J. Cryptology (1994) 7: 69–77

Journal of
CRYPTOLOGY

Proof of Soundness (Integrity) of Cryptographic Protocols[1]

G. J. Simmons
Sandia Park, NM 87047, U.S.A.

Communicated by Thomas Beth

Received 25 March 1992 and revised 24 August 1993

Abstract. In 1990 a workshop on the Mathematical Concepts (or Principles) of Dependable Systems was held at the Mathematisches Forschungsinstitut Oberwolfach in Germany. The purpose of the workshop was to examine mathematical techniques developed to make it possible to prove the proper functioning of complex systems—where the system could be either hardware, software, information protocols, etc., or combinations of these components. Principal researchers in each of these areas were invited to take part in the workshop, and did—but as is so often the case at scientific meetings, one area emerged as a clear center of gravity for the workshop. This was the topic of applying formal methods to the proof of soundness or correctness of information or cryptographic protocols. The organizers of the workshop felt that the results reported in this area were so important to the cryptology community they deserved publication as a whole, as opposed to publication as separate and isolated papers: hence this special section. This paper, which is essentially an introduction for the invited paper that follows, describes the background for the workshop and the challenge problem(s) that provided a unifying theme for the three systems of formal analysis treated in it.

Key words. Cryptography, Protocols, Protocol failures.

Introduction

One has only to read the paper "Protocol Failures in Cryptosystems" by Moore [4] to be convinced of the difficulty of either designing sound information-based protocols or of proving that a candidate protocol is sound, i.e., that it realizes only the intended function(s) and then only under prescribed circumstances. Even under

[1] For reasons beyond the control of either the authors or the editors, this special section has been unusually long delayed in publication; so long in fact that a great deal of related work has been done in the interim. It would have been desirable to summarize these developments in this preface, but this would have changed the nature of the preface from a brief note describing the circumstances that led to the invitation to the authors to prepare the paper that makes up the special section, to a full fledged survey paper in itself. After careful consideration, it was decided that this would be inconsistent with the original intent for the special section.

the very restrictive definition that Moore uses—in which a weakness of an information protocol is classified as a protocol failure only if the intended function of the protocol is completely subverted without at the same time impeaching, or even eroding, the integrity of the underlying cryptoalgorithm—there is a long list of examples. Moore exhibits key distribution protocols that fail by passing keys to unintended and unauthorized recipients, digital signature protocols in which signatures can be forged for messages that were not signed by an authorized signatory, notarization protocols in which fraudulent messages can be notarized by persons other than the notary, secrecy protocols in which the contents of (supposedly) private communications are revealed to unintended receivers, etc.; all without impeaching, or even eroding, the security of the cryptographic algorithm on which the respective protocols are based. The number of examples is enormously increased if the definitions are weakened slightly to include failures that occur as a result of the erosion of the security of a cryptographic algorithm—as, for example, through reducing a key-space search from a computationally infeasible problem to a tractable one—or in which, while the integrity of some of the intended functions of the protocol survive intact, others fail according to Moore's definition. More about this later.

The point of the remarks in the preceding paragraph is that it is extremely difficult to determine whether an information-based protocol is sound, even for very simple protocols. Traditionally, the way this has been done is for the designer (and others) to attempt to find flaws in the protocol using all of the experience and analytical tools at their disposal. This "free form" evaluation is commonly called an analysis—but the impressive array of example protocols that have survived several rounds of this process, only to have subsequently been shown to have a protocol failure, is compelling evidence of the inadequacy of this *ad hoc* approach. Furthermore, information-based protocols are becoming much more complex; for example, protocols to utilize shared capabilities in networks where the obligation of a particular server may only be to execute faithfully a blind task handed to it. A paradigm for this might be a protocol with which a customer, who has authenticated his identity with one relay node in a mobile communications network, is passed to other nodes as he moves about in the service area without having to repeat the identification procedure each time a handoff to a new node is made. In this example if an interloper could deceive a node into accepting a transfer of service for an unidentified or improperly identified caller, services could be obtained fraudulently. The integrity of such a system is certainly no better than the soundness of the overall protocol.

