Computer Science Review 57 (2025) 100758

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

# The Beauty and the Beast: A survey on process algebras and cybersecurity

Gabriele Costa, Silvia De Francisci *, Rocco De Nicola

*IMT School for Advanced Studies, Piazza S. Ponziano, 6, 55100, Lucca, Italy*

ARTICLE INFO

*Keywords:*
Process Algebra
Secure development
Attack modeling
Vulnerability Assessment

ABSTRACT

Process algebras (PAs) provide the mathematical foundation for several verification techniques and have profoundly influenced many areas of computer science. One of the main reasons for their success is their compact yet expressive and flexible syntax, which allows for the modeling of the relevant aspects of computation while abstracting away the irrelevant ones. Cybersecurity is no exception, and most authors acknowledge the importance of PAs in this field. However, estimating the impact of PAs is not trivial.

In this survey, we consider lines of research that employ PAs to address security problems. Our systematization of knowledge aims to assess and measure the impact of PAs. To achieve this goal, we start by briefly reviewing the evolution of PAs. Then, we analyze the literature by mapping each contribution to three cybersecurity sub-fields: *secure development*, *attack modeling*, and *vulnerability assessment*. Our methodology follows the chronological development of process algebras and identifies the emerging features specifically introduced for dealing with security problems. Although our analysis confirms that PAs have been greatly influential in general, it provides a fine-grained understanding of how PAs have shaped research in cybersecurity. Interestingly, our work highlights that some application areas remain underexplored, thus providing the research community with valuable insights on future directions.

**Contents**



## 1. Introduction

Process Algebras (PAs) are formal languages commonly used to model the behavior of a computational agent. Roughly speaking, PAs put most of the emphasis on the control flow structure of the modeled agent, rather than on its data flow. This is similar, for instance, to the approach followed by finite state automata [1], context-free

* Corresponding author.
*E-mail addresses:* gabriele.costa@imtlucca.it (G. Costa), silvia.defrancisci@imtlucca.it (S. De Francisci), rocco.denicola@imtlucca.it (R. De Nicola).

https://doi.org/10.1016/j.cosrev.2025.100758
Received 12 July 2024; Received in revised form 25 March 2025; Accepted 29 March 2025
Available online 23 April 2025

grammars [2], and the $\lambda$-calculus [3], where a computation amounts to a sequence of transitions between configurations. Vice versa, this perspective opposes that of other formal models of computation, such as Turing Machines, which put more emphasis on data processing.

In most PAs, computational steps also emit *observable* actions.[1] This allows one to elegantly introduce the role of an external observer, i.e., someone who aims to understand an agent's behavior but has no control over its internal structure. Under reasonable assumptions, an attacker may be represented as an external observer. As a matter of fact, an attacker is generally represented by its *capabilities* and *goals*. In terms of capabilities, the ability to partially interact with the agent's I/O mechanism is a typical setting (e.g., think of information flow [4]). In terms of goals, the attacker's aim can be modeled as a target state, e.g., denoting a failure of the computation she wants to reach. Formal models of both the attacker and the system are the fundamental building blocks of most automated, formal reasoning techniques. Thus, it is not surprising that PAs are often adopted in this field.

Although different PAs may significantly vary in terms of syntax and semantics, all of them share a few common elements, such as a succinct, yet expressive syntax. A cornerstone of any PAs is a set of actions, sometimes called *alphabet*, *emitted/accepted* by the computational steps. The alphabet can be bipartite to distinguish between observable, aka *external*, and non-observable, aka *internal*, actions. The possibility for an observer to discriminate between two agents is modeled by means of some equivalence relationship $\equiv$. The kind of equivalence depends very heavily on how the systems under consideration will be used. In spite of the general agreement on taking an extensional approach to the semantics of systems, there are different views about what "reasonable" observations are and how their outcomes can be used to distinguish or identify systems. This is mainly due to the large number of properties which may be relevant in the analysis of such systems. For instance, for any pair of agents $P$ and $Q$ such that $P \equiv Q$, an attacker cannot distinguish between them as far as her observational capabilities are properly modeled by $\equiv$. In other terms, $\equiv$-attacks that do not success against $P$ would also fail against $Q$.

The main notions of equivalence in system behavior introduced in the literature consider two systems as indistinguishable if they:

1. perform the same sequences of actions,
2. perform the same sequences of actions and, after each sequence, are ready to emit/accept the same sets of actions
3. perform the same sequences of actions and, after each sequence, exhibit, recursively, the same behavior.

These approaches give rise to three main equivalences respectively *trace equivalence* $=_T$, *testing equivalence* $\simeq$ and *bisimulation equivalence* $\sim$. All of them can be either *strong* or *weak*, depending on whether they consider or neglect, respectively, internal actions. As a consequence, strong equivalences imply their weak counterparts, but not vice versa. We propose the following classical examples to highlight the differences among the three equivalences discussed above. For a gentle introduction to these notions, we refer the reader to [5].

**Example 1.** Consider the three systems $P, Q$ and $R$ depicted in Fig. 1, where $a, b, c$ and $d$ are observable actions. These systems are represented as *labeled transition systems* (LTSs). Briefly, LTSs are a model of computation where agents are represented through states and labeled transitions, i.e., arrows denoting computational steps from a source state to a target one that fire an observable action as a side effect. LTS traces are then the sequences of actions fired during the computations of the agent. Intuitively, if $P, Q$, and $R$ have the same trace set ($T = \{a, ab, abd, abc\}$), then $P =_T Q =_T R$. However, we cannot consider them equivalent from a behavioral point of view. As a matter of fact, after choosing $b$, an agent can pick between $c$ and $d$ on system $P$ but not on system $Q$. On the other hand, $P \nsim Q \nsim R$, but from an external observer, $Q$ and $R$ appear to be acting in the same way. In fact, in the $Q$ system, choice $b$ determines action $c$ or $d$, while the second one is the $a$'s choice, but an observer cannot understand the difference between them; she cannot know what is driving if $a$ or $b$. The result is the concept of *testing equivalent*, for which $P \not\simeq Q$ and $P \not\simeq R$ but $Q \simeq R$.

The features described above make PAs particularly appropriate for reasoning about the perspective of an attacker who can (partially) observe the system. Not surprisingly, there exists a vast literature proposing PAs to deal with security-related problems. However (and perhaps consequently), determining the mutual dependency between PAs and cybersecurity, including its subfields, is difficult.

*Related work.* To the best of our knowledge, no survey that systematically examined how PAs have shaped cybersecurity practices or how much they are adopted in the field was ever published. Nevertheless, other authors revised the history of PAs and the relevant application domains. For instance, Baeten [6] surveys PAs in general, he summarizes the history of CCS, CSP, and ACP and he presents the developments of time and stochastic features. More recently, Brookes and Roscoe [7] approach from a historical point of view to CSP and, in particular, to the FDR tool. Aldini et al. [8] survey the application of PAs to software architecture, emphasizing on component-oriented modeling. Beek et al. [9] compare formal methods used on web service composition considering three features: connectivity, correctness, and quality of service. In the same domain, Eddine [10] compares the different approaches, among which are PAs, to design and implement web services focusing on choreography and orchestration. Tuan Anh et al. [11] look into the issues surrounding the security and privacy of the Internet of Mobile Things utilizing PAs, with a focus on the mobile PAs $\pi$-calculus. Wan et al. [12] survey composition mechanisms and models for cyber–physical systems (CPS). Interestingly, our results show the recent evolution Wan et al. [12] anticipated by claiming that "CPS development must be supported from the design phase by process algebras to achieve strong results on correctness, performance, cost and efficiency". Widel et al. [13] investigate the different generation and analysis approaches for Attack Trees. PAs are among the formal methods they consider, and UPPAAL, along with its variants, is assessed as one of the potential analysis methods.

Although not classified as surveys, other works propose an overview of PAs and their employment in protocol specification and verification. For instance, Ryan et al. [14] model and analyze protocols and properties through CSP, using FDR and Casper tools. Similarly, Ryan and Smyth [15], give an overview of the applied $\pi$-calculus, its application areas, and how to use it to model protocols and properties.

*Goal of this survey.* Our goal is to shed light on the deep relationship between PAs and cybersecurity. In order to achieve this goal, we reviewed a corpus of research papers using PAs to solve some security-related problems. This includes general-purpose PAs, applied to model security aspects, as well as security-specific PAs, i.e., algebras embedding security features in their syntax or semantics. Hence, we put forward a classification based on the following three crucial, macro-areas of cybersecurity.

*Secure development.* As discussed above, PAs are particularly suited for reasoning about the *correct* behavior of a system. Following the *Security-by-Design* principle [16], correctness checks allow to avoid common implementation mistakes, thus preventing security flaws. Consequently, many authors have proposed PA-based methodologies to drive system development during one or more phases of its life cycle. In this category, we explore the relationship between PA-based methodologies and the phases of a generic development life cycle (DLC).

[1] The relationship between computational steps and actions may vary depending on the specific application scenario of each PA. For instance, when modeling I/O mechanisms, an input can be modeled as an action accepted by the agent, while an output would be an emitted action.

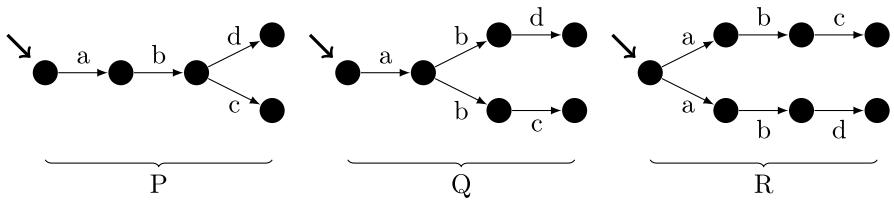


**Fig. 1.** An example of three labeled transition systems. P, Q, and R are trace equivalent; Q and R are also testing equivalent; however, none are bisimilar.

*Attack modeling.* Characterizing the attacker, for example, in terms of her motives and capabilities, is a crucial aspect of cybersecurity. Using PAs, attackers' strategies can be modeled as processes, enabling the systematic exploration of their malicious interaction with the system. Furthermore, PAs support in reasoning about the capabilities and limitations of an external observer, have lead many researchers to use them for modeling threats. While modeling the general behavior of a real adversary might be too complex, most attack modeling frameworks decompose this behavior into a few categories of threats. For this reason, we consider the relationship between existing works in the literature and the categories of a major attack modeling framework, specifically STRIDE [17].

*Vulnerability assessment.* Disclosing and fixing vulnerabilities are among the most effective and common methods to enhance system security, serving as cornerstones of activities like penetration testing. A vulnerability is disclosed only when concrete evidence, typically in the form of a proof-of-concept (PoC) exploit, is provided. Formal verification techniques, such as those based on PAs, have the capability to produce actual examples, which can often be translated into a working PoC. Consequently, many researchers have proposed PA-based techniques for disclosing vulnerabilities and automatically generating PoCs. We provide a characterization of existing works in the literature across several major technological domains.

The main reason behind these three areas is that they all have to do with the analysis of the behavior of some agents: a system under development (behavior it must have), an expected adversary (behavior she could have), and an existing system (behavior it has, but shouldn't have), respectively. As stated above, the core features of PAs make them particularly suitable for this kind of application.

Wrapping up, our systematization of knowledge aims at addressing the following *research questions*.

- Q1 How did PA theory and cybersecurity practice influence each other's development?
- Q2 How did PA-related contributions map into the three macro-areas discussed above?
- Q3 Are PA uniformly adopted for studying cybersecurity in the various application domains?

Answering these questions may help researchers to better understand the application conditions making PA suitable for dealing with security-related problems. Also, our analysis has the potential to identify research fields of cybersecurity where PA-based approaches are still unexplored.

*The main contributions of the paper* are the following.

- A brief history of the evolution of PAs and the genealogy of the PAs cited in this survey along with their distinguishing features (Section 3).
- A focus of the use of PAs in cybersecurity (Section 4) and classification of PAs according to the three macro-areas of Secure development (Section 4.1), Attack modeling (Section 4.2) and Vulnerability assessment (Section 4.3).
- A final discussion including answers to our research questions where gaps emerge in the use of PAs in specific aspects of each of the 3 macro-areas (Section 5). The areas where outlier values in the adoption of PAs were observed are as follows:
  - Planning and maintenance within the DLC on one side, compared to design, development, and testing on the other.
  - Spoofing and repudiation attacks.
  - Network in contrast to software and hardware vulnerability assessment.

*The rest of the paper is organized as follows.* Section 2 explains the research methodology and threats to validity. In Section 3 we provide some background and historical information. Section 4 presents the results of the study, and Section 5 a discussion of the results. Finally, Section 6 concludes the paper.

