

# Limitations on Design Principles for Public Key Protocols

Paul Syverson

Center for High Assurance Computer Systems

Naval Research Laboratory

Washington, DC 20375

(syverson@itd.nrl.navy.mil)

## Abstract

*Recent papers have taken a new look at cryptographic protocols from the perspective of proposing design principles. For years the main approach to cryptographic protocols has been logical, and a number of papers have examined the limitations of those logics. This paper takes a similar cautionary look at the design principal approach. Limitations and exceptions are offered on some of the previously given basic design principals. The focus is primarily on public key protocols, especially on the order of signature and encryption. But, other principles are discussed as well. Apparently secure protocols that fail to meet principles are presented. Also presented are new attacks on protocols as well as previously claimed attacks which are not.*

## 1. Introduction

Protocols employing cryptography for key distribution, authenticated and/or confidential data exchange, and a host of other applications have been around for a long time. And, analysis and modelling techniques for evaluating cryptographic protocols have been researched for well more than a decade (e.g. [9, 19]). Logical methods have been applied to protocol analysis almost as long. In fact, some of these logics (especially BAN logic [7]) have themselves been subject to a good deal of analysis, e.g., [14, 31, 26, 6, 35]. Burrows et al. were careful to provide a number of caveats about the limitations of their logic in [7]. But, this has not precluded a number of confused applications or misapplications of the logic. What the papers just cited have done is map out in more detail the boundaries of what BAN logic can be applied to and what one can conclude from such application. This can prove valuable.

For example, in [26], Snekkenes showed that the results of BAN analysis is the same for any two protocols

containing the same set of messages regardless of the order in which they are sent. (The sender and receiver of each message is nonetheless expected to stay the same.) To illustrate this, consider a coin-flip protocol. One way to flip coins in a distributed setting is to have Alice send Bob messages saying ‘*H*’ (for heads) and ‘*T*’ (for tails) each encrypted with a key *K*. He then chooses one and sends it back. Finally, Alice sends Bob *K*. Bob obviously cannot tell whether he is choosing heads or tails until he has *K*. But, if the order of the sent messages is changed so that Alice sends *K* to Bob before he chooses, he can always decide how the coin will land. (Coin flip protocols can be used for establishing keys, for certified mail, and for other applications.) Note that this is a simple description to illustrate one point and does not include all the necessary security mechanisms. We will return to coin flips below.

More recently there has been some emphasis on providing general requirements [5, 29, 30] and design principles [1, 2] for cryptographic protocols. Design principles have been put forth not only for producing secure protocols but also for producing protocols whose security is easy to evaluate [17].

This paper will attempt to explore some limitations of the design principle approach to cryptographic protocols similar to the way that limitations on logical analysis were explored in the works cited above. One cannot fairly criticize the design principle approach simply because exceptions are found to the principles. Abadi and Needham state at the outset of their paper that the principles they give are neither necessary nor sufficient. Nonetheless, it can be just as helpful to be aware of exceptions and limitations to these principles as to be aware of the principles themselves. Though we will make some other points, we will proceed primarily by examining principles involving the encryption of signed data and/or the signing of encrypted data.

The remainder of the paper proceeds as follows. In section 2 we discuss a protocol message from [34] that

putatively fails because it is signed before encrypting rather than vice versa. We provide a solution to the failure that is both more reasonable at maintaining accountability and more computationally efficient than the solution in [34]. In section 3 we discuss the often reasonable principle that signing messages before encrypting is the order to follow in maintaining nonrepudiability. We look at a putative attack from [2] on a CCITT X.509 protocol message due to encryption prior to signing. We show that it is not an effective attack. (Nonetheless, the basic attack structure is quite clever, and we use it later to construct other attacks.) We also look at protocols where encryption prior to signing is desirable, sometimes even as a mechanism to maintain nonrepudiation. In section 4 we use the technique of the just mentioned attack to construct an attack on a coin-flip protocol from [25]. We also illustrate how difficult it can be to spot oracles by using this to attack oblivious transfer in an unexpected way. In section 5 we construct the most surprising attack based on the Anderson-Needham technique. We show how to attack an auction protocol to alter a bid after it is submitted even though it is always readable by the receiver and the receiver maintains an uncompromised copy of the bid for verification. Finally, we look at the underlying basis for most of the design principles that have been given, namely, explicitness. The attack on the auction protocol shows that even full explicitness does not prevent substitution. And, we discuss protocols for which explicitness is antithetical to design goals.

## 2. Is it OK to Encrypt Signed Messages?

In [34], Tsang and John considered a protocol in which principal  $A$  sends a message to  $B$  that is signed with  $A$ 's signature key and then encrypted with  $B$ 's public key. Their analysis of this protocol is that  $B$  can attack the protocol; thus,  $B$  cannot prove that  $A$  sent the message (i.e., nonrepudiation is lost). They conclude that the problem is fixed if  $A$  encrypts before signing.

The protocol that Tsang and John consider is the following. Suppose that Alice is both Bob's boss and Chuck's boss. And, suppose that she sends to Bob a message  $\{\{M\}_{K_a^{-1}}\}_{K_b}$ . Here ' $M$ ' stands for the message "Your salary is hereby increased by \$5000.", ' $K_a^{-1}$ ' is Alice's signature key, ' $K_b$ ' is Bob's public encryption key, and subscripting with the relevant key represents signature or encryption accordingly.

Now, since Bob is able to strip off the encryption with  $K_b$ , he can send  $\{\{M\}_{K_a^{-1}}\}_{K_c}$  to Chuck (where ' $K_c$ ' is Chuck's public encryption key). Chuck now has proof that Alice authorized a salary increase for

him, and this proof is every bit as good as the legitimate 'proof' that Alice sent to Bob. Because of this possibility, Alice can deny having sent the message to Chuck. Thus, she can equally well refute having sent it to Bob. But, if she instead sends  $\{\{M\}_{K_b}\}_{K_a^{-1}}$ , Bob's attack is no longer possible.

