Security of the consumer communication systems often depends on the mechanisms introduced in the Needham-Schroeder [190] authentication protocol and the Denning Sacco protocol [81] for secret key distribution. The authors of Reference [60] have created a simplified model of the Needham-Schroeder NPSK protocol and the Denning Sacco protocol and expressed security properties using LTL. The authors have provided an efficient model for model checking of the security properties using the Spin model checker.

Implementation. The recent adoption of FM tools by large technology companies has shaken up the field. If in the past FM were tied to niche safety critical domains (e.g., aerospace, railway, medical) and fields with significant governmental regulation, the current panorama shows that the future brings the usage of FM tools in the daily practice of software engineering. No matter the intention behind the usage of FM tools, the outcome has demonstrated a contribution to increasingly secure implementations. According to recent reports, when a developer commits a code modification to one of the large technology companies' codebases, a static analysis tool is invoked and a code review is provided. The author of Reference [196] describes the process as continuous reasoning, and any change to a Facebook product is analysed by the Infer static analysis tool, which checks "small theorems" on large codebases. This approach has been shown to improve the

security of the company's own codebase and library implementations (e.g., OpenSSL). The same is reported about the software engineering practice inside Google [215], although it is not clear whether FM is used by Project Zero, its elite security team. However, the authors of Reference [24] report on how numerous security vulnerabilities were fixed by applying FUDGE, a static analysis tool based on fuzzing developed in house.

Particularly targeted to the security domain, the authors of Reference [84] report a static analysis tool, Zoncolan, in collaboration with the Facebook App Security team. Zoncolan uses abstract interpretation to analyse and issue security alerts for the implementations of the applications in the company's codebase: Messenger, WhatsApp, Instagram, or Facebook. This level of application of FM shows the implementations of software used by millions of consumers has been swept by a FM tool.

FM is also being applied to secure implementations of web browsers, which are designed with security in mind because they mediate a vast amount of personal information (e.g., credentials, banking details). Nevertheless, due to a large attack surface of a web browser, attacks are possible, and implementation flaws are not uncommon. In a bolder move, the authors of Reference [133] propose a new browser, QUARK, that follows the "kernel architecture"[7] of modern browsers, but QUARK's kernel is formally verified. The formal verification yields to the Coq theorems to assert properties as tab non-interference, or cookie confidentiality and integrity. According to work in Reference [96], the price to pay for such a prime example of functional correctness verification (above airline runtime error-free level) is 25% increase in overhead, affecting performance. Increasing browsers' security risks, browser extensions can spy and exploit users as demonstrated in Reference [109]. In Reference [218] the authors report on a verified design of an experimental browser using the Maude tool and rewriting logic, and the authors of Reference [184] show that x86 native code executed by arbitrary clients conforms with a predefined sandbox policy when using Google Chrome's Native Client service.

Hardware. In contrast to critical-system hardware (e.g., fly-by-wire hardware) attackers cannot be prevented from physical access to the consumer hardware, which provides a large attack surface. Modern consumer hardware provides hardware-level protections for critical software components. An example of this is ARM TrustZone [192], providing separation between trusted and rich software providing potentially untrusted interfaces. The authors of Reference [93] propose verification of hardware security properties by use of information flow control at the level of the **Hardware Description Language (HDL)** such as SecVeriLog [271]. The authors create *SecVeriLogBL*, an extension to SecVeriLog, by adding new types for security labels defined in SecVeriLog. This allows for static analysis at the design time, providing a lightweight verification with small effects on the hardware performance. To demonstrate this approach, the authors have designed an implementation of TrustZone, including 10+ security bugs. Similarly the authors of Reference [163] present a formally defined hardware security enforcement for x86 architecture. In this setting, the software relies on underlying hardware for security enforcement; for example, memory paging features of an x86 CPU. The authors note that incorrect implementations of hardware enforcement policies often lead to vulnerabilities [140]. The authors use Coq to model the architecture and the Coq theorem prover to prove the soundness of the security policy.

In the area of commodity hardware, the authors of Reference [239] have used model checking to determine possible attacks on smart meters, which are considered critical devices [143]. The authors have created a model of the smart meter using rewriting logic, formal definition of the

[7]Termed multi-process architecture in Google Chrome with sandboxing of untrusted code, which accesses resources through a trusted broker.

attacker's actions, and used the Maude tool to check that the attacker's actions are not able to break the security invariants. The discovered attacks were then mapped to an implementation of a smart meter, the SEGMeter, to investigate the practicality of these attacks. The authors determined that many attacks discovered by the model checker are indeed practical, despite the model being abstract and not specifically refined towards the SEGMeter implementation.

Today, hardware is often packaged as an SoC, which security may be verified using the combination of integrated theorem proving and model checking proposed in Reference [111]. Due to the hierarchical nature of SoCs, the authors propose that the design expressed in HDL is decomposed into sub-modules and security specifications into sub-specifications. The sub-specifications are then verified using the Cadence IFV model checker [1]. These verified sub-specifications are then used as proven lemmas in the Coq theorem prover [242], removing the need to prove these lemmas by hand. This simplifies the model checking as well by providing only a small specification to the model checker, avoiding the state space explosion. The authors extend their method by automated code conversion from HDL to verifiable specification [112]. SoC complexity increases in **Multi Processor SoCs (MPSoC)**, where multiple processors exchange data via **Network on a Chip (NoC)** routers. The authors of Reference [220] have used unbounded model checking to verify security properties of an NoC, which was practical due to the highly sequential behaviour of NoCs. The authors formalise the security and functionality correctness properties using LTL and use the CIP unbounded model checker [155] to verify them. As a proof of concept, the authors have analysed six different router implementations, determining the feasibility of their approach for NoC security analysis in early design stages.

2.5 Enterprise

Enterprise and large corporate computing is the backbone of large international business. In recent years, there is a trend in enterprise computing to utilise cloud solutions, while still often operating on-premises (local) data centers. These data centers and cloud clusters are utilised for a plethora of enterprise tasks such as virtualisation of collaboration platforms, company management, and hosting of corporate web portals. This section provides an overview of utilisation of FM to address security challenges of enterprise computing, ranging from secure data storage through virtualisation and software-defined networking security to strong authentication using hardware tokens. As enterprises are larger entities, changes are often slower and need to be well managed. To this end, the FM have been utilised as a booster in cloud adoption by enterprises, as several FM-based solutions have been proposed to enable enterprises' secure switch from local data centers to federated cloud solutions.

Similarly to previous sections, model checking is the most used tool in formal analysis of security in enterprise computing. Theorem proving is, however, not far behind, especially within analysis of hardware such as Trusted Platform Module chips within enterprise servers. Lightweight FM have also been significantly utilised at the implementation level of abstraction, since they are often provided as plugins to software development environments, making them easily accessible. The cyber security topics present within the enterprise section are shown in Figure 6.

Application. Enterprise applications often process and store data critical for an organisation. Nowadays, such data is carried by **Software-Defined Networking (SDN)**. The authors of Reference [226] have created a verification platform for applications utilising SDN, consisting of a modelling language that could be automatically translated and dispatched to PRISM, SPIN, and Alloy model checkers. Any counterexamples are then displayed in the tool. Similarly to SDN, **Service-Oriented Architectures (SOAs)** are often used within enterprise applications, increasing the interconnectivity of these applications. In Reference [20] the authors present a platform for security

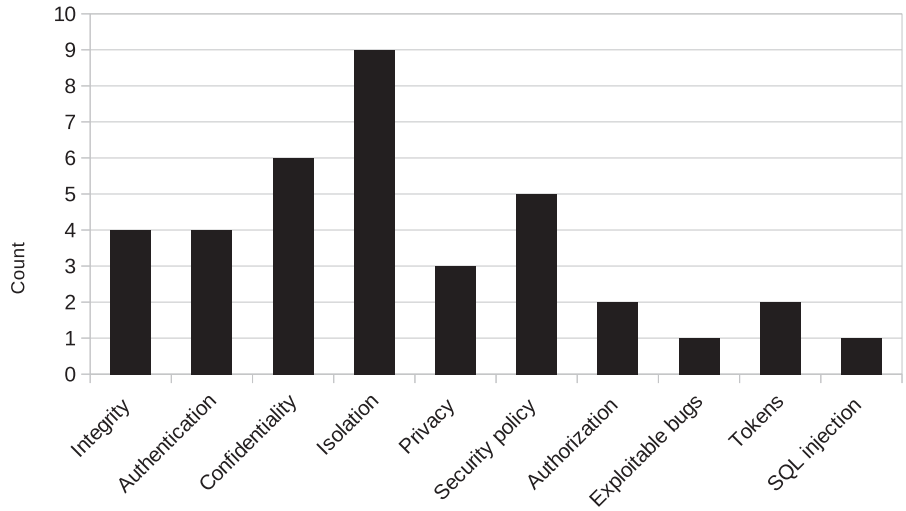

Fig. 6. Cyber security topics in the enterprise section.

assessment of SOAs utilising a formal specification language and several model checkers, namely, CL-Atse and OFMC. The authors uncovered an issue with SAML-SSO integration in Google Apps.

Enterprise data are often stored within large relational databases. In this context, the authors of Reference [61] have proposed a method for secure outsourcing of databases to untrusted servers, building upon the notion of Verifiable Databases [36], and utilised theorem proving to demonstrate their method as secure. Another often used technology in the cloud is virtualisation. The authors of Reference [222] introduce a formal analysis scheme for security of Xen hypervisor, consisting of model checking and static analysis, successfully rediscovering a known vulnerability. Finally, we would like to point to the work within Reference [72] describing how a leading cloud provider utilises FM for security of their services, noting that the benefits of FM are important to their customers.

System. Nowadays, large enterprises can either host their own infrastructure, fully utilise the cloud, or partially combine their infrastructure with the cloud, leading to federated cloud systems [183]. The authors of Reference [270] have proposed a method for analysis of federated cloud behaviour utilising CPN and CPN tools [137], creating several models for security analysis. Similarly, the authors of Reference [256] have used Z with Z/EVES theorem prover to formally analyse a data exchange system against confidentiality and integrity properties, while also generating tests utilising the domain theory [39]. In Reference [135], the authors present a formal approach to analysis of firewall rules and cloud topology based on Mobile Ambients [55] and the non-interfering Boxed Ambients calculus [51]. As cloud computing is often built utilising shared resources, the authors of Reference [169] have built an offline framework for formal analysis of network isolation properties, ensuring isolation among shared resources. This has been carried out by use of first order logic and the constraint satisfaction solver Sugar [240]. Similarly, the authors of Reference [173] have proposed a security framework for cloud complexity management (agent) system [113], utilising the Z/EVES theorem prover to analyse several cloud security properties within a NIST [43] cloud reference architecture. Also, in Reference [232], the authors have proposed a broker solution for automatically pairing cloud services with customers while managing the cloud complexity. An important part of the broker is finding a service satisfying customers

security requirements, defined in first-order relational logic [130] using KODKOD finite model finder [244].

Several works also consider virtualisation within the cloud system. For example, the authors of Reference [117] have proposed a formal framework for analysis of security and trust in virtualised system, combining hardware and software models expressed in CSP# [238] dispatched to the PAT model checker discovering a subtle bug in a real-world cloud system. Similarly, the authors of Reference [42] have proposed a security subsystem for change analysis within virtualised infrastructures in relation to security policies by utilisation of graphs and graph transformations dispatched to GROOVE model checker [102].

Protocols. Enterprise computing is moving towards the cloud, creating need for secure communication protocols, benefiting from FM analysis.

Amazon cloud services [13] use the s2n [14], the open source implementation of the TLS protocol, utilising FM to prove its correctness. For example, in Reference [64] the authors demonstrate that the **Hashed based Message Authentication Code (HMAC)** utilised by the protocol is indistinguishable from a random generator using Cryptol specification language [89] describing the HMAC, which was then dispatched to Coq theorem prover and the results are connected with the implementation by use of Software Analysis Workbench [100]. In the Microsoft cloud, the authors of Reference [136] have developed a tool for analysis of network protocols to assist with the task of network policy maintenance within data centers by use of the Z3 SMT solver, providing an important security tooling for Azure cloud services. Since the cloud services are often accessed remotely, the authors of Reference [151] have utilised Alloy analyser to find vulnerabilities in the SAML protocol [125].