## 2. Methodology

In this section, we describe the research methodology used to conduct our work. We mainly follow the Systematic Mapping Study (SMS) [18], a structured methodology used to categorize and analyze existing research within a specific domain. Unlike systematic literature reviews, which focus on evaluating the quality of studies, SMS aims to provide a broad overview of trends, classifications, and research gaps [18]. This study focuses on the application of PA in cybersecurity, identifying key areas where it has been utilized and highlighting future research opportunities. An SMS protocol is made up of three main phases *planning*, *conducting* and *reporting*, as shown in Fig. 2. In the following subsections, we describe each phase in more detail as well as a critical evaluation of our methodology (Section 2.4).

### 2.1. Planning

The goal of this phase is to identify the research scope of the work and provide a plan to conduct the research. As shown in Fig. 2, the first step was *Formulate Research Questions*. Since our aim was to explore how PA and cybersecurity practices have influenced each other over time, Q1 naturally arose. For the second research question, we were inspired by the nature and history of PAs themselves. PAs are born for formalizing systems and agent behaviors, so we focused on the behavior that a system *should*, *could* and *should not* have. These distinctions led us to identify the three macro-areas discussed in the introduction, i.e., Secure Development, Attack Modeling and Vulnerability Assessment, and addressed by Q2. Finally, to highlight trends and research gaps, we examined the adoption of PA across various domains with cybersecurity concerns, as explored in Q3. To structure this investigation, we selected relevant databases and keywords during the *Define search strategy* step. We used the following digital libraries: ACM Digital Library,[2] IEEE Explore,[3] Science Direct,[4] Springer Link[5] and Scopus.[6]

Based on our research questions, we defined two sets of keywords, one related to cybersecurity and the other to PA. We selected terms relevant to the topic, along with their synonyms, resulting in the following search query:

[2] https://dl.acm.org.
[3] https://ieeexplore.ieee.org.
[4] https://www.sciencedirect.com.
[5] https://link.springer.com.
[6] https://www.scopus.com.

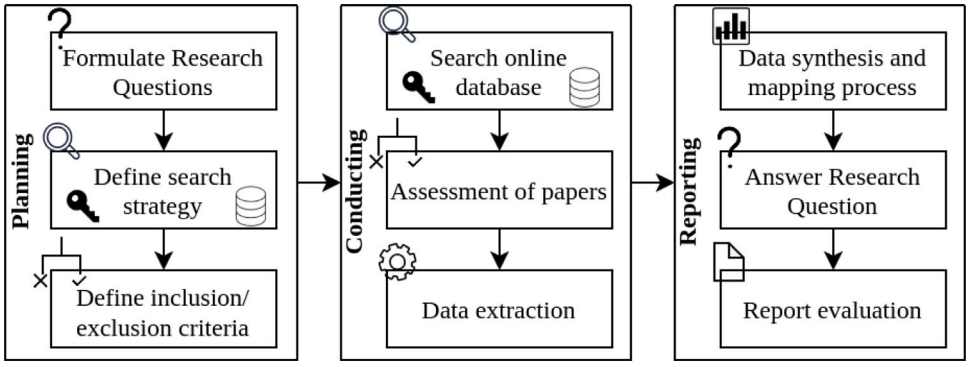


**Fig. 2.** Flowchart of our SMS methodology.

`(*secur* OR attack* OR threat* OR risk* OR vulnerabilit* OR weakness)`

`AND (process algebra OR process calculus OR formal method)`

Where '*' represents any string, e.g., `*secur*` encompasses cybersecurity, security and secure. Finally, to refine the result set and support the screening process, we defined the following *exclusion criteria*:

- Studies that do not explicitly discuss PA in a cybersecurity context.
- Studies focusing on general cybersecurity methods without using PAs.
- Non-peer-reviewed and non-English studies as well as duplicates from different databases.

### 2.2. Conducting

During this phase, we applied our planning to select studies and relevant data from them to facilitate classification and analysis. In the *Search online database* step, we applied the defined search query to the selected databases. Thus, we filtered the search results by applying the exclusion criteria to remove irrelevant or duplicate studies (*Assessment of papers*), obtaining 132 papers. Finally, from the selected studies, we *extracted key data*, including:

- Used PA: CCS, CSP, $\pi$-calculus, etc.
- Main purpose for using PA: verification, modeling, testing, etc.
- Application domain: software, hardware, CPS, networks, etc.
- Security perspective: attacker, defender, developer, etc.

### 2.3. Reporting

This step ensures that each article is systematically classified on the basis of its contributions to the field of cybersecurity and PA. In the *Data synthesis and mapping process* step, we used the information extracted to classify studies into categories. We categorized the extracted data into three macro-areas: secure development, attack modeling, and vulnerability assessment. For secure development and vulnerability assessment, we analyzed the application domain, the methodology used (verification, modeling, testing, and whether a tool was employed), and the specific PA applied. For attack modeling, which is generally independent of a specific application domain, we focused on the type of attack (following the STRIDE model), the methodology used, and the type of PA applied. After categorizing the selected studies, we analyzed the data to identify trends, gaps, and research patterns. Thus allowing us to *Answer research questions* in the second step. In the final step *Report evaluation* of our study, we reported key insights derived from the mapping process and outlined directions for future research. We used tables and graphs for better data visualization and reported the strengths and limitations of current research efforts as well as cybersecurity domains where PAs can be applied.

### 2.4. Threats to validity

Our SMS is subject to several potential threats to validity. One major concern is *selection bias*, as the choice of digital libraries that offer an adequate search engine and a wide variety of publications may still exclude relevant studies indexed in other sources. Additionally, while the search query includes multiple synonyms and Boolean operators, it may still fail to capture all relevant publications. For instance, some studies might not explicitly mention 'process algebra', 'process calculus' or 'formal method'. Another issue is *publication bias*, as prioritizing peer-reviewed sources may exclude valuable insights from technical reports and preprints. Furthermore, the exclusion of non-English studies could result in a lack of geographical diversity in the selected works. There is also a risk of *classification bias* since categorizing studies into the three macro-areas (secure development, attack modeling, and vulnerability assessment) involves subjective judgment. The classification of cybersecurity challenges depends on varying terminologies and perspectives across different research communities. *Internal validity* could be affected by potential human errors in the screening and filtering process, despite the application of predefined inclusion and exclusion criteria. Lastly, *external validity* is limited by the time-dependent nature of the findings, as new research may emerge that alters the observed trends.

To mitigate these threats, we adhered to a structured study selection protocol, implemented a systematic screening process, and performed iterative cross-checks among researchers. Future research could expand database coverage, refine classification methodologies, and incorporate expert validation to further enhance reliability.

## 3. A brief history of process algebras

In this section, we briefly survey the history of PAs and their evolution over time. For a more detailed discussion about the origin and history of PAs, we refer the interested reader to [6,19]. The genesis of PAs goes back to the 80s, when a few authors independently proposed different calculi for the specification of processes. Fig. 3 schematically depicts the genealogy of PAs considered in this survey, starting from Milner's Calculus of Communicating Systems (CCS) [20], Brookes, Hoare, and Roscoe's Communicating Sequential Processes (CSP) [21], and Bergstra's Algebra of Communicating Processes (ACP) [22], which inspired a considerable number of other process calculi. During their early days, original PAs' syntax and semantics went through a refinement process. For instance, CCS was originally formulated with

*observational* semantics and equivalence. Subsequently, after the conceptualization of bisimulation by David Park [23] and the introduction of the notion of *process testing* by De Nicola and Hennessy [24], these concepts were also proposed for CCS [24,25].

Similarly, the original CSP [26] evolved into a more abstract formalism to introduce the so-called Theoretical CSP (TCSP), which was later simply referred to as CSP, although the new formalism differed significantly from the original CSP. Brookes, Hoare, and Roscoe defined process equivalence in terms of *failures*. In their words "two processes that fail in exactly the same circumstances are indistinguishable by external observation" [21].

ACP was the first model of concurrency to include the word "Algebra" in the name. Its semantics was, in fact, expressed in terms of algebraic laws. Bergstra and Klop proposed many variants of ACP, including a basic, well-known variant, called BPA [27]. Notably, BPA included neither operators for process *synchronization* nor *communication*, making it more primitive than ACP. Synchronization devices, however, soon started receiving major attention and became the main distinguished features of PAs. In this context, synchronization generically refers to the coordinated or simultaneous execution of actions by multiple processes. It allows processes to perform computations that depend on a shared state, implemented through the controlled exchange of signals. Due to the absence of a synchronization operator, BPA is not explicitly listed in Fig. 3. Rather, it is considered among the common ancestors (root node "PA") of each PA. On the other hand, ACP, which was also later reconsidered and augmented with *bisimulation* and *branching bisimulation* [28], appears in Fig. 3.

In addition to semantics, the syntax also varies among these algebras; the main differences lie in the choice and parallel composition operators. CSP has two distinct operators for internal and external choice, while CCS and ACP have only one choice operator. Parallel composition is a two-party synchronization operator in CCS, a multiparty synchronization operator in CSP, and in ACP, there are three operators for parallel composition, one of which is a generalization of the CSP operator. The other two serve only to impose a specific step as the first action, i.e., first communication or first left process action.

Following these original proposals of PAs, a plethora of variants have been proposed in the literature. In Fig. 3, we show the cladogram for some of them. Intuitively, a PA descends from another one if it shares some common elements with its ancestor (being the abstract notion of PA — top, circular node — the universal one). Dashed arrows are used to indicate whether a PA inherits from more than one ancestor. We consider some, more recent PAs, as independent proposals, i.e., they do not derive from any pre-existing one. This is the case, for instance, of the Calculus of Wireless Systems (CWS) [56]. The reason is that the syntax and semantics of these proposals are not inspired by any other specific PA, but, rather, they are developed from scratch.

As stated above, the common elements that we are considering here include PA syntax, semantics, and equivalence relationships. Typically, derived PAs introduce some new elements w.r.t. their archetypes. These elements often aim to model specific aspects of computation. In order to better highlight these extensions, in Fig. 3 we report the paper that originally put forward a certain PA and we label each PA with icons denoting their distinguishing features. We list them below.

- *Stochastic PAs* are meant to study the behavior of a system in probabilistic terms, e.g., by introducing probabilistic equivalences.
- *Timed PAs* permit to describe the behavior of systems under real-time settings.
- *Imperative PAs* embed assignment operators to update data in memory during the execution, as real computer programs typically do [57].
- *Wireless networks PAs* combine features of broadcast, synchronous, and asynchronous communications.
- *Cyber–physical systems PAs* model hybrid, physical–digital systems and are used for reasoning about their properties.
- *Quantum computing PAs* have been proposed for modeling quantum computing enabled agents.
- *Security PAs* are used for modeling security properties, threats, and other security-critical aspects of computation.

Arguably, the main reason behind this diversification is the need to consider specific aspects of the computation of distributed agents. In this landscape, security is no exception and several security-related PAs (🔒) emerged. However, the main difference w.r.t. other domain-specific PAs is that security does not refer to some peculiar aspect of the computation. In general, security pertains to all potential issues that could arise during the execution of a process. These specific behaviors can be modeled using ad-hoc secure PAs or even general-purpose ones. As a consequence, PAs used in cybersecurity (either specific or general purpose) require particular attention for understanding which security concerns they address, as elaborated below.

## 4. Process algebras for security

As stated above, security is a multifaceted issue. It is common knowledge that there is no "silver bullet" for security and that several security tasks must be carried out to improve the robustness of a system, e.g., as in Security-by-Design [16]. Some security tasks are of particular interest for PAs, while others are less relevant. For instance, PAs are suitable for supporting the verification of correctness during the early specification and design phases. On the other hand, PAs may be less useful in the latest phases, e.g., for the secure disposal of a product. Following this line of reasoning, here we put forward a classification based on the following three areas.

- Secure development, i.e., PAs supporting the secure design, implementation, and execution of a system.
- Attack modeling, i.e., PAs used for modeling and analyzing the behavior of an attacker and her strategies.
- Vulnerability assessment, i.e., methods employing PAs for spotting out actual flaws in existing systems.

Needless to say, precisely measuring the impact of a certain PA in these areas is extremely hard or even impossible. However, a rough estimation can be obtained by considering the scientific literature. In particular, we propose the following scale that aims at estimating the impact of a PA in terms of research works adopting it in the three security areas described above.

- 0 No literature exists applying PAs in the area.
- $\frac{1}{2}$ Some papers exist, but their authors belong to a single clique.[7]
- 1 Some papers exist and their authors belong to two or more cliques.

Fig. 4 schematically depicts some PAs discussed in this section and their impact profile according to the previous metrics. From left to right, the columns are for CCS, $\pi$-calculus, CSP, ACP and CWS. For each of them we report their spider chart, as well as that of two among their most influential descendants.

In the following sections we describe each of these areas in more details.