It has been recognized for a long time that a more systematic and formal method of evaluating protocols was needed. The examples mentioned above simply show how pressing this need is. As might be expected though, such metamethods have been exceedingly difficult to devise and prove. A small community of researchers, including the three authors represented here—Kemmerer, Meadows, and Millen—have achieved surprising and satisfying results in the formal analysis of protocols. Our purpose here is to prepare the way for a detailed discussion by them of these methods—with emphasis on their applicability to cryptographic protocols.

Background

In 1982 one of the workshop organizers, in collaboration with Purdy and Studier, proposed what was essentially a key distribution protocol [5]. The basic concept was that within a community of users, the originator of an encrypted message (a single cipher) would have the capability to generate and disseminate user-unique keys that would enable only those users he wished to be able to decrypt the cipher, and hence to have access to the information the cipher concealed. An important part of the protocol was that a user had to request a key from the issuer of the cipher, i.e., he had to "subscribe," in order to be included. There was a noncryptographic portion of the protocol involving user-unique secure cryptographic modules (SCMs) which prevented a user who had subscribed and received a key from sharing his key with another user who had not subscribed. Although the SCMs play an essential role in the proposed usage of the protocol (to control the use of protected software), they played no part in the protocol failure, and are not discussed here. Shortly after the protocol was published, the present author found a (Moore-type) protocol failure in it. A pirate who wished to use the protected (encrypted) software without licensing it, i.e., without having been issued a key, could—by observing the (open) communications between another user requesting a key and the issuer of the cipher responding with that user's unique key—use this information to cause his SCM to generate the same key he would have obtained had he licensed the software. In other words, the key distribution scheme designed to put user-unique keys in only the hands of designated users failed in such a way that any user who cared to eavesdrop on open communications could generate the same key he would have been given had he subscribed to the service. All of this was accomplished without in any way degrading the integrity of the cryptoalgorithm used in the protocol, hence this failure was a dramatic example of a Moore-type protocol failure. It is important to the background for this paper to mention that in addition to the key distribution function, the Purdy–Simmons–Studier protocol also provided an authentication capability which made it possible for a user to verify that the key–cipher pair he had was a matched pair, i.e., that the key he received as a result of becoming a subscriber was a "good" key issued by the same party who created the cipher/software.

In 1985 Simmons reported the failure he had discovered and proposed a revised protocol to realize the same functions (by then referred to as a selective broadcast protocol) [7]. Because it provided such a clean example of a protocol failure, the original Purdy–Simmons–Studier protocol quickly became a test problem for formal methods of protocol evaluation. Any method that fails to discover the failure needs to be extended so that it does. Each of the authors of the following paper had a formal analysis system in development at the time, which they immediately applied to both the original flawed Purdy–Simmons–Studier protocol and to the new Simmons selective broadcast key distribution protocol. Millen was the first to modify his program (Interrogator) to deal with this problem, but Meadows's Naval Research Laboratory (NRL) Protocol Analysis Tool and Kemmerer's Formal Development Methodology both proved adequate to the task as well. The most impressive accomplishment to date using formal methods for protocol analysis, though, resulted when Meadows applied the NRL program to the new Simmons protocol. It should be remarked that this (the selective broadcast protocol) is one of the most exhaustively analyzed protocols in existence. First, having been guilty of publishing a flawed protocol once, the author was extremely careful in the analysis of the "corrected" version: subjecting it not only to careful scrutiny himself but also enlisting the aid of several prominent, i.e., often successful, cryptanalysts as well before publishing the new protocol. Second, Gemini Computers chose the revised key distribution protocol to use in their secure operating system [6] and subjected it to further analysis. Finally, the Gemini system was submitted to the National Computer Security Center for certification where the embedded key distribution protocol was subjected to yet another evaluation. It was a welcome bit of news therefore, but no great surprise, when Meadows's program was able to "prove" within the very natural rewrite rules of her program that the Simmons' selective broadcast key distribution protocol was sound [2]. The completely unexpected—and hence very surprising—result though, was her discovery of a convoluted failure in the authentication function of the protocol, i.e., a sequence of steps through which a user could be convinced that he had a legitimate key from the purported issuer when in fact he did not. Once this flaw was found it was easy to fix and then prove that the "fixed" protocol was sound in both respects. If the reader has not seen Meadows's analysis of this protocol he should get her papers [2], [3] and read them since it is a beautiful illustration of the power of formal methods. The most impressive thing about the analysis is that a flaw—albeit one that depends on a complex set of steps—which had eluded the extensive and intensive analysis this protocol had been subjected to was discovered. In a sense, this discovery marked the "coming of age" of formal methods since it is far more convincing for a program to discover an elusive and unsuspected flaw than to show that a known flaw can be found.

