

Information Flow-Based Quantification of Timing-Based Software Systems

Fatma Chebbi*

KASTEL Security Research Labs
Karlsruhe Institut of Technology
Karlsruhe, Baden-Württemberg, Germany
fatma.chebbi@kit.edu

Abstract

With the growing volume of data stored in software systems, protecting confidential information has become increasingly critical. Consequently, ensuring information secrecy should be addressed throughout the software development lifecycle. While existing approaches support confidentiality analysis at design time and runtime, practical tools for detecting and evaluating side channels, among others timing channels, lack maturity to be easily used by software developers.

Because side channels are often not explicitly represented in system artifacts, they go unnoticed even after deployment, posing a serious threat to sensitive information. In my thesis, I intend to present an approach for integrating timing channel analysis into different stages of the software system's development. The goal is to bridge theoretical foundations with practical tools that developers can use to assess and improve system confidentiality.

To support flexible decision-making, particularly in case a trade-off between confidentiality and other properties are necessary, the approach will include quantitative metrics. It will also enable modular analysis, allowing results from individual components to be aggregated and interpreted at the system level. Once developed, I will evaluate the approach through one or more case studies.

Keywords

Software Security Analysis, Timing Channels, Quantitative Information Flow, Composability of Security Analysis

ACM Reference Format:

Fatma Chebbi. 2026. Information Flow-Based Quantification of Timing-Based Software Systems. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-Companion '26)*, April 12–18, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3774748.3787670>

1 Introduction

As software systems are increasingly involved in the daily activities of people, these systems need to collect a huge amount of information, a significant portion of which is confidential and should

not be disclosed, e.g., for reasons relating to business or also the privacy protection of individuals².

To this end, several approaches, such as cryptography-based techniques, have been in the center of interest of security researchers [15]. Applying the results of these approaches in a system's development process and integrating them into different phases of the process can help to adjust the analysis to the system's needs. This allows for earlier decisions that can (a) reduce the cost of later modification and (b) improve the security of existing software systems through analysis at later stages of their development. The principal benefit of security analyses being integrated into the development process is that security is not treated as an afterthought but instead, as a main constituent of the development process.

Despite the existence of works aiming to support architecture-based analysis of software systems [4, 14], some implicit channels that may not be known to the developer may be exploited by malicious attackers in order to infer information. For instance, the timed behavior of a system or the probabilistic distribution of its output may depend on the value of its secrets. Hence, for inferences to be made about sensitive information, it may just require observations by an attacker who actually lacks access to the confidential parts of the system.

In this context, information flow analysis techniques have been developed to assess the extent of leakage through such channels [2, 16]. In my thesis, I focus on timing channel-related leakages and aim to quantify them in order to enable flexible security assessment. To this end, my approach builds on existing works [6], using timed automata as models for the system's timed behavior. These models may be constructed manually or they may be automatically extracted from existing code. For the automatic extraction, I reuse existing works [3, 6] and extend them with model transformations to build the automata and conduct the analysis.

I also consider the properties to be verified on the constructed automata, as well as the metrics to be extracted. Furthermore, to keep the automata as simple as possible, I aim to analyze modular parts of the program (e.g., methods) and then aggregate the results to assess the confidentiality of larger components or the entire system. In this case, maintaining the accuracy of the result can be challenging.

The contributions of my work are expected to be the following.

- (C1): The extension of an existing approach such as Liva et al.'s [9] to derive time-based models from code and enable the timing channel analysis for several development phases.

*Advisor: Anne Koziol¹
PhD Year: 2 out of expected 5



This work is licensed under a Creative Commons Attribution 4.0 International License. *ICSE-Companion '26, Rio de Janeiro, Brazil*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2296-7/26/04
<https://doi.org/10.1145/3774748.3787670>

¹https://mcse.kastel.kit.edu/staff_anne_koziol.php, accessed on 18 December, 2025

²<https://doi.org/10.6028/NIST.SP.1800-29>, accessed on November 14, 2025

- (C2): Extension and adjustment of the model constructed in C1 as a preparation for the subsequent analysis steps.
- (C3): Definition of properties that should be fulfilled by the extended model's version in C2 allowing to infer conclusions about the confidentiality of the software system under study.
- (C4): Definition of extractable metrics (e.g., probability of the model in C2 fulfilling the properties defined in C3) and establishing a relationship with state-of-the-art information flow quantification metrics (e.g., the information leakage).
- (C5): Providing an aggregation of the results obtained after analyzing the sub-modules in order to derive statements about the whole system's security.

This document is organized as follows: Section 2, presents a summary of concepts relevant to this thesis. Section 3 describes the planned approach, distinguishing a constructive and an analytical part. Section 4 presents the planned evaluation.

2 Background

Since my work focuses on information flow analysis, I introduce basics of qualitative information flow in Section 2.1. Then, I talk about quantitative information flow in Section 2.2. Finally, Section 2.3 defines briefly timed automata as a formalism to model a system's timed behavior.

