Poster: The Beauty and the Beast (40 years of process algebra and cybersecurity)

1st Silvia De Francisci
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
silvia.defrancisci@imtlucca.it

2nd Gabriele Costa
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
gabriele.costa@imtlucca.it

3rd Rocco De Nicola
SySMA Unit
IMT School for Advanced Studies
Lucca, Italy
rocco.denicola@imtlucca.it

***Abstract*—Process algebras provide the mathematical foundation for several formal verification techniques, and they profoundly influenced many fields, from correct design to testing. Process algebras were greatly influential also for the security community. One of the main reasons for their success is their compact, yet expressive and flexible syntax, which allows to model the relevant aspects of computation while abstracting away the secondary ones. Although most authors acknowledge the importance of process algebras for the security community, it is not trivial to estimate how they shaped the past and present researches.**

The goal of this work is to provide a comprehensive outlook on some prominent works about process algebras and security. These include both the application of process algebras to security problems and process algebras inspired by security-related aspects of computation. To achieve this, we consider three fundamental fields of cybersecurity, i.e., *secure development*, *threat modeling* and *vulnerability assessment*.

***Index Terms*—process algebra, cybersecurity, formal methods in security.**

1. Introduction

Process Algebras (PA) are formal languages commonly used to model the behavior of a computational agent. Roughly speaking, PA put most of the emphasis on the control flow structure of the modeled agent, rather than on its data flow. In most PA, computational steps also emit *observable* actions. This allows to elegantly introduce the role of an external observer, i.e., someone who aims at understanding an agent's behavior, but has no control on its internal structure. Under reasonable assumptions, an attacker is, in fact, an external observer. As a matter of fact, an attacker is generally represented by its *capabilities* and *goals*. In terms of capabilities, the possibility to partially interact with the agent's I/O mechanism is a typical setting (e.g., think of information flow [19]). In terms of goals, the attacker's aim can be modeled as a target state, e.g., denoting a failure of the agent, she wants to reach. Formal models of both the attacker and the system are the fundamental building blocks of most automated, formal reasoning techniques. Not surprisingly, a vast literature about the adoption of PA in cybersecurity exists. As a result, determining the overall influence of PA in cybersecurity and its sub-fields is difficult.

This poster proposes a systematization of the existing PA and their applications to the cybersecurity fields. Our contributions are the following.[1]

- We show significant PA from a genealogy standpoint, as well as their unique characteristics.
- We discuss the impact of some PA and its genealogy in three application scenarios for cybersecurity classification: Secure Development, Threat modeling, and Vulnerability assessment.
- We show the state-of-the-art of PA for secure Development Life Cycle (DLC).

2. Genealogy of PA

The genesis of PA goes back to the 80s, when a few authors independently proposed different calculi for the specification of processes: Milner's Calculus of Communicating Systems (CCS) [16], Brookes, Hoare and Roscoe's Communicating Seqential Processes (CSP) [8], and Bergstra's Algebra of Communicating Processes (ACP) [6], which inspired a considerable number of other PA. Typically, derived PA introduce some new elements w.r.t. their archetypes. These elements often aim to model some specific aspects of computation. In order to better highlight these extensions we label PA with icons denoting their peculiar features. Such features include the ability to model security aspects (🔒), timed computation (🕒), stochastic behaviors (🎲), quantum computing (⚛), imperative statements (❗), cyber-physical systems 🚗 and wireless networks 📶. Arguably, the main reason behind this diversification is the needing of considering specific aspects of the computation of distributed agents. In this landscape, security is no exception and several security-related PA (🔒) emerged. However, the main difference w.r.t. other domain-specific PA is that security does not refer to some peculiar aspect of the computation. In general, security has to do with all the things that might go wrong during the execution of a process. These behaviors of interest can be modeled with specific, secure PA or even with general purpose ones. As a consequence, PA used in this field require particular attention for understanding which security concerns they deal with, as explained below.

