# 07091 Abstracts Collection
# Mobility, Ubiquity and Security
## — Dagstuhl Seminar —

G. Barthe[1], H. Mantel[2], A. Myers[3], P. Müller[4] and A. Sabelfeld[5]

[1] INRIA Sophia Antipolis, FR
`Gilles.Barthe@sophia.inria.fr`
[2] RWTH Aachen, DE
`mantel@cs.rwth-aachen.de`
[3] Cornell University, US
`andru@cs.cornell.edu`
[4] ETH Zürich, CH
`peter.mueller@inf.ethz.ch`
[5] Chalmers University of Technology, Göteborg, SE
`andrei@cs.chalmers.se`

**Abstract.** From 25.02.2007 to 02.03.2007, the Dagstuhl Seminar 07091 "Mobility, Ubiquity and Security" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

**Keywords.** Mobility, confidentiality, integrity, availability, type systems, static analysis, information flow, cryptography, proof-carrying code

## 07091 Executive Summary – Mobility, Ubiquity and Security

Increasing code mobility and ubiquity raises serious concerns about the security of modern computing infrastructures. The focus of this seminar was on securing computing systems by design and by construction.

*Keywords:* Mobility, confidentiality, integrity, availability, type systems, static analysis, information flow, cryptography, proof-carrying code

*Extended Abstract:* http://drops.dagstuhl.de/opus/volltexte/2007/1101

## Gradual Release: Unifying Declassification, Encryption and Key Release Policies

*Aslan Askarov (Chalmers UT - Göteborg, SE)*

Information security has a challenge to address: enabling information-flow controls with expressive information release (or declassification) policies. Existing approaches tend to address some aspects of information release, exposing the other aspects for possible attacks. It is striking that these approaches fall into two mostly separate categories: revelation-based (e.g. information purchase, aggregate computation, moves in a game) and encryption-based declassification (e.g. sending encrypted secrets over an untrusted network, storing passwords). This paper introduces *gradual release*, a policy that unifies declassification, encryption and key release policies. We model the knowledge of an attacker by the sets of possible secret inputs as functions of publicly observable outputs. The essence of gradual release is that knowledge must remain constant between releases. Gradual release turns out to be a powerful foundation for release policies, which we demonstrate by formally connecting revelation-based and encryption-based declassification. Furthermore, we show that gradual release can be provably enforced by security types and effects.

*Keywords:* Information flow, declassification

*Joint work of:* Askarov, Aslan; Sabelfeld, Andrei

## Information Flow, Modularity, and Declassification

*Anindya Banerjee (Kansas State University, US)*

We give, via a relational Hoare-like logic, the specification of an interprocedural and flow sensitive (but termination insensitive) information flow analysis for heap-manipulating programs. Pointer aliasing is ubiquitous in such programs, and can potentially leak confidential information. Thus the logic employs agreement assertions to describe the noninterference property that formalizes confidentiality, and employs region assertions to describe possible aliasing. The logic supports local reasoning about state in the style of separation logic. In the second part of the talk, we show how the logic can be used to formalize some recent proposals for declassification. Parts of this work are joint with Torben Amtoft, Sruthi Bandhakavi, Roberto Giacobazzi, Isabella Mastroeni and David Naumann.

*Keywords:* Declassification, Hoare logic, information flow, separation logic, two-state assertions

## Certificate translation for optimizing compilers

*Gilles Barthe (INRIA Sophia Antipolis, FR)*

Certifying compilation provides trust in a mobile code infrastructure, by generating a formal proof, agreeable by the code consumer, that certifies that the delivered code satisfies appropriate conditions. Since proofs are generated automatically, such properties are restricted to sufficiently basic safety policies. On the other hand, (possibly interactive) program verification aims at proving, a broader set of safety, security, or functionality properties.

We present Certificate Translation, a novel technique that brings the benefits of interactive software verification to the code consumer. It consists of an extension for program transformations that automatically translates a formal proof of functional correctness of programs (in high-level languages) into certificates for executable code.

In this talk, we explain the convenience of the approach, and describe the proof transformations required when specific program optimizations are applied. Emphasis will be made on proof transformations that rely on what we call a Certifying Analyzer, i.e. an extension for a static analyzer that automatically returns a proof for the result of the analysis.

