Authentication Tests and Disjoint Encryption: a Design Method for Security Protocols*

Joshua D. Guttman
The MITRE Corporation
guttman@mitre.org

20 August 2003

Abstract

We describe a protocol design process, and illustrate its use by creating ATSPECT, an Authentication Test-based Secure Protocol for Electronic Commerce Transactions. The design process is organized around the *authentication tests*, a method for protocol verification based on the strand space theory. The authentication tests dictate how randomly generated values such as nonces may be combined with encryption to achieve authentication and freshness.

ATSPECT offers functionality and security guarantees akin to the *purchase request*, *payment authorization*, and *payment capture* phases of SET, the secure electronic transaction standard created by the major credit card firms.

1 Introduction

In previous work [12, 14, 9], we have developed a method—called the "authentication test" method—that can be used by hand to verify cryptographic protocols. We also pointed out that the same ideas can be used to guide the protocol development process, quickly leading to new protocols; proofs of correctness for these protocols then follow from the development process. In [12, 14] we illustrated the point by "reinventing" preexisting protocols. The purpose of this paper is to use it to create a completely new protocol with highly non-trivial functionality.

We call our new protocol ATSPECT, an Authentication Test-based Secure Protocol for Electronic Commerce Transactions. It is intended to achieve the essential security goals of the existing Secure Electronic Transaction (SET) *purchase request*, *payment authorization*, and *payment capture* phases, as we understand them.

*Supported by the National Security Agency. An earlier version appears as "Security Protocol Design via Authentication Tests"in *Proceedings, 15th IEEE Computer Security Foundations Workshop*, IEEE CS Press, June 2002.

The Secure Electronic Transaction protocol [19] was a major effort undertaken by a consortium of credit card companies and banks in the mid-90s. It was intended to provide a basis for secure electronic commerce. It is not currently in use anywhere, presumably partly as a consequence of being complex, difficult to implement, and difficult to analyze. For these reasons it was viewed as a high-risk undertaking, something that the financial industry prefers to avoid. Also, it shifts information away from merchants (for instance, information about their clients' credit cards), and resistance from the retail industry may be another reason why it languished. However, it would have provided better functionality for customers and financial institutions and better privacy protection for customers. The security goals of SET are hard to determine in a precise way, although Bella, Massacci, and Paulson have recently studied it in its own terms [2]. We will make no strong claim relating SET to ATSPECT.

2 ATSPECT Protocol Goals

Our design goals for ATSPECT are to provide authentication and pairwise confidentiality for certain values in a three-way protocol exchange. ATSPECT must also provide non-repudiation guarantees. However, we give no attention to fairness: different participants achieve their guarantees at different stages of the protocol. Analyzing fairness requires subtler methods [5, 15].

2.1 Protocol Participants

Principals playing three different roles, typically a Customer, a Merchant, and a Bank or other financial institution, desire to engage in an authenticated transaction. We will refer to the three participants as C, M, and B. Some data must be agreed among all three participants, for instance their identities and the total purchase cost for an order C places with M.

Other data must be shared between a pair, while remaining confidential from the third participant. For instance, C's credit card number must be agreed between C and B, but is best withheld from M. Otherwise, when M's systems are hacked, all its customers' credit card numbers will be stolen. The merchandise being purchased must be agreed between C and M, but is no concern of B's. Payment details such as B's fee for handling the credit card transaction for the merchant M may be confidential business information that should not be disclosed to C. No data may be disclosed to principals other than these three.

The same principal may play different roles in different protocol executions. When merchants order supplies from each other, they alternately play the roles of C and M. A bank may order supplies from a merchant, playing the role of C. Possibly it also plays the role of B in the same protocol run. Some firms may want to use different keys when playing the different roles, but the protocol should remain sound even if the same keys are used.

2.2 Protocol Goals

The goals of the participants are of four kinds:

Confidentiality, I Data intended only for a pair P, Q of participants is not disclosed to the third participant in the run, or any other principal.

Confidentiality, II Data intended to be shared among the three participants in the exchange is not disclosed to any other principal.

Authentication, I Each participant P should receive a guarantee that each partner Q has received P's data and Q accepted it.

Non-Repudiation Each participant P should be able to prove the **Authentication, I** guarantee it has received to a third party.