Nowadays, clouds collect data from small-footprint IoT devices [185], which prompted the authors of Reference [141] to propose a lightweight mutual authentication protocol and verify it against several attacks using OFMC and CL-AtSe. Similarly, the authors of Reference [212] have proposed a mobile authentication scheme verified using ProVerif. The IoT devices could take advantage of the 5G networks, where in Reference [33] the authors use Tamarin, finding an issue with the authentication sub-protocol, while the authors of Reference [8] have analysed the authentication framework protocol [269] and the mobile ethernet protocol [128] by expressing them in the CSP, which was subsequently dispatched to the FDR model checker for analysis against mutual authentication properties.

Implementation. Enterprise computing is often composed of many applications implemented using different technologies. For example, the authors of Reference [195] have created a static code analysis tool for PHP plugins, *phpSafe*, that was then utilised to discover over 580 vulnerabilities in several PHP plugins. In similar fashion, the authors of Reference [267] have created a tool utilising invariant analysis [116] for malicious behaviour detection, noting the high effectiveness of logic flaws in several web applications.

Hypervisors are an important part of enterprise computing. To the end of security implementation of hypervisors, the authors of Reference [252] have created a framework for implementation of security verified hypervisors based on behavioural contracts and verified using FRAMA-C [225] for static analysis of the behavioural contracts. Similarly, the authors of Reference [251] have created a hypervisor framework for verification of memory integrity within single guest hypervisors utilising the CBMC model checker [67] for automated analysis of most of the codebase.

Hardware. Enterprise computing requires significant cloud hardware infrastructure and assurances such as data confidentiality and computational security. Customers often consider a cloud provider as an untrusted entity, where the administrators themselves could pose a

security threat [217]. In this regard, the authors of Reference [219] created a cloud isolation system, separating the user data from administrators and limiting the operations that the administrators could take against a user's virtual machine, utilising a hardware module, that the authors named ***Trusted Cloud Module*** **(TCM)**, which provides a limited set of interfaces to the cloud administrator, manages encryption keys, and provides secure storage for the user. The module is built from off-the-shelf hardware components using the Scyther verification tool. Similarly, the basis of any trusted computing is the **Trusted Platform Module (TPM)** co-processor providing secure storage and computing environment. Unfortunately, the security of platforms using TPM is often not formally verified leading to vulnerabilities [50]. To mitigate this, the authors of Reference [27] have proposed *TRUSTFOUND,* a formal modelling framework for model checking utilising a Trusted CSP#, an extension of CSP#, and LS^2 [77], where the PAT model checker is used for verification and detecting six implied assumptions and two severe logic flaws.

Sometimes, to provide strong authentication, small **One Time Password (OTP)** generation hardware is used by enterprises to authenticate users towards cloud services [45]. One such device is *Yubikey*, a USB OTP generator. In Reference [154], the authors have formally analysed the security of the Yubikey OTP and also a security of the **Hardware Security Module (HSM)**. Another challenge in this area is addressing CPU side-channel attacks. One of these attacks is a timing channel attack, where an attacker, possibly a virtual machine, could determine the algorithm executed by another virtual machine in a shared environment. To solve this, the authors of Reference [92] have proposed *Timing Compartments*, an isolation scheme implemented in hardware isolating timing information between parties sharing the resources. The scheme was checked by information flow analysis using SecVerilog.

3 FUTURE OUTLOOK

This survey has provided an overview of use of FM within security in several domains. Based on the research conducted within these domains, it is expected that in some cases the use of FM will accelerate while, in other cases, the use will increase with a slower pace. There is, however, a general trend of increase of adoption. In cases within the financial domain, it is clear that the use of FM comes with new financial technologies such as cryptocurrencies and smart contracts. This adoption could be seen in a survey aimed specifically at the smart contracts domain [119]. The use of mobile applications in the financial domain is also spurring a demand for high security assurance that could be delivered by use of FM. Finally, with the rise of cryptocurrencies, the hardware within the financial domain is being specialised to facilitate transactions. It is expected that the security of this hardware will continue to be scrutinised formally with increasing coverage and complexity.

The industrial domain faces its own set of unique challenges in the area of cyber security. It is expected that with increase in automation complexity and use of digital technologies in critical industrial installations, FM will play a crucial role. The trend was already presented in 2015 by Reference [148], who have surveyed approaches for security and safety of industrial control systems, including informal approaches. As of now, most works within this domain are of reactive nature, i.e., analysis of existing systems and protocols. However, several works within the survey show a trend towards utilisation of FM early in the design process of new industrial installations and protocols. Another emerging trend within the industrial domain is integration of formal verification tools with the software development processes, this is clear primarily in terms of robotic applications and PLC code. It is expected that with increasing complexity of robotic applications and underlying hardware, FM will play a significant role in the future.

The domain of consumer computation is a rapidly evolving one. The consumer trends move fast; however, a somewhat surprising amount of work is already put forward to use of FM in

malware protection [37], going as far as creation of formally verified Internet browsers. Also, the shift of computation from computers to smartphones has brought new security challenges. In this area, a lot of focus has already been put on analysis of the Android OS permission system. This area shows an increase in the amount of research and as long as the mobile OSs are in use by millions of users, the formal verification will accelerate, possibly leading to full formal security verification of a popular mobile OS. Consumer hardware such as smartphones is putting into use TPMs, securing mobile computation. Also in this area, the use of FM is increasing, specifically to ensure the properties of the secure TPM enclave.

Finally, the domain of enterprise computation has uncovered several interesting trends. The first of these is a trend to use FM against virtualisation hypervisors to analyse security properties of existing virtual environments as well as use the knowledge to build fully formally verified hypervisors. Another important trend is the significant investment that major cloud computation providers are putting into formal verification of security of their products. To this end, not only have existing tools been applied, but the cloud providers have turned towards development of their own FM tools. Both of these trends are expected not only to continue but also to accelerate due to the ever-increasing popularity of cloud computing and virtualisation. Several of these trends were already mentioned in 2002 by Reference [139].

It is important to note that several authors have expressed a wish for improvement of automated formal verification tools. This is to allow for simplified entry of non-practitioners to the world of FM. Both academia and industry are moving towards addressing this wish, with tools becoming similar to software development IDEs and in some cases integration of FM toolkit directly to an existing IDE. It is expected that knowledge of FM will become important for system and software engineering disciplines in the future, and therefore collaborative projects between industry and academia shall provide experts in this domain. These issues have been discussed for several years now [79].

When it comes to FM techniques, static analysis tools are becoming popular in software development, while model checking is moving strongly towards system design and protocol verification, with model checking being designed for specific problems. Theorem proving is also showing a promise of playing a crucial role in the future, given that the perceived large learning curve could be minimised.

As the use of FM is accelerating within all of the different domains considered in this survey, it is imperative that a new updated survey is carried out in 5 to 10 years. By then it is expected that the tools will reach the quality of commercial-grade IDEs and integration with a wide variety of text editors [245], and the techniques will become a known factor when developing and designing a new system, application, or a hardware component.

4 CONCLUSIONS

More than 30 years ago, Burrows et al. published their pioneering work on the BAN logic for security protocol analysis [52]. Their work was not fully formal and was shown to permit approval of dangerous protocols. Nevertheless, they showed that their logic was good at revealing various subtle security flaws and drawbacks, specifically in authentication protocols. They set out to answer five questions:

(1) Does this protocol work?
(2) Can it be made to work?
(3) Exactly what does this protocol achieve?
(4) Does this protocol need more assumptions than another protocol?
(5) Does this protocol do anything unnecessary?

Their important paper inspired a generation of security researchers to use FM to devise and analyse security protocols and to answer similar questions. Furthermore, in describing the benefits of formal verification for software engineering, Dijkstra famously quoted, "Testing shows the presence, not the absence of bugs." More than 50 years later, in the setting of computer security, we might now have sufficient evidence to claim "Formal methods show the presence, not the absence of security flaws" [54]. That is, although FM provides rigorous tools and techniques for proving the absence of security flaws, this rigour comes with a proviso: Proofs are only possible if the security flaw is documented and specified within a formal framework. Current limitations mean that FM cannot uncover any new flaws, since we may not actually be looking for them.

Take for instance the recent Spectre and Meltdown attacks [145]. Like many vulnerabilities, both attacks have been shown to exist for a range of processors that make use of speculative execution, and mitigation against these requires software-side interventions. However, despite the widespread nature of these vulnerabilities, the attacks themselves were not discovered through formal verification, but rather through a series of experiments over the training and timing of micro-architectural components. Fortunately, once a security flaw has been uncovered, even complex attacks such as Spectre and Meltdown can be formally characterised and isolated [59]. Once this has been done, the next phase is a formal framework for reasoning about such issues, followed by more streamlined tools to scale verification to larger-scale systems. Thus, as important as it is to continue research into the practical use of FM in security, it is equally important to expand our reasoning capabilities for FM in security through the study of theoretical aspects of the discipline. Without this, there is a possibility of a new type of security flaw that falls outside the realms of current-day logics. Variants of Spectre and Meltdown, for instance, can be captured by subset-closed hyper-properties [59]. Hence, without existing works on hyper-properties [68] and subsequent works on their verification [59, 106, 107], the specification and therefore use of FM to protect against Spectre and Meltdown would have been much more difficult.

As for the utilisation of different FM techniques, we have uncovered that model checking is often a choice of practitioners when analysing wider systems. From the surveyed works, a trend could be seen where the model checking is often considered a good approach due to significant amount of tool support, plethora of modelling languages to choose from, the ability to carry out an exhaustive analysis, and even existence of model checkers specifically aimed at security analysis [200]. We could determine that even against problems where model checking would suffer from the size of the problem, the authors scope the problem such that it becomes analysed partially, specifically aiming at a critical parts of the system or an application. In some cases, either due to the large size of the problem or the problem's suitability, the FM practitioners choose to utilise theorem proving. While the tool support is not on par with model checking and the learning curve is steeper, theorem proving provides exhaustive analysis of systems that would overwhelm model checkers. We are, however, observing that theorem proving needs to become more accessible to system and software engineers to increase its utilisation. Finally, a special category of lightweight FM provides an interesting entry point to the world of FM. On one hand, the inability to carry out exhaustive analysis could be seen as a downside, but on the other hand, it could be argued that lightweight FM are reaching ubiquity within the engineering world. For example, static code analysers are not only becoming used within software-build toolchains but often integrated directly to software-integrated development environments [221], being at the fingertips of significant portion of software developers around the world. As pondered in the Section 3, this could be one of the catalysts bringing the FM to the foreground in system design and development.

In this article, we have shown how FM have had an impact on society so far and how this impact will increase in the future. In the past, security has been an optional extra that industry does not want to invest in during development. But times are changing. For example, security has become

a core selling point for Amazon Web Services (see Section 2.4). FM have been used successfully for security analysis in the financial, industrial, consumer, and enterprise sectors (see Section 2).