Tsang and John's example seems implausible. After all, any real world authorization for salary increase includes a unique identifier for the person receiving the increase such as a full name, employee ID number, etc. There are bearer authorizations, messages which apply to the bearer of that message [22], but these are used in contexts where a bearer instrument is appropriate. And, the point of the example is to show that the message is transferable in an unintended way. In other words, it is not meant to be a bearer instrument even though it appears to function as such. Tsang and John state that the problem with the message "originates from the hypothesis that the signer and sender are the same person." We will return to this. For now, we note that there is a simple way to securely indicate the intended recipient of the message, specifically, give his name. If Bob is tied to the salary increase within the signed message, the attack is not possible. This is in keeping with Abadi and Needham's third principle: "If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message."

Tsang and John consider this solution and dismiss it because "the security of the resulting scheme relies on the contents of signed messages". It is unclear why they think this should be a problem. They also cite integrity concerns, but integrity of signed and encrypted messages is generally assumed to be addressed on another level of analysis. (Cf. [28] for a discussion of integrity issues in cryptographic protocols.)

That the signer of a message need not be the originator of the message is widely recognized. Even if we do not employ the obvious solution just mentioned, we still need not resort to principals signing encrypted plaintext. Tsang and John note that someone guessing that  $\{\{M\}_{K_b}\}_{K_a^{-1}}$  contains the message "Your salary is increased by \$5000" can check this, thus compromising confidentiality. Their solution is to again encrypt this with  $K_b$ , i.e., to produce  $\{\{\{M\}_{K_b}\}_{K_a^{-1}}\}_{K_b}$ . However, note that if we reverse the order, i.e., sign then encrypt then sign, all these problems go away. The content of  $\{\{\{M\}_{K_a^{-1}}\}_{K_b}\}_{K_a^{-1}}$  cannot be confirmed by anyone but Bob. If this is the structure of salary increase notices, Bob cannot turn this into a salary increase for Chuck.

There is still a small potential danger even if Alice signs then encrypts then signs. The danger is if Bob

strips off the outer signature and the encryption, produces  $\{\{M\}_{K_a^{-1}}\}_{K_c}$ , and he or someone else somehow tricks Alice into signing this. For example, she might receive it as a nonce she is to sign to authenticate herself. In this case, even though she had once seen the inner content, it would be impossible for her to recognize it without Chuck's private key. This shows the danger of signing anything not understood: nonces, keys, etc. As Abadi and Needham note "signing before encrypting is not a bill of health". In fact this example illustrates both the wisdom and a limitation on their principle 5: "When a principal signs material that has already been encrypted it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message then encrypts it for privacy knows the content of the message." The second half of the principle must be read carefully; it only holds if the same principal who does the signing also does the encrypting. For example, the attack in this paragraph shows that the principle does not allow us to infer from  $\{\{\{M\}_{K_a^{-1}}\}_{K_b}\}_{K_a^{-1}}$  that Alice knows the contents of  $\{\{M\}_{K_a^{-1}}\}_{K_b}$  even though she may know the contents of  $\{M\}_{K_a^{-1}}$ . And, in general accord with principle 5, if we follow the Tsang and John suggestion, even if Bob produces  $\{\{M\}_{K_b}\}_{K_a^{-1}}$ , Alice can claim that she never saw the content of  $\{M\}_{K_b}$ .

In any case, three layers of signature and encryption in whatever order is clearly more computationally expensive than two. And if Alice simply sends

$\{\{ \text{At } T_1 \text{ Alice authorizes a salary increase for Bob}$   
 $\text{from } \$X \text{ to } \$X + \$5000 \}_{K_a^{-1}}\}_{K_b}$

none of these attacks are possible. Tsang and John's recommendation runs contrary to design principles of [1] (and, as we will see presently, design principles of [2]). This leads to their incorrect recommendation. In the next section, we will begin to look at protocols that are exceptions to these design principles.

### 3. Is it OK to Sign Encrypted Messages?

Expanding on Abadi and Needham's fifth principle, the first principle set forth by Anderson and Needham in [2] is:

**Principle 1:** Sign before encrypting. If a signature is affixed to encrypted data, then one cannot assume that the signer has any knowledge of the data. A third party certainly cannot assume that the signature is authentic, so nonrepudiation is lost.

This principle is predicated on the assumption that the motivation for digital signatures is to hold princi-

pals accountable for what they say in a manner that can be proven to an independent party. But, this is not their only use. There are numerous obvious examples of protocols where encryption should precede signing, e.g., voting protocols and blind signature protocols for preserving anonymity in digital cash systems. In these examples anonymity is a more important motivation than nonrepudiation; though both may be present. These examples may be considered esoteric, but accountability concerns are also not the primary current motivation for use of public keys. Though this may change in the not too distant future, to date public keys have primarily been used for key distribution. And, in a key distribution protocol the primary goal is secrecy. The idea is to keep the key, and hence the messages it protects, confidential within a limited group, typically two principals plus possibly a server. While signatures are generally used as part of authentication, authentication itself is used to keep the key within the limited group. In key distribution even integrity protections are primarily for secrecy concerns: messages are protected from alteration so that the key used and the group who has it cannot be altered in a way that allows others access to secrets. While there are important attacks on key distribution protocols that spoof authentication without revealing secrets, these are always considered less significant than attacks that reveal a key or secret data to an unintended principal.

This is not to say that nonrepudiation is a nonissue in key distribution. Showing that a principal had possession of a key makes him potentially liable for its misuse or at the very least negligent revealing of it, assuming this can be traced to him. Rather the point is that Anderson and Needham's first design principle is fairly specific. If one is not motivated by concern about nonrepudiation, then their principle may not apply. In fact, we will see below that this principle may not apply even if one is concerned about nonrepudiation.