Authentication, II Each participant Q should receive a guarantee that data purportedly from a partner P in fact originated with P, freshly in a recent run of this protocol.

Three-Party Agreement Suppose that P completes a run apparently with partners Q, R. Then Q and R have begun runs of the protocol, agreeing on the shared data.

Most of these goals concern just a pair P and Q of principals. We want to achieve the goals whichever principals P and Q may be. This observation motivates our design strategy, which treats the protocol as a collection of two party subprotocols (Section 4). When we show that the two-party protocols meet their goals (Section 5.1), we will also be more precise about which keys must be uncompromised to establish each goal.

Two of the goals are intrinsically about the interaction of all three parties. One is **Confidentiality, II**, which concerns the confidentiality of the information shared among all three participants. The other is **Three-Party Agreement**, which requires that both intended interlocutors have begun runs of the protocol with compatible shared data. The pairwise secret data of the intended interlocutors is not constrained by **Three-Party Agreement**.

We establish the three-party goals directly for the combined protocol (Section 6.4). They are easily proved given the structure of the three-party protocol, and what is already known about confidentiality and authentication for the two-party subprotocols. We explain the practical usage of the protocol in somewhat more detail in Section 6.3.

3 The Authentication Tests

In this section, we will introduce the basic ideas of the strand space theory, and then describe the authentication tests. A more precise summary is in the Appendix.

3.1 Strand Spaces

A *strand* is a sequence of transmission and reception events local to a particular run of a principal. If this principal is honest, it is a *regular strand*. If it is dishonest, it is a *penetrator strand*, taking the forms in Definition A.8.

A *bundle* $\mathcal{B}$ is a causally well-founded directed graph containing the transmission and reception events of a number of strands. It represents a global execution possible for a given protocol (with a penetrator). A node m in the graph *precedes* a node n (written $m \preceq_{\mathcal{B}} n$) if the n is accessible from m via 0 or more edges of the graph. Likewise, $m \prec_{\mathcal{B}} n$ means it is accessible via 1 or more edges. (See Definition A.5.)

We write S for *safe* keys, i.e. keys we can prove that the penetrator can never learn. In [14] we show how to define S in a useful way. In our current context, we are interested only in the private members of public-private key pairs. Since private keys are never transmitted in the protocols we will consider, they will belong to S unless compromised before execution of the protocol. Thus, we will not need any elaborate method to prove that a key is in S. If $K \in \mathsf{S}$, the penetrator can never use K for encryption or decryption.

3.2 The Authentication Test Idea

Suppose a principal in a cryptographic protocol creates and transmits a message containing a new value v, later receiving v back in a different cryptographic context. It can conclude that some principal possessing the relevant key K has received and transformed the message in which v was emitted. If $K \in \mathsf{S}$ is safe, this principal cannot be the penetrator, but instead must be a regular principal. A *transforming edge* is the action of changing the cryptographic form in which such a value v occurs. The *authentication tests* [10, 14, 18] give sufficient conditions for transforming edges being the work of regular principals. There are two main types of authentication test.

Outgoing Tests A uniquely originating value a may be transmitted only in encrypted form $\{\!|\ldots a\ldots|\!\}_K$ where the decryption key $K^{-1} \in \mathsf{S}$ is safe. If it is later received outside the context $\{\!|\ldots a\ldots|\!\}_K$, then a regular participant, not the penetrator, must have been responsible the first time it appears in a different context. We write $\{\!|\ldots a\ldots|\!\}_K \quad \rightsquigarrow \quad \ldots a\ldots$ for a transforming edge that extracts it from this form. This transforming edge occurs after the original transmission of $\{\!|\ldots a\ldots|\!\}_K$ at m_0 and before the transformed version is received back at m_1, where the temporal relations refer to the ordering $\preceq_{\mathcal{B}}$ generated by the arrows in the bundle $\mathcal{B}$.

It is an *outgoing test* because the encrypted unit goes out; see Figure 1. Figure 1 presents a theorem, Proposition 19 of [14] in simplified form.