2.1 Information Flow Analysis

The security analysis of software systems is divided into several areas, among which we can find the analysis of data flow and information flow. Despite their commonalities, information flow analysis practices tend to deliver more sound results but less practically applicable methods than data flow approaches [4]. In the area of information flow, system analyzers are concerned with verifying whether the propagation of information within a software system adheres to the requirements specified as security policies [7]. The formulation of these policies requires the definition of different confidentiality levels within the systems' domain, to which its different parts can be assigned.

The information flow between these subdomains occurs through different types of channels, which can be assigned to two main categories: The first is the overt channels that were explicitly defined during software development, such as component interfaces. In other words, such channels are intended for information exchange. The second type is that of covert channels that may be overlooked by the system developers but are used by malicious users to infer confidential information. Common information flow channels are probabilistic, power, or timing channels [13], on which I focus in my work.

Ensuring the compliance with security policies can be achieved by defining formally verifiable properties, for example, by means of model checking or language-based techniques [7]. One of the well-known security properties is non-interference [7, 8]. It implies that the observation of an attacker with low privileges does not allow him to infer confidential information about the system. This means that the observed behavior is independent of the confidential input.

2.2 Quantitative Information Flow Analysis

Qualitative information flow analysis can be too rigid, as it only allows recognizing whether unwanted information leakage exists in the system, without considering its extent or impact on the system in different security scenarios. To remedy these problems, different researchers proposed quantification approaches. For instance, Clarkson et al. [5] based their work on the notion of belief from the perspective of an attacker, representing his knowledge about the distribution of secrets within the system. The authors measure the accuracy of the attacker's belief compared to the secret distribution. In addition, work has been done on developing information-theoretic metrics, such as Alvim et al. [2] or Smith [16, 17]. Their works focus on quantifying information leakage and formalizing the notion of channels representing the worst-case leakage. Depending on the attacker model and the attack scenarios, the authors employed different entropies for the leakage's computation. Although the results can be helpful for the assessment of information leakage, most of these works are theoretical or offer algorithms that cannot be easily integrated into a software development process. Similar to the intention of my work, Vasilikos [11] used timed automata to quantify the information leakage occurring due to timing channels. In his work, Vasilikos tried to calculate the information leakage based on the information theoretic metrics provided in literature [2]. One difficulty the author pointed out, was a scalability issue, which they attribute, among other factors, to the *state-explosion* problem. Vasilikos mentions Statistical Model Checking (SMC) as an interesting possibility to be investigated in future works.

2.3 Timed Automata as a Formalism for Information Flow Analysis

Timed automata can be seen as an extension of automata with time constraints using numerical variables, whose values increase with the progress of time and can help make statements about the duration of an execution. These variables are known as clocks [1], or if their values are defined in a continuous interval, real-time clocks.

3 Approach Description

In this section, I describe the planned approach, distinguishing between a constructive and an analytical part.

3.1 Constructive Part

The constructive part focuses on building a formal model that captures the timed behavior of the software system. This model serves as the basis for the subsequent analysis. To achieve this, I reuse existing metamodels, such as Java Model Parser and Printer (JaMoPP) [3] and Palladio Component Model (PCM) [12], and implement exogenous model transformations to build a Java model (instantiating JaMoPP's metamodel) and then PCM model. As depicted in Figure 1, the Java model is directly parsed from the code (first exogenous transformation) and is used to build the corresponding timed automaton as well as the PCM model. The PCM model extends the timed automaton by missing information.

The transformation from the Java model to a timed automaton is currently being developed, based on the semantics proposed by Liva et al. [9]. For that reason, only explicit information about the

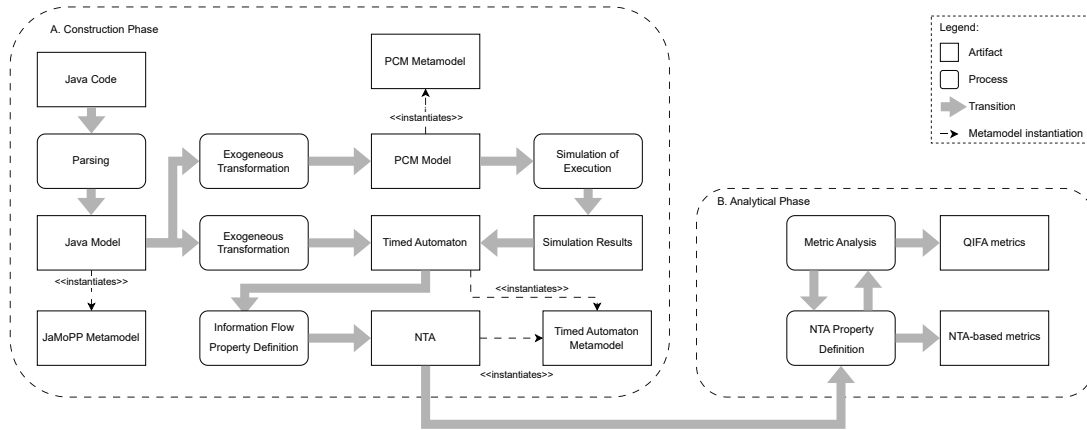


Figure 1: Process Overview

execution duration is considered for the model’s construction. I intend to address this shortcoming by performing simulations with the PCM model. The transformation from a JaMoPP model to a PCM model is achieved using the Continuous Integration of architectural Performance Models (CIPM) approach [10]. The resulting model corresponds to Contribution C2. It represents the necessary input to construct a Network of Timed Automata (NTA), i.e., a set of timed automata interacting among one another through parallel composition. These interactions are intended to facilitate the analysis of the system upon the fulfillment of a selected information flow property. Hence, the second input for the transformation building the NTA is the selection of the information flow property conducted in the *Property Definition* step. The property considered in this work is non-interference [7].