Aside from the incorrect recommendations of Tsang and John, we have also noted that their example used to motivate encrypting before signing is unrealistic. But, such criticism cuts both ways. The example that Anderson and Needham use to illustrate their Principle 1 is itself unrealistic. Unlike Tsang and John, Anderson and Needham have chosen a real protocol, or at least a real proposal, viz: CCITT X.509 [8]. What is problematic in this case is not the protocol but the application of their attack to it. The general protocol form they attack is as follows: Alice sends  $\{\{M\}_{K_b}\}_{K_a^{-1}}$ , an encrypted then signed message, to Bob. In the example, RSA [24] is used, and some particulars of the algorithm are relevant. Filling in the RSA details the sent message becomes  $(M^{e_b} \pmod{n_b})^{d_a} \pmod{n_a}$ ,

where  $n_x$ ,  $e_x$ , and  $d_x$  are respectively the modulus, public exponent, and private exponent of principal  $X$ .

Since Bob can factor  $n_b$ , and its factors are only 250–300 bits long, [assuming moduli of 500 to 600 bits] he can work out discrete logarithms with respect to them and then use the Chinese Remainder theorem to get discrete logs modulo  $n_b$ . So, if he wants to get Alice’s ‘signature’ on a different message,  $M'$ , he can find  $x$  such that

$$[M']^x = M \pmod{n_b}$$

He then registers  $(xe_b, n_b)$  as a public key with a certification authority, and claims that the message signed by Alice was not  $M$  but  $M'$ .

This provides a direct attack on CCITT X.509 in which Alice signs a message of the form  $(T_a, N_a, B, X, \{Y\}^{e_b} \pmod{n_b})$  and sends it to Bob. Here  $T_a$  is a timestamp,  $N_a$  is a serial number, and  $X$  and  $Y$  are user data. ([2], p. 237)

The basic attack presented provides important lessons, but it cannot be used as a direct attack on the CCITT X.509 protocol they mention. Certification authorities are expected to maintain copies of old certificates precisely to resolve potential disputes. So, if minimal standards of key management are maintained, Bob’s attack can be detected.

Bob’s deceit is even detectable directly from the evidence he himself presents without any help from authorities. Every CCITT X.509 certificate includes a time it becomes effective and a time it expires. The basic design does not preclude that the start time is before the present. But, allowing after-the-fact certificates is such a dubious concept that certification authorities are unlikely to specially configure so as to allow this even if they were unaware of this attack. The certificate containing  $(xe_b, n_b)$  would have to be generated after  $T_a$ . (Such a system of course assumes that clocks are maintained to tolerable synchronization levels. As Abadi and Needham note, this effectively brings the time mechanism everywhere inside the trusted computing base.) Thus, if Bob attempts to claim that Alice had signed a message containing  $\{Y'\}^{e_b}$  rather than  $\{Y\}^{e_b}$ , and if he uses the certificate containing  $(xe_b, n_b)$  as proof, then the fact that the certificate was not yet good at  $T_a$  would obviate his claim and expose his deception. So, the attack is detectable solely from data that Bob himself would produce in attempting the fraud.

Anderson and Needham also present an attack based on using ElGamal encryption [12] with ‘trapdoor’ moduli. These can be chosen independently of any message to be attacked. So, timing features of messages and certificates will not preclude such an attack. However, the consensus amongst the panelists in the paper they cite is that the odds on finding trapdoor moduli that are both useful and hard to detect are currently negligible [10].

A variant on Anderson and Needham’s RSA-based attack on X.509 might still be possible. If  $Y$  were previously known or predictable text for Bob, then he could register and obtain a certificate for  $(e_b, n_b)$  and then obtain a certificate for  $(xe_b, n_b)$  before the first expires. He could then have a certificate for  $(xe_b, n_b)$  whose time range includes  $T_a$ . Assuming that  $Y$  is known or predictable, this attack is not detectable simply from the evidence Bob himself presents. But, this attack is still infeasible unless a number of things all happen. First, Alice would have to fail to keep a copy of the key she used to encrypt for Bob. Otherwise this could be compared to the key Bob claims is the right public key and seen to be different. Second, Alice would have to fail to receive a certificate revocation for  $(e_b, n_b)$ . This puts Bob’s attack under potentially impossible time constraints. (Cf. [27] for a discussion of relevant issues.) Even if Alice does not receive such a revocation, the existence of certificates for two keys, both valid at the time Alice sends her message, should call into question any obligation Alice might have based on her signature on that message. So, this attack also assumes, contrary to standard recommendations, that the certification authority does not keep copies of old certificates. It might still be possible to mount an attack such as this if the time that certificates are held by authorities is less than the statute of limitations on any obligation implicit in the X.509 message.

The failure of the example as an actual attack on X.509 does not undermine Principle 1. Further, Anderson and Needham point out that “the weaknesses we have discussed do not necessarily imply that any given system based on a protocol criticised above is insecure, as there are many ways to implement compensating controls.” ([2], p. 245) However, in this case the controls are known and given in the source they cite. Furthermore, the attack does not illustrate any weakness in the protocol for sending confidential information to Bob. (That Bob cannot correctly infer that this information came from Alice is well known, independent of this attack [7, 21].) The Anderson-Needham attack is based on the assumption that Alice could otherwise be held accountable for the contents of  $\{Y\}_{K_b}$ . But, such an assumption is not justified. This follows

from fifth principle of Abadi and Needham on which Principle 1 is based: “When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. . . .” In any case, even if the given example does not constitute a real attack on signing encrypted data, Principle 1 remains intuitively strong. And, the Tsang and John protocol has been seen as an implausible example of the need to encrypt before signing to maintain accountability. Nonetheless, more compelling examples are available. Consider the following coin-flip protocol.

$$\begin{aligned} A \rightarrow B &: \{r_{i1}, \{H\}_K, \{T\}_K\}_{K_a^{-1}} \\ B \rightarrow A &: \{r_{i2}, X\}_{K_b^{-1}} \\ A \rightarrow B &: \{r_{i3}, K\}_{K_a^{-1}} \\ B \rightarrow A &: \{r_{i4}, K\}_{K_b^{-1}} \end{aligned}$$

Here  $K$  is a key for some symmetric key algorithm and  $X$  is randomly chosen from  $\{H\}_K$  and  $\{T\}_K$ . Note that which order  $\{H\}_K$  and  $\{T\}_K$  appear in the first message is expected to be randomly chosen.

