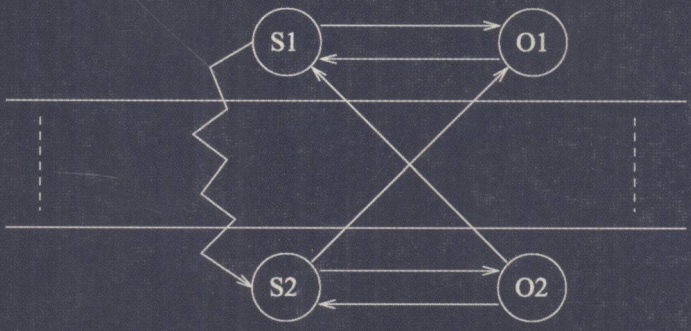
Riccardo Focardi
Roberto Gorrieri (Eds.)

Tutorial

LNCS 2946

Foundations of Security Analysis and Design II

FOSAD 2001/2002 Tutorial Lectures

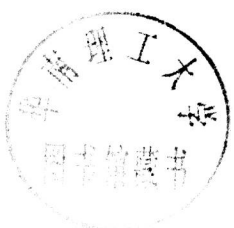


Springer

Riccardo Focardi Roberto Gorrieri (Eds.)

Foundations of Security Analysis and Design II

FOSAD 2001/2002 Tutorial Lectures



Springer

Volume Editors

Riccardo Focardi
Università Ca' Foscari di Venezia, Dipartimento di Informatica
Via Torino 155, 30172 Mestre (Venice), Italy
E-mail: focardi@dsi.unive.it

Roberto Gorrieri
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: gorrieri@cs.unibo.it

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): D.4.6, C.2, K.6.5, K.4, D.3, F.3, E.3

ISSN 0302-9743
ISBN 3-540-20955-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 10985960 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 2946

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

International School on Foundations of Security Analysis and Design

17–29 September 2001, 23–27 September 2002, Bertinoro, Italy

Security is a fast-growing area of computer science, with increasing relevance to real-life applications such as Internet transactions and electronic commerce. Foundations for the analysis and the design of security aspects of these applications are badly needed in order to validate and prove (or guarantee) their correctness. Recently an IFIP Working Group on "Theoretical Foundations of Security Analysis and Design" was established (see http://www.dsi.unive.it/IFIPWG1_7/ for more details) in order to promote research and education in security-related issues.

One of the many initiatives of the IFIP WG 1.7 has been the creation of the "International School on Foundations of Security Analysis and Design" (FOSAD) that is held annually at the Residential Centre of the University of Bologna in Bertinoro, with the goal of disseminating knowledge in this critical area, especially for participants coming from less-favored and non-leading countries. The Residential Center (see http://www.centrocongressibertinoro.it/) is a former convent and episcopal fortress that has been transformed into a modern conference facility with computing services and Internet access.

The first edition of this school (FOSAD 2000) was very successful and the collection of tutorial lectures was published in Springer LNCS volume 2171. This second volume collects some of the tutorials given at the two successive schools (FOSAD 2001 and FOSAD 2002) that attracted many participants from all over the world.

This volume collects six tutorial lectures given at these two schools. More precisely:

- Alessandro Aldini, Mario Bravetti, Alessandra Di Pierro, Roberto Gorrieri, Chris Hankin and Herbert Wiklicky (Two Formal Approaches for Approximating Noninterference Properties);
- Carlo Blundo and Paolo D'Arco (The Key Establishment Problem);
- Michele Bugliesi, Giuseppe Castagna, Silvia Crafa, Riccardo Focardi, Vladimiro Sassone (A Survey of Name-Passing Calculi and Crypto-primitives);
- Roberto Gorrieri, Riccardo Focardi and Fabio Martinelli (Classification of Security Properties – Part II: Network Security);
- Rosario Gennaro (Cryptographic Algorithms for Multimedia Traffic);
- Hanne Riis Nielson, Flemming Nielson and Mikael Buchholtz (Security for Mobility).

We want to thank all the institutions that have supported the initiatives: CNR-IAT, ONR, Università Ca' Foscari di Venezia, Università di Bologna, Progetto MURST "Metodi Formali per la Sicurezza e il Tempo" (MEFISTO), and

EU-FET project MyThS: Models and Types for Security in Mobile Distributed Systems. Moreover, the school was held under the auspices of the European Association for Theoretical Computer Science (EATCS – Italian Chapter), the International Federation for Information Processing (IFIP – WG 1.7), and the European Educational Forum. Finally, we want to warmly thank the local organizers of the school, especially Alessandro Aldini, Andrea Bandini, Chiara Braghin and Elena Della Godenza.