[7] Cliques are computed by considering the co-authoring relation induced by the literature considered in this paper.

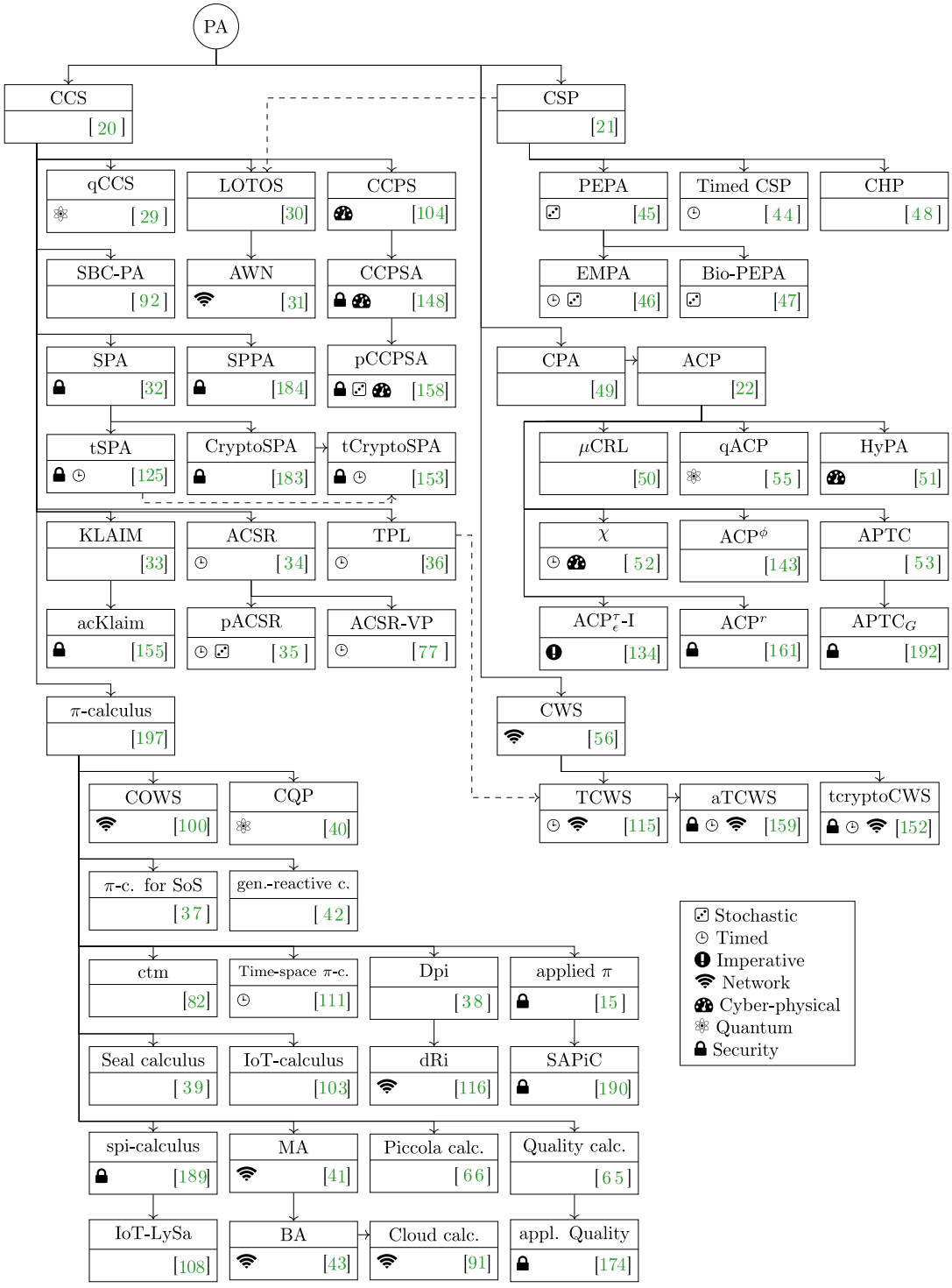


**Fig. 3.** Process algebras family tree: each PA includes its name (above), reference (right), and features (left) (see [29–55]).

### 4.1. Process algebras for secure development

In the last decades, Security-by-Design [16] has received more and more attention as the standard approach for developing secure systems. A cornerstone of Security-by-Design is that security should be considered from the very early stages of the design process. In this respect, thanks to their abstract and compact syntax, PAs have often been proposed as a design formalism. Also, their formal semantics permit to carry out verification procedures which are not natively supported by other design languages, e.g., UML [58] and BPMN [59]. Often, formal verification occurs via model checking [60]. A model checker compares the behavior of the system against a specification, e.g., expressed as a logical formula. Eventually, the model checker either verifies the system or returns a counterexample, i.e., an evidence showing that the current design violates the specification. When the model checked property encodes a security policy, the counterexample can even be interpreted as a proof-of-concept attack, i.e., instructions for an attacker to violate the policy.

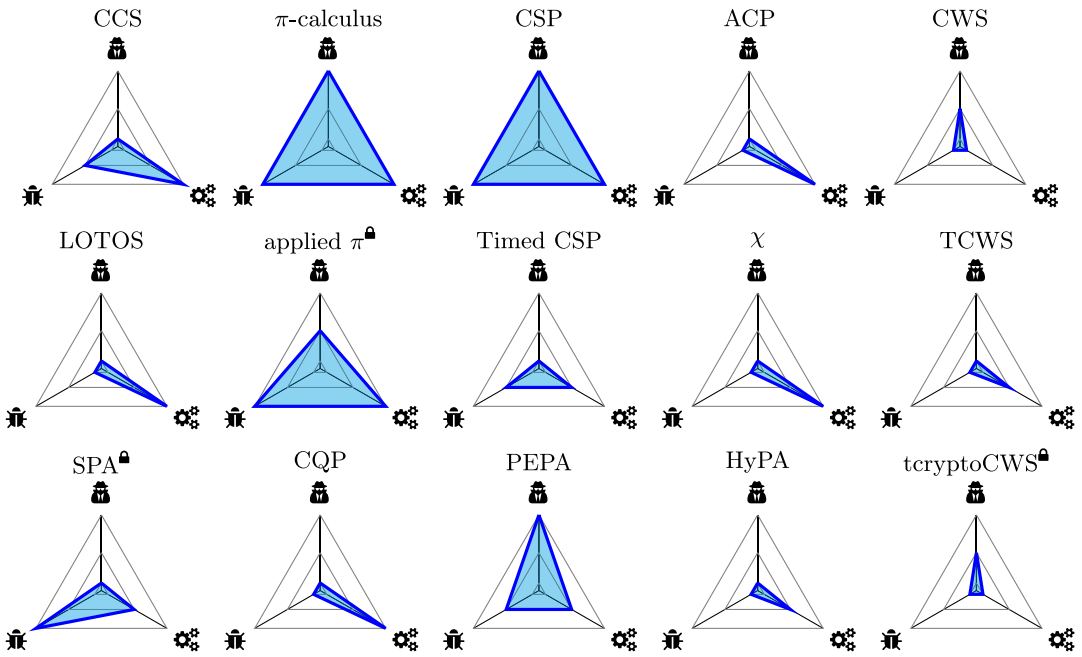


**Fig. 4.** The PA employment for Secure Development ( ), Attack Modeling ( ), and Vulnerability Assessment ( ). Secure PA are marked with .

**Table 1**
Security (top) and general-purpose (bottom) PA-based verification tools.

| Tool | Specification languages | | Tr. | Eqs. | | | C.E. |
|---|---|---|---|---|---|---|---|
| | System behavior | Properties | | $=_T$ | $\simeq$ | $\sim$ | |
| AKISS | Applied $\pi$-calculus | Built-in | | ✓ | | | ✓ |
| CoPS | SPA | Built-in | | | | ✓$^b$ | |
| CoSeC | SPA | Built-in | | ✓ | | ✓$^b$ | ✓ |
| ProVerif | Applied $\pi$-calculus | Built-in | ✓ | | | ✓$^{sb}$ | ✓ |
| StatVerif | Applied $\pi$-calculus | Built-in | ✓ | | | ✓$^{sb}$ | ✓ |
| Tamarin | SAPiC | Many-sorted FOL | ✓ | | | ✓$^{sb}$ | ✓ |
| CADP | LOTOS | $\mu$-calculus | ✓ | ✓ | | ✓ | ✓ |
| CWB-NC | CCS, CSP, LOTOS, TCCS | $\mu$-calculus | ✓ | ✓ | ✓ | ✓$^b$ | ✓ |
| Exp.Open 2.0 | CCS, CSP, LOTOS, E-LOTOS | $\mu$-calculus | | ✓ | | ✓ | ✓ |
| FDR | CSP | $\text{CSP}_M$ | ✓ | ✓ | ✓ | | ✓ |
| mCRL2 | ACP | $\mu$-calculus | ✓ | ✓ | | ✓$^w$ | ✓ |
| MWB | $\pi$-calculus | $\mu$-calculus | ✓ | | | ✓$^b$ | ✓ |
| PAT | CSP, Timed CSP | LTL | ✓ | ✓ | ✓ | | ✓ |

*b/w/s*: branching/weak/strong relation unsupported.

Table 1 lists the PA-based verification tools appearing in the articles considered in this survey.[8] Tools are grouped according to their scope. In particular, the first group consists of tools specifically proposed for the verification of security properties, while tools in the second group are general-purpose model checkers (which can be used for the verification of security properties). Below we explain the table columns in detail and we briefly comment on their content.

**System specification language**. This column reports the formal languages used by a tool for specifying the behavior of the target system (System).

As one might expect, security model checkers often rely on a single security PA such as SPA or the applied $\pi$-calculus. On the contrary, general purpose model checkers often support multiple PAs. This may be achieved by translating them into a ground formalism, e.g., LTS or another PA.

**Properties specification language.** This language is used to specify the properties that a system should comply with. In the case of security properties, these rules often encode the attacker model in terms of goals and capabilities. As a consequence, many security model checkers rely on built-in security policies encoding some relevant properties, e.g., confidentiality or integrity, under a specific attacker model, e.g., Dolev–Yao [63]. Conversely, general purpose model checkers resort to highly expressive temporal logics, e.g., the $\mu$-calculus, as the properties to be verified are arbitrary.

**Trace Properties (Tr.)**. Trace properties verification amounts to checking whether one or more sequences of events, i.e., execution traces, belong to the semantics of a target agent. For instance, trace verification is used by tools to check whether the agent can reach undesired configurations, e.g., deadlock. Since trace semantics provides a coarse-grained definition of the agent behavior, it is not surprising that most tools support it.

**Trace Equivalence** ($=_T$). In trace equivalence, systems are compared according to the sequences of actions, aka traces, they perform during a computation. Two systems are trace equivalent if they generate the exact same set of traces. Trace equivalence is a coarse-grained form of observational equivalence. As a matter of fact, it subsumes that, apart from the generated traces, the attacker does not consider other observable aspects of the computation (see below).

**Testing Equivalence** ($\simeq$). Two systems are testing equivalent if they are both *must* and *may* equivalent. *Must* equivalence means that they

[8] We intentionally omit some model checkers that are mainly devoted to education, e.g., CAAL [61] and TAPAs [62].

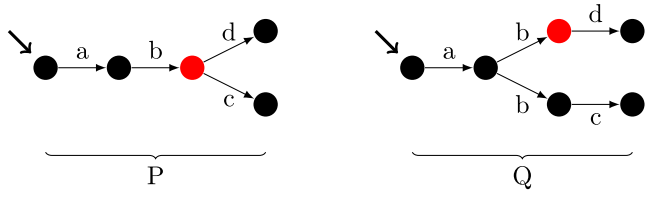


**Fig. 5.** Two processes, $P$ and $Q$, that are *may* equivalent, but not *must* equivalent.

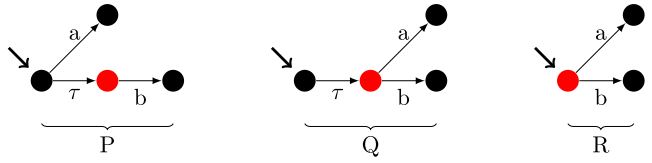


**Fig. 6.** Three agents satisfying different types of bisimulations. P, Q, and R are weak bisimilar; Q and R are also branching bisimilar; however, none are strong bisimilar.

have to satisfy the same observations, as explained in Example 2. *May* equivalence implies that the two systems can satisfy the same observations. An alternative characterization of *may* is the *trace equivalence*, while *must* corresponds to *failure*, i.e., two systems are *failure* equivalent if they always fail under the same conditions.

Clearly, w.r.t. $=_T$, $\simeq$ introduces a further capability for the attacker, i.e., distinguishing between the final, possibly failing, states of agents.