*Keywords:* Languages and compilers, optimization, program transformation, proof-carrying code, static analysis

*Joint work of:* Barthe, Gilles; Kunz, César; Grégoire Benjamin; Rezk, Tamara

*Full Paper:* http://www-sop.inria.fr/.../certtrans-SAS06.pdf

## Integration of a Security Type System into a Program Logic

*Richard Bubel (Chalmers UT - Göteborg, SE)*

Type based systems and program logics are two approaches used in program analysis. The presented work has been performed within the MOBIUS project and aims at integrating both approaches in a common logic framework in order to combine their strengthens. These are automation and efficiency on the type-based system side with the generality and completeness on the program logic side. The presented work emulates the type-based system of Hunt and Sands for analysing secure information flow within a program logic. Therefore an abstraction-based calculus has been developed. It is shown that any derivation of the type-based system can be transformed into a proof of the program logic. The proof is constructive, which allows to deduce that it is not significantly harder to find a proof in the program logic as to find a corresponding

derivation in the type-based system. We outline further how declassification can be treated within the logic framework. A unique logic framework allows also the use of uniform certificates, which is an important property for proof carrying code.

*Keywords:* Type systems, information flow, declassification

*Joint work of:* Bubel, Richard; Hänle, Reiner; Pan, Jing; Rümmer, Philipp; Walter, Dennis

## A Complete Logic of Knowledge and One-Way Computable Terms

*Mads Dam (KTH - Stockholm, SE)*

The combination of epistemic logic and formal cryptography offers a potentially very powerful framework for security protocol verification. We address two main challenges towards such a combination; First, the expressive power, specifically the epistemic modality, needs to receive concrete computational justification. Second, the logic must be shown to be, in some sense, formally tractable. Addressing the first challenge, we provide a generalized Kripke semantics that uses permutations on the underlying domain of cryptographic messages to reflect agents' limited computational power. Using this approach, we obtain logical characterizations of important concepts of knowledge in the security protocol literature, specifically Dolev-Yao style message deduction and static equivalence. Answering the second challenge, we exhibit an axiomatization which is sound and complete relative to the underlying theory of cryptographic terms, and to an omega rule for quantifiers. The axiomatization uses largely standard axioms and rules from first-order modal logic. In addition, there are some novel axioms for the interaction between knowledge and cryptography. Time permitting, we illustrate the logic on examples such as Schaum mix, a Crowds style protocol, and an electronic payments protocol reminiscent of SET.

*Keywords:* Cryptography, epistemic logic, first-order logic, multi-agent systems, static equivalence

*Joint work of:* Cohen, Mika; Dam, Mads

## A Static Approach to Secure Service Composition

*Pierpaolo Degano (Università di Pisa, IT)*

We will present a static approach for studying secure composition of software. We use $\lambda^{req}$, an extension of the $\lambda$-calculus, and we study resource usage analysis and verification.

The calculus $\lambda^{req}$ features dynamic creation of resources, and encloses security-critical code in usage policy framings with a possibly nested, local scope. Additionally, it has primitives for selecting and invoking services that respect given security requirements. The actual runtime behaviour of services is over-approximated by a type and effect system. Types are standard and effects include the actions with possible security concerns, as well as information about which services may be invoked at runtime. These approximations are model-checked to verify policy framings within their scopes. This allows for removing any runtime execution monitor, and for instrumenting the code with local checks whenever these may help. Also, our static analysis can be used to determine the plans driving the selection of those services that match the security requirements on demand.

*Keywords:* History based security, type and effect systems, web services

*Joint work of:* Degano, Pierpaolo; Bartoletti, Massimo; Ferrari, Gian-Luigi; Zunino, Roberto

*See also:* http://www.di.unipi.it/~bartolet/pubs/pubs.html

## JavaScript: Mobility & Ubiquity (Two out of Three Ain't Bad)

*Brendan Eich (MOZILLA - Mountain View, US)*

JavaScript is almost twelve years old and used on at least 500 million web pages and in other wide-scale settings. After a brief summary of its key features, we review its security models starting from 1995 with emphasis on data tainting. Since the renewal of browser competition and "Web 2.0", workloads have stressed browsers' long-standing security models, which are either sound and over-restrictive, or unsound and exploited by attackers. We list six top security problems and focus on three relevant to advanced JavaScript use-cases: high principals viewing low objects, "mashups" in hosted content, and browser extensions that rewrite web pages on the fly. It appears that all of these cases can be secured by hybrid data tainting, with static analysis to taint variables set in implicit flows, and dynamic taint propagation along explicit flows.