The Oberwolfach Meeting

In late 1987 Professor Thomas Beth, Director of the European Institute for Systems Security (EISS) and Professor of Computer Science at the University of Karlsruhe in Germany, and Gustavus J. Simmons, then Senior Fellow in national security studies at the Sandia National Laboratories in the United States, submitted a proposal to the Mathematisches Forschungsinstitut Oberwolfach for a workshop to be devoted to a new topic in applied mathematics: The Mathematical Concepts (or Principles) of Dependable Systems. The proposal was accepted and the workshop scheduled for 15–21 April 1990. Since the following paper is an outgrowth of that meeting, it may be useful to indicate a relevant portion of the intent for the meeting.

The letter of invitation to the potential participants (participation in Oberwolfach workshops is always by invitation only) by Simmons said in part:

> For several years, Prof. Beth and I have been struck by the fact that there is a remarkable commonality between several widely different

> areas of applied mathematics. Each of these areas is characterized by a need to be able to verify (prove?) the correctness of function of complex systems. Depending on the area in which the problem arises, the system itself could be software, hardware (i.e., circuits or ensembles of electronic devices), physical plants, information protocols, etc. ...
>
> I'll mention only one last example area—because this letter is already too long—that illustrates another aspect to this topic. This is the subject of information integrity protocols, which includes such topics as cryptography, authentication, etc. Here the problem is to devise protocols which can ensure that the system will only act in the intended manner; for example, if the content of a communication is supposed to be concealed from all but a designated set of recipients, that it will be concealed (in all probability), or that if it is supposed to require the concurrence of some specified grouping of persons to initiate an action, that no improper grouping should be able to do so. What is different about these protocols, in which the opponent is intelligent and capable (with memory, etc.), from the reliability or safety-like problems in which the "opponent" is a statistically described set of possible events is that the analysis becomes two-sided (i.e., game-like) as opposed to being a one-sided or statistical analysis. Irrespective of this difference, though, the underlying problem is essentially the same; namely, of how to design complex protocols so as to insure correct functioning, and then of devising means to prove the correct functioning of the resulting scheme.

It was the intention of the organizers to bring together at the workshop principal researchers in each of these areas—with particular emphasis on the topic of proving the soundness of information protocols. A preliminary version of Moore's paper on protocol failures had just appeared and the state of the art in using formal methods for the proof of soundness (correctness) of protocols was as described above. It was clear to the organizers, therefore, that the time was ripe to have such a meeting.

