Taming the Adversary

Martín Abadi

Bell Labs Research, Lucent Technologies
abadi@lucent.com
www.pa.bell-labs.com/~abadi

Abstract. While there is a great deal of sophistication in modern cryptology, simple (and simplistic) explanations of cryptography remain useful and perhaps necessary. Many of the explanations are informal; others are embodied in formal methods, particularly in formal methods for the analysis of security protocols. This note (intended to accompany a talk at the Crypto 2000 conference) describes some of those explanations. It focuses on simple models of attacks, pointing to partial justifications of these models.

1 Polite Adversaries

Some of the simplest explanations of cryptography rely on analogies with physical objects, such as safes, locks, and sealed envelopes. These explanations are certainly simplistic. Nevertheless, and in spite of the sophistication of modern cryptology, these and other simplifications can be helpful when used appropriately. The simplifications range from informal metaphors to rigorous abstract models, and include frequent omissions of detail and conceptual conflations (e.g., [28]). They commonly appear in descriptions of systems that employ cryptography, in characterizations of attackers, and correspondingly in statements of security properties. They are sometimes deceptive and dangerous. However, certain simplifications can be justified:

– on pragmatic grounds, when the simplifications enable reasoning (even automated reasoning) that leads to better understanding of systems, yielding increased confidence in some cases and the discovery of weaknesses in others;
– on theoretical grounds, when the simplifications do not hide security flaws (for example, when it can be proved that a simple attacker is as powerful as an arbitrary one).

In particular, in the design and study of security protocols, it is typical to adopt models that entail sensible but substantial simplifying restrictions on attackers. (See for example [13] and most of the references below.) In these models, an adversary may perform the same operations as other principals. For example, all principals, including the adversary, may be allowed to send and receive messages. If the protocol relies explicitly on cryptography, all principals may be allowed to perform cryptographic operations; thus, the adversary may generate keys, encrypt, decrypt, sign, verify signatures, and hash. In addition, the adversary may have some capabilities not shared by other principals, for example intercepting messages. On the other hand, the adversary may be subject to limitations that tend to make it like other principals, for example:

- it sends messages of only certain forms (e.g., [6, 18]),
- its actions are in a certain order (e.g., [10, 17]),
- it behaves "semi-honestly", that is, it follows the protocol properly but keeps records of all its intermediate computations (e.g., [14]).

Although an actual attacker need not obey them, such conditions are sometimes sound and often convenient.

An even stronger simplification is implicit in these models, namely that the adversary politely respects the constraints and abstractions built into the models. In effect, the adversary does not operate at a lower-level of abstraction than other principals.

- If other principals treat encrypted messages as indivisible units, and only process them by applying decryption operations, the adversary may proceed in the same way. Since those messages are actually bitstrings, many other transformations are possible on them. The adversary will not apply those transformations.
- Similarly, the adversary may generate keys and nonces in stylized ways, like other principals, but may not use arbitrary calculations on bitstrings for this purpose. We may even reason as though all principals obtained keys and nonces from a central entity that can guarantee their distinctness (like an object allocator in a programming language run-time system).
- More radically, the adversary may create and use secure communication channels only through a high-level interface, without touching the cryptographic implementation of these channels.
- Even detailed, concrete models often miss features that could conceivably be helpful for an actual attacker (real-time delays, power consumptions, temperature variations, perhaps others).

In support of such restrictions, we may sometimes be able to argue that operating at a lower-level of abstraction does not permit more successful attacks. In other words, security against polite adversaries that use a limited, high-level suite of operations should imply security against adversaries that use a broader, lower-level vocabulary (e.g., [1]). For instance, the adversary should gain nothing by using the lower-level vocabulary in trying to guess the keys of other principals, since those keys should be strong. (The next section says a little more about these arguments.) A complementary justification is that the restrictions make it easier to reason about protocols, and that in practice this reasoning remains fruitful because the restrictions do not hide many important subtleties and attacks.

2 From Politeness to Formality

With varying degrees of explicitness, many of these simplifications have been embodied in symbolic algorithms, proof systems, and other formal methods for the analysis of security protocols (e.g., [4, 9, 11–13, 16, 19, 21, 23, 25, 27, 29, 31]). The power of the formal methods is largely due to these simplifications.

In these methods, keys, nonces, and other fresh quantities are typically not defined as ordinary bitstrings. They may even be given a separate type. While all bitstrings can be enumerated by starting from 0 and adding 1 successively, such an enumeration need not cover keys and nonces. Moreover, an adversary that may non-deterministically choose any bitstring may be unable to pick particular keys or nonces. Keys and nonces are introduced by other means, for example by quantification ("for every key K such that the adversary does not have or invent K ...") or by a construct for generating new data ("let K be a new key in ..."). The former approach is common in logical methods (e.g., [29]); the latter, in those based on process calculi such as the pi calculus [26] (e.g., [4, 11]). They both support the separation of keys and nonces from ordinary data.

This separation is an extremely convenient stroke of simplism. Without this separation, it is hard to guarantee that the adversary does not guess keys. At best, we may expect such a guarantee against an adversary of reasonable computational power and only probabilistically. With the separation, we may prevent the adversary from guessing keys without imposing restrictions on the adversary's power to compute on bitstrings, and without mention of computational complexities or probabilities (cf. [7, 15, 32]).