*Keywords:* Data tainting, information flow, JavaScript, same origin

## Program Verification, Non-Interference, and Declassification Applied to Privacy in Data Mining

*Amy Felty (University of Ottawa, CA)*

In today's society, people have very little control over what kinds of personal data are collected and stored by various agencies in both the private and public sectors.

Moreover, the ability to infer new knowledge from existing data is increasing rapidly with advances in database and data mining technologies. We describe an approach to addressing this problem that allows individuals to specify constraints on the way their own data is used. We use program correctness methods to allow developers of software that processes personal data to provide assurances that the software meets the specified privacy constraints. Our notion of "privacy correctness" differs from general software correctness in two ways. First, properties of interest are simpler and thus their proofs are generally easier to automate. Second, this kind of correctness is stricter; in addition to showing that a certain relation between input and output is realized, we express constraints on information flow to show that only operations that respect privacy constraints are applied during execution. We consider two approaches. In the first approach, we express programs directly in Coq and state and prove privacy properties as theorems about such programs. The second approach works directly on Java programs. Specifications are written in the Java Modelling Language (JML), and Hoare-style program verification is carried out using the Krakatoa tool, which generates proof obligations in Coq and helps to automate their proofs.

*Keywords:* Data mining, declassification, information flow, non-interference, privacy, program verification

*Joint work of:* Felty, Amy; Matwin, Stan; Capretta, Venanzio; Dufay, Guillaume

## A Framework for Parameterizing Type Systems with Relational Information

*Daniel Hedin (Chalmers UT - Göteborg, SE)*

The precision of information flow analyses improves if certain properties about the analyzed program are known. Examples of such properties are aliasing information, value equality and nil pointer freeness. Recently proposed information flow analyses use an integrated alias analysis to improve their precision. In these cases the alias analysis is built in and cannot easily be replaced, something which makes the analyses hard to extend. We present a framework for type based information flow analyses parametrized over external analyses. We define a common interface between the information flow analysis and the external analysis required to reason about the store and the heap. We prove the correctness of the framework and show how it can embed a number of previously presented analyses.

*Keywords:* Information flow, type systems, static analysis

*Joint work of:* Hedin, Daniel; Gedel, Tobias

## Bytecode Modeling Language

*Marieke Huisman (INRIA Sophia Antipolis, FR)*

We present the Bytecode Modeling Language (BML), the Java bytecode cousin of JML. BML allows the application developer to specify the behaviour of an application in the form of annotations, directly at the level of the bytecode. An extension of the class file format is defined to store the specification directly with the bytecode. This is a first step towards the development of a platform for Proof Carrying Code, where applications come together with their specification and a proof of correctness. BML is designed to be closely related with JML. In particular, JML specifications can be compiled into BML specifications. We briefly discuss the tools that are currently being developed for BML, and that will result in a tool set where an application can be validated throughout its development, both at source code and at bytecode level.

*Keywords:* Specification, bytecode, proof carrying code, verification

*Joint work of:* Burdy, Lilian; Huisman, Marieke; Pavlova, Mariela

*Full Paper:* ftp://ftp-sop.inria.fr/everest/Marieke.Huisman/bml.pdf

## Preserving Privacy in Service Composition Using Information Flow Control

*Dieter Hutter (DFKI Saarbrücken, DE)*

The vision of a landscape of heterogeneous web services deployed as encapsulated business software assets in the Internet is currently becoming a reality as part of the Semantic Web. When pro-active agents handle the context-aware discovery, acquisition, composition, and management of application services and data, ensuring the security of customers' data becomes a principle task.

To dynamically compose its offered service, an agent has to process and spread confidential data to other web services demanding the required degree of security. In this talk we propose a methodology based on type-based information flow to control the privacy of data and their proliferation within a set of communicating web services.

*Keywords:* Semantic web, web services, information flow

## Dependency-graph-based protocol analysis

*Peeter Laud (University of Tartu, EE)*

In a dependency graph, nodes correspond to operations. Edges between nodes connect producers of data to their consumers.

