

Dilsun Kaynar

Research Statement

Overview

My research interests lie within the broad area of distributed computing. It is well-known that distributed computing is intrinsically complex due to factors such as concurrency, interaction, timing uncertainty, failures, and sharing of data and other resources. The benefits offered by distributed computing are so compelling that despite all the challenges we have been building distributed systems and applications that provide a wide range of functionalities in a wide range of domains. The guiding principle of my research is that it is important and necessary to build systems that perform well in practice and that meet users' expectations for the most part, but it is not always sufficient. We should also develop accompanying modeling, verification, and programming frameworks that allow us to make clear statements about why our systems work, under what circumstances we can depend on them, and predict, to a reasonable degree of certainty, how they would behave in environments that cannot be precisely known at the time of design. When we think of systems that claim to provide safety or security, the need for such frameworks becomes even more evident. My research in this area has three main threads that inform one another.

- **Security:** Communication over possibly unreliable channels and sharing of data and other resources are at the heart of distributing computing. In this setting, there is a tension between providing flexible and efficient services and providing security and privacy. It is commonly agreed that distributed system security remains to be an art and system design principles based on sound foundations are needed. My research has the objective of contributing to the foundations of security while ensuring that work on the foundational side leads to principles that can be applied in practice. One component of my research involves formulating notions of security, data privacy, and information flow using conceptual tools from concurrency theory, logics, and programming languages. This work is complemented by developing related analysis frameworks that can be used by system and protocol designers to make rigorous arguments about the properties satisfied by their systems.
- **Timing-dependency in distributed computing:** It is common for distributed systems to make use of real time, or some logical notion of time, to be able to meet their functionality and performance requirements. My research involves the development of methods for stating timing constraints and for analyzing whether timed systems meet their requirements. One component in this line of research is the development of user-friendly software tools that facilitate specifying, validating, testing, and verifying timed system designs.
- **Programming languages:** Dependability and predictability of applications for distributed systems rely heavily on having programming models that use abstractions compatible with our mental model of distributed computing, and program analysis techniques that support establishing correspondences between real implementations and specifications that describe intended behavior. I have previously been interested in programming languages that support code mobility and methods for enforcing certain security guarantees that are of concern in the mobile code setting. I have also led the development of a specification language for timing constraints in distributed system designs and algorithms. My recent interests in this area have shifted towards language-based enforcement of security and privacy policies of the kinds that arise in Web-based technologies and organizational processes such as workflows.

Summary of contributions and recent work

1. Security

Secure Systems: Contemporary secure systems exemplified by security hypervisors, virtual machine monitors, systems utilizing secure co-processors such as the Trusted Platform Module (TPM), and Web browsers are designed to provide subtle security guarantees against powerful adversaries that can use both the network

and the local system interfaces to launch attacks. Existing frameworks for security analysis, however, assume weaker adversaries that use either the network or the local system or some ad hoc combinations of these. The lack of a formal adversary model that encompasses both network and local attacks makes it difficult to make rigorous claims about security properties of systems that are used in practice. In order to address this issue, my coauthors and I developed the Logic for Secure Systems (LS^2). Systems designs in LS^2 are modeled as programs in a concurrent language with constructs for operations on shared memory, machine resets, cryptographic operations, network communication, and dynamically loading and executing unknown code. An adversary is modeled as an arbitrary program in this language running in parallel with the system. We developed a sound proof system that allows reasoning about system models without having to consider adversary actions explicitly, thereby reducing the complexity of reasoning. We used LS^2 to characterize Trusted Computing primitives and proved code integrity and execution integrity properties of two remote attestation protocols of the Trusted Computing Group [1].

This work is a first step towards an interface-based theory of system security. A central idea in this line of work is to view a system in terms of the interfaces it exposes, and an adversary in terms of the interfaces to which it has access. Adopting this view point allows us to develop a more general analysis framework that can accommodate a range of adversary classes, rather than a fixed adversary model. We are refining LS^2 so that we can express different classes of adversaries that can be ordered with respect to their power, as opposed to the fixed powerful adversary model it currently assumes. We aim our framework to ultimately support comparing the security levels of systems based on the adversary classes against which they are proved to be secure.

Privacy: One of my recent interests in the area of security is data privacy. Many modern computer systems store and manipulate large amounts of data, allowing the flow of information between different components in the system. These systems usually purport to operate in accordance with certain privacy policy specifications or guidelines. However, these specifications and guidelines do not always have unambiguous meanings and it may be hard or impossible to enforce them. It is desirable to come up with formal and enforceable definitions of what data privacy means.

