Review of **Foundations of Cryptography: Basic Tools**[*] and **Modelling and Analysis of Security Protocols**[†]

Riccardo Pucella

Department of Computer Science
Cornell University

July 14, 2003

Introduction

There are essentially two schools, advocating two general approaches to the problem of reasoning about the security properties of a system. What do I mean by a security property? Examples of these include secrecy (ensuring that only authorized parties get access to a piece of information), integrity (ensuring that messages exchanged between parties are not modified in transit), and authenticity (ensuring that parties can ascertain the origin of messages). Reasoning about the security of a system basically means figuring out whether the security properties of interest hold in the presence of an adversary that attempts to attack the system by manipulating the environment in which the system executes. For instance, the adversary may intercept, modify, and redirect messages, may pose as different parties, etc. In order to prevent the adversary from successfully attacking the system (and thereby access secret information, or corrupting messages without any party noticing, depending on what the security property is guaranteeing should not happen), the system will typically make use of various encryption schemes to encrypt the messages exchanged by the parties, signature schemes to digitally sign messages, and communication protocols indicating, say, the pattern of message exchanges that need to occur between the parties.

As I said, historically, two schools have emerged concerning the analysis of security in such systems. Very roughly speaking, the first school, which is also the oldest, focuses on the underlying mechanisms for providing secure messaging, such as encryption schemes (for instance, DES, AES, Blowfish, RSA, etc), or signature schemes. The central theme in this approach is one of complexity—how difficult is it to "break" the scheme—and reliance on probabilities. Here, the adversary is assumed to possess a certain amount of computational abilities, without going into the details of what exactly those abilities are. For instance, a common assumption is to take the adversary to be able to perform arbitrary probabilistic polynomial-time computations.

The second school has focused not so much on the properties of the underlying schemes, but rather on the way these schemes are used in communication protocols. The kind of analysis done is more combinatorial in nature. For instance, many protocols have flaws that are independent of the security of the underlying encryption schemes. To put it bluntly, no encryption scheme is secure if you stupidly reveal the key used to encrypt messages during a protocol exchange. This is of course an extreme example, but many security protocols fail due to the misuse of perfectly good encryption schemes. To help concentrate on that aspect of security protocols, the approach is to completely abstract away from the encryption or signature schemes, assuming them to be perfect, and rather than allow the adversary to

[*] O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001, 372pp, ISBN 0521791723.
[†] P. Ryan, S. Schneider, *Modelling and Analysis of Security Protocols*, Addison-Wesley, 2001, 300pp, ISBN 0201674718.

break the schemes, only allow him to intercept messages, construct new messages from messages he has intercepted, and encrypt and decrypt data for which he has the corresponding key. While this approach may seem less powerful than the preceding version (because the adversary is so limited), it has the distinct advantage of allowing for automatic verification tools.

I should emphasize that while a priori the focus of the two approaches is different—arguably, the first approach tends to address lower-level questions than the second approach—in recent years there has been an attempt at reconciling these approaches. The difficulties in this project stem from the fact that the approaches represent two completely different philosophies with respect to the meaning of security properties, and exhibit vastly different methodologies and tools. In the interim, let's look at what these two books have to offer us.

Foundations of Cryptography: Basic Tools

Goldreich's book exemplifies the computational-complexity approach to security and cryptography, what I have called the first school. By and large, this approach focuses on the properties of the building blocks of many essential security mechanisms, such as encryption schemes, or signature schemes. It focuses on an adversary essentially defined by its computational abilities.

Chapter 1: Introduction. This chapter gives a very nice overview of the philosophy underlying the computational-complexity approach to cryptography. It also reviews some of the background required for this book, namely probability theory, and computational complexity models (including the probabilistic polynomial time complexity classes, and non-uniform boolean circuits).

Chapter 2: Computational Difficulty. This chapter introduces the core construct of much of cryptography, *one-way functions*. Roughly speaking, a one-way function is a function that is efficient to compute in one direction, but difficult to invert. This ties in with complexity theory in the following way. The existence of one-way functions requires the existence of hard (on average) problems. Furthermore, it should be possible to generate hard instances of those problems, along with enough "auxiliary" information to help solve the instance of the problem if one is given that auxiliary information. Different notions of one-way functions are defined (strong, weak, non-uniform), along with candidates. (Note that the existence of one-way functions is a complexity-theoretic assumption, in the same sense that P≠NP is often taken as a complexity-theoretic assumption.) Saying that a function f is one-way basically says that given a value y, it is difficult to find the pre-image of y under f. However, it may be the case that partial information on the pre-image is easy to compute. To address this issue, the notion of *hard-core predicates* is introduced, which yields more flexibility in specifying the difficulty of inverting one-way functions.