**Example 2.** Consider again the agents of Fig. 1 reported in Fig. 5. The two systems $P$ and $Q$ are not *must* equivalent. As a matter of fact, both $P$ and $Q$ reach the red states with trace $ab$. However, from the red state $Q$ has to fire action $d$, while $P$ can also fire action $c$.

**Bisimulation** ($\sim$). Two systems are *bisimilar* if their initial computational states are bisimilar. Two states are bisimilar if $(i)$ they are the source of transitions labeled with the same actions and $(ii)$ performing a transition with the same label always leads to bisimilar states. Bisimulation can be *weak*, *strong*, and *branching*. Roughly, in weak bisimulation, one can neglect internal action $\tau$, while strong bisimulation also considers them. Branching bisimulation lays on a middle ground. Indeed, it neglects $\tau$ actions, except when they allow to discriminate between two alternative branches of the agent computation. To clarify, we propose the following example.

**Example 3.** Let us consider $P$, $Q$, and $R$ of Fig. 6. All three have the same trace set ($T = \{a, b\}$). However, the internal action $\tau$ affects their bisimulation equivalence. $P$ and $Q$ are weakly, but not branching (and thus strongly) bisimilar. As a matter of fact, one can observe that, after the $\tau$ transition, $a$ is still possible in $Q$, but not in $P$. On the other hand, $R$ and $Q$ are branching (and thus weakly) bisimilar, since after the $\tau$ transition in $Q$, both $a$ and $b$ remain possible, just as in the initial state of $R$. Nevertheless, strong bisimulation does not hold, since $R$ has no initial $\tau$ transition.

As for properties, equivalence relationships are instrumental to the specific purposes of the model checker. As a consequence, security-oriented model checkers tend to support a limited number of equivalences, while general-purpose tools support more. A further factor influencing the set of supported relationships is that some relationships were originally proposed for a specific PA. For instance, *failure* semantic, a variant of *testing* ones, were originally studied on CSP, and they are supported by PAT and FDR.

**Counterexample (C.E.).** When a specification is not satisfied, a model checker typically returns a *counterexample*, i.e., a system trace that violates the specification. The availability of counterexamples is shown in the last column of Table 1. Counterexamples are useful for several, post-analysis tasks. Often, security model checkers convert the counterexample to an attack trace, e.g., useful for vulnerability testing and training. Similarly, general-purpose model checkers can exploit counterexamples, e.g., for automatic testing procedures or refinement processes. Hence, it is not surprising that most model checkers return counterexamples from both groups.

Since they are the basis for the definition of any Security-by-Design methodology, *Development Life Cycles* (DLC) are also very relevant. Although there exist several alternative DLC, all of them include the following, few common macro-phases.

- *Planning*, i.e., the initial conceptualization of the system and its requirements.
- *Design*, i.e., the architectural modeling of the system and its components.
- *Implementation*, i.e., the actual development of the system.
- *Testing*, i.e., for validating the implementation against the expected requirements.
- *Maintenance*, i.e., for monitoring, updating, and eventually disposing the system.

Table 2 schematically reports the adoption of PAs in DLC w.r.t. some major application domains. For instance, the first group refers to the software development life cycle (SDLC). The second column displays the LC phases coverage, where colored segments are those where the proposed methodology applies. Thus, for example, the second entry under SDLC, shows techniques that apply to design, implementation, and testing phases. Then, under the third column, we list the employed PAs. Also, column Tool reports whether there exists at least one implementation among the listed methodologies. Finally, papers references are given in the last column.

A few observations arise from Table 2. We briefly discuss them below.

**No usage for planning.** No author proposes PAs for the earliest stage of the development process. Reasonably, this happens because during this phase, there is no information about the modules that will constitute the system to be implemented.

**Design, implementation, and testing.** Many authors propose approaches employing PAs during design, implementation, and testing. This is somehow expected since PAs are particularly suitable for modeling a system and its components. Moreover, their formal semantics provide the foundation for refinement methods, useful at implementation time for driving the development process from its initial specification. Also, at testing time, model checkers' counterexamples can be converted to test cases.

**Almost no maintenance.** Not surprisingly, PAs are scarcely used for maintenance, with a few, interesting exceptions mostly related to runtime enforcement. In particular, this topic is recurrent in the field of policy specification. The main reason is that PAs provide a theoretical background for policy enforcement. As a matter of fact, some authors [139–143] found it convenient to model policy monitors as agents that run in parallel with a target system. In this context, action authorization amounts to synchronous transitions between the two agents, i.e., the monitor and its target.

**Secure PA.** Interestingly, secure PAs are not commonly used for DLC. Since these PAs are formulated to model security threats, they put a strong emphasis on the attacker (see Sections 4.2 and 4.3). However, in DLC the main focus is on avoiding design and implementation flaws. Thus, the attacker's role is often marginal.

Summarizing, from Table 2, one can notice how PAs have been extensively adopted for enhancing the correctness of the development life cycles in many different contexts. Reasonably, we might expect a similar trend also for recently developing research fields, e.g., quantum computing.

**Table 2**
Process algebras usage in development life cycle. (see [64–137]).

| Domain | LC phases | PA | Tool | References |
|---|---|---|---|---|
| Software | | $\pi$-c., Quality c., Piccola c., c. for SoS | | [64–67] |
| | | ACP,CSP, LOTOS, $\pi$-c., Timed CSP | ✓ | [8,68–76] |
| | | ACSR-VP | | [77] |
| Hardware | | ACP, CCS, CHP | ✓ | [78–81] |
| Management systems | | PEPA, ctm | | [82–84] |
| | | ACP, CSP | ✓ | [85–87] |
| Cloud computing | | $\pi$-c., appl. $\pi$🔒, COWS | ✓ | [88–90] |
| | | Cloud c. | | [91] |
| Web services | | APTC, SBC-PA | | [92,93] |
| | | CCS, CSP, LOTOS, $\pi$-c., COWS | ✓ | [94–101] |
| | | appl. $\pi$🔒 | ✓ | [102] |
| Cyber–Physical systems | | IoT-c., $\chi$, CCPS, HyPA | | [103–106] |
| | | CSP, IoT-LySa, ACP, Time-Space $\pi$-c., $\chi$ | ✓ | [107–112] |
| | | TPL | | [113] |
| Network | | dRi, Seal c., TCWS | | [114–116] |
| | | appl. $\pi$🔒, ACP, CSP, AWN, pACSR | ✓ | [117–121] |
| | | appl. $\pi$🔒 | ✓ | [122] |
| Policy specification | | CSP, BA, appl. $\pi$🔒, $\pi$-c., tSPA🔒, EMPA | | [123–129] |
| | | SPA🔒, SAPiC🔒, CSP, ACP, ACP$^{\tau}_{\epsilon}$-I | ✓ | [14,130–134,138] |
| | | CSP | ✓ | [137] |
| | | SPA🔒 | | [54] |
| | | CSP, ACP$^{\phi}$, SPA🔒 | | [139–143] |
| Quantum | | CQP | ✓ | [135,136] |

🔒: Process Algebra embedding security features.

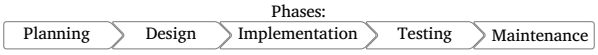


**Table 3**
Process algebras for attack modeling.

| Thrt | PA | Tool | References |
|---|---|---|---|
| S | CSP, SAPiC🔒 | FDR, Tamarin | [144,145] |
| T | CSP, CCPSA🔒, TPL, CWS, Quality calculus | FDR, UPPAAL | [146–150] |
| R | generative–reactive calculus | | [151] |
| I | CSP, PEPA, acKlaim🔒, tcryptoCWS🔒, tCryptoSPA🔒 | FDR | [14,152–155] |
| D | applied $\pi$🔒, PEPA, TPL, aTCWS🔒, pCCPSA🔒 | PEPA Plug-in | [150,156–160] |
| E | ACP$^{r}$🔒, $\pi$-calculus | MWB | [161–164] |

🔒 Process Algebra embedding security features.

### 4.2. Process algebras for attack modeling

Usually, the primary objective of attack modeling is to describe threat actors in terms of their capabilities and goals. *Attack modeling* is a complex, multifaceted task which may involve the description of the attacker from many different perspectives. Hence and not surprisingly, several alternative attack modeling frameworks have been put forward in the past. Here we follow the STRIDE [17] classification, which has a long-standing tradition in threat modeling. The acronym STRIDE stands for the categories of threats that an attacker may represent. We list them below.

- ***S**poofing*, i.e., identifying as another user.
- ***T**ampering*, i.e., malicious modification of data.
- ***R**epudiation*, i.e., deny execution of an action.
- ***I**nformation disclosure*, i.e., access to private information.
- ***D**enial of service*, i.e., deny service to valid users.
- ***E**levation of privilege*, i.e., appropriation of privileged access.

Table 3 lists the proposals that apply PAs to one of the categories presented above. By looking at Table 3, one can notice that security PAs are not common. This is somehow expected. As a matter of fact, security PAs embed their reference attacker model and they are not meant to be used for modeling new threats. On the other hand, some PAs are adequate for describing certain specific security menaces. For instance, DoS attacks aim at degrading the quality of a certain service. Probabilistic PAs such as PEPA can encode quality of service metrics, e.g., in terms of average client waiting time or server response rate. A similar argument applies to the measurement of the amount of leaked data, e.g., during information disclosure attacks, as well as other security metrics. Another example of the adequacy of certain PAs is the $\pi$-calculus, which is used for modeling privilege escalation attacks. A possible reason is that privilege escalation occurs when a certain agent interacts with a system under different contexts, i.e., the privileges that the system assigns to the agent. Context evolving over time, as a consequence of the escalation operations, is analogous to that of mobile agents where, for instance, a piece of software migrates between execution platforms. It is well known that the $\pi$-calculus has often been used for this purpose. Last, it is worth noticing that some areas appear to be relatively unexplored, as in the case of repudiation attacks.

### 4.3. Process algebras for vulnerability assessment

Vulnerabilities are the flip side of threats. Indeed, attackers search for vulnerabilities that they can exploit to pursue their goals. Hence, finding and fixing vulnerabilities is nowadays the most effective countermeasure for preventing attacks. Several tools help in disclosing vulnerabilities, e.g., by testing the target system with predefined payloads or some other heuristics. Nevertheless, these techniques are still

**Table 4**
Process algebras for vulnerability assessment.

| Domain | PA | Tool | References |
|---|---|---|---|
| Management systems | CSP | FDR | [169] |
| Cloud computing | Bio-PEPA | | [170] |
| Web services | $\pi$-calculus | | [171] |
| Cyber–Physical systems | SPA🔒 | CoPS | [172] |
| | CCPSA🔒 | UPPAAL SMC | [148] |
| | CSP | FDR | [173] |
| | Applied Quality calculus🔒 | | [174] |
| | IoT-LySa | | [175] |
| Network | Applied $\pi$-calculus🔒 | ProVerif, AKISS | [176–181] |
| | SPA🔒 | CoSeC | [182,183] |
| | SPPA🔒 | | [184] |
| | PEPA | | [185] |
| | Timed CSP | FDR | [186] |
| | CSP | FDR, PAT | [138,168,187,188] |
| | Spi-calculus🔒 | | [189] |
| | SAPiC🔒 | Tamarin | [190] |
| | CCS | CWB-NC | [191] |
| | APTC$_G$🔒 | | [192] |
| Policy specification | Applied $\pi$-calculus🔒 | ProVerif | [193] |
| | CSP | FDR | [194] |
| Quantum computing | qACP | | [195] |

🔒 Process Algebra embedding security features.

far from reaching the ability of expert penetration testers in finding new, zero-day vulnerabilities [165,166]. In this context, formal methods offer some advantages. As stated above, PAs permit to formalize security properties and check whether a system meets them, e.g., via model checking. When a property is violated, the counterexample generated by a model checker can be turned into a proof-of-concept exploit that can disclose the vulnerability in the real system. In the past, some prominent examples of zero-day vulnerabilities have been discovered in this way, e.g. [167,168]. Needless to say, this approach also has limitations. Two major ones are the high computational cost of model checking and the necessity of having accurate models, which might be difficult to obtain. Roughly, the first issue occurs since the model checking problem complexity grows exponentially w.r.t. the target model. Hence, real systems with many states and transitions may rapidly become intractable. The second issue has to do with approximations introduced in the model. For instance, approximations are necessary when the target system is only partially known or for reducing the model size (e.g., to avoid the exponential growth described above). However, as a consequence, approximations may lead to false results.