While the planning for the Oberwolfach meeting was going on, Professor David Newman of George Washington University sent a report on a new "Key Distribution Protocol for Digital Mobile Communication Systems" [8] to Simmons asking for his comments and criticism. A detailed discussion of this key distribution protocol (KDP1), and of the associated (Moore-type) protocol failure that Simmons found in it, are given in the Appendix. We discuss here only those aspects of it that are relevant to the Oberwolfach workshop, and hence to the paper by Kemmerer, Meadows, and Millen. Newman and his collaborators, Tatebayashi and Matsuzakai, were concerned with the problem of providing for secure communications in a mobile radio network. Such networks are characterized by a having a large (and changing) population of subscribers (users), each of whom has a terminal with a very limited computational capability and of low physical security, i.e., the terminal is no place to store a collection of private keys. There is a central server, the communicaton relay node, who could reasonably have both greater computational capability and physical security—in particular, while the node is active. The bare bones of the Tatebayashi–Matsuzakai–Newman (TMN) protocol is that all user terminals have the capability to carry out the easy one of a complementary pair of public key (asymmetric) cryptographic operations (forming a modular cube as opposed to extracting a modular cube root, for example) and to carry out efficiently a single key (symmetric) cryptographic operation (exclusive or-ing of a binary one-time key with a binary text to implement unconditionally secure Vernam ciphers, for example). When subscriber A wishes to communicate securely with subscriber B, A sends a (random) one-time key encrypted under the easy end of the server's public key system to the server along with a request to set up a secure channel to B. The server contacts B, who generates a (random) session key and also encrypts it under the easy end of the server's public key system. B sends this cipher to the server. The server who is able to do hard computations decrypts both ciphers to recover the session key and A's one-time key, which he then uses to encrypt the session key. The server sends this cipher to A who can decrypt it using the one-time key he chose to recover the session key that B chose. A and B now have a common key which they can use to communicate securely. The object of the protocol seems to have been achieved. Terminals having very limited capability are able to exchange session keys using two secure cryptographic algorithms: Vernam encryption with an appropriately generated one-time key is unconditionally secure, while extracting modular cube roots (for appropriately chosen prime factors in the modulus) has been proven by Williams [9] and by Seberry *et al.* [1] to be equivalent in difficulty to factoring the modulus. Since factoring can be made to be a computationally infeasible task, neither cryptoalgorithm can be impeached. Nonetheless, Simmons discovered that the TMN protocol had a beautifully simple Moore-type protocol failure. Since the TMN protocol is a key distribution protocol intended to pass a session key generated by B to the requester A securely, a Moore-type protocol failure would be the distribution of the key to someone other than A (or, of course to the unconditionally trusted server S). This is precisely what Simmons showed was possible—the details of the failure are also given in the Appendix. Since the failure does not erode the security of either of the cryptoalgorithms, it is a Moore-type protocol failure. There may be other modes of failure as well; for example, the server's security might be breached or suspect, or the key generation process (employed by the users) might not be random, etc., but weaknesses of these types do not concern us here. The relevant point to this paper and to the Oberwolfach workshop was that the TMN key distribution protocol contained a clean and previously unknown Moore-type protocol falure which was discovered just in time to be set as a challenge problem for all of the authors of papers on formal methods of proof of soundness for protocols who were going to take part in the workshop.

Conclusion

The performance of various formal method programs in the analysis of protocols having Moore-type protocol failures is a convincing demonstration that these methods have finally come of age in helping prove the soundness of protocols. Since the general cryptographic community is largely unaware of how capable and sophisticated these methods have become, it is hoped that the following paper, unified by the treatment of a single real-world challenge problem, will provide a useful introduction to the subject. In a real sense these methods are maturing just in the nick of time since information protocols are rapidly becoming too complex to rely on classic "free form" analyses for proofs of their soundness.

Appendix

Protocol Failure in the Tatebayashi–Matsuzakai–Newman Key Distribution Protocol

In the TMN key distribution protocol there is a large (and changing) community of users, each of whom has a mobile communication terminal with limited computational and cryptographic capability, and a single central server with greater computational capability. It is assumed that the server is secure when in operation, but not sufficiently secure to store private cryptographic keys for all of the users, etc. The server is, however, assumed to be able to protect one key—namely his own. Tatebayashi *et al.* assume, in particular, that each terminal can encrypt and decrypt using the data encryption system (DES), and compute a modular cube, i.e., $x^3 \bmod n$ where $n = pq$ is a product of two suitably chosen primes so that n is infeasible (impossible) to factor. This is the easy end of an RSA system. The hard problem is to extract cube roots modulo n (decryption) which is what the server is assumed to be able to do. Subject to these assumptions, when A wishes to set up a secure communication with B—for which they must both share a common DES key—he executes the following protocol. A and B are the two users, S is the trusted server.