1. Tables and figures highlighting our results are omitted in this abstract and will be included in the poster above.

3. Secure PA

We now focus on PA that have been specifically proposed for tackling security aspects. 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 [11].

Here we put forward a classification based on the following three areas.

- ⚙ Secure development, i.e., PA supporting the secure design, implementation and execution of a system.
- 🕵 Threat modeling, i.e., PA used for modeling and analyzing the behavior of an attacker and her strategies.
- 🐞 Vulnerability assessment, i.e., methods employing PA 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, to the best of our knowledge, we can measure for a any given PA and security area.

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

Below we discuss some observations we consider more interesting.

Secure Development According to the considered literature, most PA-based proposals focus on secure design and development. According to our analysis, for each considered PA, ⚙ ¿ 0. However, when only considering security PA (🔒), the trend changes significantly.

Secure PA Secure PA are usually employed for modeling threats and identifying vulnerabilities. Interestingly, SPA with its extensions and especially applied π-calculus with its extensions are used for the secure development of systems.

4. PA for Secure Development

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, PA have often been proposed as a design formalism. Also, their formal semantics permits to carry out verification procedures which are not natively supported by other design languages, e.g., UML [7] and BPMN [26]. Often, formal verification occurs via model checking [10].

We reports the adoption of PA in DLC w.r.t. some major application domains, specifying whether there exists at least one implementation among PA-based tools.

We consider the following DLC macro-phases:

2. Cliques are computed by considering the co-authoring relation induced by the literature considered in this poster.

- *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.

To the best of our knowledge, we conclude the following use of PA related to DLC phases:

No usage for planning. No author proposes PA 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 PA during design, implementation and testing. This is somehow expected since PA 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, PA 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 PA provide a theoretical background for policy enforcement. As a matter of fact, some authors [1], [4], [5], [15], [14] 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.

Types of PA When only the Design phase is studied, the PA employed are frequently new, generated for the article's purpose. While, when the Design, Implementation, and Testing phases (D-I-T) are considered together, the PA used are typically those associated with a tool. An ad hoc PA could better model the architecture under consideration, but at the same time, it could be challenging to verify the model's correctness. As a result, sometimes authors translate a PA or a programming language into another PA to take advantage of a model checker. For example, on [20], CHP is translated into LOTOS in order to allow the application of CADP; Ferrara [13] also shows a translation from BPEL into LOTOS for the same reason.

5. Related Work

Some authors revised the history of PA and the relevant application domains. For instance, Baeten [3] surveys PA in a 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 [9] approach from a historical point of view to CSP and in particular to the FDR tool. Wang presents ATCP and its variants in [25] and [24], considering cryptographic properties, abstraction and introducing guards. In the first article, he uses ATCP to analyze several protocols, while in the second one, ATCP is used to model Map-Reduce, Google File System, cloud resource management, Web Service Composition, and QoS-aware Web Service orchestration. Aldini et al. [2] survey the application of PA to software architecture, emphasizing on component-oriented modeling. Beek et al. [21] compare formal methods used on web service composition considering three features: connectivity, correctness, and quality of services. In the same domain Eddine [12] compares the different approaches, among which are PA, to design and implement Web services focusing on choreography and orchestration. Tuan Anh et al. [22] look into the issues surrounding the security and privacy of the Internet of Mobile Things utilizing PA, with a focus on the mobile PA π-calculus. Wan et al. [23] survey composition mechanisms and then models for cyber-physical systems, concluding that "CPS development must be supported from the design phase by process algebras to achieve strong results on correctness,performance, cost and efficiency." Related to protocol specification and verification, Ryan et al. [18] model and analyze protocols and properties through CSP, using FDR and Casper tools. Ryan and Smyth [17], present the applied pi-calculus, in which areas it was used, and how to use it to model protocols and properties. Wideł et al. [27] investigate the different generation and analysis approaches for Attack Trees; among the formal methods studied are PA, while among the analysis methods is the tool UPPAAL and its variants.