Table 4 shows applications of PAs for discovering vulnerabilities in some major technological domains. Each row reports the used PA and whether there exist tools associated with the proposed technique. From Table 4 we observe a few facts. First of all, network vulnerability assessment is the most studied field. The reason is that, under this entry, we also list works dealing with vulnerabilities in network protocols. This field is notoriously of great interest for the formal methods community, in general, and for the PAs community, in particular. Indeed, network protocols are compact, distributed algorithms with formal security goals. These features are a good match for most, general purpose PAs. Furthermore, several PAs were specifically proposed for the sake of assessing the vulnerabilities of security protocols, e.g., the Spi calculus [189]. On the other hand, some domains seem to be unexplored. This is the case, for instance, of the assessment of software vulnerabilities that, to the best of our knowledge, is never considered in the literature. Similarly, only a few authors rely on PAs for dealing with vulnerabilities of Web services and Cloud applications. The reason might be that software components, e.g., web services, are implemented with high-level programming languages such as Java or Python which may be complex to model using PAs. Languages like Rust and Go, with their process-based concurrency models and message-passing mechanisms, are more naturally aligned with PAs, making translation more feasible [196] and potentially reducing this gap. However, PAs are still not widely used due to their abstract nature and the difficulty of directly capturing real-world implementation details, such as memory management. Also hardware vulnerability detection using PAs seems to be unexplored. Again, a motivation behind this phenomenon could be the substantial gap between PAs and hardware specification languages.

## 5. Discussion

In this section we answer the questions raised in Section 1. For each answer, we also provide a brief discussion based on the results presented above.

*Q1. How did PAs theory and cybersecurity practice influence each other's development?* The recent developments in the theory of PAs and practice of cybersecurity *significantly* influenced each other.

On the one hand, many security problems are related to the verification that a certain system guarantees security properties, like confidentiality, integrity and authenticity. Formalizing this verification problem offers several advantages (see Section 4.1), yet it also requires considerable effort. As a result, many authors developed PAs with suitable syntax and semantics, making it easier to model and reason about security-related verification problems. This phenomenon is highlighted in Fig. 3, which shows how security-oriented PA (🔒) frequently stem. For instance, the applied pi-calculus [15] enriches the terms algebra of the $\pi$-calculus [197] for modeling security protocols. Furthermore, as depicted in Fig. 4, also general-purpose PAs were used for modeling and addressing some important security problems. For instance, CSP [21] was widely applied.

On the other hand, PA-based verification methods have also changed the approach of security professionals toward certain challenges. Notably, the adoption of formally verifiable specifications for security-critical network protocols is becoming more and more common. For instance, the specification of Transport Layer Security (TLS) version 1.3 [198], formal modeling and verification of the protocol were carried out using Tamarin [199].

Fig. 7 shows the time distribution of the papers considered in this survey. In particular, we group them according to the three macro-areas discussed above, i.e., Secure Development, Attack Modeling, and Vulnerability Assessment. The total number of papers per interval is represented by the circular markers. Overall, the number of publications tends to increase over the considered time span. Also, there are

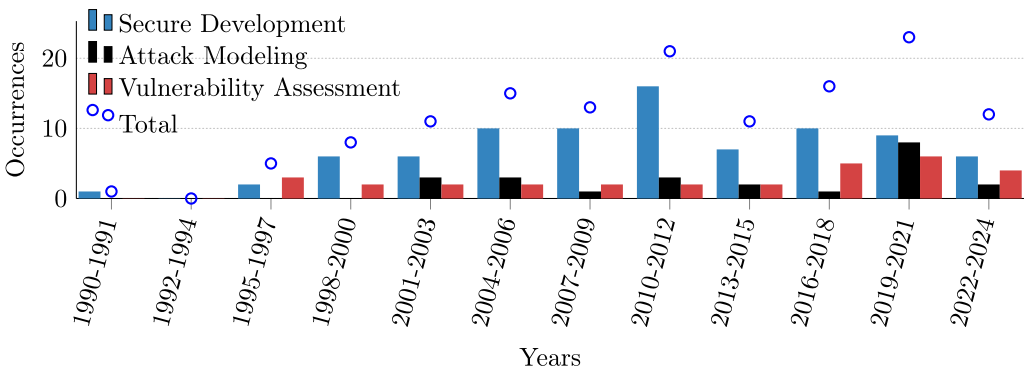


**Fig. 7.** Distribution of published papers over the years, classified into the three considered macro-areas.

some trends that emerge. One of them is that the number of papers on Secure Development had a significant growth from the early 2000s and a peak on 2010–2012, followed by a small decline and a period of stability. This trend resembles a Gartner Hype Cycle [200] and it might be the result of some triggering events that influenced the research community. For instance, some remarkable developments of new software verification techniques, e.g., think of model checking [60,201], occurred in those years. In contrast, Attack Modeling and Vulnerability Assessment have remained relatively stable.

*Q2. How did PA-related contributions map into the three macro-areas discussed in this paper?* According to the considered literature, most PA-based proposals focus on secure design and development. More precisely, among the papers listed in Tables 2, 3 and 4, 83 of them deal with secure design, 23 pertain to attack modeling and 30 are focused on vulnerability assessment. This also emerges from Fig. 4, where one can observe that, for each considered PA but CWS and tcryptoCWS, ⚙ > 0. However, the situation changes significantly when focusing solely on security PAs (🔒). As a matter of fact, security PAs are relatively predominant in vulnerability assessment and, even more, in attack modeling. Arguably, this trend may be attributed to the central role of the attacker in these contexts, while in secure design most of the emphasis is on the security specifications.

Since security PAs incorporate their own attacker models, analysts are relieved from the cumbersome and error-prone task of defining them manually. Needless to say, attacker models can vary significantly based on the specific security context. Therefore, we foresee the emergence of new PAs tailored to model evolving threats that will arise over time.

*Q3. Are PAs uniformly adopted for studying cybersecurity in the various application domains?* As discussed earlier, differences exist among the three macro-areas considered in this paper, and further heterogeneity emerges across various application domains. For instance, Table 4, illustrates that many authors use PA-based methodologies for assessing network vulnerabilities. This is probably due to the well-defined working assumptions (as discussed in Section 4.3) and attacker models (e.g., Dolev–Yao [63]). Conversely, none of the surveyed papers addresses hardware/software vulnerability assessment, although some proposals focus on hardware/software design (as shown in Table 2). One possible explanation could be the lack of efficient and accurate model extraction algorithms applicable to real-life, complex hardware/software systems. At present, PAs for hardware/software vulnerability assessment remain relatively unexplored.

## 6. Conclusion

In this paper, we revised the state of the art of PA-based approaches in cybersecurity. In particular, we focused on three security domains, i.e., secure development, attack modeling and vulnerability assessment. For each domain, we discussed the evolution and the impact of PA. Our work leads to a better understanding of the past and present trends in this research field and highlights areas where further research could drive significant advancements. In particular, some key findings have been identified in our work. Although PAs and cybersecurity have mutually influenced each other's development, there are still areas where the usage of PAs still needs to be explored. Also, the emerging security threats stimulated the evolution of PAs, leading researchers to develop algebras with features tailored to address specific security challenges. Not surprisingly, we observed that security-specific PAs are relatively predominant in vulnerability assessment (39%) and, even more so, in attack modeling (53%), while their adoption in secure development is significantly lower (12%). Conversely, PAs have influenced security by enabling, for instance, the discovery of unsafe protocols and promoting the adoption of formally verifiable specifications throughout the DLC. However, significant gaps remain in each of the three macro-areas. Some of these gaps may be attributed to the inherent nature of PAs. For instance, the limited adoption for the planning and maintenance phase of the DLC might be because PAs are particularly adequate for defining the semantics of agents. During planning, the semantics is still under definition, while, for maintenance, it may be scarcely relevant. Instead, the absence of PAs for software/hardware vulnerability assessment may derive from the complexity of modeling low-level, technology-specific security flaws. For instance, buffer overflow and data races stem from the way memory and parallelism are actually implemented in computers. As PAs abstract from these details, they might be not particularly suitable for the task. Nevertheless, the investigation on modeling attackers during development and repudiation attacks do not suffer from this issue and, thus, we expect them to be further developed in future researches. Regarding future research directions, following our study, we envisage novel advancements in some areas while little to no progress in others. For instance, PA has seen increasing adoption in IoT and CPS security. While PA has already demonstrated its utility in these areas, future research may focus on refining existing frameworks to handle the complexity of large-scale IoT networks, real-time security verification, and automated threat mitigation. Similarly, research on the application of PA in quantum systems to address cybersecurity challenges is expected to grow. While several quantum PAs have already been developed, their use in tackling cybersecurity problems remains limited. Additionally, integrating quantum PA with classical cybersecurity frameworks may provide novel approaches for modeling and analyzing threats in hybrid quantum–classical systems. Blockchain and decentralized security mechanisms also present a promising avenue for the application of PA. Given the formal and structured nature of blockchain protocols, PA can be used to model and analyze the correctness and security of these processes. On the other hand, it is important to recognize the limitations of PA in certain domains. Specifically, its application to the security of machine learning-based systems remains highly challenging, if even viable. Reasonably, this is due to the fact that most security issues in this context are related to global, emerging behaviors, e.g., hallucinations and backdoors [202]. However, PAs are best for analyzing the local behavior of agents.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gabriele Costa, Silvia De Francisci, Rocco De Nicola reports financial support was provided by NextGenerationEU. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was partially supported by project SEcurity and RIghts in the CyberSpace SERICS PE0000014 - PNRR M4C2 I.1.3, Financed by the European union Next Generation EU - CUP: D67G22000340001.

## Data availability

No data was used for the research described in the article.