November 2003

Riccardo Focardi
Roberto Gorrieri

Lecture Notes in Computer Science

For information about Vols. 1–2856
please contact your bookseller or Springer-Verlag

Vol. 2857: M.A. Nascimento, E.S. de Moura, A.L. Oliveira (Eds.), String Processing and Information Retrieval. Proceedings, 2003. XI, 379 pages. 2003.

Vol. 2858: A. Veidenbaum, K. Joe, H. Amano, H. Aiso (Eds.), High Performance Computing. Proceedings, 2003. XV, 566 pages. 2003.

Vol. 2859: B. Apolloni, M. Marinaro, R. Tagliaferri (Eds.), Neural Nets. Proceedings, 2003. X, 376 pages. 2003.

Vol. 2860: D. Geist, E. Tronci (Eds.), Correct Hardware Design and Verification Methods. Proceedings, 2003. XII, 426 pages. 2003.

Vol. 2861: C. Bliek, C. Jermann, A. Neumaier (Eds.), Global Optimization and Constraint Satisfaction. Proceedings, 2002. XII, 239 pages. 2003.

Vol. 2862: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. Proceedings, 2003. VII, 269 pages. 2003.

Vol. 2863: P. Stevens, J. Whittle, G. Booch (Eds.), «UML» 2003 – The Unified Modeling Language. Proceedings, 2003. XIV, 415 pages. 2003.

Vol. 2864: A.K. Dey, A. Schmidt, J.F. McCarthy (Eds.), UbiComp 2003: Ubiquitous Computing. Proceedings, 2003. XVII, 368 pages. 2003.

Vol. 2865: S. Pierre, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2866: J. Akiyama, M. Kano (Eds.), Discrete and Computational Geometry. Proceedings, 2002. VIII, 285 pages. 2003.

Vol. 2867: M. Brunner, A. Keller (Eds.), Self-Managing Distributed Systems. Proceedings, 2003. XIII, 274 pages. 2003.

Vol. 2868: P. Perner, R. Brause, H.-G. Holzhütter (Eds.), Medical Data Analysis. Proceedings, 2003. VIII, 127 pages. 2003.

Vol. 2869: A. Yazici, C. Şener (Eds.), Computer and Information Sciences – ISCIS 2003. Proceedings, 2003. XIX, 1110 pages. 2003.

Vol. 2870: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003. Proceedings, 2003. XV, 931 pages. 2003.

Vol. 2871: N. Zhong, Z.W. Raś, S. Tsumoto, E. Suzuki (Eds.), Foundations of Intelligent Systems. Proceedings, 2003. XV, 697 pages. 2003. (Subseries LNAI)

Vol. 2873: J. Lawry, J. Shanahan, A. Ralescu (Eds.), Modelling with Words. XIII, 229 pages. 2003. (Subseries LNAI)

Vol. 2874: C. Priami (Ed.), Global Computing. Proceedings, 2003. XIX, 255 pages. 2003.

Vol. 2875: E. Aarts, R. Collier, E. van Loenen, B. de Ruyter (Eds.), Ambient Intelligence. Proceedings, 2003. XI, 432 pages. 2003.

Vol. 2876: M. Schroeder, G. Wagner (Eds.), Rules and Rule Markup Languages for the Semantic Web. Proceedings, 2003. VII, 173 pages. 2003.

Vol. 2877: T. Böhme, G. Heyer, H. Unger (Eds.), Innovative Internet Community Systems. Proceedings, 2003. VIII, 263 pages. 2003.

Vol. 2878: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part I. Proceedings, 2003. XXXIII, 819 pages. 2003.

Vol. 2879: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part II. Proceedings, 2003. XXXIV, 1003 pages. 2003.

Vol. 2880: H.L. Bodlaender (Ed.), Graph-Theoretic Concepts in Computer Science. Proceedings, 2003. XI, 386 pages. 2003.

Vol. 2881: E. Horlait, T. Magedanz, R.H. Glitho (Eds.), Mobile Agents for Telecommunication Applications. Proceedings, 2003. IX, 297 pages. 2003.

Vol. 2882: D. Veit, Matchmaking in Electronic Markets. XV, 180 pages. 2003. (Subseries LNAI)

Vol. 2883: J. Schaeffer, M. Müller, Y. Björnsson (Eds.), Computers and Games. Proceedings, 2002. XI, 431 pages. 2003.