References

[1] Kamel Adi, Lamia Hamza, and Liviu Pene. Automatic security policy enforcement in computer systems. *computers & security*, 73:156–171, 2018.

[2] Alessandro Aldini, Marco Bernardo, and Flavio Corradini. *A process algebraic approach to software architecture design*. Springer Science & Business Media, 2010.

[3] Jos CM Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[4] David Basin, Samuel J Burri, and Günter Karjoth. Dynamic enforcement of abstract separation of duty constraints. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):1–30, 2012.

[5] David Basin, Samuel J Burri, and Günter Karjoth. Obstruction-free authorization enforcement: Aligning security and business objectives. *Journal of Computer Security*, 22(5):661–698, 2014.

[6] Jan A Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and control*, 60(1-3):109–137, 1984.

[7] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[8] Stephen D Brookes, Charles AR Hoare, and Andrew W Roscoe. A theory of communicating sequential processes. *Journal of the ACM (JACM)*, 31(3):560–599, 1984.

[9] Stephen D Brookes and AW Roscoe. Csp: a practical process algebra. In *Theories of Programming: The Life and Works of Tony Hoare*, pages 187–222. 2021.

[10] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[11] Daniel Deogun, Dan Johnsson, and Daniel Sawano. *Secure by Design*. Manning Publications, 2019.

[12] Meftah Mohammed Charaf Eddine et al. A comparative study of formal approaches for web service oriented architecture. *Netw. Commun. Technol.*, 5(2):15–33, 2020.

[13] Andrea Ferrara. Web services: a process algebra approach. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 242–251, 2004.

[14] Mahjoub Langar, Mohamed Mejri, and Kamel Adi. Formal enforcement of security policies on concurrent systems. *Journal of Symbolic Computation*, 46(9):997–1016, 2011.

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

[16] Robin Milner et al. A calculus of communicating systems. *Springer Verlag*, 1980.

[17] Mark D Ryan and Ben Smyth. Applied pi calculus. In *Formal Models and Techniques for Analyzing Security Protocols*, pages 112–142. Ios Press, 2011.

[18] Peter Ryan, Steve A Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley Professional, 2001.

[19] Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1):5–19, 2003.

[20] Gwen Salaun, Wendelin Serwe, Yvain Thonnart, and 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)*, pages 73–82. IEEE, 2007.

[21] Maurice H Ter Beek, Antonio Bucchiarone, and Stefania Gnesi. Formal methods for service composition. *Annals of Mathematics, Computing & Teleinformatics*, 1(5):1–10, 2007.

[22] Vu Tuan Anh, Pham Quoc Cuong, and Phan Cong Vinh. Context-aware mobility based on π-calculus in internet of thing: A survey. In *Context-Aware Systems and Applications, and Nature of Computation and Communication*, pages 38–46. Springer, 2019.

[23] Kaiyu Wan, Danny Hughes, Ka Lok Man, and Tomas Krilavičius. Composition challenges and approaches for cyber physical systems. In *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, pages 1–7. IEEE, 2010.

[24] Yong Wang. Actors–a process algebra based approach. *arXiv preprint arXiv:2104.05438*, 2021.

[25] Yong Wang. Secure process algebra. *arXiv preprint arXiv:2101.05140*, 2021.

[26] Stephen A White. Introduction to bpmn. *BPTrends*, 2004.

[27] Wojciech Wideł, Maxime Audinot, Barbara Fila, and Sophie Pinchinat. Beyond 2014: Formal methods for attack tree–based security modeling. *ACM Computing Surveys (CSUR)*, 52(4):1–36, 2019.

THE BEAUTY AND THE BEAST

40 YEARS OF PROCESS ALGEBRA AND CYBERSECURITY

Silvia De Francisci, Gabriele Costa, Rocco De Nicola

IMT School for Advanced Studies Lucca

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

Abstract

Recently, process algebras were greatly influential in many fields, including cybersecurity. One of the main reasons for their success is their compact, yet expressive and flexible syntax, which allows to model the relevant aspects of computation while abstracting away the secondary ones. Although most authors acknowledge the importance of process algebras for the security community, it is not trivial to estimate how they are shaped the past and present researches. The goal of this work is to provide a comprehensive outlook of the past and present researches about process algebras and security.