## References

[1] Michael O. Rabin, Dana Scott, Finite automata and their decision problems, IBM J. Res. Dev. 3 (2) (1959) 114–125.
[2] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Introduction to automata theory, languages, and computation, Acm Sigact News 32 (1) (2001) 60–65.
[3] Alonzo Church, The Calculi of Lambda Conversion. (AM-6), vol. 6, Princeton University Press, 2016.
[4] Andrei Sabelfeld, Andrew C. Myers, Language-based information-flow security, IEEE J. Sel. Areas Commun. 21 (1) (2003) 5–19.
[5] Rocco De Nicola, Behavioral equivalences, in: David A. Padua (Ed.), Encyclopedia of Parallel Computing, Springer, 2011, pp. 120–127.
[6] Jos C.M. Baeten, A brief history of process algebra, Theoret. Comput. Sci. 335 (2–3) (2005) 131–146.
[7] Stephen D. Brookes, A.W. Roscoe, CSP: a practical process algebra, in: Theories of Programming: The Life and Works of Tony Hoare, Association for Computing Machinery, 2021, pp. 187–222.
[8] Alessandro Aldini, Marco Bernardo, Flavio Corradini, A Process Algebraic Approach to Software Architecture Design, Springer Science & Business Media, 2010.
[9] Maurice H. Ter Beek, Antonio Bucchiarone, Stefania Gnesi, Formal methods for service composition, Ann. Math. Comput. Teleinform. 1 (5) (2007) 1–10.
[10] Meftah M.C. Eddine, A comparative study of formal approaches for web service oriented architecture, Netw. Commun. Technol. 5 (2) (2020) 15–33.
[11] Vu Tuan Anh, Pham Quoc Cuong, Phan Cong Vinh, Context-aware mobility based on $\pi$-calculus in Internet of Thing: A survey, in: Context-Aware Systems and Applications, and Nature of Computation and Communication, Springer, 2019, pp. 38–46.
[12] Kaiyu Wan, Danny Hughes, Ka L Man, Tomas Krilavičius, Composition challenges and approaches for cyber physical systems, in: 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, IEEE, 2010, pp. 1–7.
[13] Wojciech Wideł, Maxime Audinot, Barbara Fila, Sophie Pinchinat, Beyond 2014: Formal methods for attack tree–based security modeling, ACM Comput. Surv. 52 (4) (2019) 1–36.
[14] Peter Ryan, Steve A Schneider, Michael Goldsmith, Gavin Lowe, Bill Roscoe, The Modelling and Analysis of Security Protocols: The CSP Approach, Addison-Wesley Professional, 2001.
[15] Mark D. Ryan, Ben Smyth, Applied pi calculus, in: Formal Models and Techniques for Analyzing Security Protocols, Ios Press, 2011, pp. 112–142.
[16] Daniel Deogun, Dan Johnsson, Daniel Sawano, Secure by Design, Manning Publications, 2019.
[17] Adam Shostack, Experiences threat modeling at microsoft, MODSEC@ Models 2008 (2008) 35.
[18] Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering, EASE, BCS Learning & Development, 2008, p. 10.
[19] Rocco De Nicola, Process algebras, in: David A. Padua (Ed.), Encyclopedia of Parallel Computing, Springer, 2011, pp. 1624–1636.
[20] Robin Milner, A Calculus of Communicating Systems, Springer Verlag, 1980.
[21] Stephen D. Brookes, Charles A.R. Hoare, Andrew W. Roscoe, A theory of communicating sequential processes, J. ACM 31 (3) (1984) 560–599.
[22] Jan A. Bergstra, Jan Willem Klop, Process algebra for synchronous communication, Inf. Control 60 (1–3) (1984) 109–137.
[23] David Park, Concurrency and automata on infinite sequences, in: Theoretical Computer Science, Springer, 1981, pp. 167–183.
[24] Rocco De Nicola, Matthew C.B. Hennessy, Testing equivalences for processes, Theoret. Comput. Sci. 34 (1–2) (1984) 83–133.
[25] Robin Milner, Communication and Concurrency, vol. 84, Prentice hall Englewood Cliffs, 1989.
[26] C.A.R. Hoare, Communicating sequential processes, Commun. ACM 21 (8) (1978) 666–677.
[27] Jan A. Bergstra, Jan Willem Klop, Fixed Point Semantics in Process Algebras, Technical Report IW 208, Mathematical Centre, Amsterdam, 1982.
[28] Jan A. Bergstra, Jan Willem Klop, Process algebra: specification and verification in bisimulation semantics, Math. Comput. Sci. II 4 (1986).
[29] Mingsheng Ying, Yuan Feng, Runyao Duan, Zhengfeng Ji, An algebra of quantum processes, ACM Trans. Comput. Log. (TOCL) 10 (3) (2009) 1–36.
[30] Ed Brinksma, LOTOS A formal description technique based on the temporal ordering of observational behaviour, Int. Organ. Stand. 8807 (1988).
[31] Ansgar Fehnker, Rob van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, Wee Lum Tan, A process algebra for wireless mesh networks, in: European Symposium on Programming, Springer, 2012, pp. 295–315.
[32] Riccardo Focardi, Roberto Gorrieri, A classification of security properties for process algebras 1, J. Comput. Secur. 3 (1) (1995) 5–33.
[33] Rocco De Nicola, Gian Luigi Ferrari, Rosario Pugliese, KLAIM: A kernel language for agents interaction and mobility, IEEE Trans. Softw. Eng. 24 (5) (1998) 315–330.
[34] Insup Lee, Hanene Ben-Abdallah, Jin-Young Choi, A process algebraic method for the specification and analysis of real-time systems, Form. Methods Real-Time Comput. (1996) 167–194.
[35] Anna Philippou, Rance Cleaveland, Insup Lee, Scott Smolka, Oleg Sokolsky, Probabilistic resource failure in real-time process algebra, in: International Conference on Concurrency Theory, Springer, 1998, pp. 389–404.
[36] Matthew Hennessy, Tim Regan, A process algebra for timed systems, Inform. and Comput. 117 (2) (1995) 221–239.
[37] Flavio Oquendo, $\pi$-Calculus for SoS: A foundation for formally describing software-intensive systems-of-systems, in: 2016 11th System of Systems Engineering Conference (SoSE), IEEE, 2016, pp. 1–6.
[38] Peter Sewell, Global/local subtyping and capability inference for a distributed $\pi$-calculus, in: International Colloquium on Automata, Languages, and Programming, Springer, 1998, pp. 695–706.
[39] Jan Vitek, Giuseppe Castagna, Seal: A framework for secure mobile computations, in: International Conference on Computer Languages, Springer, 1998, pp. 47–77.
[40] Simon J. Gay, Rajagopal Nagarajan, Communicating quantum processes, in: Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2005, pp. 145–157.
[41] Luca Cardelli, Andy Gordon, Mobile ambients, Theoret. Comput. Sci. 240 (1) (2000) 177–213.
[42] Alessandro Aldini, Mario Bravetti, An asynchronous calculus for generative-reactive probabilistic systems, in: Proceedings of the 8th International Workshop on Process Algebra and Performance Modeling, Carleton Scientific, Citeseer, 2000, pp. 591–605.
[43] Michele Bugliesi, Giuseppe Castagna, Silvia Crafa, Boxed ambients, in: International Symposium on Theoretical Aspects of Computer Software, Springer, 2001, pp. 38–63.
[44] Andrew W. Roscoe, Geoffrey M. Reed, A timed model for communicating sequential processes, Theoret. Comput. Sci. 58 (1988).
[45] Jane Hillston, A Compositional Approach to Performance Modelling (Ph.D. thesis), Cambridge University Press, 1996.
[46] Marco Bernardo, Roberto Gorrieri, Extended Markovian process algebra, in: International Conference on Concurrency Theory, Springer, 1996, pp. 315–330.
[47] Federica Ciocchetta, Jane Hillston, Bio-PEPA: A framework for the modelling and analysis of biological systems, Theoret. Comput. Sci. 410 (33–34) (2009) 3065–3084.
[48] Alain J. Martin, Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits, Technical Report, California Institute of Technology Pasadena Department of Computer Science, 1989.
[49] Jan A. Bergstra, Jan W. Klop, Process Algebra for Communication and Mutual Exclusion, Technical Report IW 218/83, Stichting Mathematisch Centrum, Amsterdam, 1983.
[50] Jan Friso Groote, Michel A. Reniers, Algebraic process verification, in: Handbook of Process Algebra, Elsevier, 2001, pp. 1151–1208.
[51] Pieter Jan Laurens Cuijpers, Michel A. Reniers, Hybrid process algebra, J. Log. Algebr. Program. 62 (2) (2005) 191–245.
[52] Dirk A van Beek, Ka L Man, Michel A Reniers, Jacobus E Rooda, Ramon RH Schiffelers, Syntax and consistent equation semantics of hybrid Chi, J. Log. Algebr. Program. 68 (1–2) (2006) 129–210.
[53] Yong Wang, Algebraic laws for true concurrency, 2016, arXiv preprint arXiv:1611.09035.
[54] Fabio Martinelli, Ilaria Matteucci, Partial model checking, process algebra operators and satisfiability procedures for (automatically) enforcing security properties, in: Foundations of Computer Security, 2005, pp. 133–144.

[55] Yong Wang, An axiomatization for quantum processes to unifying quantum and classical computing, Internat. J. Theoret. Phys. 58 (10) (2019) 3295–3322.
[56] Nicola Mezzetti, Davide Sangiorgi, Towards a calculus for wireless systems, Electron. Notes Theor. Comput. Sci. 158 (2006) 331–353.
[57] Rocco De Nicola, Rosario Pugliese, Testing semantics of asynchronous distributed programs, in: LOMAPS Workshop on Analysis and Verification of Multiple-Agent Languages, Springer, 1996, pp. 320–344.
[58] Grady Booch, The Unified Modeling Language User Guide, Pearson Education India, 2005.
[59] Stephen A. White, Introduction to BPMN, BPTrends (2004).
[60] Edmund M Clarke Jr., Orna Grumberg, Daniel Kroening, Doron Peled, Helmut Veith, Model Checking, MIT Press, 2018.
[61] Jesper R Andersen, Nicklas Andersen, Søren Enevoldsen, Mathias M Hansen, Kim G Larsen, Simon R Olesen, Jirí Srba, Jacob K Wortmann, CAAL: concurrency workbench, in: International Colloquium on Theoretical Aspects of Computing, Springer, 2015, pp. 573–582.
[62] Francesco Calzolai, Rocco De Nicola, Michele Loreti, Francesco Tiezzi, TAPAs: A tool for the analysis of process algebras, in: Transactions on Petri Nets and Other Models of Concurrency I, Springer, 2008, pp. 54–70.
[63] Danny Dolev, Andrew Yao, On the security of public key protocols, IEEE Trans. Inform. Theory 29 (2) (1983) 198–208.
[64] Christelle Chaudet, Flavio Oquendo, /Spl pi/-SPACE: a formal architecture description language based on process algebra for evolving software systems, in: Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering, IEEE, 2000, pp. 245–248.
[65] Hanne R. Nielson, Flemming Nielson, Roberto Vigo, A calculus for quality, in: International Workshop on Formal Aspects of Component Software, Springer, 2012, pp. 188–204.
[66] Oscar Nierstrasz, Franz Achermann, A calculus for modeling software components, in: International Symposium on Formal Methods for Components and Objects, Springer, 2002, pp. 339–360.
[67] Flavio Oquendo, Formally describing the software architecture of systems-of-systems with SosADL, in: 2016 11th System of Systems Engineering Conference (SoSE), IEEE, 2016, pp. 1–6.
[68] Alessandro Aldini, Marco Bernardo, On the usability of process algebra: An architectural view, Theoret. Comput. Sci. 335 (2–3) (2005) 281–329.
[69] Zhiru Hou, Lili Xiao, Huibiao Zhu, Phan Cong Vinh, Formalization and analysis of aeolus-based file system from process algebra perspective, Mob. Netw. Appl. 29 (1) (2024) 273–285.
[70] Vitus S.W. Lam, Julian Padget, Consistency checking of sequence diagrams and statechart diagrams using the $\pi$-calculus, in: International Conference on Integrated Formal Methods, Springer, 2005, pp. 347–365.
[71] Jonathan Lawrence, Practical application of CSP and FDR to software design, in: Communicating Sequential Processes. the First 25 Years, Springer, 2005, pp. 151–174.
[72] Radu Mateescu, Pascal Poizat, Gwen Salaün, Behavioral adaptation of component compositions based on process algebra encodings, in: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, 2007, pp. 385–388.
[73] Radu Mateescu, Pascal Poizat, Gwen Salaün, Adaptation of service protocols using process algebra and on-the-fly reduction techniques, IEEE Trans. Softw. Eng. 38 (4) (2011) 755–777.
[74] Sjouke Mauw, Gerrit Jan Veltink, A process specification formalism, Fund. Inform. 13 (2) (1990) 85–139.
[75] Luu Anh Tuan, Man Chun Zheng, Quan Thanh Tho, Modeling and verification of safety critical systems: A case study on pacemaker, in: 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, IEEE, 2010, pp. 23–32.
[76] Xi Wu, Huibiao Zhu, Formalization and analysis of the REST architecture from the process algebra perspective, Future Gener. Comput. Syst. 56 (2016) 153–168.
[77] Hee-Hwan Kwak, Insup Lee, Anna Philippou, Jin-Young Choi, Oleg Sokolsky, Symbolic schedulability analysis of real-time systems, in: Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279), IEEE, 1998, pp. 409–418.
[78] Dominique Borrione, Menouer Boubekeur, Laurent Mounier, Marc Renaudin, Antoine Siriani, Validation of asynchronous circuit specifications using IF/CADP, in: VLSI-SOC: From Systems to Chips, Springer, 2006, pp. 85–100.
[79] Hemangee K. Kapoor, Mark B. Josephs, Modelling and verification of delay-insensitive circuits using CCS and the Concurrency Workbench, Inform. Process. Lett. 89 (6) (2004) 293–296.
[80] Ka L. Man, J. van der Wulp, Specification and analysis of hardware designs using mCRL2, in: 2008 Canadian Conference on Electrical and Computer Engineering, IEEE, 2008, pp. 000211–000214.
[81] Gwen Salaun, Wendelin Serwe, Yvain Thonnart, Pascal Vivet, Formal verification of CHP specifications with CADP illustration on an asynchronous network-on-chip, in: 13th IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC'07, IEEE, 2007, pp. 73–82.
[82] Marco Carbone, Mogens Nielsen, Vladimiro Sassone, A calculus for trust management, in: International Conference on Foundations of Software Technology and Theoretical Computer Science, Springer, 2004, pp. 161–173.
[83] Xiao Chen, Jie Ding, Zhenyu Lu, A decentralized trust management system for intelligent transportation environments, IEEE Trans. Intell. Transp. Syst. (2020).
[84] Yuli Yao, Wendong Chen, Xiao Chen, Jie Ding, Senshan Pan, A blockchain-based privacy preserving scheme for vehicular trust management systems, in: 2020 International Conference on Internet of Things and Intelligent Applications, ITIA, IEEE, 2020, pp. 1–5.
[85] Wolfgang Boehmer, Christoph Brandt, Jan Friso Groote, Evaluation of a business continuity plan using process algebra and modal logic, in: 2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH), IEEE, 2009, pp. 147–152.
[86] Ali N Haidar, Ali E Abdallah, PYA Ryan, PV Coveney, B Beckles, JM Brooke, M AS Jones, Formal modelling of a usable identity management solution for virtual organisations, Proc. Form. Asp. Virtual Organ. (2009) 17.
[87] Ali N. Haidar, Ali E. Abdallah, Formal modelling of pki based authentication, Electron. Notes Theor. Comput. Sci. 235 (2009) 55–70.
[88] Myrto Arapinis, Sergiu Bursuc, Mark Ryan, Privacy supporting cloud computing: Confichair, a case study, in: International Conference on Principles of Security and Trust, Springer, 2012, pp. 89–108.
[89] Yongxiang Li, Xifan Yao, Cloud manufacturing service composition and formal verification based on extended process calculus, Adv. Mech. Eng. 10 (6) (2018) 1687814018781287.
[90] Alireza Souri, Amir Masoud Rahmani, Nima Jafari Navimipour, Reza Rezaei, A hybrid formal verification approach for qos-aware multi-cloud service composition, Clust. Comput. 23 (4) (2020) 2453–2470.
[91] Yosr Jarraya, Arash Eghtesadi, Mourad Debbabi, Ying Zhang, Makan Pourzandi, Cloud calculus: Security verification in elastic cloud computing platform, in: 2012 International Conference on Collaboration Technologies and Systems, CTS, IEEE, 2012, pp. 447–454.
[92] Keng-Pei Lin, Chih-Ya Shen, William Chao, Enriching UML from model multiplicity to model singularity with structure-behavior coalescence, in: 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC, IEEE, 2018, pp. 1970–1975.
[93] Guiping Dai, Yong Wang, Relation of web service orchestration, abstract process, web service and choreography, in: 2020 7th International Conference on Information Science and Control Engineering, ICISCE, IEEE, 2020, pp. 1082–1084.
[94] Javier Cámara, Carlos Canal, Javier Cubo, Antonio Vallecillo, Formalizing wsbpel business processes using process algebra, Electron. Notes Theor. Comput. Sci. 154 (1) (2006) 159–173.
[95] Christophe Dumez, Mohamed Bakhouya, Jaafar Gaber, Maxime Wack, Pascal Lorenz, Model-driven approach supporting formal verification for web service composition protocols, J. Netw. Comput. Appl. 36 (4) (2013) 1102–1115.
[96] Jiaqi Yin, Huibiao Zhu, Phan Cong Vinh, Formalization and analysis of haystack architecture from process algebra perspective, Mob. Netw. Appl. 25 (3) (2020) 1125–1139.
[97] Gwen Salaun, Lucas Bordeaux, Marco Schaerf, Describing and reasoning on web services using process algebra, Int. J. Bus. Process. Integr. Manag. 1 (2) (2006) 116–128.
[98] Gwen Salaün, Tevfik Bultan, Nima Roohi, Realizability of choreographies using process algebra encodings, IEEE Trans. Serv. Comput. 5 (3) (2011) 290–304.
[99] Yong-Lian Wang, Xue-Li Yu, Formalization and verification of automatic composition based on pi-calculus for semantic web service, in: 2009 Second International Symposium on Knowledge Acquisition and Modeling, Vol. 1, IEEE, 2009, pp. 103–106.
[100] Alessandro Lapadula, Rosario Pugliese, Francesco Tiezzi, Service discovery and negotiation with COWS, Electron. Notes Theor. Comput. Sci. 200 (3) (2008) 133–154.
[101] Junya Xu, Jiaqi Yin, Huibiao Zhu, Lili Xiao, Modeling and verifying producer-consumer communication in Kafka using CSP, in: 7th Conference on the Engineering of Computer Based Systems, 2021, pp. 1–10.
[102] Karthikeyan Bhargavan, Cédric Fournet, Andrew D Gordon, Stephen Tse, Verified interoperable implementations of security protocols, ACM Trans. Program. Lang. Syst. (TOPLAS) 31 (1) (2008) 1–61.
[103] Ivan Lanese, Luca Bedogni, Marco Di Felice, Internet of things: a process calculus approach, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013, pp. 1339–1346.
[104] Ruggero Lanotte, Massimo Merro, A calculus of cyber-physical systems, in: International Conference on Language and Automata Theory and Applications, Springer, 2017, pp. 115–127.
[105] Dirk A. van Beek, Ann-Katrin van den Ham, Jacobus E. Rooda, Modelling and control of process industry batch production systems, IFAC Proc. Vol. 35 (1) (2002) 403–408.
[106] Yang Zhang, Li Duan, Jun Liang Chen, Event-driven soa for iot services, in: 2014 IEEE International Conference on Services Computing, IEEE, 2014, pp. 629–636.
[107] Hamed Arshad, Ross Horne, Christian Johansen, Olaf Owe, Tim AC Willemse, Process algebra can save lives: static analysis of XACML access control policies using mCRL2, in: International Conference on Formal Techniques for Distributed Objects, Components, and Systems, Springer, 2022, pp. 11–30.