The  $r_{ij}$  serve, amongst other things, as a protocol run number to guard against replay attacks. An important feature for messages in this protocol is that they be current (from the given run). As far as Bob is concerned messages need not be virgin (not previously used) or fresh (from the present epoch). Alice, on the other hand, requires that  $K$  be virgin, lest Bob be able to determine what he has received in the first message. The  $r_{ij}$  also prevent an outside intruder from substituting messages from parallel sessions between Alice and Bob to cause them to get different results from the flip. (Roughly similar attacks on roughly similar protocols are discussed in [33].) The second subscript in the  $r_{ij}$  indicates additional information that is assumed to be included (clearly) in the message. For example,  $r_{i2}$  might represent “This is Bob’s choice between messages received from Alice for a coin flip in protocol run  $r_i$ .” Note that we follow Abadi and Needham’s Principle 10, which tells us that it should be possible to tell the encoding used for  $r_{ij}$ . The English is clear enough, but more efficient encodings of these meanings may be possible.  $r_{i2}$  will also keep Bob from being held accountable for signing something unintended, e.g. if the content of  $X$  is “Bob owes Alice \$100”. The protocol itself is a variant on early protocols that have been attributed to Blum and to Even.

The important thing to note about this protocol is that Alice’s signature on the unencrypted  $H$  and  $T$  is useless. If Bob wants to argue before a judge that Alice engaged in this protocol run, it will not help to prove that she once said “heads” and once said “tails”. What will help is that he has her signature on the two

encrypted messages and he has her signature on the key. Nonrepudiation is only preserved in this case if Alice encrypts these messages before signing. There are numerous obvious examples of protocols where encryption should precede signing, e.g., voting protocols and blind signature protocols for preserving anonymity in digital cash systems. But, these examples are examples where anonymity is more important than nonrepudiation. The lesson of this example is that there are cases where encryption should precede signing, even cases in which nonrepudiation is a concern. This does not mean that Principle 1 is incorrect for most straightforward cases. It does mean that Principle 1 should not be followed blindly.

#### 4. Is it OK to Encrypt Encrypted Messages?

Consider the following coin-flip protocol given in [25]. Alice and Bob choose keys  $K_a$  and  $K_b$ . Alice appends a random string to ‘heads’ and another to ‘tails’ forming  $M_1$  and  $M_2$  (not necessarily respectively). She encrypts these with  $K_a$  and sends them to Bob. He then picks one, encrypts it with  $K_b$  to form  $\{\{M_i\}_{K_a}\}_{K_b}$  and sends this to Alice. The protocol assumes that encryption commutes; so Alice can form  $\{\{\{M_i\}_{K_a}\}_{K_b}\}_{K_a^{-1}} = \{M_i\}_{K_b}$  which she send to Bob. He now decrypts this and sends it to Alice. If the random string matches she sends him her key. He then sends her his key.

$$\begin{aligned} A \rightarrow B &: \{M_1\}_{K_a}, \{M_2\}_{K_a} \\ B \rightarrow A &: \{\{M_i\}_{K_a}\}_{K_b} \\ A \rightarrow B &: \{M_i\}_{K_b} \\ B \rightarrow A &: M_i \\ A \rightarrow B &: K_a \\ B \rightarrow A &: K_b \end{aligned}$$

The protocol does not have to use public keys; anything will do as long as the encryption algorithm commutes. Of course this does not hold for most symmetric key algorithms such as DES. When presenting this protocol Schneier offers RSA as an example for which it does hold. This is an overstatement. It only holds for RSA if there is a common modulus. Normally, a common modulus can lead to attacks [20]. In this particular application none of those attacks apply; however, it is possible to construct one that does.

Before discussing the attack, we note that the random strings in  $M_1$  and  $M_2$  are unnecessary. For, if Bob substitutes, e.g., ‘heads’ for ‘tails’ when he finally sends cleartext to Alice, she will be able to detect this when the keys are revealed. And, she can

prove it to a third party; so, the protocol preserves nonrepudiation—assuming, as in the last example, that signatures are added to each message. Note that for all the messages save one, the signature would be on encrypted text or on keys, i.e., on apparently random strings. Nonetheless, nonrepudiation is preserved because of the connection between the three messages each principal sends. Of course, there should ordinarily be sufficient identifying plaintext attached within the signature lest a principal construct another explanation for the messages he or she sent.

If we assume that Alice and Bob are using RSA with a modulus in the 500–600 bit range, then Bob can mount an attack similar to the one Anderson and Needham directed at CCITT X.509. Specifically, using the technique of Anderson and Needham, he simply chooses two keys  $K_{b_1}$  and  $K_{b_2}$  such that  $\{\{M_1\}_{K_a}\}_{K_{b_1}} = \{\{M_2\}_{K_a}\}_{K_{b_2}}$  and  $\{\{M_2\}_{K_a}\}_{K_{b_1}} = \{\{M_1\}_{K_a}\}_{K_{b_2}}$ . Then, when he receives  $\{M_i\}_{K_b}$ , he can decrypt it using both  $K_{b_1}$  and  $K_{b_2}$  and determine which decryption corresponds to  $M_1$  and which to  $M_2$ . He can then send whichever message  $M_i$  he chooses, sending the corresponding key in the final message. He can choose to return the  $M_i$  from that message or the other. Unlike the attack on CCITT X.509 this is completely undetectable in any circumstance since the keys used are not revealed to anyone until after the fact.

Notice that, without the last message, the protocol attacked has the same basic structure as some oblivious transfer protocols. (Oblivious transfer protocols transfer a message from Alice to Bob with a fifty-fifty chance of success in such a way that Alice cannot tell if it has succeeded. They can be used amongst other things to construct contract signing protocols.) Surprisingly, this attack allows Bob to use Alice as an oracle so that the chance of transfer becomes 100%. Specifically, suppose that each of  $M_1$  and  $M_2$  contain either a message to be transferred to Bob or random garbage. Once Alice has sent Bob the third message, if he decrypts using one key he gets  $M_1$  and if he decrypts using the other he gets  $M_2$ . So, he can always obtain the desired message.