Two prominent lines of work in this area focus on two different aspects of data privacy, which are to this day investigated independently from one another. The first line of work focuses on organizational processes composed of multiple communicating agents and defines privacy in terms of the allowable flows of information about individuals. For example, in a healthcare information system, allowing a secretary to see some sensitive parts of a patient's medical record may be considered a violation of privacy while there are no restrictions for doctors. The second line of work focuses on databases that are intended to provide aggregate information about a population, and defines privacy in terms of limitations on what an adversary can infer about an individual by combining what they learn from the database with background knowledge obtained from other sources. For example, if a hospital database publishes the number and age of patients at a hospital having a certain illness, it must also make it hard, by means of some sanitization mechanism, for an adversary to infer that someone whose age they know has a certain illness.

My work on data privacy is aimed at bringing these two lines of work together, by focusing on systems in which information about individuals and databases providing aggregate information co-exist, such as healthcare systems in reality. I am investigating what is a reasonable notion of privacy in this setting that provides a desirable level of privacy without severely undermining the utility of databases, and how we can prove that systems satisfy that definition. My coauthors and I recently completed a paper that formalizes differential privacy, which was originally introduced as an abstract quantitative property for sanitization functions used to answer queries in statistical databases. In our work we modeled databases as interactive probabilistic processes and formulated differential privacy as a property of its traces. We also presented a proof-technique based on unwinding relations [2]. We are currently investigating how the privacy guarantees offered by differential privacy in the context of stand-alone statistical databases can be carried over to systems composed of interacting components, where the database is a part of the system. Since adversaries in these systems can combine what they learn from the database with what they learn through other legitimate channels in the system, we need to consider sophisticated information flow properties in conjunction with database privacy properties.

Protocol analysis: I developed (with Canetti et. al.) the task-PIOA framework for the analysis of cryptographic security protocols [3]. Our motivation for this work was to explore how well the traditional methods for the analysis of randomized and timing-dependent distributed algorithms, which are based on reasoning with invariants, simulation relations, and composition theorems would apply in the setting of cryptographic protocols. Our approach is to directly model the cryptographic primitives with probabilistic automata and to bound the success probability of attacks against a given protocol, rather than to represent the security properties of primitives in an idealized way that involves no error probabilities or computational issues.

In the first phase of this work, which involved developing the basic untimed framework, we focused on the analysis of cryptographic protocols that are designed to be secure against computationally-bounded (polynomial time-bounded) adversaries. The technical challenges included finding an appropriate scheduling mechanism in the setting of cryptographic protocols so that schedulers are not given power to reveal hidden information and coming up with the appropriate abstractions for representing computationally bounded entities. We have also introduced a notion of approximate implementation relation that captures the possible discrepancies between protocol implementations and specifications, and defined a compositional notion of secure emulation that deems a protocol secure provided that the discrepancy between the implementation and specification is negligible in an appropriate computational sense [4]. We have analyzed an Oblivious Transfer protocol using this framework.

In the second phase of this work we explored the analysis of long-lived cryptographic systems such as digital archives, whose lifetimes are not necessarily bounded by a polynomial as is typically assumed in computational protocol analyses. In such systems security must be provided throughout the lifetime of the system even though cryptographic primitives used may have shorter lifetimes. To enable the analysis of such systems, we extended the basic task-PIOA framework with a mechanism for imposing a bound on the number of steps performed per unit real-time and defined a new notion of long-term implementation, where entities may live for an unbounded amount of real time, subject to the condition that only a polynomial amount of work can be done per unit time. We have analyzed a long-lived timestamping protocol using this protocol [5].

2. Timing-dependency in distributed computing

Theory: I developed (with Lynch et. al.) the timed I/O automaton (TIOA) modeling framework, which is a basic mathematical framework to support description and analysis of timed systems [6, 7, 8]. It supports reasoning using invariants, multiple levels of abstraction, and compositional methods that have proved to be useful in reasoning about asynchronous distributed systems.

The Timed I/O automaton framework has been developed based on the assumption that a timed system can be viewed as a special case of a hybrid system in which continuous transformation is limited to internal system components that determine the timing of events. This makes our timed I/O automaton modeling framework more expressive than existing timed automaton models and makes it possible to model complex timing behavior naturally, such as clocks drifting at varying rates. The timed I/O automaton modeling framework also serves as a general semantic framework in which previous results about timed automata and related models can be summarized.