[108] Chiara Bodei, Pierpaolo Degano, Gian-Luigi Ferrari, Letterio Galletta, Tracing where IoT data are collected and aggregated, Log. Methods Comput. Sci. 13 (2017).

[109] Huailin Li, Qinsen Liu, Mengnan Liu, Bangyong Sun, Bin Du, Robust deadlock control for reconfigurable printing manufacturing system based on process algebra, IEEE Access 11 (2023) 42473–42484.

[110] Rubai Luo, Shasha Gao, Huailin Li, Shisheng Zhou, Modeling and verification of reconfigurable printing system based on process algebra, Math. Probl. Eng. 2018 (2018).

[111] Peng Wang, Yang Xiang, Shao Hua Zhang, Cyber-physical system components composition analysis and formal verification based on service-oriented architecture, in: 2012 IEEE Ninth International Conference on E-Business Engineering, IEEE, 2012, pp. 327–332.

[112] Cong Xinyu, Yu Huiqun, Xu Xin, Verification of Hybrid Chi model for cyber-physical systems using PHAVer, in: 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE, 2013, pp. 122–128.

[113] Ruggero Lanotte, Massimo Merro, Andrei Munteanu, A process calculus approach to correctness enforcement of PLCs, in: ICTCS, 2020, pp. 81–94.

[114] Giuseppe Castagna, Giorgio Ghelli, Francesco Zappa Nardelli, Typing mobility in the seal calculus, in: International Conference on Concurrency Theory, Springer, 2001, pp. 82–101.

[115] Massimo Merro, Francesco Ballardin, Eleonora Sibilio, A timed calculus for wireless systems, Theoret. Comput. Sci. 412 (47) (2011) 6585–6611.

[116] Parul Yadav, Manish Gaur, Process Calculi for intrusion detection system in mobile Ad-hoc networks, J. Commun. 13 (November) (2018) 635–647.

[117] Ed Kamya Kiyemba Edris, Mahdi Aiash, Mohammad Ali Khoshkholghi, Ranesh Naha, Abdullahi Chowdhury, Jonathan Loo, Performance and cryptographic evaluation of security protocols in distributed networks using applied pi calculus and Markov chain, Internet Things 24 (2023) 100913.

[118] Ansgar Fehnker, Rob Van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, Wee Lum Tan, Automated analysis of AODV using UPPAAL, in: Tools and Algorithms for the Construction and Analysis of Systems: 18th International Conference, TACAS 2012, Held As Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24–April 1, 2012. Proceedings 18, Springer, 2012, pp. 173–187.

[119] Myung-Ki Shin, Yunchul Choi, Hee Hwan Kwak, Sangheon Pack, Miyoung Kang, Jin-Young Choi, Verification for NFV-enabled network services, in: 2015 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2015, pp. 810–815.

[120] Dmitry A. Zaitsev, Tatiana R. Shmeleva, Jan Friso Groote, Verification of hypertorus communication grids by infinite petri nets and process algebra, IEEE/ CAA J. Autom. Sin. 6 (3) (2019) 733–742.

[121] Jiaqi Yin, Huibiao Zhu, Yuan Fei, Yucheng Fang, Modeling and verifying spark on YARN using process algebra, in: 2019 IEEE 19th International Symposium on High Assurance Systems Engineering, HASE, IEEE, 2019, pp. 208–215.

[122] Tao Xia, Menglin Wang, Jun He, Shaofeng Lin, Yongqi Shi, Liyuan Guo, Research on identity authentication scheme for uav communication network, Electronics 12 (13) (2023) 2917.

[123] Michele Bugliesi, Giuseppe Castagna, Silvia Crafa, Reasoning about security in mobile ambients, in: International Conference on Concurrency Theory, Springer, 2001, pp. 102–120.

[124] Matthew Collinson, David Pym, Algebra and logic for access control, Form. Asp. Comput. 22 (2) (2010) 83–104.

[125] Riccardo Focardi, Roberto Gorrieri, Fabio Martinelli, Information flow analysis in a discrete-time process algebra, in: Proceedings 13th IEEE Computer Security Foundations Workshop. CSFW-13, IEEE, 2000, pp. 170–184.

[126] Kohei Honda, Vasco Vasconcelos, Nobuko Yoshida, Secure information flow as typed process behaviour, in: European Symposium on Programming, Springer, 2000, pp. 180–199.

[127] Steve Kremer, Mark Ryan, Ben Smyth, Election verifiability in electronic voting protocols, in: European Symposium on Research in Computer Security, Springer, 2010, pp. 389–404.

[128] Peter Y.A. Ryan, Mathematical models of computer security, in: International School on Foundations of Security Analysis and Design, Springer, 2000, pp. 1–62.

[129] Peter Y.A. Ryan, Steve A. Schneider, Process algebra and non-interference, J. Comput. Secur. 9 (1–2) (2001) 75–103.

[130] Riccardo Focardi, Roberto Gorrieri, Classification of security properties, in: International School on Foundations of Security Analysis and Design, Springer, 2000, pp. 331–396.

[131] Robert Künnemann, Automated backward analysis of PKCS# 11 v2. 20, in: International Conference on Principles of Security and Trust, Springer, 2015, pp. 219–238.

[132] Gavin Lowe, A hierarchy of authentication specifications, in: Proceedings 10th Computer Security Foundations Workshop, IEEE, 1997, pp. 31–43.

[133] Sjouke Mauw, Jan H.S. Verschuren, Erik P. de Vink, A formalization of anonymity and onion routing, in: European Symposium on Research in Computer Security, Springer, 2004, pp. 109–124.

[134] C.A. Middelburg, Imperative process algebra with abstraction, Sci. Ann. Comput. Sci. 32 (1) (2022) 137–179.

[135] Timothy AS Davidson, Simon J Gay, Hynek Mlnarik, Rajagopal Nagarajan, Nick Papanikolaou, Model checking for communicating quantum processes, Int. J. Unconv. Comput. 8 (1) (2012) 73–98.

[136] Takahiro Kubota, Yoshihiko Kakutani, Go Kato, Yasuhito Kawano, Hideki Sakurada, Semi-automated verification of security proofs of quantum cryptographic protocols, J. Symbolic Comput. 73 (2016) 192–220.

[137] Peter Y.H. Wong, Jeremy Gibbons, Property specifications for workflow modelling, in: International Conference on Integrated Formal Methods, Springer, 2009, pp. 56–71.

[138] Steve Schneider, Security properties and CSP, in: Proceedings 1996 IEEE Symposium on Security and Privacy, IEEE, 1996, pp. 174–187.

[139] Kamel Adi, Lamia Hamza, Liviu Pene, Automatic security policy enforcement in computer systems, Comput. Secur. 73 (2018) 156–171.

[140] David Basin, Samuel J. Burri, Günter Karjoth, Dynamic enforcement of abstract separation of duty constraints, ACM Trans. Inf. Syst. Secur. (TISSEC) 15 (3) (2012) 1–30.

[141] David Basin, Samuel J. Burri, Günter Karjoth, Obstruction-free authorization enforcement: Aligning security and business objectives, J. Comput. Secur. 22 (5) (2014) 661–698.

[142] Fabio Martinelli, Ilaria Matteucci, Charles Morisset, From qualitative to quantitative enforcement of security policy, in: International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, Springer, 2012, pp. 22–35.

[143] Mahjoub Langar, Mohamed Mejri, Kamel Adi, Formal enforcement of security policies on concurrent systems, J. Symbolic Comput. 46 (9) (2011) 997–1016.

[144] Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, Daniel Loebenberger, A formal analysis of IKEv2's post-quantum extension, in: Annual Computer Security Applications Conference, 2021, pp. 91–105.

[145] Steve Schneider, Rob Delicata, Verifying security protocols: An application of CSP, in: Communicating Sequential Processes. the First 25 Years, Springer, 2005, pp. 243–263.

[146] Jeremy Bryans, Hoang Nga Nguyen, Siraj Ahmed Shaikh, Attack defense trees with sequential conjunction, in: 2019 IEEE 19th International Symposium on High Assurance Systems Engineering, HASE, IEEE, 2019, pp. 247–252.

[147] Roberto Vigo, Flemming Nielson, Hanne R. Nielson, Automated generation of attack trees, in: 2014 IEEE 27th Computer Security Foundations Symposium, IEEE, 2014, pp. 337–350.

[148] Ruggero Lanotte, Massimo Merro, Andrei Munteanu, Luca Viganò, A formal approach to physics-based attacks in cyber-physical systems, ACM Trans. Priv. Secur. (TOPS) 23 (1) (2020) 1–41.

[149] Ivan Lanese, Davide Sangiorgi, An operational semantics for a calculus for wireless systems, Theoret. Comput. Sci. 411 (19) (2010) 1928–1948.