Genealogy of Process Algebras

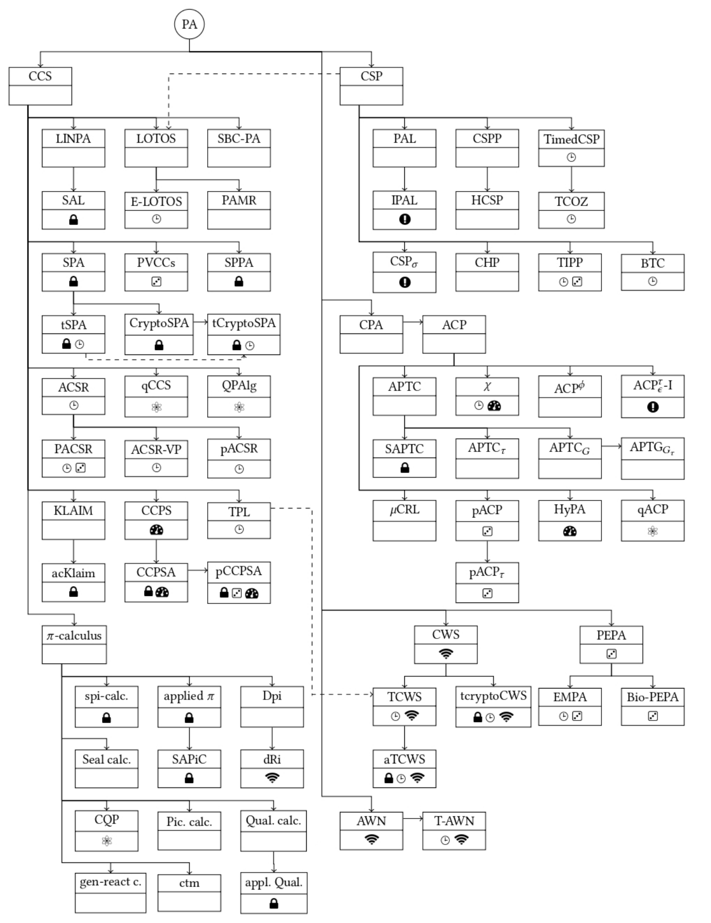

Fig. 1: Process algebras family tree.

🔒 Security aspects	◷ Timed computation	⚛ Quantum computing.
⚂ Stochastic behaviors	ᯤ Wireless networks	⚙ Cyber-physical systems
❗ Imperative statements	$\rightarrow$ Derived from	$\dashrightarrow$ Inspired by

Process Algebras for Security

⚙ Secure development.	0 No literature exists.
🕵 Threat modeling.	$\frac{1}{2}$ Papers belong to a single clique.
🐞 Vulnerability assessment.	1 Papers belong to two or more cliques.

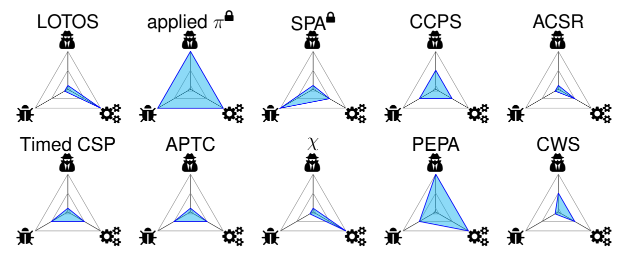

Fig. 2: The PA employment for Secure Development (⚙), Threat Modelling (🕵), and Vulnerability Assessment (🐞).

Figure 2 schematically depicts some Process Algebras (PA) and their impact profile according to the previous metrics. Each spider chart considers not only the PA mentioned but also its genealogy.