Software tools: I have been involved in the development of a set of software tools – the Tempo Toolset – for the design and analysis of timed systems [9]. The Tempo Language is a high-level specification language that allows description of system designs using the mathematical objects provided by the underlying TIOA modeling framework. In addition to the specification language, the toolset includes a simulator and interfaces to the model checker UPPAAL and the theorem prover PVS. This work gave me many insights about transferring ideas from theory to practical language and tool development. In the process of developing Tempo, a lot of thought went into deciding what modeling idioms are common, what can be sacrificed from expressivity in the interest of the clarity of models, amenability to static analysis, simulation, and automated translation into other formalisms that have their own language constraints and semantics [10].

3. Programming languages

Functional mobile code languages: In my PhD thesis [11], I explored distributed computing with languages that adopt functions as the main programming abstraction and support code mobility through the mobility of functions between remote sites. I argued that having functions at the core of a mobile computation language would support dynamism in a simple and elegant way, and also would allow reasoning about correctness, safety, and security rigorously.

Predicting mobility: I observed that the ability to statically predict which values may become mobile at runtime could serve as a valuable asset for compilers of functional mobile code languages. Compilers could benefit from these predictions in dealing with heterogeneity of network nodes, in providing static profiling tools, and in estimating resource-consumption of programs. For example, once an estimate of mobile values is available, compilers can perform optimizations to minimize transmission overheads.

I designed type systems, in which types not only classify values in the traditional sense but also estimate the identities of potentially mobile values, and the potential invocation sites of functions where applicable [12, 13]. The key idea was to make use of annotations in the code to trace the flow of values through the computation and to rely on the compositional nature of typing rules for functional languages which allows the derivation of the type for an expression from those of its subexpressions. In particular, these type systems were designed for the languages Mobile- λ and rEval- λ , which were inspired by Facile and Programming Language for Active Networks respectively.

Language-based security: The research area of language-based security is concerned with the problem of designing “secure” languages so that legal programs of such languages provably meet a predefined security requirement. Annotated type and effect systems appear to be a useful tool in enforcing security requirements in this fashion. One can design a type and effect system that captures the desirable security property, prove its soundness once and then can rely on the fact that any program that is accepted by the type system is secure according to the predefined criteria.

I investigated the problem of controlling the flow of values in a system where users have different trust levels. I designed a language called Confined- λ , which allows programmers to declare a mobility region for the services and the information they provide. The mobility region of an entity determines the subsystem in which it can flow freely. I designed a type system for this language that guarantees that entities created and manipulated by well-typed programs will remain within their mobility regions at runtime [14]. I also investigated the use of type and effect systems in enforcing a confidentiality property based on noninterference that aims to prevent undesirable, possibly indirect, flows of private information to public channels [15].

Future research

As hardware and software systems become more complex, connected, and extensible, the problem of providing security becomes more challenging. This fact makes it evident that security will continue to be an area of active research for many years to come, both in theory and practice. The state of the art in protocol and secure systems analysis methods suggests that we have made notable progress on applying basic modeling and analysis methods from the distributed computing theory, programming languages, and logics in the context of security. On the other hand, there is much more to be done.

Web security: Newly emerging computing paradigms such as cloud computing and other business models based on Web services utilize the Internet as a part of their distributed computing infrastructure, and the World Wide Web and Web browsers as a means of presenting and accessing the provided resources. Security support in Web applications and Web browsers are therefore central to the security of many new technologies. Web security constitutes a rich and high-impact application area for my research interests. My work on secure systems and protocol analysis is directly relevant to ensuring that browsers, applications, and protocols work as expected. Security analysis of plug-ins for Web browsers, for example, is a short-term project of mine that will exercise our interface-based theory for secure systems, which will likely expose some practical problems in this area and also provide feedback about possible improvements to our techniques. My work on privacy

is directly relevant to studying how privacy policies that are supposed to protect personal data collected via the Web are enforced.

Embedded system security: On the foundational side, a specific future interest of mine in the area of security is incorporating explicit modeling of timing constraints into the analysis of systems and protocols for security, and doing so within a model in which the security of cryptographic primitives is modeled quantitatively, using success probabilities of potential adversaries. I think that having timed and probabilistic models for security protocol analysis will not only facilitate more accurate analysis of protocols and systems that have been subject to study so far but also open the way for rigorously analyzing security of systems that interact with the physical world such as embedded systems.