Vol. 2884: E. Najm, U. Nestmann, P. Stevens (Eds.), Formal Methods for Open Object-Based Distributed Systems. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2885: J.S. Dong, J. Woodcock (Eds.), Formal Methods and Software Engineering. Proceedings, 2003. XI, 683 pages. 2003.

Vol. 2886: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Discrete Geometry for Computer Imagery. Proceedings, 2003. XII, 556 pages. 2003.

Vol. 2887: T. Johansson (Ed.), Fast Software Encryption. Proceedings, 2003. IX, 397 pages. 2003.

Vol. 2888: R. Meersman, Zahir Tari, D.C. Schmidt et al. (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Proceedings, 2003. XXI, 1546 pages. 2003.

Vol. 2889: Robert Meersman, Zahir Tari et al. (Eds.), On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops. Proceedings, 2003. XXI, 1096 pages. 2003.

Vol. 2890: M. Broy, A.V. Zamulin (Eds.), Perspectives of System Informatics. Proceedings, 2003. XV, 572 pages. 2003.

Vol. 2891: J. Lee, M. Barley (Eds.), Intelligent Agents and Multi-Agent Systems. Proceedings, 2003. X, 215 pages. 2003. (Subseries LNAI)

Vol. 2892: F. Dau, The Logic System of Concept Graphs with Negation. XI, 213 pages. 2003. (Subseries LNAI)

Vol. 2893: J.-B. Stefani, I. Demeure, D. Hagimont (Eds.), Distributed Applications and Interoperable Systems. Proceedings, 2003. XIII, 311 pages. 2003.

Vol. 2894: C.S. Laih (Ed.), Advances in Cryptology - ASIACRYPT 2003. Proceedings, 2003. XIII, 543 pages. 2003.

Vol. 2895: A. Ohori (Ed.), Programming Languages and Systems. Proceedings, 2003. XIII, 427 pages. 2003.

Vol. 2896: V.A. Saraswat (Ed.), Advances in Computing Science – ASIAN 2003. Proceedings, 2003. VIII, 305 pages. 2003.

Vol. 2897: O. Balet, G. Subsol, P. Torguet (Eds.), Virtual Storytelling. Proceedings, 2003. XI, 240 pages. 2003.

Vol. 2898: K.G. Paterson (Ed.), Cryptography and Coding. Proceedings, 2003. IX, 385 pages. 2003.

Vol. 2899: G. Ventre, R. Canonico (Eds.), Interactive Multimedia on Next Generation Networks. Proceedings, 2003. XIV, 420 pages. 2003.

Vol. 2900: M. Bidoit, P.D. Mosses, CASL User Manual. XIII, 240 pages. 2004.

Vol. 2901: F. Bry, N. Henze, J. Maluszyński (Eds.), Principles and Practice of Semantic Web Reasoning. Proceedings, 2003. X, 209 pages. 2003.

Vol. 2902: F. Moura Pires, S. Abreu (Eds.), Progress in Artificial Intelligence. Proceedings, 2003. XV, 504 pages. 2003. (Subseries LNAI).

Vol. 2903: T.D. Gedeon, L.C.C. Fung (Eds.), AI 2003: Advances in Artificial Intelligence. Proceedings, 2003. XVI, 1075 pages. 2003. (Subseries LNAI).

Vol. 2904: T. Johansson, S. Maitra (Eds.), Progress in Cryptology – INDOCRYPT 2003. Proceedings, 2003. XI, 431 pages. 2003.

Vol. 2905: A. Sanfeliu, J. Ruiz-Shulcloper (Eds.), Progress in Pattern Recognition, Speech and Image Analysis. Proceedings, 2003. XVII, 693 pages. 2003.

Vol. 2906: T. Ibaraki, N. Katoh, H. Ono (Eds.), Algorithms and Computation. Proceedings, 2003. XVII, 748 pages. 2003.

Vol. 2908: K. Chae, M. Yung (Eds.), Information Security Applications. Proceedings, 2003. XII, 506 pages. 2004.

Vol. 2910: M.E. Orlowska, S. Weerawarana, M.P. Papazoglou, J. Yang (Eds.), Service-Oriented Computing – ICSOC 2003. Proceedings, 2003. XIV, 576 pages. 2003.