Chapter 3: Pseudorandom Generators. This chapter introduces fundamental concepts in the modern theory of cryptography. Roughly speaking, a *pseudorandom generator* is an efficient (polynomial-time) deterministic algorithm that transforms a short randomly chosen string into a much longer "pseudorandom" string. A pseudorandom string is a string that is *computationally indistinguishable* from a true random string by efficient algorithms. One consequence is that for any string generated by a pseudorandom generator, no efficient algorithm can predict the bit following any given prefix of the string. The first part of the chapter defines computational indistinguishability, then pseudorandom generators. The bulk of the chapter investigates the relationship between pseudorandom generators and one-way functions. Pseudorandom generators exist if and only if one-way functions exist. The construction of pseudorandom generators from a special class of one-way functions is presented. (The derivation for arbitrary one-way functions, much more complex, is sketched.) The generalization of pseudorandom generators to pseudorandom functions is also discussed. Intuitively, a pseudorandom function is a function that cannot be distinguished from a truly random function by any efficient procedure that can sample the function at arbitrary points.

Chapter 4: Zero-Knowledge Proofs. This chapter (which takes up half the book) presents one of the most remarkable devices in complexity theory, zero-knowledge interactive proof systems. Roughly speaking, the setting is one in which one party A proves an assertion to another party B, so that B is convinced of the validity of the assertion, but where B *does not learn anything* beyond the fact that the assertion is true (and its consequences). This has direct applications to security, where different parties may have access to different pieces of information that they may want to share, but without revealing anything beyond the fact that they have that piece of information. Defining zero-knowledge proofs requires introducing the notion of an interactive proof system, a topic with many complexity theoretic applications beyond cryptography. Informally, an interactive proof system consists of two parties, a "prover" whose task is to produce a proof of an assertion, and a "verifier" whose task is to verify the validity of a proof. The parties can interact—for instance, the verifier can interrogate the prover during the verification process. From this, one can define a zero-knowledge interactive proof system, which requires defining the notion of knowledge derived from a proof. Following these definitions, the main result of the chapter is presented: a method for constructing zero-knowledge proofs for every language in NP. In other words, it is possible to construct zero-knowledge interactive proof systems for any given language L in NP, that proves queries about language membership in L. For instance, one can construct a zero-knowledge proof system that answers queries as to whether a graph is 3-colorable, in such a way that the verifier is convinced that the graph is indeed 3-colorable, but does not learn anything beyond that, including how to 3-color the graph. This construction relies on one-way functions (in the form of bit commitment protocols). A discussion of the limitations of zero-knowledge proof systems follows. Some further topics explored in this chapter include:

– proofs of knowledge, where the prover not only asserts the existence of some object—such as a 3-coloring of a graph—by also knowledge of that object;

– computationally sound proofs, which relax some of the correctness requirements of zero-knowledge proofs system;

– constant-round zero-knowledge proofs, where one restricts the number of rounds of interaction between the prover and the verifier to a constant;

– non-interactive zero-knowledge proofs, where the prover and verifier do not closely interact. Rather, there is a single message sent from the prover to the verifier. However, both parties have access to a uniformly chosen random string of bits (chosen by a trusted third party, for instance);

– multi-prover zero-knowledge proofs, a generalization of interactive proof systems where a verifier is allowed to interact with multiple provers.

Opinion. This is a very complete introduction to the basics of modern complexity-theoretic cryptography. It is a solid foundation for understanding much of the current work in this area. Note that this is the first volume of a planned three-volume series. Used by itself, this volume is possibly more suited for a first or second-year graduate course on complexity theory than a course on straight cryptography. This should improve once volume 2 and 3 are out. Volume 2 (which is outlined in an appendix of this book) will apply the ideas of this book to the basic problems of defining encryption schemes, signature schemes, and more general cryptographic protocols. Note that draft chapters of the second volume are available online.

Modelling and Analysis of Security Protocols

Ryan and Schneider's book exemplifies the formal methods approach to security, what I have called the second school. Here, the focus is much more on the combinatorial issues surrounding the use of cryptographic primitives. The idea is that once you have cryptography, and you want to securely exchange information with other parties, then having cryptographic schemes is but a first step. After that, you want to actual *use* these schemes in such a way that you can achieve some higher-level goals, such as mutual authentication, or secret sharing. How can you do that? You need to construct communication protocols. What's interesting is that some of these protocols have problems *irrespectively* of the strength of the cryptographic primitives. More to the point, even if we assume perfect cryptography, there are protocols that can be broken based only on the way the messages are constructed. This is what the formal methods approach takes as a starting point: assume we do have perfect cryptography (so that the adversary will not attempt to “break” the crypto), represent the protocol through one of many notations people have devised for studying communication protocols, and analyze the protocol in the presence of the kind of adversary alluded to above.