Timing-dependency arise in protocols in several ways, for example, through the assumptions about message delays and processor speeds, clock synchronization, freshness of messages, and lifetimes of cryptographic primitives and services. The question of reasoning about such uses of time in security protocols cannot simply be reduced to reasoning within one's favorite real-time formalism because a faithful modeling of cryptographic primitives and computational security introduces new subtleties. For example, one needs to take into account that adversaries can interfere with the timing of events in more intricate ways than is assumed in the asynchronous case.

Privacy and programs: Privacy is an aspect of security in distributed systems that has proved to be particularly challenging. On the one hand, there are efforts in developing computational foundations for privacy. On the other hand, there are efforts in the language community to develop policy specification languages that enable precise expression of one's privacy expectations. Ideally, given a program that operates on data and a policy specification, one would like to have high-level reasoning methods (potentially automatable) to conclude that the specification is not violated by the program. Such methods would provide the required assurance only if they are sound with respect to the relevant foundational model of privacy. Development of such methods is precisely what is needed to bridge the gap between foundational work and languages for privacy, and I intend to pursue research in this direction.

In the longer term, I expect the wide availability of multicore processors to have a strong impact on distributed computing by giving rise to new programming models, new types of algorithms, and data structures. This will in turn require us to revisit our approach to modeling and analysis of distributed systems and deal with new challenges in dependability and predictability of applications.

References

- [1] Anupam Datta, Jason Franklin, Deepak Garg, and Dilsun Kaynar. A logic of secure systems and its application to trusted computing. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland 2009)*, pages 221–236, May 2009.
- [2] Michael Tschantz, Anupam Datta, and Dilsun Kaynar. Towards differential privacy for systems: Formal model and proof technique. Under submission.
- [3] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Analyzing security protocols using time-bounded Task-PIOAs. *Discrete Event Dynamic Systems*, 18(1):111–159, 2008.
- [4] Ran Canetti, Ling Cheung, Dilsun Kaynar, Nancy Lynch, and Olivier Pereira. Compositional security for Task-PIOAs. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF)*, pages 125–139. IEEE Computer Society, 2007.
- [5] Ran Canetti, Ling Cheung, Dilsun Kaynar, Nancy Lynch, and Olivier Pereira. Modeling computational security in long-lived systems. In *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR)*, pages 114–130, Toronto, Canada, July 2008. Springer-Verlag.

- [6] Dilsun K. Kaynar, Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. *The Theory of Timed I/O automata*. Morgan and Claypool Publishers, 2006. Synthesis Lectures on Computer Science.
- [7] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. Timed I/O automata: A framework for modeling and analyzing real-time systems. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS)*, pages 166–177. IEEE Computer Society, December 2003.
- [8] Dilsun Kaynar and Nancy Lynch. Decomposing verification of timed i/o automata. In *Proceedings of the Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) Formal Techniques in Real-Time and Fault Tolerant System (FTRTFT)*, volume 3253 of *Lecture Notes in Computer Science*, pages 84–101, Grenoble, France, September 2004. Springer-Verlag.
- [9] Tempo-toolset. Downloads and documentation available at <http://www.veromodo.com/Veromodo/Tempo-Toolset.html>.
- [10] Hongping Lim, Dilsun Kaynar, Nancy Lynch, and Sayan Mitra. Translating timed i/o automata specifications for theorem proving in PVS. In *Proceedings of the International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS)*, volume 3829 of *Lecture Notes in Computer Science*, pages 17–31, Uppsala, Sweden, September 2005. Springer-Verlag.
- [11] Dilsun Kirli. *Mobile computation with functions*. PhD thesis, The University of Edinburgh, 2002. Also published as a book by Kluwer Academic Publishers.
- [12] Dilsun Kirli. A static type system for detecting potentially transmissible functions. In *Proceedings of the 5th Mobile Object Systems Workshop: Programming Languages for Wide Area Networks*, 1999.
- [13] Dilsun Kirli. Distributed call-tracking for security. *Computer Languages, Systems and Structures*, 28:129–154, 2002. Invited paper for the special issue on Computer Languages and Security.
- [14] Dilsun Kirli. Confined mobile functions. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 283–294, Nova Scotia, June 2001. IEEE Computer Society. Currently known as IEEE Computer Security Foundations Symposium (CSF).
- [15] Dilsun Kirli. Secure information flow for mobile functions. In *Proceedings of the Workshop on Issues in the Theory of Security(WITS)*, pages 30–35, 2000.