Incoming Tests If, instead, a is received in encrypted form $\{\!|\ldots a\ldots|\!\}_K$ although it was not sent in that context, and the encryption key $K \in \mathsf{S}$ is safe, then a regular participant must have been responsible when a entered this

$$\begin{tikzcd}
m_0 \arrow[rr, dashed, "K^{-1} \in \mathsf{S} \qquad a \sqsubset \{\!|h|\!\}_K"] \arrow[d, Rightarrow] & & n_0 \arrow[d, Rightarrow] \\
m_1 & {} \arrow[l, dashed, "a \sqsubset \operatorname{term}(n_1)"'] & n_1 \arrow[l, dashed, "a \sqsubset t'"']
\end{tikzcd}$$

Assume	$\{\!\mid h\mid\!\}_K \not\sqsubset \operatorname{term}(m_1)$ a originates uniquely at m_0, a contained only in $\{\!\mid h\mid\!\}_K$
Conclude	nodes n_0, n_1 exist in $\mathcal{B}$ and are regular $\{\!\mid h\mid\!\}_K \not\sqsubset t'$ $m_0 \prec n_0 \prec n_1 \prec m_1$

Figure 1: Outgoing Authentication Test

$$\begin{tikzcd}
m_0 \arrow[r, dashed, "a \sqsubset \operatorname{term}(m_0)"] \arrow[d, Rightarrow] & {} \arrow[r, dashed] & n_0 \arrow[d, Rightarrow] \\
m_1 & {} \arrow[l, dashed, "\{\!|h|\!\}_K \sqsubset \operatorname{term}(m_1) \quad K \in \mathsf{Safe}"'] & n_1 \arrow[l, dashed, "a \sqsubset \{\!|h|\!\}_K"']
\end{tikzcd}$$

Assume	$\{\!\mid h\mid\!\}_K \not\sqsubset \operatorname{term}(m_0)$ a originates uniquely at m_0
Conclude	nodes n_0, n_1 exist in $\mathcal{B}$ and are regular $m_0 \prec n_0 \prec n_1 \prec m_1$

Figure 2: Incoming Authentication Test

Test	Test edge	Constraint	Transforming edge	Bound
Outgoing	$+\{\|h\|\}_K \Rightarrow -\ldots a \ldots$	$K^{-1} \in \mathsf{S}$	$\{\|h\|\}_K \rightsquigarrow a$	$\times$
Incoming	$+\ldots a \ldots \Rightarrow -\{\|h\|\}_K$	$K \in \mathsf{S}$	$a \rightsquigarrow \{\|h\|\}_K$	$\times$
Unsolicited	$-\{\|h\|\}_K$	$K \in \mathsf{S}$	$\rightsquigarrow \{\|h\|\}_K$	

Table 1: The Authentication Tests

context. We refer to this transforming edge as $\ldots a \ldots \rightsquigarrow \{\!|\ldots a \ldots|\!\}_K$. As with an outgoing test, the transformation producing $\{\!|\ldots a \ldots|\!\}_K$ must occur after m_0 and before m_1. We call this an *incoming test* because the encrypted unit comes in, as shown in Figure 2, representing Proposition 20 of [14]. In public key cryptography, K is serving as a signature key.

Sometimes a uniquely originating value a is transmitted in one encrypted form $\{\!|h|\!\}_K$ and received back in a different $\{\!|h'|\!\}_{K'}$. If $K^{-1} \in \mathsf{S}$ and $K' \in \mathsf{S}$, then this is *both* an outgoing test and an incoming test. However, these two views may have different consequences. As an outgoing test, it implies a regular transforming edge that accepts $\{\!|h|\!\}_K$ and extracts a from it. This may be of some form other than $\{\!|h'|\!\}_{K'}$, since another principal may later transform it again. The incoming test yields a transforming edge creating $\{\!|h'|\!\}_{K'}$, although it may have received a in a form other than $\{\!|h|\!\}_K$.

Unsolicited Tests A third, related but weaker, type of test is the *unsolicited test*. If a term $\{\!|t|\!\}_K$ is received, and $K \in \mathsf{S}$ is safe, then $\{\!|t|\!\}_K$ originated on some regular strand. After all, it originated somewhere, and that can not have been a penetrator strand if $K \in \mathsf{S}$. Here we know only that the regular node originating $\{\!|t|\!\}_K$ is before the node on which it is received. We do not know any node after which it must have occurred. We write $\rightsquigarrow \{\!|t|\!\}_K$ for the positive node that must exist as a result of an unsolicited test.