Vol. 2911: T.M.T. Sembok, H.B. Zaman, H. Chen, S.R. Urs, S.H.Myaeng (Eds.), Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access. Proceedings, 2003. XX, 703 pages. 2003.

Vol. 2912: G. Liotta (Ed.), Graph Drawing. Proceedings, 2003. XV, 542 pages. 2004.

Vol. 2913: T.M. Pinkston, V.K. Prasanna (Eds.), High Performance Computing – HiPC 2003. Proceedings, 2003. XX, 512 pages. 2003.

Vol. 2914: P.K. Pandya, J. Radhakrishnan (Eds.), FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science. Proceedings, 2003. XIII, 446 pages. 2003.

Vol. 2916: C. Palamidessi (Ed.), Logic Programming. Proceedings, 2003. XII, 520 pages. 2003.

Vol. 2918: S.R. Das, S.K. Das (Eds.), Distributed Computing – IWDC 2003. Proceedings, 2003. XIV, 394 pages. 2003.

Vol. 2919: E. Giunchiglia, A. Tacchella (Eds.), Theory and Applications of Satisfiability Testing. Proceedings, 2003. XI, 530 pages. 2004.

Vol. 2920: H. Karl, A. Willig, A. Wolisz (Eds.), Wireless Sensor Networks. Proceedings, 2004. XIV, 365 pages. 2004.

Vol. 2921: G. Lausen, D. Suciu (Eds.), Database Programming Languages. Proceedings, 2003. X, 279 pages. 2004.

Vol. 2922: F. Dignum (Ed.), Advances in Agent Communication. Proceedings, 2003. X, 403 pages. 2004. (Subseries LNAI).

Vol. 2923: V. Lifschitz, I. Niemelä (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 2004. IX, 365 pages. 2004. (Subseries LNAI).

Vol. 2924: J. Callan, F. Crestani, M. Sanderson (Eds.), Distributed Multimedia Information Retrieval. Proceedings, 2003. XII, 173 pages. 2004.

Vol. 2926: L. van Elst, V. Dignum, A. Abecker (Eds.), Agent-Mediated Knowledge Management. Proceedings, 2003. XI, 428 pages. 2004. (Subseries LNAI).

Vol. 2927: D. Hales, B. Edmonds, E. Norling, J. Rouchier (Eds.), Multi-Agent-Based Simulation III. Proceedings, 2003. X, 209 pages. 2003. (Subseries LNAI).

Vol. 2928: R. Battiti, M. Conti, R. Lo Cigno (Eds.), Wireless On-Demand Network Systems. Proceedings, 2004. XIV, 402 pages. 2004.

Vol. 2929: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments. Proceedings. VII, 273 pages. 2003.

Vol. 2931: A. Petrenko, A. Ulrich (Eds.), Formal Approaches to Software Testing. Proceedings, 2003. VIII, 267 pages. 2004.

Vol. 2932: P. Van Emde Boas, J. Pokorný, M. Bieliková, J. Štuller (Eds.), SOFSEM 2004: Theory and Practice of Computer Science. Proceedings, 2004. XIII, 385 pages. 2004.

Vol. 2933: C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. Proceedings, 2003. VIII, 383 pages. 2004.

Vol. 2935: P. Giorgini, J.P. Müller, J. Odell (Eds.), Agent-Oriented Software Engineering IV. Proceedings, 2003. X, 247 pages. 2004.

Vol. 2937: B. Steffen, G. Levi (Eds.), Verification, Model Checking, and Abstract Interpretation. Proceedings, 2004. XI, 325 pages. 2004.

Vol. 2938: Z. Zhang, C. Zhang, Agent-Based Hybrid Intelligent Systems. XV, 196 pages. 2004. (Subseries LNAI).

Vol. 2944: K. Aberer, M. Koubarakis, V. Kalogeraki (Eds.), Databases, Information Systems, and Peer-to-Peer Computing. Proceedings, 2003. X, 249 pages. 2004.

Vol. 2946: R. Focardi, R. Gorrieri (Eds.), Foundations of Security Analysis and Design II. VII, 267 pages. 2004.

Vol. 2950: N. Jonoska, G. Păun, G. Rozenberg (Eds.), Aspects of Molecular Computing. XI, 391 pages. 2004.