This reflects on two other Anderson and Needham principles. Principle 3 says, “Be careful when signing or decrypting data that you never let yourself be used as an oracle by your opponent.” This attack shows how tricky spotting an oracle attack can be. There is no reason to suspect an oracle attack here. Decrypting one message encrypted with Alice’s public RSA key should have nothing to do with decrypting the other. Yet, Bob creates a relationship between the two messages that has nothing to do with their explicit internal structure, i.e., with their being encrypted by Alice, such that her decrypting either of them is effectively for him the same

act.

Principle 6 says, “Do not assume that a message you receive has a particular form (such as  $g^r$  for known  $r$ ) unless you can check this.” In the above attack Alice does receive  $\{\{M_i\}_{K_a}\}_{K_b}$  and can even verify this at the end of the protocol. Nonetheless, Bob is able to attack. This is because in the attack  $\{\{M_i\}_{K_a}\}_{K_b} = \{\{M_1\}_{K_a}\}_{K_{b_1}} = \{\{M_2\}_{K_a}\}_{K_{b_2}}$ . Unless the mathematical structure for encryption, decryption, and signature is a free algebra there is no guarantee that form is unique.

This paper is primarily about limitations and exceptions to previously stated design principles. However, in light of the last paragraph we offer one new principle at which others can take a crack.

Do not assume that a message you receive has *only* a particular form (such as  $g^r$  for known  $r$ ) even if you can check this.

One might suggest protecting the basic coin-flip protocol by increasing the size of the modulus to a point where Bob cannot calculate discrete logarithms on the factors. However, the protocol does not need to rely on factoring. Despite being suggested as having the appropriate structure, RSA is therefore not an appropriate algorithm for this protocol. It is more reasonable to simply use a prime modulus sufficiently large to make finding discrete logarithms hard. This would be simpler to implement, computationally cheaper, and not subject to the above attack.

## 5. Is it OK to Sign Signed Messages?

The next type of attack we discuss is the most surprising developed from the basic Anderson-Needham technique. All of the previous attacks involved messages that could not be read at the time they were received. In this type of attack, the receiver can read the entire content of the message; yet, he is still tricked into later thinking he had received a different message. (What is not surprising is that these most surprising of attacks require the most restrictive implementation assumptions; nonetheless, the examples we present are still instructive. We will discuss implementation assumptions below.)

Consider a closed-bid auction. The auction protocol should keep the bids of bidders private until after the bidding is closed. It should also provide each bidder with a receipt for his bid to be used in resolving any disputes that may arise once the winning bid is announced. The following protocol is designed to meet these requirements. Principals directly involved in the protocol are the bidders, a bid registrar who certifies

and registers the bids, and an auctioneer who decides the winning bid and announces the result. (The auctioneer and the registrar could be the same principal.)

An invitation for bids is posted to a public site including: a description of the item up for bid, an identifier for the item, and an expiration time after which bids will not be accepted. (This posting is assumed to be protected for authenticity and integrity.) Alice sends to the bid registrar her bid signed with her key and encrypted with the registrar's public key. The registrar checks the signature and that the bid is of the right form (e.g., constitutes a bid on an item for which bidding is not closed.) If the bid is acceptable, the registrar sends an acceptance receipt to the bidder, logs the bid and forwards it to the auctioneer. If the bid is not acceptable, the registrar again sends a receipt to the bidder and logs the bid, only now the receipt and log reflect that the bid is not acceptable. In this case the bid is not given to the auctioneer. After bidding is closed, the registrar sends a message to the auctioneer telling him that all bids have been sent. The auctioneer then chooses the winning bid on that item and announces the winning bidder and bid.

Such a protocol could be used both when offering items bidders would like to obtain and when requesting bids for contracts bidders hope to get, i.e., in cases where high bid wins and in cases where low bid wins. Depending which of these is the intended application the protocol might be further tailored to meet additional specific requirements.

Here is a basic description of the interaction between bidder Alice and the bid registrar in the submitting of an acceptable bid.

$$\begin{aligned} A \rightarrow R &: \{Cert_a, M_1\}_{K_r} \\ R \rightarrow A &: \{M_2, M_1\}_{K_r^{-1}} K_a \end{aligned}$$

Here  $Cert_a$  is Alice's public key certificate from a relevant authority,  $M_1$  is

$\{\text{Alice bids } \$X \text{ on item } Y \text{ at } T_a.\}_{K_a^{-1}}$ , and  $M_2$  is "Alice's bid of  $\$X$  on item  $Y$  is accepted at  $T_r$ ." If Alice wants to contest the outcome of the auction, say because she feels that she had outbid the winning bidder, she can use the (unencrypted) contents of her receipt from the registrar to appeal. Note that it is up to Alice to produce a receipt and a key certificate; the registrar only keeps a copy of the bid to guard against fraudulent claims. If he were aware of the following potential attack, he could detect it by simply keeping a copy of  $Cert_a$  together with the bid. In the absence of such concern he has no reason to keep certificates (unless he has some key management responsibilities as well). We are assuming that he does not keep the certificate Alice sent to him.

Suppose Alice would really like to obtain an item up for bid, but she wants to get it as cheaply as possible. In basic RSA, signing and encrypting are mathematically the same; they are differentiated only by whether the exponent is publically known or not. So, if signatures in this protocol are based on RSA with a modulus of 500–600 bits, then she can use the basic Anderson-Needham technique to mount an attack as follows. She prepares two bids, one at the low end of what she expects is sufficient to win the auction and the other at the high end. She then finds two keys using the Anderson-Needham technique such that one is an integer power of the other (modulo her modulus). She obtains certificates for each of them and sends the low bid in to the registrar. If she wins the bid, she does nothing more. If she loses and the winning bid is lower than her high bid, she submits her receipt along with the key certificate for the high bid as evidence that the registrar recorded the wrong bid value for her. When this is checked against the signed bid logged for her, it is determined she is right, and she wins the bid. (It is conceivable that in the face of such an error, instead of her winning the bid the auction is declared invalid. This at least gives her another opportunity to bid on the item rather than losing it.)