Key Distribution Protocol 1 (KDP1). The protocol for the key distribution can be summarized as follows:

1. First, A generates a random 64-bit number r_1 (a key-encryption key).
2. A encrypts r_1 with S's public key ($e = 3$) and sends $r_1^3 (\bmod n)$ to S.
3. S decrypts $r_1^3 (\bmod n)$ using its secret key $d = 3^{-1}$, i.e., $ed \equiv 1 \pmod{\varphi(n)}$ so that 3^{-1} is the multiplicative inverse of 3 modulo $\varphi(n)$, and gets $(r_1^3 (\bmod n))^{3^{-1}}$ $(\bmod n) = r_1$.
4. S calls B, asking B to generate a session key for a secure communication with A.
5. B generates a random 56-bit number which with the appropriate eight parity bits for a DES key becomes r_2 (a session key between A and B).
6. B encrypts r_2 with S's public key and sends $r_2^3 (\bmod n)$ to S.
7. S decrypts $r_2^3 (\bmod n)$ using its secret key $d = 3^{-1}$ to get $(r_2^3 (\bmod n))^{3^{-1}} (\bmod n)$ $= r_2$.
8. S encrypts r_2 using the key-encryption key r_1 supplied by A and sends the cipher $E(r_2, r_1)$ to A. Tatebayashi *et al.* suggests that the operation E be Vernam encryption so that $E(r_2, r_1) = r_1 \oplus r_2$.
9. A decrypts $E(r_2, r_1)$ using the key-encryption key r_1 which he generated in the first place to get $D(E(r_2, r_1), r_1) = r_2$. r_2 is now shared by A and B and can be used as a session key for secure (DES encrypted) communications between A and B.

The Flaw

An eavesdropper, E, wishes to listen in on the (supposedly) private communication between pairs of users in the mobile communication system. To do this, he must get access to the session key r_2 (we assume that the symmetric cryptographic scheme in which r_2 is used is unbreakable). In the protocol failure to be described, the eavesdropper must himself be a user of the system and have an accomplice, D, who is also a user of the system. E makes an advance arrangement with D which will make it possible for either of them to eavesdrop on anyone else's communications. They arrange that whenever one of them is asked by the server to provide a session key to set up a channel with the other that he will return a key, r_k, known to the other party, instead of a randomly chosen key as expected in the protocol. E and D could generate a table of randomly chosen keys, r_i, and each keep a copy, or they could agree to use any of the many available pseudorandom deterministic key stream generators with both of them knowing the initial values, etc. The point is that they both know what the key, r_k, will be, but an outsider will be unable to detect that they are violating the intent of the protocol, irrespective of how often they do so.

As has already been mentioned, E must learn the session key r_2 if he is to be able to eavesdrop on the communication between A and B. He has observed three encrypted transmissions on the open channel during the setup of the secure session between A and B.

$$
\begin{array}{ll}
A \text{ to } S & r_1^3\ (\mathit{mod}\ n), \\
B \text{ to } S & r_2^3 \pmod n, \\
S \text{ to } A & r_1 \oplus r_2,
\end{array}
$$

the latter two of which are both encryptions of the key r_2. Both cryptographic systems, however, are assumed to be secure, i.e., E cannot break $E(\cdot, k)$ nor can he extract cube roots modulo n. Since only the server can extract cube roots, the essential point to the protocol failure is that D can get the server to do for him what he cannot do for himself. D's protocol is the following:

1. E oberves B's response ($r_2^3 \pmod n$) to the server.
2. E chooses a random 64-bit number r and forms its cube which he multiplies with $r_2^3 \pmod n$ which he has just observed. He reduces this product modulo n to form $r^3(r_2^3 \pmod n) \bmod n$ which is the same as $(rr_2)^3 \pmod n$.
3. E sends $(rr_2)^3 \pmod n$ to the server along with a request to set up a secure channel to D.
4. S decrypts $(rr_2)^3 \pmod n$ to get rr_2 (mod n), *i.e.*, r_2 encrypted with r used as a random one-time key, so there is no way for S to detect that the key r_2 which he has just passed to A is involved in this new request.
5. S calls D, asking D to generate a session key for a secure communication with E.
6. D sends $r_k^3 \pmod n$ to S.
7. S decrypts $r_k^3 \pmod n$ to recover r_k which he believes to be the session key for the requested session between E and D. Because of the way in which r_k

was chosen there is no way for S to detect that r_k is not what it purports to be.

8. S encrypts the "session key" r_k using the one-time key rr_2 supplied to him by E to form the cipher $r_k \oplus rr_2$ which he sends to D.
9. Since D knows r_k he computes

$$r_k \oplus (r_k \oplus rr_2) = rr_2.$$

10. Since D chose r in the first place, he can easily compute its multiplicative inverse r^{-1} modulo n using the Euclidean algorithm. He now multiplies rr_2 by r^{-1} to recover r_2.

This has all taken place in the time required to initiate a session through the server. Using this protocol, E has obtained the session key that A and B are using and can thereafter eavesdrop on their communication in real time. The whole protocol failure depends on E being able to trick S into providing him with the modular cube root of r_2^3 which he could not do for himself. It is possible that a few seconds of encrypted communication could have occurred between A and B before E is able to get the key r_2. If it is vital to the eavesdropper that he not miss anything in the private communication, he could record the initial portion of the encrypted communication and decrypt it after the session is completed. The essential point is that E has learned the session key that A and B are using and hence can do anything that either of them can do.

References

[1] J. Loxton, D. S. P. Khoo, G. J. Bird, and J. Seberry, A Cubic Residue Code Equivalent to Factorization, *Journal of Cryptology*, Vol. 5, 1992, pp. 139–150.

[2] C. Meadows, Applying Formal Methods to the Analysis of a Key Management Protocol, NRL Report 9265, Naval Research Laboratory, September 19, 1990.

[3] C. Meadows, A System for the Specification and Verification of Key Management Protocols, *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, 1991, pp. 182–195.

[4] J. H. Moore, Protocol Failures in Cryptosystems, in *Contemporary Cryptology: The Science of Information Integrity*, G. J. Simmons, ed., IEEE, New York, 1991, pp. 541–558; also in *Proceedings of the IEEE*, Vol. 76, No. 5, May 1988, pp. 594–602.

[5] G. B. Purdy, G. J. Simmons, and J. A. Studier, A Software Protection Scheme, *Proceedings of the IEEE Computer Society 1982 Symposium on Security and Privacy*, Oakland, CA, April 26–28, 1982, pp. 99–103.

[6] R. Schell, Letter to G. J. Simmons, January 10, 1989.

[7] G. J. Simmons, How To (Selectively) Broadcast a Secret, *Proceedings of the IEEE Computer Society 1985 Symposium on Security and Privacy*, Oakland, CA, April 22–25, 1985, pp. 108–113; also in *Security & Privacy*, Vol. 2 (Compendium), IEEE Computer Society, New York, 1990.

[8] M. Tatebayashi, N. Matsuzakai, and D. B. Newman, Key Distribution Protocol for Digital Mobile Communication Systems, in *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Vol. 435, G. Brassard, ed., Springer-Verlag, New York, 1991, pp. 324–333.

[9] H. C. Williams, An M^3 Public-Key Encryption Scheme, in *Advances in Cryptology—CRYPTO '85*, Lecture Notes in Computer Science, Vol. 218, H. C. Williams, ed., Springer-Verlag, New York, 1986, pp. 358–368.