Table of Contents

Two Formal Approaches for Approximating Noninterference Properties

Alessandro Aldini[1], Mario Bravetti[2], Alessandra Di Pierro[3], Roberto Gorrieri[2], Chris Hankin[4], and Herbert Wiklicky[4]

[1] Istituto STI, Università di Urbino *Carlo Bo*, Italy
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
[3] Dipartimento di Informatica, Università di Pisa, Italy
[4] Department of Computing, Imperial College, London, UK

Abstract. The formalisation of security properties for computer systems raises the problem of overcoming also in a formal setting the classical view according to which confidentiality is an absolute property stating the complete absence of any unauthorised disclosure of information. In this paper, we present two formal models in which the notion of noninterference, which is at the basis of a large variety of security properties defined in the recent literature, is approximated. To this aim, the definition of indistinguishability of process behaviour is replaced by a similarity notion, which introduces a quantitative measure ε of the behavioural difference among processes. The first model relies on a programming paradigm called Probabilistic Concurrent Constraint Programming, while the second one is presented in the setting of a probabilistic process algebra. In both models, appropriate notions of distance provide information (the ε) on the security level of the system at hand, in terms of the capability of an external observer of identifying illegal interferences.

1 Introduction

The exact estimation of properties of computer systems is a problem that was widely and successfully attacked via several different formal approaches (see, e.g., [CT02,BHK01,HHHMR94,HS95,Hil96,BDG98,Ber99,BB00,Bra02]). However, a number of factors make the use of approximation techniques necessary to enhance the reliability of "exact" solutions obtained through the formal analysis of the mathematical model of a real, complex system. On the one hand, the confidence we can have in the answers computed by a software tool, which are delivered with certainty, strictly depends on the likelihood of obtaining precise information needed to formally specify the system at hand. On the other hand, even when such information is exact, the results of the mathematical analysis definitely assert that the considered property is or is not satisfied by the system model, while in practice it often happens that a system that approximately behaves like a perfect one is not only acceptable but also the only possible implementation. In practice, in a realistic scenario, a qualitative binary answer to the classical question "does the system satisfy my property?" is too restrictive and, in many cases, not significant.

R. Focardi and R. Gorrieri (Eds.): FOSAD 2001/2002, LNCS 2946, pp. 1–43, 2004.

In this work, we concentrate on formal techniques that employ probabilistic information to give a *quantitative* answer to the kind of question above in the restricted framework of security properties. Indeed, the motivations surveyed above apply also to the problem of verifying the security requirements of real systems. It is well-accepted that the unauthorised disclosure of confidential information cannot be completely avoided in real, complex systems, where typically the interplay between the portion of the system handling secrets and the other components that instead manage public information is more tight than that we expect [RMMG01]. In practice, part of the information flowing through the system cannot be controlled, and a portion of such an unavoidable information flow is sometimes illegal, in the sense that it reveals confidential data to unauthorised users. In such a case, the goal of the designer consists of minimising the illegal information leakage, and, as a consequence, the aim of the analyst must be the provision of an approximated estimation of such an information leakage. As a simple, real example, consider a password-based authentication system, like, e.g., an automatic teller machine. It is trivial to verify that absolute secrecy cannot be guaranteed. In fact, a brute-force based attack has the possibility, even if negligible, of guessing the password, thus violating the secrecy requirements. The analysis of such a kind of system is beyond the scope of possibilistic information flow techniques, which reject programs that do not guarantee absolute secrecy. A more interesting analysis should state that a potential information leakage is not troubling. From a quantitative viewpoint, this corresponds to verify whether or not the probability of detecting a potential illegal information flow is beyond a threshold for which the observer considers the system to be secure "enough". In case of the automatic teller machine, the probability of cracking the system depends on the length of the password and on the number of attempts at disposal of the attacker. By playing on these parameters, the designer can limit to a negligible (as desired) value the probability of accessing the system without knowing the appropriate password.

The approach to information flow analysis we consider is based on the idea of noninterference, originally proposed in [GM82], which states that "one group of users, using a certain set of commands, is noninterfering with another group of users if what the first group does with those commands has no effect on what the second group of users can see". In a security context, the first group is represented by the high-level users, which execute confidential, secret activities, while the second group is given by the low-level users, which instead see public data only. The intuition is that the low-level view of the system to be analysed is not to be altered by the behaviour of the high-level users. If this is the case, we say that any covert channel cannot be set up from the high level to the low level. The verification of the condition above is based on the idea of indistinguishability of behaviours: in order to establish that there is no information flow between a high-level component H and a low-level object L, it is sufficient to check if for any pair of behaviours of the system that differ only in H's behaviour, L's observations cannot distinguish these two behaviours. Depending on the nature of the information flow, an external observer can characterise differ-