Specification Languages and Associated Tools

Our survey covers more than a decade of the use of FM in security. It reveals the rich variety of formal specification languages and their tools, theorem provers, model checkers, and verification frameworks. We have recorded more than 40 different specification languages and more than 40 different verification tools. These include the following:

Specification languages. **AADL (Architecture Analysis & Design Language)** [18, 70], **ASF (Anonymous Secure Framework)** [153], **ASLan++ (AVANTSSAR Specification Language)** [20], BAN logic [52, 228], Boogie [31], Boxed Ambients [135], **CASM (ASM-based SL for compilers)** [252], **CCS (Calculus of Communicating Systems)** [152], **COVERT (compositional analysis of Android apps)** [26], **CSP (Communicating Sequential Systems)** [120], **CSP# (shared variables CSP)** [237], **CTL (Computation tree temporal logic)** [229], Cloud Calculus [135], Cryptol [89], Dynamic State Machine [187], ERC20 token contracts [199], Event-B [83], **HLPSL (High Level Protocol Specification Language)** [44], Hoare logic [108], **LS2 (Logic of Secure Systems)** [27], **LTL (linear-time temporal logic)** [266], Markov Decision Process [182], Petri nets [15], π-calculus [40], PlusCal [9], Promela [172], **RTL (real-time logic)** [110], **SPDL (Security Protocol Description Language)** [168], SysML-Sec [18], **TLA+ (Temporal Logic of Actions)** [72], Trusted CSP# [27], überSpark [252], VDM [98], Verilog [164], VHDL [111], VML [226], vTRUST [117], **XMHF (eXtensible and Modular Hypervisor Framework)** [251], Z [265].

Model checkers. **AVISPA (Automated Validation of Internet Security Protocols and Applications)** [21], Alloy [83], **CBMC (Bounded Model Checker for C and C++)** [58], **CWB-NC (Concurrency Workbench of New Century)** [152], **Cadence IFV (RTL block-level verifier)** [111], FDR [103], GROOVE [102], jKind [99], NuSMV [65], **OFMC (on-the-fly model checker)** [34], **PAT (Process Analysis Toolkit for CSP#)** [237], **PRISM (probabilistic model checker)** [182], **SATMC (SAT-based model checker for security protocols)** [44], SPIN [247], TRUSTFOUND [27], UPPAAL [177], **UVHM (formal analysis scheme for hypervisors)** [251].

Theorem provers. Coq [72], Isabelle/HOL [144], K-framework [199], TAMARIN [154], Why [95].

Verification tools and frameworks. **AndroBugs (Framework For Android Vulnerability Scanning)** [241], **Cl-Atse (protocol analyser)** [141], **FUDGE (Fuzz driver generator)** [24], Frama-C [252], Krakatau [152], **Maude (rewrite engine)** [193], **MobSF (mobile security framework)** [127], phpSAFE [195], ProVerif [40], Quark [134], **SAW (Software Analysis Workbench)** [72], SMACK [72], SecGuru [136], SecVeriLog [92], **Sugar (SAT-based)** [169], **TTool (translator from SysML-Sec to π-calculus)** [18], Z3 [22].

This shows how research and application in FM for security has developed since Burrows et al.'s seminal paper [52]. Our survey concentrated in particular on the practical application of these techniques, especially on an industrial scale. Today, it seems inconceivable that a company would produce a commercial secure system without subjecting it to formal analysis. We also suspect that hackers use formal techniques to crack supposedly secure systems.

As a final statement, we need to acknowledge that a survey provides a snapshot in time within a developing field. It is therefore necessary to return to a survey work every decade, something we are planning on doing. Despite this shortcoming, it is the opinion of the authors that a survey work is an important part of the research, as it provides a starting point and a direction indicator for new and experienced practitioners looking for works under the large field of formal methods in security.

APPENDIX

This appendix provides an overview of the several surveyed research works in the form of a cross-tabulation according to Reference [166]. We group the 115 covered references by Domain, **Level Of Abstraction (LOA)** and **Approach (App.)**.

Table 1. The Sorting of the 120 Works Leading to the Taxonomy Displayed in Figure 2

Domain	LOA	App.	Works
Financial	Application	MC	[63, 126, 152, 249]
		TP	[17, 198]
	System	LW	[2]
		MC	[7, 87, 197, 216, 268, 272]
	Protocol	MC	[5, 11, 44, 118, 266]
		TP	[168]
	Implementation	LW	[98, 127, 241]
		MC	[16]
		TP	[94]
	Hardware	MC	[22, 62]
		TP	[19, 171]
Industrial	Application	MC	[120, 146, 177, 193, 254, 273]
	System	LW	[246]
		MC	[82, 114, 150, 211, 214, 255]
	Protocol	LW	[15, 224]
		MC	[49, 187, 203]
		TP	[86]
	Implementation	LW	[196]
		MC	[18, 70, 223, 247]
		TP	[70]
	Hardware	MC	[3]
		TP	[110, 124, 162, 164]
Consumer	Application	LW	[75, 76, 167, 174]
		MC	[178, 229–231]
		TP	[227]
	System	MC	[25, 26, 53, 83, 153, 182]
		TP	[170]
	Protocol	MC	[60]
		TP	[71]
	Implementation	LW	[24, 84]
		MC	[218]
		TP	[109, 133, 184]
	Hardware	LW	[93]
		MC	[111, 220, 239]
		TP	[111, 163]
Enterprise	Application	MC	[20, 72, 222, 226]
		TP	[61, 72]
	System	LW	[270]
		MC	[42, 117, 169, 232]
		TP	[135, 173, 256]
	Protocol	MC	[8, 141, 151, 212]
		TP	[33, 64, 136]
	Implementation	LW	[195, 252, 267]
		MC	[251]
		TP	[199]
	Hardware	LW	[92]
		MC	[27]
		TP	[154, 219]

ACKNOWLEDGMENTS

We would like to thank Nick Battle, Jaco van de Pol, and Bas Spitters for reviews of earlier versions of this article. We would also very much like to thank the anonymous reviewers of an earlier version of this article for their valuable input, which definitely improved it.

REFERENCES

[1] Jasper RTL Apps. 2020. Cadence IFV Model Checker. Retrieved from www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform.html.

[2] Fides Aarts, Joeri De Ruiter, and Erik Poll. 2013. Formal models of bank cards for free. In *IEEE 6th International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 461–468.

[3] Imran Hafeez Abbasi, Faiq Khalid Lodhi, Awais Mehmood Kamboh, and Osman Hasan. 2017. Formal verification of gate-level multiple side channel parameters to detect hardware trojans. In *Formal Techniques for Safety-critical Systems*, Cyrille Artho and Peter Csaba Ölveczky (Eds.). Springer International Publishing, Cham, 75–92.

[4] Jean-Raymond Abrial. 2010. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, Cambridge, UK. DOI:https://doi/10.1017/CBO9781139195881

[5] S. Abughazalah, K. Markantonakis, and K. Mayes. 2014. Secure mobile payment on NFC-enabled mobile phones formally analysed using CasperFDR. In *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 422–431.

[6] Sarita V. Adve and Kourosh Gharachorloo. 1996. Shared memory consistency models: A tutorial. *IEEE Comput.* 29, 12 (1996), 66–76.

[7] Shakeel Ahamad, Siba Udgata, and V. Sastry. 2012. A new mobile payment system with formal verification. *Int. J. Internet Technol. Secur. Trans.* 4 (01 2012), 71–103. DOI:https://doi/10.1504/IJITST.2012.045153

[8] Mahdi Aiash. 2015. A formal analysis of authentication protocols for mobile devices in next generation networks. *Concurr. Computat.: Pract. Exper.* 27, 12 (2015), 2938–2953. DOI:https://doi/doi.org/10.1002/cpe.3260 arXiv:onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3260.

[9] Sabina Akhtar, Ehtesham Zahoor, and Olivier Perrin. 2017. Formal verification of authorization policies for enterprise social networks using PlusCal-2. In *Collaborative Computing: Networking, Applications and Worksharing - 13th International Conference, CollaborateCom 2017, Edinburgh, UK, December 11–13, 2017, Proceedings (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 252)*, Imed Romdhani, Lei Shu, Takahiro Hara, Zhangbing Zhou, Timothy J. Gordon, and Deze Zeng (Eds.). Springer, 530–540.

[10] Jade Alglave, Alastair F. Donaldson, Daniel Kroening, and Michael Tautschnig. 2011. Making software verification tools really work. In *Automated Technology for Verification and Analysis*, Tevfik Bultan and Pao-Ann Hsiung (Eds.). Springer Berlin, 28–42.

[11] A. Alshehri, J. A. Briffa, S. Schneider, and S. Wesemeyer. 2013. Formal security analysis of NFC M-coupon protocols using Casper/FDR. In *5th International Workshop on Near Field Communication (NFC)*. 1–6. DOI:https://doi/10.1109/NFC.2013.6482439

[12] R. Alur, C. Courcoubetis, and D. Dill. 1990. Model-checking for real-time systems. In *5th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 414–425. DOI:https://doi/10.1109/LICS.1990.113766

[13] Amazon.com Inc. 2019. Amazon Simple Storage Service (S3). Retrieved from http://www.aws.amazon.com/s3/.

[14] Amazon.com Inc. 2019. s2n. Retrieved from http://www.github.com/awslabs/s2n.

[15] Raphael Amoah, Seyit Camtepe, and Ernest Foo. 2016. Formal modelling and analysis of DNP3 secure authentication. *J. Netw. Comput. Applic.* 59 (2016), 345–360. Retrieved from http://www.sciencedirect.com/science/article/pii/S1084804515001228.

[16] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. 2014. Modeling Bitcoin contracts by timed automata. In *Formal Modeling and Analysis of Timed Systems*, Axel Legay and Marius Bozga (Eds.). Springer International Publishing, Cham, 7–22.

[17] Danil Annenkov and Martin Elsman. 2018. Certified compilation of financial contracts. In *20th International Symposium on Principles and Practice of Declarative Programming (PPDP'18)*. Association for Computing Machinery, New York, NY. DOI:https://doi/10.1145/3236950.3236955

[18] L. Apvrille, L. Li, and Y. Roudier. 2016. Model-driven engineering for designing safe and secure embedded systems. In *Architecture-Centric Virtual Integration (ACVI)*. IEEE, 4–7. DOI:https://doi/10.1109/ACVI.2016.6

[19] Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. 2019. A formal treatment of hardware wallets. In *Financial Cryptography and Data Security*, Ian Goldberg and Tyler Moore (Eds.). Springer International Publishing, Cham, 426–445.

[20] Alessandro Armando, Wihem Arsac, Tigran Avanesov, Michele Barletta, Alberto Calvi, Alessandro Cappai, Roberto Carbone, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Gabriel Erzse, Simone Frau, Marius Minea, Sebastian Mödersheim, David von Oheimb, Giancarlo Pellegrino, Serena Elisa Ponta, Marco Rocchetto, Michael Rusinowitch, Mohammad Torabi Dashti, Mathieu Turuani, and Luca Viganò. 2012. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Tools and Algorithms for the Construction and Analysis of Systems*, Cormac Flanagan and Barbara König (Eds.). Springer Berlin, 267–282.

[21] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò,

and L. Vigneron. 2005. The AVISPA tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification*, Kousha Etessami and Sriram K. Rajamani (Eds.). Springer Berlin, 281–285.

[22] Anish Athalye, Adam Belay, M. Frans Kaashoek, Robert Morris, and Nickolai Zeldovich. 2019. Notary: A device for secure transaction approval. In *27th ACM Symposium on Operating Systems Principles (SOSP'19)*. Association for Computing Machinery, New York, NY, 97–113. DOI:https://doi/10.1145/3341301.3359661

[23] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. 2012. PScout: Analyzing the Android permission specification. In *ACM Conference on Computer and Communications Security (CCS'12)*. ACM, New York, NY, 217–228. DOI:https://doi/10.1145/2382196.2382222

[24] Domagoj Babić, Stefan Bucur, Yaohui Chen, Franjo Ivančić, Tim King, Markus Kusano, Caroline Lemieux, László Szekeres, and Wei Wang. 2019. FUDGE: Fuzz driver generation at scale. In *2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, 975–985. DOI:https://doi/10.1145/3338906.3340456

[25] Hamid Bagheri, Eunsuk Kang, Sam Malek, and Daniel Jackson. 2018. A formal approach for detection of security flaws in the Android permission system. *Form. Asp. Comput.* 30, 5 (01 Sep. 2018), 525–544. DOI:https://doi/10.1007/s00165-017-0445-z

[26] H. Bagheri, A. Sadeghi, J. Garcia, and S. Malek. 2015. COVERT: Compositional analysis of Android inter-app permission leakage. *IEEE Trans. Softw. Eng.* 41, 9 (Sep. 2015), 866–886. DOI:https://doi/10.1109/TSE.2015.2419611

[27] Guangdong Bai, Jianan Hao, Jianliang Wu, Yang Liu, Zhenkai Liang, and Andrew Martin. 2014. TrustFound: Towards a formal foundation for model checking trusted computing platforms. In *FM 2014: Formal Methods*, Cliff Jones, Pekka Pihlajasaari, and Jun Sun (Eds.). Springer International Publishing, Cham, 110–126.

