

Michael Clarkson

Research Statement

January 2, 2009

We, as computer scientists, do not understand the foundations of computer security: *What does it mean for a system to be secure? How can we specify systems' security requirements? How can we gain assurance that systems meet those requirements?* We have not yet developed sufficient systematic knowledge and principles, or theory and practice, to answer these questions. In short, we lack a science of security. Such a science could free us from focusing on the attack *du jour*. Instead, we could predict future attacks and construct systems proven secure against entire classes of attacks.

The long-term goal of my research is to develop the science of security. I want to identify principles and methods for defining security and for building systems that offer security assurance. In my research so far, I have developed theories that leverage formal methods, and I have also built an electronic voting system, *Civitas*, that puts theory into practice. *Civitas* offers strong assurance of the correctness of tallying and of the secrecy of votes. In my theoretical work, I have created a mathematical framework that unifies seemingly disparate kinds of security requirements by enabling the expression of all security policies. Using programming language theory, I have also invented an improved model for quantitative information-flow policies that rectifies anomalies found in previous models. In my future research, I want to improve our understanding of security specification and assurance by developing methods for building secure cryptographic systems.

Research Summary

Foundations of security policies. A scientific framework for security policies should enable all possible policies to be expressed, provide insight into the classification and structure of policies, and suggest how to verify that systems satisfy policies. These issues, especially the first, have been problematic for the frameworks currently in use. My research on **hyperproperties** addresses this problem [1, 2]. Hyperproperties can express all security policies (on systems represented as traces), including information flow and service guarantees. This was a surprising result, because various kinds of security requirements—for example, confidentiality, integrity, and availability—had seemed too different to be unified by a single framework. Hyperproperties also enable classification of security policies in terms of *safety* and *liveness*, which originate from the study of properties of concurrent systems. Safety hyperproperties prohibit bad behaviors of systems, and liveness hyperproperties prescribe good behaviors. This classification enjoys a deep characterization in terms of topology: safety corresponds to closed sets, and liveness to dense sets. The hyperproperties framework also sheds light on verification of security policies, in three ways. First, there is a complete verification technique for safety hyperproperties that have finitely bounded bad behaviors. Second, safety and liveness explain why some information-flow policies are harder to verify than others. Third, hyperproperties clarify the exact class of security policies to which stepwise refinement is applicable, as well as why refinement is problematic for all other policies.

Information-flow security policies usually view systems in black and white, classifying them qualitatively as either completely secure or insecure. But complete security is something that real systems rarely (if ever) offer. Security definitions based on quantitative measurements of information flow could capture this intuition and enable us to compare the security of systems. My research on **quantitative information flow** addresses this problem. I proposed a new model for measurement of the amount of confidential information that leaks to an attacker as he interacts with a program [3, 4]. The key innovation in this model is explicit representation of the

attacker's beliefs. This representation enables compositional reasoning about leakage resulting from sequences of interactions—something not handled by previous models. Another innovation is measurement of leakage using *accuracy* of the attacker's beliefs. Previous models instead used the attacker's uncertainty, which resulted in anomalies—for example, attackers could be deemed to learn no information despite an interaction revealing that the confidential information was not equal to a particular value. Accuracy repairs those anomalies and even enables analysis of misinformation given to attackers by probabilistic programs.

Implementation of secure systems. Systems building is a laboratory in which foundational theories can be tested and in which we learn new, and sometimes unexpected, facts and principles. Systems designers and programmers need such principled, scientific techniques to build secure systems and to convey security assurance. For my own laboratory, I chose to explore building secure electronic voting systems, because they have strong, conflicting requirements: voters must be convinced that their votes are tallied correctly, while the secrecy of those votes must also be maintained—even when someone tries to buy votes or to coerce voters.

I designed and helped to build **Civitas** [5], an electronic voting system, to meet these requirements. Civitas is a remote voting system, meaning that voters can vote from anywhere—not just supervised polling booths. Voters can check that their votes are included in the tally, and anyone can check that the tally includes only authorized votes and that no votes were added, changed, or deleted during tallying. Moreover, voters cannot sell their votes, and voters can cast “fake” votes to appear as if complying with the demands a (remote or physical) coercer. I used principled techniques to design and implement Civitas: The design extends the protocol of Juels et al. [6] to improve its confidentiality, availability, and performance; the cryptographic security proofs for this protocol yield assurance. And the implementation was carried out in Jif, a security-typed language, yielding additional assurance that the code implementing the protocols is itself secure. (Of the 21K LOC in Civitas, about 13K are in Jif, 8K are in Java, and a negligible amount is in C.)