Accordingly, keys and cryptographic operations are manipulated through symbolic rules. These rules do not expose the details of the definitions of cryptographic operations. They reflect only essential properties, for example that decryption can undo encryption. This property is easy to express through an equation, such as $d(e(K, x), K) = x$. The treatment of other properties is sometimes more delicate. For example, suppose that the symbol f represent a one-way function. The one-wayness of f may be modeled, implicitly, by the absence of any sequence of operations that an adversary can use to recover the expression x from the expression $f(x)$.

The set of rules is extensible. For example, it is often possible to incorporate special properties of particular cryptographic functions, such as the commutation of two exponentiations with the same modulus. The awareness of such extensions is quite old—it appears already in Merritt's dissertation [24, page 60]. Nevertheless, we still seem to lack a method for deciding whether a given set of rules captures "enough" properties of an underlying cryptosystem.

More broadly, even if these formal methods are consistent and useful, their account of cryptography is (deliberately) partial; it may not even be always sound. The simplifications undoubtedly imply inaccuracies, perhaps mistakes. Thus, we may wonder whether the separation of keys and nonces from ordinary data does not have any unintended consequences. (In the *Argumentum Ornithologicum* [8], the difference of a fresh quantity from particular integers leads to the conclusion that God exists.)

Some recent research efforts provide limited but rigorous justifications for abstract treatments of cryptography [2, 3, 5, 22, 30] (see also [20]). They establish relations between:

– secure channels, secure message transmission, and other high-level notions,
– formal accounts of cryptography, of the kind discussed in this section, and
– lower-level accounts of cryptography, based on standard concepts of computation on bitstrings (rather than ad hoc concepts of computation on symbolic expressions).

In particular, a formal treatment of encryption is sound with respect to a lower-level computational model [5]. The formal treatment is small but fairly typical: simplistic and symbolic. In the computational model, on the other hand, all keys and other cryptographic data are bitstrings, and adversaries have access to the rich, low-level vocabulary of algorithms on bitstrings. Despite these additional capabilities of the adversaries, the assertions that can be proved formally are also valid in the computational model, not absolutely but with high probability and against adversaries of reasonable computational power (under moderate, meaningful hypotheses). Thus, at least in this case, we obtain a computational foundation for the tame, convenient adversaries of the formal world.

References

1. Martín Abadi. Protection in programming-language translations. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 868–883. Springer-Verlag, July 1998. Also Digital Equipment Corporation Systems Research Center report No. 154, April 1998.
2. Martín Abadi, Cédric Fournet, and Georges Gonthier. Secure implementation of channel abstractions. In *Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 105–116, June 1998.
3. Martín Abadi, Cédric Fournet, and Georges Gonthier. Authentication primitives and their compilation. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 302–315, January 2000.
4. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, January 1999. An extended version appeared as Digital Equipment Corporation Systems Research Center report No. 149, January 1998.
5. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). In *Proceedings of the First IFIP International Conference on Theoretical Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, August 2000. To appear.
6. Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. Technical Report 3915, INRIA, March 2000. Extended abstract to appear in the Proceedings of CONCUR 2000.
7. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 82)*, pages 112–117, 1982.
8. Jorge Luis Borges. Argumentum Ornithologicum. In *Obras completas 1923–1972*, page 787. Emecé Editores, Buenos Aires, 1974.
9. Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989. A preliminary

version appeared as Digital Equipment Corporation Systems Research Center report No. 39, February 1989.

10. Edmund Clarke, Somesh Jha, and Will Marrero. Partial order reductions for security protocol verification. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *Lecture Notes in Computer Science*, pages 503–518. Springer-Verlag, March/April 2000.
11. Mads Dam. Proving trust in systems of second-order processes. In *Proceedings of the 31th Hawaii International Conference on System Sciences*, volume VII, pages 255–264, 1998.
12. Richard A. DeMillo, Nancy A. Lynch, and Michael Merritt. Cryptographic protocols. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 383–400, 1982.
13. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, March 1983.
14. Oded Goldreich. Secure multi-party computation (working draft). On the Web at http://theory.lcs.mit.edu/~oded/frag.html, 1998.
15. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.
16. James W. Gray, III and John McLean. Using temporal logic to specify and verify cryptographic protocols (progress report). In *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pages 108–116, 1995.
17. Joshua D. Guttman and F. Javier Thayer Fábrega. Authentication tests. In *Proceedings 2000 IEEE Symposium on Security and Privacy*, pages 96–109, May 2000.
18. Antti Huima. Efficient infinite-state analysis of security protocols. Presented at the 1999 Workshop on Formal Methods and Security Protocols, 1999.
19. Richard A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, May 1989.
20. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 112–121, 1998.
21. Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.
22. Nancy Lynch. I/O automaton models and proofs for shared-key communication systems. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 14–29, 1999.
23. Catherine Meadows. A system for the specification and analysis of key management protocols. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 182–195, 1991.
24. Michael J. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Institute of Technology, February 1983.
25. Jonathan K. Millen, Sidney C. Clark, and Sheryl B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.
26. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100:1–40 and 41–77, September 1992.

27. John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murϕ. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 141–151, 1997.
28. R. M. Needham. Logic and over-simplification. In *Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 2–3, June 1998.
29. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.
30. Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Cryptographic security of reactive systems (extended abstract). *Electronic Notes in Theoretical Computer Science*, 32, April 2000.
31. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings 1998 IEEE Symposium on Security and Privacy*, pages 160–171, May 1998.
32. Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 82)*, pages 80–91, 1982.