[28] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. 2019. SoK: Computer-Aided Cryptography. Cryptology ePrint Archive. Retrieved from http://www.eprint.iacr.org/2019/1393.

[29] John Barnes. 2012. *Spark: The Proven Approach to High Integrity Software*. Altran Praxis, UK.

[30] Janet Barnes, Rod Chapman, Randy Johnson, James Widmaier, David Cooper, and Bill Everett. 2006. Engineering the Tokeneer enclave protection system. In *1st IEEE International Symposium on Secure Software Engineering*. IEEE Computer Society Press.

[31] Michael Barnett, Bor-Yuh Evan Chang, Robert DeLine, Bart Jacobs, and K. Rustan M. Leino. 2005. Boogie: A modular reusable verifier for object-oriented programs. In *Formal Methods for Components and Objects, 4th International Symposium, FMCO 2005, Amsterdam, The Netherlands, November 1–4, 2005, Revised Lectures (Lecture Notes in Computer Science, Vol. 4111)*, Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever (Eds.). Springer, Berlin, 364–387.

[32] David Basin. 2021. Formal methods for security. In *The Cyber Security Body of Knowledge v1.1*, Awais Rashid, Howard Chivers, Emil Lupu, Andrew Martin, and Steve Schneider (Eds.). University of Bristol. Retrieved from http://www.www.cybok.org.

[33] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A formal analysis of 5G authentication. In *ACM SIGSAC Conference on Computer and Communications Security*. 1383–1396.

[34] David A. Basin, Sebastian Mödersheim, and Luca Viganò. 2005. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.* 4, 3 (2005), 181–208.

[35] Elyes Ben Hamida, Kei Leo Brousmiche, Hugo Levard, and Eric Thea. 2017. Blockchain for enterprise: Overview, opportunities and challenges. In *13th International Conference on Wireless and Mobile Communications (ICWMC'17)*. IARIA XPS Press. Retrieved from http://www.hal.archives-ouvertes.fr/hal-01591859.

[36] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. 2011. Verifiable delegation of computation over large datasets. In *Advances in Cryptology – CRYPTO 2011*, Phillip Rogaway (Ed.). Springer Berlin, 111–131.

[37] Fabrizio Biondi, Thomas Given-Wilson, Axel Legay, Cassius Puodzius, and Jean Quilbeuf. 2018. Tutorial: An overview of malware detection and evasion techniques. In *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 565–586.

[38] Dines Bjørner. 1979. The Vienna development method (VDM). In *Mathematical Studies of Information Processing*, E. K. Blum, M. Paul, and S. Takasu (Eds.). Springer Berlin, 326–359.

[39] M. R. Blackburn, Ramaswamy Chandramouli, and Robert Busser. 2001. Model-based approach to security test automation. *Qual. Week* (01 2001).

[40] Bruno Blanchet. 2016. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Found. Trends Priv. Secur.* 1, 1–2 (2016), 1–135.

[41] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. 2018. *ProVerif 2.00: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*. INRIA.

[42] Sören Bleikertz, Carsten Vogel, Thomas Groß, and Sebastian Mödersheim. 2015. Proactive security analysis of changes in virtualized infrastructures. In *31st Annual Computer Security Applications Conference (ACSAC'15)*. Association for Computing Machinery, New York, NY, 51–60. DOI:https://doi/10.1145/2818000.2818034

[43] R. Bohn, John Messina, Fang Liu, Jin Tong, and Jian Mao. 2011. NIST cloud computing reference architecture. 594–596. DOI:https://doi/10.1109/SERVICES.2011.105

[44] Sriramulu Bojjagani and V. N. Sastry. 2015. SSMBP: A secure SMS-based mobile banking protocol with formal verification. In *WiMob Conference*. IEEE Computer Society, 252–259.

[45] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*. IEEE, 553–567.

[46] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. 2014. Towards secure distance bounding. In *Fast Software Encryption*, Shiho Moriai (Ed.). Springer Berlin, 55–67.

[47] Alejandro Bracho, Can Saygin, HungDa Wan, Yooneun Lee, and Alireza Zarreh. 2018. A simulation-based platform for assessing the impact of cyber-threats on smart manufacturing systems. *Procedia Manuf.* 26 (2018), 1116–1127. Retrieved from http://www.sciencedirect.com/science/article/pii/S2351978918308242.

[48] Sergey Bratus, Nihal D'Cunha, Evan R. Sparks, and Sean W. Smith. 2008. TOCTOU, traps, and trusted computing. In *Trusted Computing - Challenges and Applications, First International Conference on Trusted Computing and Trust in Information Technologies, Trust 2008, Villach, Austria, March 11–12, 2008, Proceedings (Lecture Notes in Computer Science, Vol. 4968)*, Peter Lipp, Ahmad-Reza Sadeghi, and Klaus-Michael Koch (Eds.). Springer, Berlin, 14–32.

[49] Alessandro Bruni, Michal Sojka, Flemming Nielson, and Hanne Riis Nielson. 2014. Formal security analysis of the MaCAN protocol. In *Integrated Formal Methods*, Elvira Albert and Emil Sekerinski (Eds.). Springer International Publishing, Cham, 241–255.

[50] D. Bruschi, L. Cavallaro, A. Lanzi, and M. Monga. 2005. Replay attack in TCG specification and solution. In *21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE. https://doi/10.1109/CSAC.2005.47

[51] Michele Bugliesi, Silvia Crafa, Massimo Merro, and V. Sassone. 2005. Communication and mobility control in boxed ambients. *Inf. Computat.* 202 (10 2005), 39–86. DOI:https://doi/10.1016/j.ic.2005.06.002

[52] Michael Burrows, Martín Abadi, and Roger M. Needham. 1990. A logic of authentication. *ACM Trans. Comput. Syst.* 8, 1 (1990), 18–36.

[53] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, and Ahmad-Reza Sadeghi. 2013. Smart keys for cyber-cars: Secure smartphone-based NFC-enabled car immobilizer. In *3rd ACM Conference on Data and Application Security and Privacy (CODASPY'13)*. Association for Computing Machinery, New York, NY, 233–242. DOI:https://doi/10.1145/2435349.2435382

[54] J. N. Buxton and B. Randell. 1970. *Software Engineering Techniques: Report of a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO*. https://dl.acm.org/doi/10.5555/1102021

[55] Luca Cardelli and Andrew D. Gordon. 2000. Mobile ambients. *Theoret. Comput. Sci.* 240, 1 (2000), 177–213. Retrieved from http://www.sciencedirect.com/science/article/pii/S0304397599002315.

[56] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. 2014. The nuXmv symbolic model checker. In *International Conference on Computer -aided Verification*. Springer, 334–342.

[57] Common Criteria Recognition Agreement CCRA. 2006. *Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and General Model*. Technical Report. Tech. Rep. CCMB-2006-09-001, Version 3.1, Revision 1. Common Criteria Management Board.

[58] Sudipta Chattopadhyay and Abhik Roychoudhury. 2018. Symbolic verification of cache side-channel freedom. *IEEE Trans. Comput.-aided Des. Integr. Circ. Syst.* 37, 11 (2018), 2812–2823.

[59] K. Cheang, C. Rasmussen, S. Seshia, and P. Subramanyan. 2019. A formal approach to secure speculation. In *IEEE 32nd Computer Security Foundations Symposium (CSF)*. 288–28815. DOI:https://doi/10.1109/CSF.2019.00027

[60] S. Chen, H. Fu, and H. Miao. 2016. Formal verification of security protocols using Spin. In *IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE, 1–6. DOI:https://doi/10.1109/ICIS.2016.7550830

[61] Xiaofeng Chen, Jin Li, Jian Weng, Jianfeng Ma, and Wenjing Lou. 2014. Verifiable computation over large database with incremental updates. In *Computer Security - ESORICS 2014*, Mirosław Kutyłowski and Jaideep Vaidya (Eds.). Springer International Publishing, Cham, 148–162.

[62] Tom Chothia, Flavio D. Garcia, Joeri De Ruiter, Jordi Van Den Breekel, and Matthew Thompson. 2015. Relay cost bounding for contactless EMV payments. In *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, 189–206.

[63] Tom Chothia, Flavio D. Garcia, Chris Heppel, and Chris McMahon Stone. 2017. Why Banker Bob (still) can't get TLS right: A security analysis of TLS in leading UK banking apps. In *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, 579–597.

[64] Andrey Chudnov, Nathan Collins, Byron Cook, Joey Dodds, Brian Huffman, Colm MacCárthaigh, Stephen Magill, Eric Mertens, Eric Mullen, Serdar Tasiran, Aaron Tomb, and Eddy Westbrook. 2018. Continuous formal verification of Amazon s2n. In *Computer Aided Verification*, Hana Chockler and Georg Weissenbacher (Eds.). Springer International Publishing, Cham, 430–446.

[65] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. 1999. NUSMV: A new symbolic model verifier. In *11th International Conference on Computer Aided Verification (CAV'99)*. Springer-Verlag, Berlin, 495–499.

[66] Aniello Cimitile, Francesco Mercaldo, Vittoria Nardone, Antonella Santone, and Corrado Aaron Visaggio. 2018. Talos: No more ransomware victims with formal methods. *Int. J. Inf. Secur.* 17, 6 (01 Nov. 2018), 719–738. DOI:https://doi/10.1007/s10207-017-0398-5

[67] Edmund Clarke, Daniel Kroening, and Flavio Lerda. 2004. A tool for checking ANSI-C programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, Kurt Jensen and Andreas Podelski (Eds.). Springer Berlin, 168–176.

[68] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *J. Comput. Secur.* 18, 6 (Sep. 2010), 1157–1210.

[69] Rance Cleaveland and Steve Sims. 1996. The NCSU concurrency workbench. In *Computer Aided Verification*, Rajeev Alur and Thomas A. Henzinger (Eds.). Springer Berlin, 394–397.

[70] D. Cofer, A. Gacek, J. Backes, M. W. Whalen, L. Pike, A. Foltzer, M. Podhradsky, G. Klein, I. Kuz, J. Andronick, G. Heiser, and D. Stuart. 2018. A formal approach to constructing secure air vehicle software. *Computer* 51, 11 (Nov. 2018), 14–23. DOI:https://doi/10.1109/MC.2018.2876051

[71] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. 2017. A formal security analysis of the signal messaging protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 451–466. DOI:https://doi/10.1109/EuroSP.2017.27

[72] Byron Cook. 2018. Formal reasoning about the security of Amazon Web Services. In *Computer Aided Verification*, Hana Chockler and Georg Weissenbacher (Eds.). Springer International Publishing, Cham, 38–47.

[73] Cas J. F. Cremers. 2008. The scyther tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification*, Aarti Gupta and Sharad Malik (Eds.). Springer Berlin, 414–418.

[74] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. 2012. Frama-C — A software analysis perspective. In *Software Engineering and Formal Methods - 10th International Conference, SEFM 2012, Thessaloniki, Greece, October 1–5, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7504)*, George Eleftherakis, Mike Hinchey, and Mike Holcombe (Eds.). Springer, Berlin, 233–247.