Research Plans

During the next five years, I am eager to work on research problems that will advance the science of security. In my previous work, I did this by addressing the questions about specification and assurance that I raised at the beginning of this statement. In particular, I invented new ways to specify security policies, and I built a cryptographic voting system as a testbed for methods of obtaining security assurance. But the problems of specification and assurance have not been fully solved—for example, when building Civitas in Jif, I could not precisely model the security requirements of cryptographic primitives in Jif's type system. So one very interesting open question is how to assure that cryptographic systems satisfy specified security policies. Answers to this question could advance our understanding of security specification and assurance.

Verifying cryptographic systems. Cryptographic systems implement cryptographic primitives, such as encryption schemes and zero-knowledge proofs. For assurance of a system's security, the design of each primitive should be verified (typically this is done with a pen-and-paper proof), and the implementation of each primitive, as well as the composite system, should be verified against a security policy. Our understanding of how to accomplish both verification tasks is incomplete.

The pen-and-paper proofs that cryptographers construct for cryptographic primitives are hard to check and easy to get wrong. However, such proofs often have a highly stylized structure, usually involving a computational reduction or hybrid argument. So I believe that a special-purpose logic (perhaps similar to existing logics of indistinguishability or protocol composition)

could be developed for formalizing cryptographic proofs, and this logic could then be mechanized in a theorem prover (with which I already have some experience [7]).

I see two approaches to verifying the security of the implementation of cryptographic systems. The first approach is to employ security-by-construction, in which a compiler would generate cryptographic code based on policies expressed by the programmer in source code. The main challenge with this approach is to make it possible to add new cryptographic primitives to the compiler's repertoire without requiring substantial changes to the policy language, the compiler, or the soundness proofs for the approach. The second approach is to employ scalable static analysis, such as type systems. The main challenge with this approach is to enable enforcement of sophisticated security policies, including information flow. Another challenge is to make the analysis relatively complete, so that programmers can always verify correct protocols.

These two verification tasks are not independent. For example, the proofs used to establish the security of a primitive will make assumptions, which become requirements of the policy against which the implementation is verified. Ultimately, the two tasks might be unified: I imagine that a cryptographic scheme could be verified, and that a secure implementation could be extracted from that verification, much like programs can be extracted from correctness proofs in constructive logic.

Secure electronic voting. Electronic voting systems could make an ideal testbed for my future work on verifying cryptographic systems: building a verified voting system would test my new theories and hopefully reveal new principles and problems.

Moreover, secure electronic voting is interesting in and of itself. One issue that I discovered in my work on Civitas is that cryptographic voting protocols often strive to satisfy the strongest possible security requirements, which typically exceed those of polling-place voting systems. Such protocols thus incur additional cost, complexity, and trust requirements—perhaps unnecessarily. Part of this problem is that we do not fully understand the formal security policies that polling-place systems satisfy, nor do we know how to construct electronic systems that satisfy the best such policies. Addressing these issues could be an important step toward finally solving the problem of secure electronic voting. Moreover, we could learn new principles that could be applied to building other systems that must guarantee strong forms of anonymity and correctness.

References

- [1] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proc. IEEE Computer Security Foundations Symposium*, pages 51–65, July 2008. One of three conference papers invited to special (peer-reviewed) issue of *Journal of Computer Security*.
- [2] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*. Submitted for publication, December 2008. Computing and Information Science Technical Report, <http://hdl.handle.net/1813/11660>, Cornell University, December 2008.
- [3] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. In *Proc. IEEE Computer Security Foundations Workshop*, pages 31–45, June 2005. One of three conference papers invited to special (peer-reviewed) issue of *Journal of Computer Security*.
- [4] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*. Accepted for publication, December 2006. Computing and Information Science Technical Report, <http://hdl.handle.net/1813/5766>, Cornell University, March 2007.
- [5] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *Proc. IEEE Symposium on Security and Privacy*, pages 354–368, May 2008.
- [6] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. Workshop on Privacy in the Electronic Society*, pages 61–70, November 2005.
- [7] Denis L. Bueno and Michael R. Clarkson. Hyperproperties: Verification of proofs. Computing and Information Science Technical Report, <http://hdl.handle.net/1813/11153>, Cornell University, July 2008.