3.2 Analytical Part

In the analytical part of my approach, different properties of the constructed NTA (e.g., the reachability of certain states in the NTA) are defined to extract NTA-based metrics (e.g., the probability of reaching the state). These properties and corresponding metrics are then analyzed and compared with the Quantitative Information Flow Analysis (QIFA) metrics, e.g., non-interference related metrics from the literature. These comparisons should help not only to establish a relationship between the metrics produced by this approach and those found in the QIFA literature, but also to confirm the suitability of the NTA property definition.

4 Evaluation

To evaluate the approach described in Section 3, I plan to apply it to different case studies drawn from different domains. This will ensure that the results are generalizable beyond a single context. A potential threat to the validity is the restriction to Java programs in the implementation. However, the extraction of timed automata should also be possible for other programming languages similarly.

Acknowledgments

This work was funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe. This paper has been edited by our textician, Daniel Shea.

References

- [1] Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235. doi:10.1016/0304-3975(94)90010-8
- [2] Mário S Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. 2020. *The science of quantitative information flow*. Springer.
- [3] Martin Armbruster. 2022. *Parsing and Printing Java 7-15 by Extending an Existing Metamodel*. Technical Report. Tech. rep. July 28.
- [4] Musard Balliu, Daniel Schoepe, and Andrei Sabelfeld. 2017. We Are Family: Relating Information-Flow Trackers. In *Computer Security – ESORICS 2017*, Simon N. Foley, Dieter Gollmann, and Einar Sneekens (Eds.). Springer International Publishing, Cham, 124–145.
- [5] M. R. Clarkson, A. C. Myers, and F. B. Schneider. null. Belief in Information Flow. In *18th IEEE Computer Security Foundations Workshop (CSFW’05)*. IEEE. doi:10.1109/csfw.2005.10
- [6] Christopher Gerking, David Schubert, and Eric Bodden. 2018. Model checking the information flow security of real-time systems. In *Engineering Secure Software and Systems: 10th International Symposium, ESSoS 2018, Paris, France, June 26-27, 2018, Proceedings 10*. Springer, 27–43.
- [7] J. A. Goguen and J. Meseguer. 1982. Security Policies and Security Models. In *1982 IEEE Symposium on Security and Privacy*. 11–11. doi:10.1109/SP.1982.10014
- [8] Joseph A Goguen and José Meseguer. 1984. Unwinding and inference control. In *1984 IEEE Symposium on Security and Privacy*. IEEE, 75–75.
- [9] Giovanni Liva, Muhammad Taimoor Khan, and Martin Pinzger. 2019. Semantics-driven extraction of timed automata from Java programs. *Empirical Software Engineering* 24, 5 (2019), 3114–3150.
- [10] Manar Mazkatli, David Monschein, Martin Armbruster, Robert Heinrich, and Anne Koziolk. 2025. Continuous integration of architectural performance models with parametric dependencies—the CIPM approach. *Automated Software Engineering* 32, 2 (2025), 1–79.
- [11] Panagiotis Vasilikos. 2019. Timed Automata for Security of Real-Time Systems. (2019). <https://orbit.dtu.dk/en/publications/timed-automata-for-security-of-real-time-systems>
- [12] Ralf Reussner, Steffen Becker, Erik Burger, Jens Happe, Michael Hauck, Anne Koziolk, Heiko Koziolk, Klaus Krogmann, and Michael Kuperberg. 2011. The palladio component model. (2011).
- [13] Andrei Sabelfeld and Andrew C Myers. 2003. Language-based information-flow security. *IEEE Journal on selected areas in communications* 21, 1 (2003), 5–19.
- [14] Stephan Seifermann, Robert Heinrich, and Ralf Reussner. 2019. Data-driven software architecture for analyzing confidentiality. In *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 1–10.
- [15] Yifan Shen, Zhaochun Sun, and Tian Zhou. 2021. Survey on Asymmetric Cryptography Algorithms. In *2021 International Conference on Electronic Information*

- Engineering and Computer Science (EIECS)*, 464–469. doi:10.1109/EIECS53707.2021.9588106
- [16] Geoffrey Smith. 2009. On the Foundations of Quantitative Information Flow. In *Foundations of software science and computational structures*, Luca de Alfaro (Ed.). Lecture notes in computer science Theoretical computer science and general issues, Vol. 5504. Springer, Berlin and Heidelberg, 288–302. doi:10.1007/978-3-642-00596-1_21
- [17] Geoffrey Smith. 2015. Recent Developments in Quantitative Information Flow (Invited Tutorial). 23–31 pages. doi:10.1109/LICS.2015.13