[75] Khanh-Huu-The Dam and Tayssir Touili. 2017. Learning Android malware. In *12th International Conference on Availability, Reliability and Security (ARES'17)*. ACM, New York, NY. DOI:https://doi/10.1145/3098954.3105826

[76] Khanh Huu The Dam and Tayssir Touili. 2018. Learning malware using generalized graph kernels. In *13th International Conference on Availability, Reliability and Security (ARES'18)*. ACM, New York, NY. DOI:https://doi/10.1145/3230833.3230840

[77] Anupam Datta, Jason Franklin, Deepak Garg, and Dilsun Kaynar. 2009. A logic of secure systems and its application to trusted computing. In *IEEE Symposium on Security and Privacy*. IEEE, 221–236. DOI:https://doi/10.1109/SP.2009.16

[78] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Marcel Winandy. 2011. Privilege escalation attacks on Android. In *Information Security*, Mike Burmester, Gene Tsudik, Spyros Magliveras, and Ivana Ilić (Eds.). Springer Berlin, 346–360.

[79] Jennifer A. Davis, Matthew Clark, Darren Cofer, Aaron Fifarek, Jacob Hinchman, Jonathan Hoffman, Brian Hulbert, Steven P. Miller, and Lucas Wagner. 2013. Study on the barriers to the industrial adoption of formal methods. In *Formal Methods for Industrial Critical Systems*, Charles Pecheur and Michael Dierkes (Eds.). Springer Berlin, 63–77.

[80] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer Berlin, 337–340.

[81] Dorothy E. Denning and Giovanni Maria Sacco. 1981. Timestamps in key distribution protocols. *Commun. ACM* 24, 8 (Aug. 1981), 533–536. DOI:https://doi/10.1145/358722.358740

[82] Michael Denzel, Mark Ryan, and Eike Ritter. 2017. A malware-tolerant, self-healing industrial control system framework. In *ICT Systems Security and Privacy Protection*, Sabrina De Capitani di Vimercati and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 46–60.

[83] Petr N. Devyanin, Alexey V. Khoroshilov, Victor V. Kuliamin, Alexander K. Petrenko, and Ilya V. Shchepetkov. 2014. Formal verification of OS security model with alloy and event-B. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Yamine Ait Ameur and Klaus-Dieter Schewe (Eds.). Springer Berlin, 309–313.

[84] Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. 2019. Scaling static analyses at Facebook. *Commun. ACM* 62, 8 (2019), 62–70.

[85] Danny Dolev and Andrew Chi-Chih Yao. 1981. On the security of public key protocols (extended abstract). In *22nd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 350–357.

[86] Jannik Dreier, Maxime Puys, Marie-Laure Potet, Pascal Lafourcade, and Jean-Louis Roch. 2017. Formally verifying flow properties in industrial systems. In *SECRYPT 2017 - 14th International Conference on Security and Cryptography* (Madrid, Spain) *(Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24–26, 2017.)*. SCITEPRESS Science And Technology Publications, Portugal, 55–66. DOI:https://doi/10.5220/0006396500550066

[87] Zhangbo Duan, Hongliang Mao, Zhidong Chen, Xiaomin Bai, Kai Hu, and Jean-Pierre Talpin. 2018. Formal modeling and verification of blockchain system. In *10th International Conference on Computer Modeling and Simulation (ICCMS'18)*. Association for Computing Machinery, New York, NY, 231–235. DOI:https://doi/10.1145/3177457.3177485

[88] Andrea Enrici, Ludovic Apvrille, and Renaud Pacalet. 2014. TTool/DiplodocusDF: A UML Environment for Hardware/Software Co-Design of Data-Dominated Systems-on-Chip. DOI : 10.1007/978-3-319-11653-2_23

[89] Levent Erkök and John Matthews. 2009. Pragmatic equivalence and safety checking in Cryptol. In *3rd Workshop on Programming Languages Meets Program Verification*. ACM, New York, NY, 73–82.

[90] Fabian Vogelsteller and Vitalik Buterin. 2020. ERC20 Token Standard. Retrieved from http://www.github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[91] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. 2011. Android permissions demystified. In *18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM, New York, NY, 627–638. DOI:https://doi/10.1145/2046707.2046779

[92] Andrew Ferraiuolo, Yao Wang, Rui Xu, Danfeng Zhang, Andrew C. Myers, and G. Edward Suh. 2017. *Full-processor Timing Channel Protection with Applications to Secure Hardware Compartments*. Technical Report. Cornell University Library.

[93] Andrew Ferraiuolo, Rui Xu, Danfeng Zhang, Andrew C. Myers, and G. Edward Suh. 2017. Verification of a practical hardware security architecture through static information flow analysis. *SIGARCH Comput. Archit. News* 45, 1 (Apr. 2017), 555–568. DOI:https://doi/10.1145/3093337.3037739

[94] D. Fett, P. Hosseyni, and R. Küsters. 2019. An extensive formal security analysis of the OpenID financial-grade API. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 453–471. DOI:https://doi/10.1109/SP.2019.00067

[95] Jean-Christophe Filliâtre and Claude Marché. 2007. The Why/Krakatoa/Caduceus platform for deductive program verification. In *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3–7, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4590)*, Werner Damm and Holger Hermanns (Eds.). Springer, Berlin, 173–177.

[96] Kathleen Fisher, John Launchbury, and Raymond Richards. 2017. The HACMS program: Using formal methods to eliminate exploitable bugs. *Philos. Trans. Roy. Societ. A: Math., Phys. Eng. Sci.* 375, 2104 (2017), 20150401.

[97] Mike Flynn, Tim Hoverd, and David Brazier. 1989. Formaliser — An interactive support tool for Z. In *Proceedings of the Fourth Annual Z User Meeting, Oxford, UK, December 15, 1989 (Workshops in Computing)*, John E. Nicholls (Ed.). Springer, Berlin, 128–141. DOI:https://doi/10.1007/978-1-4471-3877-8_8

[98] Leo Freitas. 2018. VDM at large: Modelling the EMV® 2nd generation kernel. In *Brazilian Symposium on Formal Methods*. Springer, Berlin, 109–125.

[99] Andrew Gacek, John Backes, Mike Whalen, Lucas G. Wagner, and Elaheh Ghassabani. 2017. The JKind Model Checker. Retrieved from http://arxiv.org/abs/1712.01222.

[100] Galois Inc. 2019. The Software Analysis Workbench. Retrieved from http://www.saw.galois.com/index.html.

[101] R. Gandhi, A. Sharma, W. Mahoney, W. Sousan, Q. Zhu, and P. Laplante. 2011. Dimensions of cyber-attacks: Cultural, social, economic, and political. *IEEE Technol. Societ. Mag.* 30, 1 (Spring 2011), 28–38. DOI:https://doi/10.1109/MTS.2011.940293

[102] A. H. Ghamarian, M. J. de Mol, Arend Rensink, Eduardo Zambon, and M. V. Zimakova. 2010. *Modelling and Analysis Using GROOVE*. Number TR-CTIT-10-18 in CTIT Technical Report Series. Centre for Telematics and Information Technology (CTIT), Netherlands.

[103] Thomas Gibson-Robinson. 2019. *FDR4: The CSP Refinement Checker*. Oxford University Department of Computer Science. Retrieved from www.cs.ox.ac.uk/projects/fdr/.

[104] Mario Gleirscher and Diego Marmsoler. 2020. Formal methods in dependable systems engineering: A survey of professionals from Europe and North America. *Empir. Softw. Eng.* 25, 6 (2020), 4473–4546.

[105] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *J. Comput. Syst. Sci.* 28, 2 (1984), 270–299.

[106] Matt Griffin and Brijesh Dongol. 2021. Verifying secure speculation in Isabelle/HOL. In *Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20–26, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13047)*, Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan (Eds.). Springer, 43–60. DOI:https://doi/10.1007/978-3-030-90870-6_3

[107] Roberto Guanciale, Musard Balliu, and Mads Dam. 2020. InSpectre: Breaking and fixing microarchitectural vulnerabilities by formal analysis. In *CCS'20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, November 9–13, 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM, 1853–1869. DOI:https://doi/10.1145/3372297.3417246

[108] Roberto Guanciale, Hamed Nemati, Mads Dam, and Christoph Baumann. 2016. Provably secure memory isolation for Linux on ARM: Submission to special issue on verified information flow security. *J. Comput. Secur.* 24 (12 2016), 793–837. DOI:https://doi/10.3233/JCS-160558

[109] Arjun Guha, Matthew Fredrikson, Benjamin Livshits, and Nikhil Swamy. 2011. Verified security for browser extensions. In *IEEE Symposium on Security and Privacy*. IEEE, 115–130.

[110] X. Guo, R. G. Dutta, P. Mishra, and Y. Jin. 2016. Automatic RTL-to-formal code converter for IP security formal verification. In *17th International Workshop on Microprocessor and SOC Test and Verification (MTV)*. IEEE, 35–38. DOI:https://doi/10.1109/MTV.2016.23

[111] X. Guo, R. G. Dutta, P. Mishra, and Y. Jin. 2016. Scalable SoC trust verification using integrated theorem proving and model checking. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 124–129. DOI:https://doi/10.1109/HST.2016.7495569

[112] X. Guo, R. G. Dutta, P. Mishra, and Y. Jin. 2017. Automatic code converter enhanced PCH framework for SoC trust verification. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 25, 12 (Dec. 2017), 3390–3400. DOI:https://doi/10.1109/TVLSI.2017.2751615

[113] J. Octavio Gutierrez-Garcia and Kwang Sim. 2010. Agent-based service composition in cloud computing. *Commun. Comput. Inf. Sci.* 121, 1–10. DOI:https://doi/10.1007/978-3-642-17625-8_1

[114] Muluken Hailesellasie and Syed Rafay Hasan. 2018. Intrusion detection in PLC-based industrial control systems using formal verification approach in conjunction with graphs. *J. Hardw. Syst. Secur.* 2, 1 (01 Mar. 2018), 1–14. DOI:https://doi/10.1007/s41635-017-0017-y

[115] Anthony Hall. 2005. Realising the benefits of formal methods. In *Formal Methods and Software Engineering*, Kung-Kiu Lau and Richard Banach (Eds.). Springer, Berlin, 1–4.

[116] Dick Hamlet. 2005. Invariants and state in testing and formal methods. *SIGSOFT Softw. Eng. Notes* 31, 1 (Sep. 2005), 48–51. DOI:https://doi/10.1145/1108768.1108806

[117] Jianan Hao, Yang Liu, Wentong Cai, Guangdong Bai, and Jun Sun. 2013. vTRUST: A formal modeling and verification framework for virtualization systems. In *Formal Methods and Software Engineering*, Lindsay Groves and Jing Sun (Eds.). Springer, Berlin, 329–346.

[118] Daniel Hartung and Christoph Busch. 2012. Biometric transaction authentication protocol: Formal model verification and "four-eyes" principle extension. In *Financial Cryptography and Data Security*, George Danezis, Sven Dietrich, and Kazue Sako (Eds.). Springer, Berlin, 88–103.

[119] Dominik Harz and William Knottenbelt. 2018. Towards Safer Smart Contracts: A Survey of Languages and Verification Methods. arXiv:1809.09805v4. https://arxiv.org/abs/1809.09805.

[120] J. Heneghan, S. A. Shaikh, J. Bryans, M. Cheah, and P. Wooderson. 2019. Enabling security checking of automotive ECUs with formal CSP models. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 90–97. DOI:https://doi/10.1109/DSN-W.2019.00025

[121] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu, and G. Rosu. 2018. KEVM: A complete formal semantics of the Ethereum virtual machine. In *IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 204–217.

[122] C. A. R. Hoare. 1978. Communicating sequential processes. *Commun. ACM* 21, 8 (Aug. 1978), 666–677. DOI:https://doi/10.1145/359576.359585

[123] C. A. R. Hoare. 1985. *Communicating Sequential Processes*. Prentice-Hall, USA.

[124] Bo-Yuan Huang, Sayak Ray, Aarti Gupta, Jason M. Fung, and Sharad Malik. 2018. Formal security verification of concurrent firmware in SoCs using instruction-level abstraction for hardware. In *55th Annual Design Automation Conference (DAC'18)*. ACM, New York, NY, 91:1–91:6. DOI:https://doi/10.1145/3195970.3196055

[125] John Hughes and Eve Maler. 2005. Security assertion markup language (SAML) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08* 13 (2005).