A dependency graph makes explicit, which data is used where; this information is not so clearly presented in other types of program representation, such as abstract syntax trees or control flow graphs. Certain forms of program dependency graphs can also be given semantics, although it is in general more tricky than for more orthodox representations.

Program (or protocol) dependency graphs make certain program transformations convenient, in particular those that correspond to definitions of cryptographic primitives. These transformations do not observably change the semantics of a protocol, but may change it syntactically in such a way that its security becomes more obvious. In our talk, we present a prototype protocol analyser, the transformations that we have implemented, and the insights gained.

*Keywords:* Dependecy graph, protocol analysis, program transformation

*Joint work of:* Laud, Peeter; Tahhirov, Ilja

## Modeling and Enforcing Security & Trust Management Policies (on JVM)

*Fabio Martinelli (CNR - Pisa, IT)*

We present part of our ongoing research for defining an integrated framework for the specification, analysis and enforcement of security and trust in mobile and dynamic scenarios. We aim at showing how the same machinery applied for the formal verification of security protocols may be useful to model and analyze trust management procedures. From another perspective, we show also how the same modeling language may be applied as a local policy for run-time enforcement mechanisms for Java computational services.

*Keywords:* Trust policies, run time enforcement, mobile applications

*Joint work of:* Martinelli, Fabio; Mori, Paolo; Petrocchi, Marinella; Vaccarelli, Anna

## Ensuring Confidentiality, Integrity, and Availability by Construction

*Andrew Myers (Cornell University, US)*

Trustworthy computing systems must provide data confidentiality and data integrity, and must be available. These security properties can be provided by construction, by compiling high-level, security-typed source code into explicitly distributed, security-typed target code. This code transformation provably preserves the confidentiality, integrity, and availability properties of the source. A

key technical contribution is the new target language, which describes distributed computation. In this language, any well-typed program satisfies noninterference properties that ensure confidentiality and integrity. Further, the language supports the distribution and replication of code and data using quorum replication, which enables simultaneous enforcement of integrity and availability. A novel timestamp scheme handles out-of-order accesses by concurrent distributed threads without creating covert channels. Complex security mechanisms are automatically deployed by the compiler and run-time system in response to explicit security policies.

*Keywords:* Information flow, confidentiality, integrity, distributed systems, security, compiler

## Generic Universe Types

*Peter Müller (ETH Zürich, CH)*

Ownership is a powerful concept to structure the object store and to control aliasing and modifications of objects. This talk presents an ownership type system for a Java-like programming language with generic types. Like our earlier Universe type system, Generic Universe Types enforce the owner-as-modifier discipline. This discipline does not restrict aliasing, but requires modifications of an object to be initiated by its owner. This allows owner objects to control state changes of owned objects, for instance, to maintain invariants. Generic Universe Types require a small annotation overhead and provide strong static guarantees. They are the fist type system that combines the owner-as-modifier discipline with type genericity.

*Keywords:* Ownership, universe types

*Joint work of:* Dietl, Werner; Drossopoulou, Sophia; Müller, Peter

*See also:* http://sct.ethz.ch/publications/index.html

## Static Analysis for DRM

*Flemming Nielson (Technical University of Denmark, DK)*

Digital Rights Management is a security property gaining in importance due to commercial interest. It is well known that static analysis can be used to validate a number of more classical security policies, such as discretionary and mandatory access control policies, as well as communication protocols using symmetric and asymmetric cryptography.

We show how to develop a staged Flow Logic for validating the conformance of client software with respect to a Digital Rights Management policy and without imposing any cryptographic protection. Our approach is sufficiently flexible that it extends to fully open systems that can admit new services on the fly.

*Keywords:* Digital Rights Management, static analysis, trusted computing base, distributed services

*Joint work of:* Hansen, René Rydhof; Nielson, Flemming; Riis Nielson, Hanne; Probst, Christian

## A Certified Lightweight Non-Interference Java Bytecode Verifier

*David Pichardie (IRISA - Rennes, FR)*

Non-interference is a semantical condition on programs that guarantees the absence of illicit information flow throughout their execution, and that can be enforced by appropriate information flow type systems. Much of previous work on type systems for non-interference has focused on calculi or high-level programming languages, and existing type systems for low-level languages typically omit objects, exceptions, and method calls, and/or do not prove formally the soundness of the type system. We define an information flow type system for a sequential JVM-like language that includes classes, objects, arrays, exceptions and method calls, and prove that it guarantees non-interference. For increased confidence, we have formalized the proof in the proof assistant Coq; an additional benefit of the formalization is that we have extracted from our proof a certified lightweight bytecode verifier for information flow. Our work provides, to our best knowledge, the first sound and implemented information flow type system for such an expressive fragment of the JVM.