If key certificates do not contain relevant temporal information, then Alice can even wait until after she finds out the value of the winning bid before choosing her high bid and corresponding key. However, if such information is included (as it is in most systems) then her claim in this case would reveal her fraud, just as in the Anderson-Needham attack on CCITT X.509. Even if the two bids are chosen prior to submission, there are a number of assumptions that must hold for the attack to succeed. She is still subject to some of the impediments to success and vulnerabilities to discovery discussed in section 3 in connection with the variant attack on X.509. She need not worry about the registrar keeping the submitted certificate since this was explicitly ruled out. Note that in the variant attack on X.509 the party being defrauded had a vested interest in keeping the key around for liability reasons. In this case, the auction house has no such motivation. In fact, from a financial point of view, they have an interest in having the attack succeed even if they know about it—provided that their tail is covered against liability, which it would be, and provided that this possibility of attack is not known to other participants. Thus, unless another bidder suspects fraud *and* can find the related key certificates at the certification authority she will not be detected solely because 'duplicate' certificates exist. And, since the fraud is directed at the registrar, who accepts responsibility for the error, there is some-

what less chance of such a search than in the X.509 attack. Alice may also be able to time the submission of key certificate requests and submission of her bid so as to minimize the chance that prior to logging her bid the registrar receives a certificate revocation for the accompanying key. We must relatedly assume that the registrar does not check key certificates of bids logged since the last revocation list against the current list.

An important issue for the feasibility of this attack is the nature of digital signatures. Current typical digital signature algorithms work by taking a hash or checksum of a message and effectively signing this. The plaintext message is then sent with the signed hash of the message. Thus, for a normally implemented RSA signature it is the hash that is raised to the private exponent of the signer. (The standard protocol notation for representing signing as the inverse of encryption is therefore somewhat deceptive.)

In the auction protocol example the bid registrar signs a message already signed by the bidder. If the registrar applies the hashing function and exponentiation only to the bidder's signed hash, then the attack is not affected by the hashing prior to exponentiation. More typically, however, the registrar takes a hash of the plaintext together with the bidder's signature on a hash of it and signs this. In this case the attack is less likely to succeed. For, the receipt that the bidder has will not be able to show the registrar's signature on the claimed bid. She could still claim that the registrar made a mistake not just when logging the bid but when copying the plaintext of the signed message. This would imply either that the registrar had not properly checked the bidder's signature or that the exact sequence of events at the registrar were such that the registrar was able to first confirm the bidder's signature on the bid and then accidentally alter the signed bid and log the incorrect value.

The attack could still proceed if the bidder were able to construct two bids such that the hash of one together with her signature was the same as the hash of the other together with her signature. While this is theoretically possible, hash functions are designed to make it difficult to find such collisions. Because of the complications introduced by hashing, any attack based on finding collisions in signed messages using the Anderson-Needham technique is likely to be harder than an attack based on finding collisions in encrypted messages using the technique. This provides further rule-of-thumb incentive to follow Anderson and Needham's recommendation to sign before encrypting.

Note that the secure auction service of Franklin and Reiter [13] is not vulnerable to such an attack. First, since digital money must be put up against any bid

in their auction, the attack requires the active participation of a bank. Second, since they protect against certain attacks by agents of the auction house, the bid is submitted in multiple places. It is thus unlikely that one could sustain a claim that all receivers of a bid made the same error in logging its value.

The above auction protocol is new; whatever plausibility or value it might have, an obvious objection to this attack is that it is on an example that has never been seriously put forth for use. One response would be to fix the protocol and/or implementation assumptions to avoid the attack and posit it for use. That is outside the scope of this paper. Another response is to demonstrate a similar attack on a published protocol.

The Internet Billing Server (IBS), under development at Carnegie Mellon is intended to provide billing services for service providers on the Internet. In [23], a number of alternative transaction protocols for IBS are presented including some based entirely on public key cryptography. The protocol we examine was not ultimately recommended for use; however, this was due to considerations other than security vulnerabilities. It was also examined in [18] for accountability guarantees. Some redundancies were found but no security or accountability problems. This protocol takes place between three principals: end user,  $E$ , service provider,  $S$ , and billing server,  $B$ . It occurs in three phases, pricing assurance, service provision, and invoice confirmation and payment. In pricing assurance,  $E$  requests and receives a signed price per item statement from  $S$ . In service provision,  $E$  requests a specific number of items at the quoted price, receives this from  $S$ , and acknowledges their receipt. In invoice confirmation and payment, invoices of the sale are sent by  $B$  to  $E$  and  $S$ . The three phases of the protocol run as follows:

1.  $E \rightarrow S : \{Price\ Request\}_{K_e^{-1}}$
2.  $S \rightarrow E : \{Price/item\}_{K_s^{-1}}$
3.  $E \rightarrow S : \{\{Price/item\}_{K_s^{-1}}\ item@price\}_{K_e^{-1}}$
4.  $S \rightarrow E : \{Service\}_{K_s^{-1}}$
5.  $E \rightarrow S : \{Service\ Acknowledgement\}_{K_e^{-1}}$
6.  $E \rightarrow S : \{Invoice\ Request\}_{K_e^{-1}}$
7.  $S \rightarrow B : \{\{Invoice\}_{K_b}\}_{K_s^{-1}}$
8.  $B \rightarrow S : \{\{Invoice\}_{K_s}\}_{K_b^{-1}}, \{\{Invoice\}_{K_e}\}_{K_b^{-1}}$
9.  $S \rightarrow E : \{\{Invoice\}_{K_s}\}_{K_e^{-1}}$

*Invoice* here includes message 3, and  $S$ 's signature on message 5. If  $E$  would like to get goods without paying for them, he can use the Anderson-Needham technique to construct two keys. One will reveal message 6 to acknowledge less than the actual amount of service,

and the other will show the actual amount. Once the full protocol completes,  $E$  can complain that he has not received everything he paid for. The invoice will back him up on this, and  $S$  will send him more items. If  $S$  would like to cheat  $E$ , he can use the Anderson-Needham technique to construct two keys: one revealing a low *Price/item* in message 2, the other revealing a high *Price/item*. The ‘low price’ key is given to  $E$ , and the ‘high price’ key is registered for use with *Invoice*.