[126] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone. 2019. Formal methods for Android banking malware analysis and detection. In *6th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 331–336.

[127] Fahad Ibrar, Hamza Saleem, Sam Castle, and Muhammad Zubair Malik. 2017. A study of static analysis tools to detect vulnerabilities of branchless banking applications in developing countries. In *9th International Conference on Information and Communication Technologies and Development (ICTD'17)*. Association for Computing Machinery, New York, NY. DOI:https://doi/10.1145/3136560.3136595

[128] Daisuke Inoue and Masahiro Kuroda. 2006. Secure service framework on mobile ethernet. *J. Nat. Inst. Inf. Commun. Technol.* 53 (12 2006), 61–71.

[129] ITSEC. 1991. *Information Technology Security Evaluation Criteria (ITSEC): Preliminary Harmonised Criteria*. Document COM(90) 314, Version 1.2. Commission of the European Communities.

[130] Daniel Jackson. 2000. Automating first-order relational logic. *ACM SIGSOFT Symp. Found. Softw. Eng.* 25 (09 2000). DOI:https://doi/10.1145/357474.355063

[131] Daniel Jackson. 2012. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, Cambridge, MA.

[132] Daniel Jackson and Jeanette Wing. 1996. Lightweight formal methods. *IEEE Comput.* 29, 4 (Apr. 1996), 22–23.

[133] Dongseok Jang, Zachary Tatlock, and Sorin Lerner. 2012. Establishing browser security guarantees through formal shim verification. In *21st USENIX Conference on Security Symposium (Security'12)*. USENIX Association, USA.

[134] Dongseok Jang, Zachary Tatlock, and Sorin Lerner. 2012. Establishing browser security guarantees through formal shim verification. In *21st USENIX Security Symposium (USENIX Security'12)*. USENIX, 113–128. Retrieved from www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/jang.

[135] Y. Jarraya, A. Eghtesadi, M. Debbabi, Y. Zhang, and M. Pourzandi. 2012. Cloud calculus: Security verification in elastic cloud computing platform. In *International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 447–454. DOI:https://doi/10.1109/CTS.2012.6261089

[136] Karthick Jayaraman, Nikolaj Bjørner, Geoff Outhred, and Charlie Kaufman. 2014. *Automated Analysis and Debugging of Network Connectivity Policies*. Technical Report. Tech. Rep. MSR-TR-2014-102. MSR, Seattle, WA.

[137] Kurt Jensen, Lars Michael Kristensen, and Lisa Marie Wells. 2007. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf.* 9, 3/4 (2007), 213–254. DOI:https://doi/10.1007/s10009-007-0038-x

[138] Kurt Jensen and Lars M. Kristensen. 2009. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems* (1st ed.). Springer Publishing Company, Incorporated, Berlin.

[139] Richard Jüllig. 2002. Formal methods in enterprise computing. In *Formal Methods and Software Engineering*, Chris George and Huaikou Miao (Eds.). Springer, Berlin, 22–23.

[140] Corey Kallenberg, Sam Cornwell, Xeno Kovah, and John Butterworth. 2014. Setup for failure: Defeating secure boot. The MITRE Corporation. Retrieved https://infocon.org/cons/SyScan/SyScan%202014%20Singapore/SyScan%202014%20presentations/SyScan2014_CoreyKallenberg_SetupforFailureDefeatingSecureBoot_WP.pdf.

[141] Sheetal Kalra and Sandeep K. Sood. 2015. Secure authentication scheme for IoT and cloud servers. *Pervas. Mob. Comput.* 24 (2015), 210–223. Retrieved from http://www.sciencedirect.com/science/article/pii/S1574119215001510.

[142] Michael Kenney. 2015. Cyber-terrorism in a post-Stuxnet world. *Orbis* 59 (12 2015). DOI:https://doi/10.1016/j.orbis.2014.11.009

[143] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke. 2010. Smart-grid security issues. *IEEE Secur. Priv.* 8, 1 (Jan. 2010), 81–85. DOI:https://doi/10.1109/MSP.2010.49

[144] Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2010. seL4: Formal verification of an operating-system kernel. *Commun. ACM* 53, 6 (2010), 107–115.

[145] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. 2019. Spectre attacks: Exploiting speculative execution. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 1–19.

[146] S. Kottler, M. Khayamy, S. R. Hasan, and O. Elkeelany. 2017. Formal verification of ladder logic programs using NuSMV. In *SoutheastCon*. IEEE, 1–5. DOI:https://doi/10.1109/SECON.2017.7925390

[147] Simon Kramer and Julian C. Bradfield. 2010. A general definition of malware. *J. Comput. Virol.* 6, 2 (2010), 105–114.

[148] Siwar Kriaa, Ludovic Pietre-Cambacedes, Marc Bouissou, and Yoran Halgand. 2015. A survey of approaches combining safety and security for industrial control systems. *Reliab. Eng. Syst. Safety* 139 (2015), 156–178. Retrieved from http://www.sciencedirect.com/science/article/pii/S0951832015000538.

[149] Tomas Kulik, Brijesh Dongol, Peter Gorm Larsen, Hugo Daniel Macedo, Steve Schneider, Peter Würtz Vinther Tran-Jørgensen, and Jim Woodcock. 2021. A Survey of Practical Formal Methods for Security. arXiv:2109.01362 [cs.FL].

[150] Tomas Kulik, Peter W. V. Tran-Jørgensen, and Jalil Boudjadar. 2019. Formal security analysis of cloud-connected industrial control systems. In *Innovative Security Solutions for Information Technology and Communications*, Jean-Louis Lanet and Cristian Toma (Eds.). Springer International Publishing, Cham, 71–84.

[151] Apurva Kumar. 2014. A lightweight formal approach for analyzing security of web protocols. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 192–211.

[152] N. Kumar, V. Kumar, and M. Gaur. 2019. Banking trojans APK detection using formal methods. In *4th International Conference on Information Systems and Computer Networks (ISCON)*. IEEE, 606–609.

[153] P. Kumar, A. Braeken, A. Gurtov, J. Iinatti, and P. H. Ha. 2017. Anonymous secure framework in connected smart home environments. *IEEE Trans. Inf. Forens. Secur.* 12, 4 (Apr. 2017), 968–979. DOI:https://doi/10.1109/TIFS.2016.2647225

[154] Robert Künnemann and Graham Steel. 2013. YubiSecure? Formal security analysis results for the Yubikey and YubiHSM. In *Security and Trust Management*, Audun Jøsang, Pierangela Samarati, and Marinella Petrocchi (Eds.). Springer Berlin, 257–272.

[155] Stefan Kupferschmid, Matthew Lewis, Tobias Schubert, and Bernd Becker. 2011. Incremental preprocessing methods for use in BMC. *Form. Meth. Syst. Des.* 39, 2 (2011), 185–204.

[156] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.

[157] Akash Lal, Shaz Qadeer, and Shuvendu K. Lahiri. 2012. A solver for reachability modulo theories. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, July 7–13, 2012 Proceedings (Lecture Notes in Computer Science, Vol. 7358)*, P. Madhusudan and Sanjit A. Seshia (Eds.). Springer, Berlin, 427–443.

[158] Leslie Lamport. 2002. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[159] Kim G. Larsen, Paul Pettersson, and Wang Yi. 1997. Uppaal in a nutshell. *Int. J. Softw. Tools Technol. Transfer* 1, 1 (01 Dec. 1997), 134–152. DOI:https://doi/10.1007/s100090050010

[160] P. G. Larsen, B. S. Hansen, H. Brunn, N. Plat, H. Toetenel, D. J. Andrews, J. Dawes, G. Parkin, et al. 1996. Information Technology – Programming Languages, Their Environments and System Software Interfaces – Vienna Development Method – Specification Language – Part 1: Base language. ISO/IEC 13817-1:1996.

[161] Peter Gorm Larsen, Kenneth Lausdahl, and Nick Battle. 2010. Combinatorial testing for VDM. In *8th IEEE International Conference on Software Engineering and Formal Methods (SEFM'10)*. IEEE Computer Society, Washington, DC, 278–285. DOI:https://doi/10.1109/SEFM.2010.32. ISBN 978-0-7695-4153-2.

[162] Lee W. Lerner, Zane R. Franklin, William T. Baumann, and Cameron D. Patterson. 2014. Using high-level synthesis and formal analysis to predict and preempt attacks on industrial control systems. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'14)*. Association for Computing Machinery, New York, NY, 209–212. DOI:https://doi/10.1145/2554688.2554759

[163] Thomas Letan, Pierre Chifflier, Guillaume Hiet, Pierre Neron, and Benjamin Morin. 2016. SpecCert: Specifying and verifying hardware-based security enforcement. In *Proceedings of the FM 2016: Formal Methods, John Fitzgerald, Constance Heitmeyer*, Stefania Gnesi, and Anna Philippou (Eds.). 496–512.

[164] E. Love, Y. Jin, and Y. Makris. 2011. Enhancing security via provably trustworthy hardware intellectual property. In *IEEE International Symposium on Hardware-oriented Security and Trust*. IEEE, 12–17. DOI:https://doi/10.1109/HST.2011.5954988

[165] Gavin Lowe. 1995. An attack on the Needham-Schroeder public-key authentication protocol. *Inf. Process. Lett.* 56, 3 (1995), 131–133.

[166] Hugo Daniel Macedo and José Nuno Oliveira. 2015. A linear algebra approach to OLAP. *Form. Asp. Comput.* 27, 2 (2015), 283–307.

[167] Hugo Daniel Macedo and Tayssir Touili. 2013. Mining malware specifications through static reachability analysis. In *European Symposium on Research in Computer Security*. Springer Berlin, 517–535.

[168] N. E. Madhoun, F. Guenane, and G. Pujolle. 2016. An online security protocol for NFC payment: Formally analyzed by the Scyther tool. In *2nd International Conference on Mobile and Secure Services (MobiSecServ)*. IEEE, 1–7.

[169] Taous Madi, Yosr Jarraya, Amir Alimohammadifar, Suryadipta Majumdar, Yushun Wang, Makan Pourzandi, Lingyu Wang, and Mourad Debbabi. 2018. ISOTOP: Auditing virtual networks isolation across cloud layers in OpenStack. *ACM Trans. Priv. Secur.* 22, 1 (Oct. 2018). DOI:https://doi/10.1145/3267339

[170] Haohui Mai, Edgar Pek, Hui Xue, Samuel Talmadge King, and Parthasarathy Madhusudan. 2013. Verifying security invariants in ExpressOS. In *18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'13)*. Association for Computing Machinery, New York, NY, 293–304. DOI:https://doi/10.1145/2451116.2451148

[171] Antonio Marcedone, Rafael Pass, and Abhi Shelat. 2019. Minimizing trust in hardware wallets with two factor signatures. In *Financial Cryptography and Data Security*, Ian Goldberg and Tyler Moore (Eds.). Springer International Publishing, Cham, 407–425.

[172] Fabio Martinelli, Francesco Mercaldo, and Vittoria Nardone. 2018. Identifying insecure features in android applications using model checking. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22–24, 2018*, Paolo Mori, Steven Furnell, and Olivier Camp (Eds.). SciTePress, 589–596.

[173] Fatma Masmoudi, Monia Loulou, and Ahmed Hadj Kacem. 2014. Formal security framework for agent based cloud systems. In *International Workshop on Advanced Information Systems for Enterprises*. DOI:https://doi/10.1109/IWAISE.2014.15

[174] Jackson R. Mayo, Robert C. Armstrong, and Geoffrey C. Hulette. 2015. Digital system robustness via design constraints: The lesson of formal methods. In *Annual IEEE Systems Conference (SysCon)*. IEEE, 109–114.