*Keywords:* Non-interference, information flow, type systems, bytecode verifier

## Contextual Modal Logic

*Brigitte Pientka (McGill University - Montreal, CA)*

The constructive modal logic of necessity allows us to distinguish between validity and truth, and has been for example used as a logical foundation for distributed computing and information flow. In this talk, I investigate the consequences of relativizing these concepts to explicitly specified contexts. We obtain contextual modal logic and its type-theoretic analogue. Contextual modal type theory provides an elegant, uniform foundation for understanding and reasoning with open code, and hence has the potential to for example justify type-safe linking of code.

*Keywords:* Type theory, logical frameworks, intuitionistic modal logic

*Full Paper:* http://www.cs.mcgill.ca/~bpientka/papers/tocl.pdf

## A Behavioral Semantics of Object-Oriented Components

*Arnd Poetzsch-Heffter (TU Kaiserslautern, DE)*

Behavioral semantics for software components abstract from implementation details and describe the components' behavior in a representation-independent way. Behavioral semantics provide an important foundation for behavioral substitutability and for behavioral interface specifications. In this paper, we develop a formal behavioral semantics for class-based object-oriented languages with aliasing, subclassing, and dynamic dispatch. The code of an object-oriented component consists of a class and the classes used by it. A component instance is realized by a dynamically evolving set of objects with a clear boundary to the environment. The behavioral semantics is expressed in terms of the messages crossing the boundary. It is defined as an abstraction of an operational semantics based on an ownership-structured heap. We show how the semantics can be used to define substitutability of components without the need of state-based simulation relations. Furthermore, we demonstrate how the semantics can provide a foundation for behavioral interface specification.

*Keywords:* Semantics, object-oriented programming, encapsulation, representation independence

*Joint work of:* Poetzsch-Heffter, Arnd; Schäfer, Jan

*See also:* A representation-independent behavioral semantics for object-oriented components. In Proceedings of the FMOODS 2007, Cyprus.

## Controlling the What and Where in Language-Based Security

*Alexander Reinhard (RWTH Aachen, DE)*

While a rigorous information flow analysis is a key step in obtaining meaningful end-to-end confidentiality guarantees, one must also permit possibilities for declassification. Sabelfeld and Sands categorized the existing approaches to controlling declassification in their overview along four dimensions and according to four prudent principles.

In this article, we propose three novel security conditions for controlling the dimensions where and what, and we explain why these conditions constitute improvements over prior approaches.

The conditions for what and where are compatible to each other and can be applied together for a comprehensive control of declassification.

*Keywords:* Information flow, language-based security, declassification, program security

*Joint work of:* Mantel, Heiko; Reinhard, Alexander

## Cryptographically Sound Implementations for Language-Based Information Flow Security

*Tamara Rezk (INRIA Sophia Antipolis, FR)*

Computational non-interference allows one to specify preservation of confidentiality and integrity properties, using computational hypotheses about the cryptography used in the language.

In this work we propose a sound translation from programs satisfying non-interference, to programs with shared-memory communication and cryptography. To this end, we propose type systems for each language, and prove that the translation is typability preserving.

*Keywords:* Language-based information flow security, computational model of cryptography

## Flow Sensitive Analysis of Security Properties

*Hanne Riis Nielson (Technical University of Denmark, DK)*

In this paper we consider service oriented architectures where many components interact with one another using a wireless network. We are interested in questions like: Can I be sure that I do not get unsolicited information from some service? — unless I give my permission? Can I be sure that information I send to some service never is leaked to another service? — unless I give my permission? We shall develop a static program analysis for the $\pi$-calculus and show how it can be used to give privacy guarantees like the ones requested above. The analysis records how information is flowing through the system and keeps track of, not only the potential configurations of the system, but also the order in which they may be encountered.