Chapter 0: Introduction. This chapter is just an introduction to the problem of reasoning about security protocols, of the kind I described earlier. It also includes a discussion of the kind of properties one may be interested in proving about security protocols.

Chapter 1: An Introduction to CSP. This chapter introduces CSP, Hoare’s Calculus of Sequential Processes, the language in which systems are described in the book. Roughly, the language allows one to define a system as the parallel composition of processes that communicate via shared channels. A key feature of CSP, in the view of the authors, is that the same language used to describe a system can be used to specify properties of that system. Intuitively, a property can be described by giving the “ideal” behavior of the system, by abstracting away the details of the implementation. One can then show that the process representing the implementation of the system “refines” the process representing the property. This notion of refinement is given a formal definition in this chapter, and is central to the whole approach of reasoning about security protocol in CSP.

Chapter 2: Modelling Security Protocols in CSP. In this chapter, CSP is put to the use of describing security protocols. The idea is straightforward: put in parallel processes representing the behavior of all the parties of the system, including any trusted servers. Since security protocols typically rely on cryptographic primitives, we need to define in CSP a suitable datatype for representing the values exchanged by the parties. We also need to add a process representing the adversary. Such a process simply captures the fact that the adversary can intercept and redirect messages, compose new messages from old, and apply encryption and decryption operators.

Chapter 3: Expressing Protocol Goals. This chapter focuses on the problem of expressing security goals in the CSP notation. The goals covered include secrecy goals (that a given value is never revealed to the adversary), authentication goals (verification of a party’s identity), non-repudiation (that no party can deny having sent or received a message), and anonymity (protecting the identity of the parties involved in the protocol). All of these properties are expressed as CSP processes, and a protocol satisfies a property if it refines that property, as described in Chapter 1.

Chapter 4: Overview of FDR. This chapter introduces a tool to automatically check that a process refines another. As we have seen, since we represent properties as CSP processes, this gives a way to check that a process meets a specification. The tool is FDR, a commercial tool from Formal Systems (Europe). The underlying mechanisms of the tool are surveyed in the chapter.

Chapter 5: Casper. While FDR provides a mechanical way to check that a CSP process satisfies a specification itself written as a CSP process, it is time-consuming and error-prone to directly manipulate the full CSP process describing a particular protocol, and the CSP process corresponding to the particular security property to verify. To simplify this verification, a tool called Casper is introduced, that takes as input a script representing a protocol as a sequence of message exchanges, as well as a succinct representation of the property to check, and automatically generates the CSP code corresponding to the protocol and the property, suitable for checking via FDR.

Chapter 6: Encoding Protocols and Intruders for FDR. This chapter is a careful study of the encoding of the protocols and properties performed by Casper. Many optimizations are performed in order to produce code that can be checked efficiently by FDR. These include the modelling of the deductive system used by the adversary to derive new facts from intercepted messages, as well as the treatment of algebraic equivalences between values exchanged by the different parties.

Chapter 7: Theorem Proving. The approach to verifying protocols described in the previous chapters can only deal with systems with finitely many states. This imposes a restriction on, say, the number of parties in a protocol, or the number of simultaneous instances of the protocol being executed in the system. In this chapter, a general proof technique is presented to establish properties even when the system under consideration is not finite. This relies on the notion of a rank function, which can be used as a sort of invariant to be preserved in the analysis of the protocol.

Chapter 8: Simplifying Transformations. The methods introduced in the previous chapters work well for small protocols. For dealing with more involved protocols (including most realistic ones), this chapter examines transformations that can be applied to protocols. These transformations have the property that if an attack is possible in the original protocol, this attack will also be possible in the simplified protocol. Therefore, studying the simplified protocol will not make one miss a flaw of the original protocol. The CyberCash Main Sequence protocol is used as a case study.

Chapter 9: Other Approaches. This chapter compares the CSP approach to other approaches to analyze security protocols. These include BAN logic, the NRL protocol analyzer, Strand Spaces, Paulson's inductive assertions approach, and the spi-calculus, to name but the most popular.

Chapter 10: Prospects and Wider Issues. This chapter concludes the book by describing some issues that remain to be addressed. These include a more realistic treatment of cryptography (hence bringing in complexity-theoretic issues), as well as a discussion of the combinatorial size explosion that occurs when analyzing protocols in the style advocated by the book.

Opinion. This introductory book, suitable for an advanced undergraduate course, gives a good overview of a particular way to apply formal methods to study security protocols. This book is not foundational in any sense of the word. Rather, it shows how to analyze protocols given a particular way to represent protocols (CSP) and a particular tool for verifying properties (FDR). However, in the process, it does present most of the issues common to all formal method approaches to security protocol analysis. Perhaps more importantly, it is to the best of my knowledge the only book on the market that addresses in depth the topic of formal methods applied to security protocol analysis.