[150] Ruggero Lanotte, Massimo Merro, Andrei Munteanu, A process calculus approach to detection and mitigation of PLC malware, Theoret. Comput. Sci. 890 (2021) 125–146.

[151] Alessandro Aldini, Roberto Gorrieri, Security analysis of a probabilistic non-repudiation protocol, in: Joint International Workshop Von Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Springer, 2002, pp. 17–36.

[152] Francesco Ballardin, Massimo Merro, A calculus for the analysis of wireless network security protocols, in: International Workshop on Formal Aspects in Security and Trust, Springer, 2010, pp. 206–222.

[153] Roberto Gorrieri, Enrico Locatelli, Fabio Martinelli, A simple language for real-time cryptographic protocol analysis, in: European Symposium on Programming, Springer, 2003, pp. 114–128.

[154] John McDermott, Attack-potential-based survivability modeling for high-consequence systems, in: Third IEEE International Workshop on Information Assurance, IWIA'05, IEEE, 2005, pp. 119–130.

[155] Christian W. Probst, René R. Hansen, Flemming Nielson, Where can an insider attack? in: International Workshop on Formal Aspects in Security and Trust, Springer, 2006, pp. 127–142.

[156] Jeremy T. Bradley, Stephen T. Gilmore, Jane Hillston, Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models, J. Comput. System Sci. 74 (6) (2008) 1013–1032.

[157] Stephen Gilmore, Jane Hillston, Natalia Zoń, Abstract interpretation of PEPA models, in: Semantics, Logics, and Calculi, Springer, 2016, pp. 140–158.

[158] Ruggero Lanotte, Massimo Merro, Andrei Munteanu, Simone Tini, Formal impact metrics for cyber-physical attacks, in: 2021 IEEE 34th Computer Security Foundations Symposium, CSF, IEEE, 2021, pp. 1–16.

[159] Damiano Macedonio, Massimo Merro, A semantic analysis of key management protocols for wireless sensor networks, Sci. Comput. Program. 81 (2014) 53–78.

[160] Livinus O Nweke, Goitom K Weldehawaryat, Stephen D Wolthusen, Threat modelling of Cyber–Physical Systems using an applied $\pi$-calculus, Int. J. Crit. Infrastruct. Prot. 35 (2021) 100466.

[161] Faiza Benmenzer, Rachid Beghdad, An adaptive formal parallel technique with reputation integration for the enforcement of security policy in the cloud environment, Comput. Commun. 196 (2022) 207–228.

[162] Thomas R. McEvoy, Stephen D. Wolthusen, A formal adversary capability model for SCADA environments, in: International Workshop on Critical Information Infrastructures Security, Springer, 2010, pp. 93–103.
[163] Limin Shen, Hui Li, Hongyi Wang, Yihuan Wang, Jiayin Feng, Yuqing Jian, Risk measurement method for privilege escalation attacks on android apps based on process algebra, Information 11 (6) (2020) 293.
[164] Dongkui Liang, Limin Shen, Zhen Chen, Chuan Ma, Jiayin Feng, A formal method for description and decision of android apps behavior based on process algebra, IEEE Access 10 (2022) 108668–108683.
[165] Adam Doupé, Marco Cova, Giovanni Vigna, Why Johnny can't pentest: An analysis of black-box web vulnerability scanners, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2010, pp. 111–131.
[166] Jan Friso Groote, Aad Mathijssen, Michel Reniers, Yaroslav Usenko, Muck van Weerdenburg, The Formal Specification Language mCRL2, in: Ed Brinksma, David Harel, Angelika Mader, Perdita Stevens, Roel Wieringa (Eds.), Methods for Modelling Software Systems, MMOSS, in: Dagstuhl Seminar Proceedings (DagSemProc), 6351, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, (ISSN: 1862-4405) 2007, pp. 1–34, http://dx.doi.org/10.4230/DagSemProc.06351.12, URL https://drops.dagstuhl.de/opus/volltexte/2007/862.
[167] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, Llanos Tobarra, Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps, in: Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering, 2008, pp. 1–10.
[168] Gavin Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, in: International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 1996, pp. 147–166.
[169] Gordon Thomas Rohrmair, Gavin Lowe, Using data-independence in the analysis of intrusion detection systems, Theoret. Comput. Sci. 340 (1) (2005) 82–101.
[170] Guosheng Zhao, Xiaofeng Qu, Yuting Liao, Tiantian Wang, Jingting Zhang, Cloud service security adaptive target detection algorithm based on bio-inspired performance evaluation process algebra, Wuhan Univ. J. Nat. Sci. 24 (3) (2019) 185–193.
[171] Chuan Ma, Tao Wang, Limin Shen, Dongkui Liang, Shuping Chen, Dianlong You, Communication-based attacks detection in android applications, Tsinghua Sci. Technol. 24 (5) (2019) 596–614.
[172] Ravi Akella, Han Tang, Bruce M. McMillin, Analysis of information flow security in cyber–physical systems, Int. J. Crit. Infrastruct. Prot. 3 (3–4) (2010) 157–173.
[173] Colin O'Halloran, Tom Gibson Robinson, Neil Brock, Verifying cyber attack properties, Sci. Comput. Program. 148 (2017) 3–25.
[174] Roberto Vigo, Flemming Nielson, Hanne R. Nielson, Broadcast, denial-of-service, and secure communication, in: International Conference on Integrated Formal Methods, Springer, 2013, pp. 412–427.
[175] Chiara Bodei, Gian-Luigi Ferrari, Letterio Galletta, Pierpaolo Degano, Risk estimation in IoT systems, in: Challenges of Software Verification, Springer, 2023, pp. 221–242.
[176] Abu Shohel Ahmed, Aleksi Peltonen, Mohit Sethi, Tuomas Aura, Security analysis of the consumer remote SIM provisioning protocol, ACM Trans. Priv. Secur. 27 (3) (2024) 1–36.
[177] Myrto Arapinis, Tom Chothia, Eike Ritter, Mark Ryan, Analysing unlinkability and anonymity using the applied pi calculus, in: 2010 23rd IEEE Computer Security Foundations Symposium, IEEE, 2010, pp. 107–121.
[178] Michael Backes, Matteo Maffei, Dominique Unruh, Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol, in: 2008 IEEE Symposium on Security and Privacy (S&P 2008), IEEE, 2008, pp. 202–215.
[179] Stéphanie Delaune, Steve Kremer, Ludovic Robin, Formal verification of protocols based on short authenticated strings, in: 2017 IEEE 30th Computer Security Foundations Symposium, CSF, IEEE, 2017, pp. 130–143.
[180] Stéphanie Delaune, Steve Kremer, Mark Ryan, Coercion-resistance and receipt-freeness in electronic voting, in: 19th IEEE Computer Security Foundations Workshop, CSFW'06, IEEE, 2006, pp. 12–pp.
[181] Lucca Hirschi, Cas Cremers, Improving automated symbolic analysis of ballot secrecy for e-voting protocols: A method based on sufficient conditions, in: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2019, pp. 635–650.
[182] Riccardo Focardi, Anna Ghelli, Roberto Gorrieri, Using non interference for the analysis of security protocols, in: Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols, Citeseer, 1997, pp. 3–5.
[183] Riccardo Focardi, Roberto Gorrieri, Fabio Martinelli, Non interference for the analysis of cryptographic protocols, in: International Colloquium on Automata, Languages, and Programming, Springer, 2000, pp. 354–372.
[184] Stéphane Lafrance, John Mullins, Using admissible interference to detect denial of service vulnerabilities, in: IWFM, 2003, p. 17.
[185] Jane Hillston, Andrea Marin, Carla Piazza, Sabina Rossi, Information flow security for stochastic processes, in: European Workshop on Performance Engineering, Springer, 2018, pp. 142–156.
[186] Andrew W. Roscoe, Jian Huang, Checking noninterference in timed CSP, Form. Asp. Comput. 25 (1) (2013) 3–35.
[187] Shuangqing Xiang, Huibiao Zhu, Xi Wu, Lili Xiao, Marcello Bonsangue, Wanling Xie, Lei Zhang, Modeling and verifying the topology discovery mechanism of OpenFlow controllers in software-defined networks using process algebra, Sci. Comput. Program. 187 (2020) 102343.
[188] Chao Xu, Huibiao Zhu, Wanling Xie, Modeling and verifying identity authentication security of HDFS using CSP, in: 2017 24th Asia-Pacific Software Engineering Conference, APSEC, IEEE, 2017, pp. 259–268.
[189] Martın Abadi, Andrew D. Gordon, A calculus for cryptographic protocols: The spi calculus, Inform. and Comput. 148 (1) (1999) 1–70.
[190] Steve Kremer, Robert Künnemann, Automated analysis of security protocols with global state, J. Comput. Secur. 24 (5) (2016) 583–616.
[191] Giuseppe Crincoli, Giacomo Iadarola, Piera Elena La Rocca, Fabio Martinelli, Francesco Mercaldo, Antonella Santone, Vulnerable smart contract detection by means of model checking, in: Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure, 2022, pp. 3–10.
[192] Heyang Pang, Yong Wang, Verification of the wide-mouth frog protocol based on APTC, in: Proceedings of the International Conference on Algorithms, Software Engineering, and Network Security, 2024, pp. 513–518.
[193] Bruno Blanchet, Avik Chaudhuri, Automated formal analysis of a protocol for secure file sharing on untrusted storage, in: 2008 IEEE Symposium on Security and Privacy (S&P 2008), IEEE, 2008, pp. 417–431.
[194] C.R. Ramakrishnan, R. Sekar, Model-based analysis of configuration vulnerabilities 1, J. Comput. Secur. 10 (1–2) (2002) 189–209.
[195] Yong Wang, Entanglement in quantum process algebra, Internat. J. Theoret. Phys. 58 (11) (2019) 3611–3626.
[196] Bob Diertens, Process algebra based tool coordination architectures in Raku and go, 2024, arXiv preprint arXiv:2406.16614.
[197] Robin Milner, Communicating and Mobile Systems: The Pi Calculus, Cambridge University Press, 1999.
[198] Internet Engineering Task Force, The transport layer security (TLS) protocol version 1.3, 2018, Available from https://datatracker.ietf.org/doc/html/rfc8446.
[199] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, Thyla van der Merwe, A comprehensive symbolic analysis of TLS 1.3, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450349468, 2017, pp. 1773–1788, http://dx.doi.org/10.1145/3133956.3134063.
[200] Alexander Linden, Jackie Fenn, et al., Understanding Gartner's Hype Cycles, Strategic Analysis Report Nº R- 20- 1971, 88, Gartner Inc, 2003, p. 1423.
[201] Yinyu Jin, Sha Yuan, Zhou Shao, Wendy Hall, Jie Tang, Turing award elites revisited: patterns of productivity, collaboration, authorship and impact, Scientometrics 126 (2021) 2329–2348.
[202] Rajesh Gupta, Sudeep Tanwar, Sudhanshu Tyagi, Neeraj Kumar, Machine learning models for secure data analytics: A taxonomy and threat model, Comput. Commun. 153 (2020) 406–440.

**Gabriele Costa** I am an Associate Professor in Computer Science at SySMA research unit of IMT School for Advanced Studies Lucca. My previous appointments include a position as Assistant Professor at the Department of Computer Science and System Engineering (DIBRIS) of the University of Genova and Researcher at the Institute of Informatics and Telematics (IIT) of the National Research Council of Italy (CNR).

I am co-founder of the Computer Security Laboratory (CSec Lab) of the University of Genova, co-founder and CRO of the innovative start-up Talos https://www.talos-sec.com/, and co-founder of the CTF team born2scan.

**Silvia De Francisci** I am a Ph.D. Student of Computer Science and Systems Engineering within the SySMA research unit of IMT School for Advanced Studies Lucca.

I graduated in Mathematics and Computational Science at the University of Rome 'Roma Tre'.

**Rocco De Nicola** De Nicola graduated in "Scienze dell'Informazione" at Università di Pisa in 1978 and received a Ph.D. in "Computer Science" at the University of Edinburgh (UK) in 1985. De Nicola is a full professor of "Computer Science" at IMT School for Advanced Studies Lucca. From 1995 until 2011, he was full professor at Dipartimento di Sistemi e Informatica, Università di Firenze, and, from 1990 until 1995, at Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza". Previously, he was a full-time researcher at IEI-CNR in Pisa and worked at Edinburgh University, Italtel in Milano and Olivetti in Pisa. He also has been visiting professor at Ecole Normale Supérieure in Paris and at Ludwig Maximilian University of Munich and visiting scholar at Microsoft Research Cambridge. Rector of the IMT School since November 1st, 2021, De Nicola has also served as a member of the School's Academic Senate and Board of Governors.