Summary The authentication tests are summarized in Table 1. The last column contains $\times$ if the first node on the test edge is a lower bound (in the ordering $\preceq$) constraining when the transforming edge occurs.

We will design ATSPECT so that incoming tests are sufficient to achieve all the authentication properties. A second, alternative justification of the goal **Authentication, I** uses an outgoing test. An unsolicited test achieves the non-repudiation goal.

3.3 Recency

In [9] we study recency as a means for ensuring that protocols cannot be undermined by key compromise. In the current paper, we use the same notion of recency for a different purpose, namely to ensure that a transaction is not caused by a dishonest party replaying a stale message.

Regular strands provide a way to measure recency. Implementers always ensure that a local protocol run will timeout long before cryptanalysis could have succeeded. Thus, a principal engaged in a strand knows that an event is recent if it happened after an earlier event on the same strand.

Definition 3.1 ***(Recency)*** *A node n is* recent for *a regular node m_1 in $\mathcal{B}$ if there is a regular node $m_0 \in \mathcal{B}$ such that $m_0 \Rightarrow^+ m_1$ and $m_0 \preceq_{\mathcal{B}} n \prec_{\mathcal{B}} m_1$.*

The incoming and outgoing tests entail recency. That is, if $m_0 \Rightarrow^+ m_1$ is a test edge, and $n_0 \rightsquigarrow n_1$ is the corresponding transforming edge in $\mathcal{B}$, then $m_0 \prec n_0 \prec n_1 \prec m_1$, so that n_0 and n_1 are recent for m_1. By contrast, the unsolicited test establishes nothing about recency. If we assume that each regular strand is implemented so as to time out before the total elapsed time between its first node and last node reaches some bound b, then recency in the sense of Definition 3.1 entails that n happened less than b time units before m_1.

In some cases, we need a more inclusive, "extension ladder" notion of recency.

Definition 3.2 ***(i-Recency)*** *A node n is 1-recent for m_1 if n is recent for m_1 as in Definition 3.1. A node n is $(i+1)$-recent for m_1 if there exists a node m_0 such that n is i-recent for m_0 and m_0 is recent for m_1.*

If n is i-recent for m, then there are i strands, each overlapping a portion of the preceding one. From beginning to end, at most i times b, the time-out bound for a single regular strand, can have elapsed. In the **Authentication, II** goal of ATSPECT, we will be interested in 2-recency. We will arrange that Q, executing a strand s_Q, can be sure that P's data originated on a strand s_P, such that some node of s_P comes after some node of s_Q. The data may have originated before any node of s_Q, but not more than b before.

4 Authentication Tests and Protocol Design

The authentication tests suggest a protocol design process. At our level of abstraction, authentication protocol design is largely a matter of selecting authentication tests, and constructing a unique regular transforming edge to satisfy each. We will now examine our security goals and consider how to achieve them using authentication tests.

Cryptographic Assumptions We will assume that each principal has two public-private key pairs. In one, the public key is used for encryption and the private key is used for decryption. In the other, the private key is used for signatures, and the public key for verification. We assume that the public keys for any participant can be determined reliably, e.g. via a public key infrastructure. When P is a principal with public encryption key K_P, we write $\{\!|t|\!\}_P$ to stand for $\{\!|t|\!\}_{K_P}$. Assuming K_P is uncompromised (i.e. $K_P^{-1} \in \mathsf{S}$), only P can tractably recover t from this encryption.

The encryption may be implemented via familiar mechanisms, such as the following. The sender chooses a temporary symmetric key K, encrypting the payload t with that key; K is transmitted encrypted with the public key of the recipient. Some additional information tying K to t may also be included. Thus, to transmit t securely to P, one transmits $\{\!|K \hat{} \ldots|\!\}_{K_P} \hat{} \{\!|t|\!\}_K$. It would be interesting to develop a general theorem justifying this sort of transformation, particularly in connection with a model of the underlying cryptography [3, 11].

Likewise, $[\![t]\!]_P$ is the result of signing t using P's private signature key. We assume that only P can tractably construct $[\![t]\!]_P$ from a new message t.

One other cryptographic-quality primitive is needed, namely a hash function; $h(t)$ is the result of applying the hash function to t. We assume that no principal can tractably find a pair of values t_1, t_2 such that $h(t_1) = h(t_2)$, or, given v, can tractably find t such that $h(t) = v$.