Both of these attacks involve significant implementation assumptions. Keys are expected to be registered and handled along the lines of X.509 [8]. Thus, timing issues like those that affected the Anderson-Needham attack on X.509 are a factor here as well. For  $E$ ’s attack, he would need to make sure that the key given to  $S$  is revoked and replaced prior to service provision by the key used to acknowledge fewer items. If  $S$  does not check the revocation list (e.g., if replacement occurs after the whole protocol has started, particularly after he has obtained the end user’s key to confirm message 1), then  $E$ ’s complaint will be corroborated by the invoice. Unless  $S$  is keeping records that he is supposed to rely on the key server/billing server to keep, the evidence he has will show him that he must have sent too few items.  $S$ ’s attack is even trickier since the key associated with the invoice might need to be valid at the time that  $E$  signed message 3. All the details of key management are not discussed in [23]. The protocol has not been pursued in development of the IBS, and the ones that have been pursued are not subject to this kind of attack. Thus, any speculation we make about the feasibility of these attacks on this protocol must remain on the hypothetical level. Furthermore, the same problem that arose in the auction attack because of the use of hashing in digital signatures arises for these attacks as well.

## 6. Does it Pay to be Explicit?

Explicitness is the “overarching principle of which others are in some sense instances” [2]. In [1] it is given as Principle 10:

If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

A limitation that we saw above on this principle is that a protocol can meet this requirement and still be sub-

ject to attacks because the same bit string can have two meanings, even within the same basic encoding. And, whether the meaning is eventually revealed, as in the coin-flip of section 4, or even explicitly given, as in the above auction protocol, it is still subject to this ambiguity attack. This reflects the principle we enunciated in section 4: don’t assume that a message you receive has *only* a particular form even if you can check this. This is, however, more of an implementation than a design principle. In fact, on the usual level of protocol description Abadi and Needham’s principle is probably as good as one can get because there is no way to represent this kind of ambiguity that depends on features of the encryption algorithm used. (We have decided to beat others to the punch by criticizing our own principle.) There is, however, another limitation that occurs at the appropriate level of description.

Abadi and Needham introduce Principle 10 by noting, “It seems important that principals recognize messages for what they are, and can associate them correctly with the current step of whatever protocol they are executing.” This is also the guiding idea behind other explicitness principles, e.g., Gong’s fail-stop protocols [17] and the causal consistency criterion of [32]. But, there are perfectly good protocols that violate these principles. In fact, some even derive their strength largely from their violation of these principles. Protocols have been proposed that are resistant to attacks based on guessing poorly chosen keys or passwords [3, 4, 16, 15]. These work largely by having the plaintext contents of encrypted messages be unrecognizable. For example, in the basic EKE protocol of Bellare and Merritt the only contents of any encrypted message is either a key or a nonce (that has not appeared as cleartext) or the encryption of a key or such a nonce. No principal can tell prior to the last message he receives that any message is from a given protocol run. Which message is which is discernible after the fact by the legitimate principals, but, during a run no message can be verified to be part of the protocol or even linked to any other message by means of encrypted text. The protocol proceeds effectively on faith until the end. Such a protocol might seem to be outside the scope of all the usual design principles; however, it is really fundamentally guided in design by Anderson and Needham’s Principle 3: “Be careful when signing or decrypting data that you never let yourself be used as an oracle by your opponent.” In fact, such protocols show that similar caution should be followed when encrypting data. Note that even when designing a protocol specifically to guard against guessing attacks spotting the oracle can be difficult [11].

## 7. Conclusions

We have looked at a number of protocols and attacks thereon, ones that illustrate stated design principles and ones that serve as exceptions to them. In section 2 we saw that, contrary to the recommendations of [34], nonrepudiation of basic message content is usually better preserved by encrypting before signing, rather than vice versa. In section 3 we saw that there are other motivations for protocols involving digital signatures than nonrepudiation. Thus, the sign-before-encrypting principle of Anderson and Needham may not be fully general. Further, we saw in the coin-flip example of that same section that it may be necessary to encrypt before signing to preserve more subtle nonrepudiation. Leaving nonrepudiation aside, we saw in section 4 that spotting oracles can be very difficult because, even if a message has a verifiable form, it may have another form as well. In fact, in section 5 we saw that a protocol can be attacked for this reason even if message content is not only verifiable but always readable. Finally, in section 6 we looked at the explicitness principle: the significance of a message should be clear to the intended recipient of that message. We saw that this may run contrary to other design principles such as not allowing yourself to be used as an oracle.

One may come to the end of this paper with a feeling of frustration born of looking in vain for a single recommendation that is offered without a countervailing caution against it. As the last section illustrates, following one design principle will sometimes lead to violating another. If the reader has such a feeling of disquiet, then we have done our job. Design principles are rules of thumb. It is tempting to use them simply as an after-the-fact idiot check. They can be very useful in that regard for catching simple errors. But, we should not infer from the fact that we meet even all the ones we have that the result is a good design. This motivates us to offer in parting one overarching design principle:

Use design principles at the beginning, middle, and end of designing a protocol. First, use them to guide your preliminary design. Then, when you have a specification, go through them all and look at the motivation for applying the principle in the given context. Is the motivation best served by following the principle, and if not, how might it better be served? When you have a final design, go through the principles again and look at those you violated. Make sure you have a good reason for doing so in each case.