*Keywords:* Pi-calculus, information flow, service oriented architectures

*Joint work of:* Riis Nielson, Hanne; Nielson, Flemming

## Closing Internal Timing Channels by Transformation

*Alejandro Russo (Chalmers UT - Göteborg, SE)*

A major difficulty for tracking information flow in multithreaded programs is due to the internal timing covert channel. Information is leaked via this channel when secrets affect timing behavior of a thread, which, via the scheduler, affects the interleaving of assignments to public variables. This channel is particularly dangerous because, in contrast to external timing, the attacker does not need to observe the actual execution time. This paper presents a compositional transformation that closes the internal timing channel for multithreaded programs (or rejects the program if there are symptoms of other flows). The transformation is based on spawning dedicated threads, whenever computation may affect secrets, and carefully synchronizing them. The target language features semaphores, which have not been previously considered in the context of termination-insensitive security.

*Keywords:* Information flow, concurrency, covert channel

*Joint work of:* Russo, Alejandro; Hughes, John; Naumann, David; Sabelfeld, Andrei

*Full Paper:* http://www.cs.chalmers.se/~andrei/asian06.pdf

## Trustworthy Elections

*Peter Ryan (University of Newcastle, UK)*

For centuries, we have taken democratic process for granted and placed trust in the paper ballot approach to casting and counting votes. In reality, the democratic process is one of considerable fragility. This was recognized at the dawn of democracy: the Ancient Greeks devised mechanical devices to try to sidestep the need to place trust in officials.

For over a century, the US has been using technological approaches to recording and counting votes, level machines, punch cards, optical readers, touch screen machines, largely in response to widespread corruption with paper ballots. All have been prey to various scams and forms of corruption or just plain malfunctions and have resulted in serious questions as to the legitimacy of the US 2000 and 2004 presidential elections. In the last few years, the UK has been flirting with alternative voting technologies.

In this talk I will discuss the requirements for voting systems, such as universal verifiability, ballot secrecy and coercion-resistance and the challenges these pose. In particular, I will describe a cryptographic scheme, Prï£¡ ï£¡Voter, which has the seemingly paradoxical property of providing voters with the ability to verify that their vote is accurately counted whilst still ensuring the secrecy of their ballot. Furthermore, this is achieved with minimal trust in the technology or officials.

*Keywords:* Voting systems, ballot privacy, accuracy, universal verifiability, voter verifiability

*Full Paper:* http://www.cs.ncl.ac.uk/research/pubs/trs/papers/988.pdf

## Information Flow Control for Java Based on Path Conditions in Dependence Graphs

*Gregor Snelting (Universität Passau, DE)*

Language-based information flow control (IFC) is a powerful tool to discover security leaks in software. Most current IFC approaches are however based on non-standard type systems. Type-based IFC is elegant, but not precise and can lead to false alarms.

We present a more precise approach to IFC which exploits active research in static program analysis. Our IFC approach is based on path conditions in program dependence graphs (PDGs). PDGs are a sophisticated and powerful analysis device, and today can handle realistic programs in full C or Java. We first recapitulate a theorem connecting the classical notion of noninterference to PDGs.

We then introduce path conditions in Java PDGs. Path conditions are necessary conditions for information flow; today path conditions can be generated and solved for realistic programs. We show how path conditions can produce *witnesses* for security leaks.

The approach has been implemented for full Java and augmented with classical security level lattices. Examples and case studies demonstrate the feasibility and power of the method.

*Keywords:* Information flow control, language-based security, noninterference, program slicing, dependency graph, path conditions, Java

*Joint work of:* Snelting, Gregor; Hammer, Christian; Krinke, Jens

*Full Paper:* http://www.infosun.fim.uni-passau.de/.../issse06.pdf

## Optimality and Condensing of Information Flow through Linear Refinement

*Fausto Spoto (Università di Verona, IT)*

Detecting *information flows* inside a program is useful to check *non-interference* or *independence* of program variables, an important aspect of software security. In this paper we present a new abstract domain C expressing *constancy* of program variables. We then apply Giacobazzi and Scozzari's *linear refinement* to build a domain C→C which contains all input/output dependences between the constancy of program variables. We show that C→C is optimal, in the sense that it cannot be further linearly refined, and condensing, in the sense that a compositional, input-independent static analysis over C → C has the same precision of a non-compositional, input-driven analysis. Moreover, we show that C→C has a *natural* representation in terms of Boolean formulas, which is important since it allows one to use the efficient binary decision diagrams in its implementation.