[175] Maryam Mehrnezhad, Mohammed Aamir Ali, Feng Hao, and Aad van Moorsel. 2016. NFC payment spy: A privacy attack on contactless payments. In *Security Standardisation Research*, Lidong Chen, David McGrew, and Chris Mitchell (Eds.). Springer International Publishing, Cham, 92–111.

[176] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013, Proceedings (Lecture Notes in Computer Science, Vol. 8044)*, Natasha Sharygina and Helmut Veith (Eds.). Springer, Berlin, 696–701. DOI:https://doi/10.1007/978-3-642-39799-8_48

[177] F. Mercaldo, F. Martinelli, and A. Santone. 2019. Real-time SCADA attack detection by means of formal methods. In *IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 231–236. DOI:https://doi/10.1109/WETICE.2019.00057

[178] Francesco Mercaldo, Vittoria Nardone, Antonella Santone, and Corrado Aaron Visaggio. 2016. Download malware? No, thanks: How formal methods can block update attacks. In *4th FME Workshop on Formal Methods in Software Engineering (FormaliSE'16)*. ACM, New York, NY. DOI:https://doi/10.1145/2897667.2897673

[179] José Meseguer. 2000. Rewriting logic and Maude: A wide-spectrum semantic framework for object-based distributed systems. In *Formal Methods for Open Object-based Distributed Systems IV*, Scott F. Smith and Carolyn L. Talcott (Eds.). Springer US, Boston, MA, 89–117.

[180] Andrew Miller, Zhicheng Cai, and Somesh Jha. 2018. Smart contracts and opportunities for formal methods. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 280–299.

[181] R. Milner. 1989. *Communication and Concurrency*. Prentice-Hall, Inc., USA.

[182] M. Mohsin, M. U. Sardar, O. Hasan, and Z. Anwar. 2017. IoTRiskAnalyzer: A probabilistic model checking based framework for formal risk analytics of the Internet of Things. *IEEE Access* 5 (2017), 5494–5505. DOI:https://doi/10.1109/ACCESS.2017.2696031

[183] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. 2012. IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer* 45, 12 (2012), 65–72.

[184] Greg Morrisett, Gang Tan, Joseph Tassarotti, Jean-Baptiste Tristan, and Edward Gan. 2012. RockSalt: Better, faster, stronger SFI for the X86. In *33rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'12)*. Association for Computing Machinery, New York, NY, 395–404. DOI:https://doi/10.1145/2254064.2254111

[185] Sascha Mühlbach and Sebastian Wallner. 2008. Secure communication in microcomputer bus systems for embedded devices. *J. Syst. Archit.* 54, 11 (2008), 1065–1076. Retrieved from http://www.sciencedirect.com/science/article/pii/S1383762108000611.

[186] Roberto Nardone, Ugo Gentile, Adriano Peron, Massimo Benerecetti, Valeria Vittorini, Stefano Marrone, Renato De Guglielmo, Nicola Mazzocca, and Luigi Velardi. 2015. Dynamic state machines for formalizing railway control system specifications. In *Formal Techniques for Safety-critical Systems*, Cyrille Artho and Peter Csaba Ölveczky (Eds.). Springer International Publishing, Cham, 93–109.

[187] R. Nardone, R. J. Rodríguez, and S. Marrone. 2016. Formal security assessment of Modbus protocol. In *11th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 142–147. DOI:https://doi/10.1109/ICITST.2016.7856685

[188] Zainalabedin Navabi. 1993. *VHDL: Analysis and Modeling of Digital Systems*, Vol. 2. McGraw-Hill New York.

[189] George C. Necula. 2011. Proof-carrying code. In *Encyclopedia of Cryptography and Security, 2nd ed.*, Henk C. A. van Tilborg and Sushil Jajodia (Eds.). Springer, Berlin, 984–986.

[190] Roger M. Needham and Michael D. Schroeder. 1978. Using encryption for authentication in large networks of computers. *Commun. ACM* 21, 12 (1978), 993–999.

[191] Matias Negrete-Pincetic, Felipe Yoshida, and George Gross. 2009. Towards quantifying the impacts of cyber attacks in the competitive electricity market environment. In *IEEE Bucharest PowerTech Conference*. IEEE, 1–8. DOI:https://doi/10.1109/PTC.2009.5282237

[192] B. Ngabonziza, D. Martin, A. Bailey, H. Cho, and S. Martin. 2016. TrustZone explained: Architectural features and use cases. In *IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 445–451. DOI:https://doi/10.1109/CIC.2016.065

[193] V. Nigam and C. Talcott. 2019. Formal security verification of industry 4.0 applications. In *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1043–1050. DOI:https://doi/10.1109/ETFA.2019.8869428

[194] Tobias Nipkow and Gerwin Klein. 2014. *Concrete Semantics — with Isabelle/HOL*. Springer, Berlin.

[195] P. J. C. Nunes, J. Fonseca, and M. Vieira. 2015. phpSAFE: A security analysis tool for OOP web application plugins. In *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 299–306. DOI:https://doi/10.1109/DSN.2015.16

[196] Peter W. O'Hearn. 2018. Continuous reasoning: Scaling the impact of formal methods. In *33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'18)*. Association for Computing Machinery, New York, NY, 13–25. DOI:https://doi/10.1145/3209108.3209109

[197] Iqra Obaid, Syed Kazmi, and Awais Qasim. 2017. Modeling and verification of payment system in E-banking. *Int. J. Adv. Comput. Sci. Applic.* 8 (01 2017). DOI:https://doi/10.14569/IJACSA.2017.080825

[198] Daejun Park, Yi Zhang, Manasvi Saxena, Philip Daian, and Grigore Roşu. 2018. A formal verification tool for Ethereum VM bytecode. In *26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*. Association for Computing Machinery, New York, NY, 912–915. DOI:https://doi/10.1145/3236024.3264591.

[199] Daejun Park, Yi Zhang, Manasvi Saxena, Philip Daian, and Grigore Roşu. 2018. A formal verification tool for ethereum VM bytecode. In *2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, 912–915. DOI:https://doi/10.1145/3236024.3264591

[200] Jianhua Peng, Feng Liu, Zhenju Zhao, Danqing Huang, and Rui Xue. 2010. ASM-SPV: A model checker for security protocols. In *6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 458–461. DOI:https://doi/10.1109/IIHMSP.2010.117

[201] Uta Priss. 2006. Formal concept analysis in information science. *Ann. Rev. Inf. Sci. Technol.* 40 (01 2006). DOI:https://doi/10.1002/aris.1440400120

[202] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., USA.

[203] Maxime Puys, Marie-Laure Potet, and Pascal Lafourcade. 2016. Formal analysis of security properties on the OPC-UA SCADA protocol. In *Computer Safety, Reliability, and Security*, Amund Skavhaug, Jérémie Guiochet, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 67–75.

[204] Davide Quarta, Marcello Pogliani, Mario Polino, Federico Maggi, Andrea Maria Zanchettin, and Stefano Zanero. 2017. An experimental security analysis of an industrial robot controller. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 268–286. DOI:https://doi/10.1109/SP.2017.20

[205] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: An Open-source Robot Operating System. Retrieved http://robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf.

[206] Zvonimir Rakamaric and Michael Emmi. 2014. SMACK: Decoupling source language details from verifier implementations. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18–22, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8559)*, Armin Biere and Roderick Bloem (Eds.). Springer, Berlin, 106–113.

[207] R. Rana, M. Staron, C. Berger, A. Nilsson, R. Scandariato, A. Weilenmann, and M. Rydmark. 2015. On the role of cross-disciplinary research and SSE in addressing the challenges of the digitalization of society. In *6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 1106–1109. DOI:https://doi/10.1109/ICSESS.2015.7339245

[208] Awais Rashid, Howard Chivers, Emil Lupu, Andrew Martin, and Steve Schneider (Eds.). 2021. *The Cyber Security Body of Knowledge v1.1*. University of Bristol. Retrieved from www.cybok.org.

[209] R. Rieke, M. Zhdanova, J. Repp, R. Giot, and C. Gaber. 2013. Fraud detection in Mobile payments utilizing process behavior analysis. In *International Conference on Availability, Reliability and Security*. IEEE, 662–669.

[210] Marco Rocchetto and Nils Ole Tippenhauer. 2016. CPDY: Extending the Dolev-Yao Attacker with Physical-Layer Interactions. Retrieved from http://arxiv.org/abs/1607.02562.

[211] Marco Rocchetto and Nils Ole Tippenhauer. 2017. Towards formal security analysis of industrial control systems. In *ACM on Asia Conference on Computer and Communications Security (ASIA CCS'17)*. ACM, New York, NY, 114–126. DOI:https://doi/10.1145/3052973.3053024

[212] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos. 2017. On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services. *IEEE Access* 5 (2017), 25808–25825. DOI:https://doi/10.1109/ACCESS.2017.2764913

[213] Peter Y. A. Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. 2001. *Modelling and Analysis of Security Protocols*. Addison-Wesley-Longman, USA.

[214] O. Rysavy, J. Rab, and M. Sveda. 2013. Improving security in SCADA systems through firewall policy analysis. In *Federated Conference on Computer Science and Information Systems*. IEEE, 1435–1440.

[215] Caitlin Sadowski, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon, and Ciera Jaspan. 2018. Lessons from building static analysis tools at Google. *Commun. ACM* 61, 4 (Mar. 2018), 58–66. DOI:https://doi/10.1145/3188720

[216] Antonella Santone, Valentina Intilangelo, and Domenico Raucci. 2013. Efficient formal verification in banking processes. In *IEEE 9th World Congress on Services*. IEEE, 325–332.

[217] N. Santos, Krishna P. Gummadi, and Rodrigo Rodrigues. 2009. Towards trusted cloud computing. In *Conference on Hot Topics in Cloud Computing*. USENIX Association.

[218] Ralf Sasse, Samuel T. King, José Meseguer, and Shuo Tang. 2012. IBOS: A correct-by-construction modular browser. In *International Workshop on Formal Aspects of Component Software*. Springer, Berlin, 224–241.

[219] Jinho Seol, Seongwook Jin, Daewoo Lee, Jaehyuk Huh, and Seungryoul Maeng. 2015. A trusted IaaS environment with hardware security module. *IEEE Trans. Serv. Comput.* 9 (01 2015), 1–1. DOI:https://doi/10.1109/TSC.2015.2392099

[220] J. Sepulveda, D. Aboul-Hassan, G. Sigl, B. Becker, and M. Sauer. 2018. Towards the formal verification of security properties of a Network-on-Chip router. In *IEEE 23rd European Test Symposium (ETS)*. IEEE, 1–6. DOI:https://doi/10.1109/ETS.2018.8400692

[221] Rida Shaukat, Arooba Shahoor, and Aniqa Urooj. 2018. Probing into code analysis tools: A comparison of C# supporting static code analyzers. In *15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. 455–464. DOI:https://doi/10.1109/IBCAST.2018.8312264

[222] Yuchao She, Hui Li, and Hui Zhu. 2013. UVHM: Model checking based formal analysis scheme for hypervisors. In *Information and Communication Technology*, Khabib Mustofa, Erich J. Neuhold, A. Min Tjoa, Edgar Weippl, and Ilsun You (Eds.). Springer, Berlin, 300–305.

[223] Roshan Shrestha, Hoda Mehrpouyan, and Dianxiang Xu. 2018. Model checking of security properties in industrial control systems (ICS). In *8th ACM Conference on Data and Application Security and Privacy (CODASPY'18)*. Association for Computing Machinery, New York, NY, 164–166. DOI:https://doi/10.1145/3176258.3176949

[224] Irfan Siddavatam, Sachin Parekh, Tanay Shah, and Faruk Kazi. 2017. Testing and validation of Modbus/TCP protocol for secure SCADA communication in CPS using formal methods. *Scalable Comput.: Pract. Exper.* 18 (11 2017). DOI:https://doi/10.12694/scpe.v18i4.1331

[225] Julien Signoles, Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, and Boris Yakobowski. 2012. Frama-C: A software analysis perspective. *Form. Asp. Comput.* 27 (10 2012). DOI:https://doi/10.1007/s00165-014-0326-7

[226] R. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury. 2014. A verification platform for SDN-enabled applications. In *IEEE International Conference on Cloud Engineering*. IEEE, 337–342. DOI:https://doi/10.1109/IC2E.2014.72

[227] Eric Smith and Alessandro Coglio. 2016. Android platform modeling and Android app verification in the ACL2 theorem prover. In *Verified Software: Theories, Tools, and Experiments*, Arie Gurfinkel and Sanjit A. Seshia (Eds.). Springer International Publishing, Cham, 183–201.