## Acknowledgements

I would like to thank Cathy Meadows and the anonymous referees for helpful comments on a draft of this paper, Li Gong for making me aware of [34], and Paul van Oorschot, Li Gong and Cathy Meadows for helpful discussions.

## References

- [1] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. In *Proceedings of the 1994 IEEE Computer Society Symposium on Security and Privacy*, pages 122–136. IEEE Computer Society Press, Los Alamitos, California, 1994.
- [2] R. Anderson and R. Needham. Robustness Principles for Public Key Protocols. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247. Springer Verlag, Berlin, 1995.
- [3] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84. IEEE CS Press, Los Alamitos, California, May 1992.
- [4] S. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 244–250. ACM Press, New York, November 1993.
- [5] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic Design of a Family of Attack-Resistant Protocols. *IEEE Journal on Selected Areas in Communication*, 11(5):679–693, June 1993.
- [6] C. Boyd and W. Mao. On a Limitation of BAN Logic. In T. Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 240–247. Springer Verlag, Berlin, 1994.
- [7] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. Research Report 39, Digital Systems Research Center, February 1989. Parts and versions of this material have been presented in many places including *ACM Transactions on Computer Systems*, 8(1): 18–36, Feb. 1990. All references herein are to the SRC Research Report 39 as revised Feb. 22, 1990.
- [8] CCITT. X.509 and ISO 9594-8 The Directory—Authentication Framework, March 1988.
- [9] R. DeMillo, N. Lynch, and M. Merritt. Cryptographic Protocols. In *Proceedings of the Fourteenth ACM Symposium on the Theory of Computing*, pages 383–400. ACM Press, New York, 1982.
- [10] Y. Desmedt, P. Landrock, A. Lenstra, K. McCurley, A. Odlyzko, R. Rueppel, and M. Snid. The Eurocrypt

- '92 Controversial Issue: Trapdoor Primes and Moduli. In R. Reuppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 194–199. Springer Verlag, Berlin, 1993.
- [11] Y. Ding and P. Horster. Undetectable On-line Password Guessing Attacks. *Operating Systems Review*, 29(4):77–86, 1995.
- [12] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
- [13] M. Franklin and M. Reiter. The Design and Implementation of a Secure Auction Service. In *Proceedings of the 1995 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–14. IEEE Computer Society Press, Los Alamitos, California, 1995.
- [14] V. Gligor, R. Kailar, S. Stubblebine, and L. Gong. Logics for Cryptographic Protocols — Virtues and Limitations. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 219–226. IEEE Computer Society Press, Los Alamitos, California, 1991.
- [15] L. Gong. Optimal Authentication Protocols Resistant to Password Guessing Attacks. In *Proceedings of the Eighth Computer Security Foundations Workshop*, pages 24–29. IEEE Computer Society Press, Los Alamitos, California, 1995.
- [16] L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets Against Guessing Attacks. *IEEE Journal on Selected Areas in Communication*, 11(5):648–656, June 1993.
- [17] L. Gong and P. Syverson. Fail-Stop Protocols: An Approach to Designing Secure Protocols. In *Proceedings of DCCA-5: Fifth International Working Conference on Dependable Computing for Critical Applications*, pages 44–55, Urban-Champaign, Illinois, September 1995.
- [18] R. Kailar. Reasoning About Accountability in Protocols for Electronic Commerce. In *Proceedings of the 1995 IEEE Symposium on Research in Security and Privacy*, pages 236–250. IEEE Computer Society Press, Los Alamitos, California, 1995.
- [19] M. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Institute of Technology, 1983.
- [20] J. Moore. Protocol Failures in Cryptosystems. In *Contemporary Cryptology*, chapter 11, pages 541–558. IEEE Press, Los Alamitos, California, 1992.
- [21] J. Nechvatal. Public Key Cryptography. In *Contemporary Cryptology*, chapter 4, pages 177–288. IEEE Press, Los Alamitos, California, 1992.
- [22] B. C. Neuman. Proxy-based Authorization and Accounting for Distributed Systems. In *Proceedings of the 13<sup>th</sup> Int. Conf. on Distributed Computing Systems*, May 1993.
- [23] K. R. O’Toole. The Internet Billing Server: Transaction Protocol Alternatives. Technical Report INI TR 1994-1, Carnegie Mellon University Information Networking Institute, April 1994.
- [24] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures. *Communications of the ACM*, 21:120–126, 1978.
- [25] B. Schneier. *Applied Cryptography*. John Wiley and Sons, New York, 1994.
- [26] E. Sneekenes. Exploring the BAN Approach to Protocol Analysis. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 171–181. IEEE Computer Society Press, Los Alamitos, California, 1991.
- [27] S. Stubblebine. Recent-Secure Authentication: Enforcing Revocation in Distributed Systems. In *Proceedings of the 1995 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 224–235. IEEE Computer Society Press, Los Alamitos, California, 1995.
- [28] S. Stubblebine and V. Gligor. On Message Integrity in Cryptographic Protocols. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 85–104. IEEE Computer Society Press, Los Alamitos, California, 1992.
- [29] P. Syverson and C. Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177. IEEE Computer Society Press, Los Alamitos, California, 1993.
- [30] P. Syverson and C. Meadows. Formal Requirements for Key Distribution Protocols. In A. D. Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1995.
- [31] P. F. Syverson. The Use of Logic in the Analysis of Cryptographic Protocols. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 156–170. IEEE Computer Society Press, Los Alamitos, California, 1991.
- [32] P. F. Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, New York, November 1993.
- [33] M.-J. Toussaint. Separating the Specification and Implementation Phases in Cryptology. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *Computer Security — ESORICS 92*, volume 648 of *Lecture Notes in Computer Science*, pages 77–101. Springer Verlag, Berlin, 1992.
- [34] W. Tsang and J. John. A Neglected Aspect of Digital Signatures. Technical Report, Dept. of Information Systems and Computer Science, National University of Singapore, May 1987.
- [35] P. C. van Oorschot. An Alternate Explanation of two BAN-logic “failures”. In T. Hellesest, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 443–447. Springer Verlag, Berlin, 1994.