We then prove that C→C coincides with Genaim, Giacobazzi and Mastroeni's IF domain for information flows and with Amtoft and Banerjee's Independ domain for independence. This lets us extend to IF and Independ the properties that we proved for C→C: optimality, condensing and representation in terms of Boolean formulas. As a secondary result, it lets us conclude that IF and Independ are actually the same abstract domain, although completely different static analyses have been based on them.

*Keywords:* Information flow, abstract interpretation, static analysis, compositionality

## Resource Type Checking for Database Queries

*Ian Stark (University of Edinburgh, UK)*

Use of large databases in scientific research has increased to the point where scientists can test hypotheses, perform research, and make original discoveries entirely within the database. Several projects in medicine, biology, biochemistry, physics and astronomy provide very large repositories (multi-terabyte) of raw data for other scientists to then carry out experiments "inside" the database. The Sloan Digital Sky Survey"SkyServer" is one example of this, with 14TB of image data and spectroscopic analyses freely available for access by any internet user. Many of these projects, as a matter of principle, offer extremely liberal access policies:

SkyServer, for example, accepts and executes arbitrary SQL queries from any remote user without authentication. The only constraints are some fairly generous runtime timeouts, where queries are abandoned after 10min–1hr if they have not completed.

We outline one possible contribution of programming language technology to managing resource and availability issues in this model: specifically, a type and effect system for a nested relational query language that provides asymptotic complexity bounds for query execution time and result size. The goal is that such bounds can form the basis of useful feedback to users about potentially expensive queries, as well as a portable and transparent policy language for resource management in globally-accessible scientific databases.

*Keywords:* SQL types effects resources availability e-Science SkyServer mobility

*Joint work of:* Stark, Ian; Cheney, James; MacKenzie, Kenneth

## A Probabilistic Justification of the Combining Calculus

*Henning Sudbrock (RWTH Aachen, DE)*

When giving a program access to secret information, one must ensure that the program does not leak the secrets to untrusted sinks.

For reducing the complexity of the analysis, one can employ compositional proof techniques. The Combining Calculus allows a compositional analysis while keeping the choice of the proof technique variable at the beginning of a verification. During the verification, adequate proof techniques can be chosen and combined via plugin rules. In this talk we present the combining calculus and how it can be applied to a probabilistic security property.

*Joint work of:* Kraußer, Tina; Mantel, Heiko; Sudbrock, Henning

## Towards Language-Based Cryptographic Proofs

*Santiago Zanella Béguelin (INRIA Sophia Antipolis, FR)*

Cryptographers recognize that their field has reached a crisis of rigor where it is no longer viable to construct cryptographic proofs by hand, nor even verify them. Of all cryptographic proofs techniques, a technique known as the game-playing technique is particularly amenable to automated verification since it can be recast in terms of transformation of probabilistic programs. We present a language-based framework that assists the construction and automated verification of cryptographic proofs as sequences of games. Unsurprisingly, most of the game-rewriting transformations used by cryptographers in their proofs happen to be common compiler optimizations that can be entirely automated (e.g. dead-code elimination, constant propagation, code-motion). In addition, the framework supports the application and verification of domain specific and non-semantic preserving transformations, and therefore provides a way to machine-check proofs as a whole.

*Keywords:* Automated verification, cryptographic proofs, language-based security, compiler optimizations

*Joint work of:* Zanella Béguelin, Santiago; Barthe, Gilles; Grégoire, Benjamin; Janvier, Romain

## Combining Access Control and Information Flow in DCC

*Steve Zdancewic (University of Pennsylvania, US)*

The Dependency Core Calculus (DCC) was introduced in 1999 by Abadi, Banerjee, Heintze, and Riecke to provide a framework for studying a variety of dependency analyses such as information flow, slicing, and binding time. The key construct in DCC is an indexed family of monads, where the indices are security levels and a type like “$T_l$ int” can be interpreted as "an integer value with security level l". In 2006, Abadi discovered a remarkable connection between DCC and authorization logics, which are used to express access-control policies. The insight is that the type "$T_l\ \phi$" can be read as "l says $\phi$". Here, l is treated as a principal rather than a security level.

This talk will explain these two interpretations of DCC and describe some preliminary work on trying to combine them in a language that can express both information flow and access control policies.

*Keywords:* Access control, information flow, Dependency Core Calculus