[228] E. Snekkenes. 1991. Exploring the BAN approach to protocol analysis. In *IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE, 171–181. DOI:https://doi/10.1109/RISP.1991.130785

[229] Fu Song and Tayssir Touili. 2012. Efficient malware detection using model-checking. In *International Symposium on Formal Methods*. Springer, Berlin, 418–433.

[230] Fu Song and Tayssir Touili. 2014. Model-checking for Android malware detection. In *Programming Languages and Systems*, Jacques Garrigue (Ed.). Springer International Publishing, Cham, 216–235.

[231] Fu Song and Tayssir Touili. 2014. Pushdown model checking for malware detection. *Int. J. Softw. Tools Technol. Transfer* 16, 2 (2014), 147–173.

[232] S. Souaf, P. Berthome, and F. Loulergue. 2018. A cloud brokerage solution: Formal methods meet security in cloud federations. In *International Conference on High Performance Computing Simulation (HPCS)*. 691–699. DOI:https://doi/10.1109/HPCS.2018.00113

[233] J. M. Spivey. 1989. *The Z Notation: A Reference Manual*. Prentice-Hall, USA.

[234] Andrei Stefanescu, Daejun Park, Shijiao Yuwen, Yilong Li, and Grigore Roşu. 2016. Semantics-based program verifiers for all languages. In *ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'16)*. Association for Computing Machinery, New York, NY, 74–91. DOI:https://doi/10.1145/2983990.2984027

[235] Susan Stepney, David Cooper, and Jim Woodcock. 2000. *An Electronic Purse: Specification, Refinement, and Proof*. Technical Monograph PRG-126. Oxford University Computing Laboratory.

[236] C. Steward Jr., L. A. Wahsheh, A. Ahmad, J. M. Graham, C. V. Hinds, A. T. Williams, and S. J. DeLoatch. 2012. Software security: The dangerous afterthought. In *9th International Conference on Information Technology - New Generations*. IEEE, 815–818. DOI:https://doi/10.1109/ITNG.2012.60

[237] Jun Sun, Yang Liu, and Jin Song Dong. 2008. Model checking CSP revisited: Introducing a process analysis toolkit. In *Leveraging Applications of Formal Methods, Verification and Validation*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer Berlin, 307–322.

[238] J. Sun, Y. Liu, J. S. Dong, and C. Chen. 2009. Integrating specification and programs for system modeling and verification. In *3rd IEEE International Symposium on Theoretical Aspects of Software Engineering*. IEEE, 127–135.

[239] Farid Molazem Tabrizi and Karthik Pattabiraman. 2016. Formal security analysis of smart embedded systems. In *32nd Annual Conference on Computer Security Applications (ACSAC'16)*. Association for Computing Machinery, New York, NY, 1–15. DOI:https://doi/10.1145/2991079.2991085

[240] Naoyuki Tamura, Tomoya Tanjo, and Mutsunori Banbara. 2008. System Description of a SAT-based CSP Solver Sugar., 71–75 pages. Retrieved https://tamura70.gitlab.io/papers/pdf/cpai08t.pdf.

[241] Vincent F. Taylor and Ivan Martinovic. 2017. Short paper: A longitudinal study of financial apps in the Google Play store. In *Financial Cryptography and Data Security*, Aggelos Kiayias (Ed.). Springer International Publishing, Cham, 302–309.

[242] The Coq Development Team. 2019. *The Coq Reference Manual*. LogiCal Project. Retrieved from http://coq.inria.fr. Version 8.9.1.

[243] Donald Thomas and Philip Moorby. 2008. *The Verilog® Hardware Description Language*. Springer Science & Business Media.

[244] Emina Torlak and Daniel Jackson. 2007. Kodkod: A relational model finder. In *Tools and Algorithms for the Construction and Analysis of Systems*, Orna Grumberg and Michael Huth (Eds.). Springer Berlin, 632–647.

[245] Peter Würtz Vinther Tran-Jørgensen and Tomas Kulik. 2019. Migrating overture to a different IDE. In *17th Overture Workshop (Technical Report Series, CS-TR- 1530 - 2019)*, Carl Gamble and Luis Diogo Couto (Eds.). Newcastle University, UK, 32–47.

[246] Peter W. V. Tran-Jørgensen, Tomas Kulik, Jalil Boudjadar, and Peter Gorm Larsen. 2019. Security analysis of cloud-connected industrial control systems using combinatorial testing. In *17th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE'19)*. Association for Computing Machinery, New York, NY. DOI:https://doi/10.1145/3359986.3361211

[247] K. Tsukada, K. Sawada, and S. Shin. 2016. A toolchain on model checking SPIN via Kalman Decomposition for control system software. In *IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 300–305. DOI:https://doi/10.1109/COASE.2016.7743421

[248] Mathieu Turuani. 2006. The CL-Atse protocol analyser. In *Term Rewriting and Applications*, Frank Pfenning (Ed.). Springer, Berlin, 277–286.

[249] Mathieu Turuani, Thomas Voegtlin, and Michael Rusinowitch. 2016. Automated verification of Electrum wallet. In *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, 27–42.

[250] Nils Urbach and Frederik Ahlemann. 2019. *Digitalization as a Risk: Security and Business Continuity Management Are Central Cross-Divisional Functions of the Company*. Springer International Publishing, Cham, 85–92. DOI:https://doi/10.1007/978-3-319-96187-3_9

[251] A. Vasudevan, S. Chaki, L. Jia, J. McCune, J. Newsome, and A. Datta. 2013. Design, implementation and verification of an eXtensible and modular hypervisor framework. In *IEEE Symposium on Security and Privacy*. IEEE, 430–444. DOI:https://doi/10.1109/SP.2013.36

[252] Amit Vasudevan, Sagar Chaki, Petros Maniatis, Limin Jia, and Anupam Datta. 2016. überSpark: Enforcing verifiable object abstractions for automated compositional security analysis of a hypervisor. In *25th USENIX Security Symposium (USENIX Security'16)*. USENIX Association, 87–104. Retrieved from www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/vasudevan.

[253] David von Oheimb and Sebastian Mödersheim. 2012. ASLan++ — A formal security specification language for distributed systems. In *Formal Methods for Components and Objects*, Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue (Eds.). Springer, Berlin, 1–22.

[254] R. Wang, Y. Guan, H. Song, X. Li, X. Li, Z. Shi, and X. Song. 2019. A formal model-based design method for robotic systems. *IEEE Syst. J.* 13, 1 (Mar. 2019), 1096–1107. DOI:https://doi/10.1109/JSYST.2018.2867285

[255] T. Wang, Q. Su, and T. Chen. 2017. Formal analysis of security properties of cyber-physical system based on timed automata. In *IEEE 2nd International Conference on Data Science in Cyberspace (DSC)*. 534–540. DOI:https://doi/10.1109/DSC.2017.44

[256] W. Wang, Q. Zeng, and A. P. Mathur. 2012. A security assurance framework combining formal verification and security functional testing. In *12th International Conference on Quality Software*. 136–139. DOI:https://doi/10.1109/QSIC.2012.34

[257] Dean C. Wardell, Robert F. Mills, Gilbert L. Peterson, and Mark E. Oxley. 2016. A method for revealing and addressing security vulnerabilities in cyber-physical systems by modeling malicious agent interactions with formal verification. *Procedia Comput. Sci.* 95 (2016), 24–31. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877050916324619.

[258] Sharon Weinberger. 2011. Computer security: Is this the start of cyberwarfare? *Nature* 474 (06 2011), 142–5. DOI:https://doi/10.1038/474142a

[259] Tobias Wich, Daniel Nemmert, and Detlef Hühnlein. 2017. Towards secure and standard-compliant implementations of the PSD2 directive. In *Open Identity Summit 2017, October 5–6, 2017, Karlstad University, Sweden (LNI, Vol. P-277)*, Lothar Fritsch, Heiko Roßnagel, and Detlef Hühnlein (Eds.). Gesellschaft für Informatik, Bonn, DE, 63–80. Retrieved from http://www.dl.gi.de/20.500.12116/3581.

[260] M. Williams, L. Axon, J. R. C. Nurse, and S. Creese. 2016. Future scenarios and challenges for security and privacy. In *IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI)*. IEEE, 1–6. DOI:https://doi/10.1109/RTSI.2016.7740625

[261] J. M. Wing. 1990. A specifier's introduction to formal methods. *Computer* 23, 9 (Sep. 1990), 8–22. DOI:https://doi/10.1109/2.58215

[262] Jeannette M. Wing. 1998. A symbiotic relationship between formal methods and security. In *Proceedings Computer Security, Dependability, and Assurance: From Needs to Solutions (Cat. No. 98EX358)*. IEEE, 26–38.

[263] Jim Woodcock and Jim Davies. 1996. *Using Z: Specification, Refinement, and Proof*. Prentice-Hall, USA.

[264] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. 2009. Formal methods: Practice and experience. *ACM Comput. Surv.* 41, 4 (Oct. 2009). DOI:https://doi/10.1145/1592434.1592436

[265] Jim Woodcock, Susan Stepney, David Cooper, John Clark, and Jeremy Jacob. 2008. The certification of the Mondex electronic purse to ITSEC Level E6. *Formal Asp. Comput.* 20, 1 (2008), 5–19.

[266] Meihua Xiao, Zilong Wan, and Hongling Liu. 2014. The formal verification and improvement of simplified SET protocol. *J. Softw.* 9 (09 2014). DOI:https://doi/10.4304/jsw.9.9.2302-2308

[267] Luyi Xing, Yangyi Chen, XiaoFeng Wang, and Shuo Chen. 2013. InteGuard: Toward automatic protection of third-party web service integrations. In *20th Annual Network and Distributed System Security Symposium*. The Internet Society. Retrieved from www.ndss-symposium.org/ndss2013/integuard-toward-automatic-protection-third-party-web-service-integrations.

[268] J. Yoo, Y. Jung, D. Shin, M. Bae, and E. Jee. 2019. Formal modeling and verification of a federated byzantine agreement algorithm for blockchain platforms. In *IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 11–21.

[269] Yu Zheng, D. He, Xiaohu Tang, and Hongxia Wang. 2005. AKA and authorization scheme for 4G Mobile networks based on trusted mobile platform. In *5th International Conference on Information Communications Signal Processing*. 976–980. DOI:https://doi/10.1109/ICICS.2005.1689196

[270] Wen Zeng, Maciej Koutny, Paul Watson, and Vasileios Germanos. 2016. Formal verification of secure information flow in cloud computing. *J. Inf. Secur. Applic.* 27 (2016), 103–116.

[271] Danfeng Zhang, Yao Wang, G. Edward Suh, and Andrew C. Myers. 2015. A hardware design language for timing-sensitive information-flow security. In *20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'15)*. Association for Computing Machinery, New York, NY, 503–516. DOI:https://doi/10.1145/2694344.2694372

[272] Wei Zhang, Wenke Ma, Huiling Shi, and Fu-qiang Zhu. 2012. Model checking and verification of the internet payment system with SPIN. *JSW* 7, 9 (2012), 1941–1949.

[273] S. Zonouz, J. Rrushi, and S. McLaughlin. 2014. Detecting industrial control malware using automated PLC code analytics. *IEEE Secur. Priv.* 12, 6 (Nov. 2014), 40–47. DOI:https://doi/10.1109/MSP.2014.113

Received April 2021; accepted January 2022