ent kinds of interference, due, e.g., to the deterministic, nondeterministic, timed, or probabilistic behaviour of the system. In particular, possibilistic noninterference for nondeterministic programs is weaker than probabilistic noninterference, which helps to reveal those covert channels that arise from the analysis of the frequency of the possible observations in several consecutive executions of the system [Gra90,McL90]. Consider, e.g., a program P that handles pin numbers needed to access the automatic teller machine mentioned above. At a certain point of the execution, the following statement is executed:

$$low\_variable := \mathrm{PIN}_i +^p \mathtt{rand}(9999)$$

where $+^p$ is a probabilistic choice operator that selects the left-hand command (which assigns a secret pin to a public, low variable) with probability p and the right-hand command (which assigns a random value from the range $[0 \ldots 9999]$ to the low variable) with probability $1 - p$. According to a purely nondeterministic behaviour, the program above is secure, since the set of possible outcomes does not change depending on which command will be executed. However, statistical inferences derived from the relative frequency of outcomes of repeated executions of the program allow an external observer to disclose the secret pin with a confidence that depends on the number of executed experiments.

Probabilistic noninterference also offers the means for approximating noninterference properties, by quantifying the real effectiveness of each possibilistic covert channel. More precisely, the key idea of an approach based on probabilistic noninterference is to replace the notion of indistinguishability by an appropriate notion of similarity. For instance, consider again program P and assume that parameter p is a value very close to 0. Obviously, the behaviour of P is not the same as that of the following secure program P':

$$low\_variable := \mathtt{rand}(9999)$$

since if we execute "infinitely often" both programs, then the limit of the frequencies of the possible outcomes allow the observer to distinguish P from P'. However, in practice we have that P and P' are similar and the probability of distinguishing the two programs is still negligible even after a large number n of experiments. In other words, P is considered to be an acceptable approximation of a secure program. As a result of an approach that replaces the restrictive idea of indistinguishability by a relaxed, more realistic notion of similarity, we can accept as secure systems a number of programs that somehow suffer from an information leakage but in practice offer sufficient security guarantees.

In this work, we survey two semantics-based security models (i.e., models that analyse the program behaviour to verify security properties) in which the notion of noninterference is approximated in the sense that they allow for some exactly quantified information leakage. The first one formalises such an approach in the context of a particular probabilistic declarative language, while the second one is based on a probabilistic process algebraic framework.

Language-based formalisms provide a suitable framework for analysing the confidentiality properties of real, complex computing systems. Particularly promising is the use of program semantics and analysis for the specification of

information-flow policies and information-flow controls which guarantee data confidentiality (see, e.g., [SM03] for a survey).

On the other hand, process algebras provide all the main ingredients needed to specify and analyse noninterference properties of computer systems (see, e.g., the several process algebraic approaches described in [FG01]). They are designed with the aim of describing concurrent systems that may interact through the exchange of messages, so that they can be used to naturally express each information flow occurring within the system to be modeled. They deal with both nondeterminism and, as we will focus in this work, probability, so that several kinds of information leakage can be revealed. They also deal in an elegant way with abstraction thanks to the hiding operator, which can be used to specify the observational power of each external observer, depending on the security level of such an observer. Last but not least, there exists a strong, natural similarity between the notion of indistinguishability for processes and semantic equivalences over process algebraic terms.

In the following (Sect. 2), we first introduce the language-based approach by presenting a formalisation of a noninterference property called *confinement* together with its probabilistic and approximated versions in the setting of the probabilistic programming language PCCP (Probabilistic Concurrent Constraint Programming) [DW98a,DW98b]. In this language nondeterminism is completely replaced by probabilistic choice, which makes it possible to develop a statistical interpretation of the approximation of the security property. Moreover, the different role played by variables in imperative and constraint programming hinders a direct translation of previous formalisation of noninterference based on the imperative paradigm into the PCCP setting, where a more appropriate notion must consider process identity rather than variables values.

Then (Sect. 3), we introduce a process algebraic framework for approximating probabilistic noninterference [ABG03]. The basic calculus integrates the characteristics of the classical CCS [Mil89] and CSP [Hoa85] and employs the probabilistic model introduced in [BA03], which is a mixture of the reactive and generative models of probability [GSS95]. Such an approach permits the modeler to specify both nondeterministic behaviour and probabilistic information in the same system model. The behavioural equivalence of process expressions is defined in terms of weak probabilistic bisimulation [BH97], a probabilistic extension of the classical weak bisimulation by Milner [Mil89]. Moreover, the behavioural similarity among processes is defined in terms of a relation called weak probabilistic bisimulation with ε-precision, an approximated version of the weak probabilistic bisimulation, where ε provides information on "how much" two behaviours differ from each other.