Secure Development

In the last decades, Security-by-Design [3] 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, PA have often been proposed as a design formalism. Also, their formal semantics enable verification procedures not natively supported by other design languages, e.g., UML [1] and BPMN [5]. Often, formal verification occurs via model checking [2].

Domain	LC phases	PA	Tool	# ref
Software	▷▶▷▷▷	π-c., Piccola c., Quality c.,c. for SoS		4
	▷▶▶▶▷	ACP, CSP, LOTOS, π-calculus	✓	9
	▷▷▷▶▷	ACSR-VP		1
Hardware	▷▶▶▶▷	ACP, CCS, CHP	✓	4
Management systems	▷▶▷▷▷	PEPA, ctm		3
	▷▶▶▶▷	ACP, CSP	✓	3
Cloud computing	▷▶▶▶▷	π-c., applied π🔒, COWS	✓	3
	▷▷▷▶▷	Cloud c.		1
Web services	▷▶▷▷▷	APTC, SBC-PA		2
	▷▶▶▶▷	CCS, CSP, LOTOS, π-c.	✓	6
	▷▷▷▶▷	applied π🔒	✓	1
Cyber-Physical systems	▷▶▷▷▷	χ, CCPS, IoT-c.		4
	▷▶▶▶▷	ACP, χ, IoT-LySa, Time-Space π-c.	✓	4
	▷▷▷▷▶	TPL		1
Network	▷▶▷▷▷	dRi, Seal c., TCWS		3
	▷▶▶▶▷	ACP, CSP, AWN, pACSR	✓	4
Policy specification	▷▶▷▷▷	CSP, applied π🔒, π-c., tSPA🔒, ACCRP		8
	▷▶▶▶▷	ACP, CSP, ACP$^\tau_\epsilon$-I, SPA🔒, SAPiC🔒	✓	7
	▷▷▷▶▷	CSP	✓	1
	▷▷▷▶▶	SPA🔒		1
	▷▷▷▷▶	CSP, ACP$^\phi$, SPA🔒		5
Quantum	▷▶▶▶▷	CQP	✓	2

Phases:

Planning › Design › Implementation › Testing › Maintenance

PA related to DLC

Usage: No Usage. During the planning phase there is no information about the modules that will constitute the system to be implemented.
Widely Used. PA are particularly suitable for designing a system and its components. Their formal semantics lay the foundation for refinement methods, useful at implementation time. Also, at testing time, model checkers' counterexamples can be converted to test cases.
Almost No. The PA' use in the maintenance phase is mostly related to runtime enforcement, particularly in policy specification. PA provide a theoretical background for policy enforcement.
Types: When only the design phase is studied, the PA employed are frequently new, generated for the purpose. When considering the design, implementation, and testing phases all at once, the PA used are typically those associated with a tool. While an ad hoc PA could better model the architecture under consideration, verifying the model's validity could be challenging. As a result, sometimes authors translate a PA or a programming language into another PA to take advantage of a model checker. For example, Ferrara [4] shows a translation from BPEL into LOTOS that allows the use of CADP.

Secure PA

Secure PA are not commonly used for DLC. Since these PA are formulated to model security threats, they put a strong emphasis on the attacker, i.e., *Threat modeling* and *Vulnerability assessment*). However, in DLC the main focus is on avoiding design and implementation flaws. Thus, the attacker role is often marginal.

References

[1] Grady Booch. *The unified modeling language user guide.* Pearson Education India, 2005.
[2] Edmund M Clarke Jr et al. *Model checking.* MIT press, 2018.
[3] Daniel Deogun, Dan Johnsson, and Daniel Sawano. *Secure by Design.* Manning Publications, 2019.
[4] Andrea Ferrara. "Web services: a process algebra approach". In: *Proceedings of the 2nd international conference on Service oriented computing.* 2004, pp. 242–251.
[5] Stephen A White. "Introduction to BPMN". In: *BPTrends* (2004).