Finally (Sect. 4), some conclusions and comments about related work terminate the paper.

2 Language-Based Approach to Noninterference

2.1 Probabilistic Concurrent Constraint Programming

Probabilistic Concurrent Constraint Programming (PCCP) [DW98a,DW98b] is a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [SRP91,SR90]. This can be seen as a kind of process algebra enhanced with a notion of computational state. More precisely, CCP as well as PCCP are based on the notion of a generic *constraint system* $\mathcal{C}$, defined as a cylindric algebraic complete partial order (see [SRP91,dDP95] for more details), which encodes the information ordering. This is referred to as the *entailment* relation $\vdash$ and is sometimes denoted by $\sqsupseteq$. A cylindric constraint system includes constraints of the form $\exists_x c$ (cylindric elements) to model *hiding* of local variables, and constraints of the form d_{xy} (diagonal elements) to model *parameter passing*. The axioms of the constraint system include laws from the theory of cylindric algebras [HMT71] which model the cylindrification operators $\exists_x$ as a kind of first-order existential quantifiers, and the diagonal elements d_{xy} as the equality between x and y.

Table 1. The Syntax of PCCP Agents

$A ::= \mathbf{tell}(c)$	adding a constraint
$\mid \Box_{i=1}^{n} \mathbf{ask}(c_i) \rightarrow p_i : A_i$	probabilistic choice
$\mid \|_{i=1}^{n} q_i : A_i$	prioritised parallelism
$\mid \exists_x A$	hiding, local variables
$\mid p(x)$	procedure call, recursion

In PCCP probability is introduced via a probabilistic choice and a form of probabilistic parallelism. The former replaces the nondeterministic choice of CCP, while the latter replaces the pure nondeterminism in the interleaving semantics of CCP by introducing a probabilistic scheduling. This allows us to implement mechanisms for differentiating the relative advancing speed of a set of agents running in parallel.

The concrete syntax of a PCCP agent A is given in Table 1, where c and c_i are *finite* constraints in $\mathcal{C}$, and p_i and q_i are real numbers representing probabilities. Note that at the syntactic level no restrictions are needed on the values of the numbers p_i and q_i; as explained in the next section, they will be turned into probability distributions by a normalisation process occurring during the computation. The meaning of $p(x)$ is given by a procedure declaration of the form $p(y) : -A$, where y is the formal parameter. We will assume that for each procedure name there is at most one definition in a fixed set of declarations (or program) P.

2.2 Operational Semantics

The operational model of PCCP can be intuitively described as follows. All processes share a common store consisting of the least upper bound, denoted by $\sqcup$, (with respect to the inverse $\sqsubseteq$ of the entailment relation) of all the constraints established up to that moment by means of **tell** actions. These actions allow for communication. Synchronisation is achieved via an **ask** guard which tests whether the store entails a given constraint. The probabilistic choice construct allows for a random selection of one of the different possible synchronisations making the program similar to a *random walk*-like stochastic process. Parts of the store can be made local by means of a *hiding operator* corresponding to a logical existential quantifier.

The operational semantics of PCCP is formally defined in terms of a probabilistic transition system, $(\mathrm{Conf}, \longrightarrow_p)$, where Conf is the set of configurations $\langle A, d\rangle$ representing the state of the system at a certain moment and the transition relation $\longrightarrow_p$ is defined in Table 2. The state of the system is described by the agent A which has still to be executed, and the common store d. The index p in the transition relation indicates the probability of the transition to take place. In order to describe all possible stages of the evolution of agents, in Table 2 we use an extended syntax by introducing an agent **stop** which represents successful termination, and an agent $\exists_x^d A$ which represents the evolution of an agent of the form $\exists_x B$ where d is the local information on x produced during this evolution. The agent $\exists_x B$ can then be seen as the particular case where the local store is empty, that is $d = \mathit{true}$. In the following we will identify all agents of the form $\|_{i=1}^n\ q_i : \mathbf{stop}$ and $\exists_x^d \mathbf{stop}$ with the agent **stop** as they all indicate a successful termination.

The rules of Table 2 are closely related to the ones for nondeterministic CCP, and we refer to [dDP95] for a detailed description. The rules for probabilistic choice and prioritised parallelism involve a normalisation process needed to redistribute the probabilities among those agents A_i which can actually be chosen for execution. Such agents must be enabled (i.e. the corresponding guards $\mathbf{ask}(c_i)$ succeed) or active (i.e. able to make a transition). This means that we have to re-define the probability distribution so that only enabled/active agents have non-zero probabilities and the sum of these probabilities is one. The probability after normalisation is denoted by $\tilde{p}_j$. For example, in rule **R2** the normalised transition probability can be defined for all enabled agents by

$$\tilde{p}_i = \frac{p_i}{\sum_{\vdash c_j} p_j},$$

where the sum $\sum_{\vdash c_j} p_j$ is over all enabled agents. When there are no enabled agents normalisation is not necessary. We treat a zero probability in the same way as a non-entailed guard, i.e. agents with zero probability are not enabled; this guarantees that normalisation never involves a division by a zero value. Analogous considerations apply to the normalisation of active agents in **R3**. It might be interesting to note that there are alternative ways to deal with the situation where $\sum_{\vdash c_j} p_j = 0$ (all enabled agents have probability zero). In

[DW00] normalisation is defined in this case as the assignment of a uniform distribution on the enabled agents; such a normalisation procedure allows, for example, to introduce a quasi-sequential composition.

The meaning of rule **R4** is intuitively explained by saying that the agent $\exists_x^d A$ behaves "almost" like A, with the difference that the variable x which is possibly present in A must be considered local, and that the information present in d has to be taken into account. Thus, if the store which is visible at the external level is c, then the store which is visible internally by A is $d \sqcup (\exists_x c)$. Now, if A is able to make a step, thus reducing itself to A' and transforming the local store into d', what we see from the external point of view is that the agent is transformed into $\exists_x^{d'} A'$, and that the information $\exists_x d$ present in the global store is transformed into $\exists_x d'$.

The semantics of a procedure call $p(x)$, modelled by Rule **R5**, consists in the execution of the agent A defining $p(x)$ with a parameter passing mechanism similar to call-by-reference: the formal parameter x is linked to the actual parameter y in such a way that y inherits the constraints established on x and vice-versa. This is realised in a way to avoid clashes between the formal parameter and occurrences of y in the agent via the operator Δ_y^x defined by:

$$\Delta_y^x A = \begin{cases} \exists_y^{d_{xy}} A & \text{if } x \neq y \\ A & \text{if } x = y. \end{cases}$$

Table 2. The Transition System for PCCP

R1 $\langle \mathbf{tell}(c), d\rangle \longrightarrow_1 \langle \mathbf{stop}, c \sqcup d\rangle$

R2 $\langle []_{i=1}^n \mathbf{ask}(c_i) \to p_i : A_i, d\rangle \longrightarrow_{\tilde{p}_j} \langle A_j, d\rangle \quad j \in [1,n]$ and $d \vdash c_j$

R3 $$\frac{\langle A_j, c\rangle \longrightarrow_p \langle A_j', c'\rangle}{\langle \|_{i=1}^n \ p_i : A_i, c\rangle \longrightarrow_{p\cdot\tilde{p}_j} \langle \|_{j\neq i=1}^n \ p_i : A_i \parallel p_j : A_j', c'\rangle} \ j \in [1,n]$$

R4 $$\frac{\langle A, d \sqcup \exists_x c\rangle \longrightarrow_p \langle A', d'\rangle}{\left\langle \exists_x^d A, c\right\rangle \longrightarrow_p \left\langle \exists_x^{d'} A', c \sqcup \exists_x d'\right\rangle}$$

R5 $\langle p(y), c\rangle \longrightarrow_1 \langle \Delta_y^x A, c\rangle \qquad p(x) :- A \in P$

Observables. We will consider a notion of observables which captures the probabilistic input/output behaviour of a PCCP agent. We will define the observables $\mathcal{O}(A,d)$ of an agent A in store d as a probability distribution on constraints. Formally, this is defined as an element in the real vector space:

$$\mathcal{V}(\mathcal{C}) = \left\{ \sum x_c c \;\middle|\; x_c \in \mathbb{R},\ c \in \mathcal{C} \right\},$$

that is the free vector space obtained as the set of all formal linear combinations of elements in $\mathcal{C}$. The coefficients x_c represent the probability associated to constraints c.