We model the cryptographic operators following Dolev and Yao [6], as formalized in the strand space theory [21, 14]. We regard hashing as encryption with a key for which no one knows the matching decryption key.

4.1 Payloads and Confidentiality

We will not specify the payloads fully. However, we allow one confidential payload to originate at each principal P, intended for each partner Q. We refer to it as $\mathsf{sec}_{P.Q}$, and a goal of the protocol is to provide a confidentiality protection for its contents against any principal other than P, Q. We also allow for a shared payload shared_P sent by P to both other principals. Confidentiality of shared_P against any principal other than C, M, B is required.

We assume that the identities of the intended principals may be recovered from shared_P, as well as other core data about the transaction, via a function $\mathsf{core}(\mathsf{shared}_P)$. To avoid guessing attacks [16], some unpredictable value should occur either in shared_P or in $\mathsf{sec}_{P.Q}$.

Each principal P, having received shared data from Q and R and having constructed the shared data P it will itself transmit, checks that

$$\mathsf{core}(\mathsf{shared}_P) = \mathsf{core}(\mathsf{shared}_Q) = \mathsf{core}(\mathsf{shared}_R)$$

Since we expect to implement the confidentiality requirements using public key cryptography, we will need to have P encrypt $\mathsf{sec}_{P.Q}$, together with shared_P and possibly other ingredients, using K_Q the public key of the recipient Q.

4.2 Designing the Two-Party Subprotocols

To simplify our problem, we will regard the full, three-party protocol as being composed out of simpler subprotocols that involve pairs of parties. This is natural because our authentication goals are pairwise goals; we simply want to achieve them for all six ordered pairs of the three principals. Thus, we focus on an arbitrary pair P, Q. When we have seen how to achieve the authentication goals for P, Q in a subprotocol, we will then piece the subprotocols together to form the full protocol (Section 6), there being several ways to do this. Our work on protocol independence [13] will justify the composition.

$$\begin{tikzcd}
P & & Q \\
m_0 \arrow[rr, dashed, "\{\!|\ldots \,\hat{}\, N_{P.Q} \,\hat{}\, \mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P|\!\}_Q"] \arrow[d, Rightarrow] & & n_0 \arrow[d, Rightarrow] \\
m_1 & & n_1 \arrow[ll, dashed, "\llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \rrbracket_Q"']
\end{tikzcd}$$

Figure 3: Edges Achieving Authentication, I

4.2.1 Achieving Authentication, I

Our first authentication goal is the assertion:

Authentication, I Each participant P should receive a guarantee that each partner Q has received P's data and Q accepted it.

P's data means the two values $\mathsf{sec}_{P.Q}$ and shared_P, which we know must be transmitted encrypted with Q's public key. The incoming authentication test tells us that one way to ensure this is to prepare a new value $N_{P.Q}$, transmitting $N_{P.Q}$ with $\{\!|\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P|\!\}_Q$. After receiving and processing this unit, Q returns an authenticating message $A_{P.Q}$ containing $\llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, \ldots \rrbracket_Q$, which proves that $N_{P.Q}$ reached Q and was accepted as part of a successful strand.

We also want to ensure that $N_{P.Q}$ was accompanied by the payloads $\mathsf{sec}_{P.Q}$ and shared_P when it was processed. Therefore we will require the authenticating message $A_{P.Q}$ to take the form $\llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, t \rrbracket_Q$ where t contains the payloads in some form. Specifically, we require that they be decrypted and hashed, so that we have $A_{P.Q} = \llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \rrbracket_Q$. We now have the behavior shown in Figure 3. This is evidently an incoming test assuming that Q's signature key is uncompromised and $N_{P.Q}$ is uniquely originating.

However, the outbound message also contains a uniquely originating value, namely $N_{P.Q}$, and this value is encrypted with Q's public key. If we assume that Q's decryption key is also uncompromised, then this is also an outgoing test. Only Q can decrypt the payload to extract $N_{P.Q}$.

This is not merely redundant. It may correspond to a meaningful work-flow within the principal Q. For instance, if $P = C$ and $Q = M$, then the transforming edge for this outgoing test may be performed in the sales department. They check that the customer's order is valid, that the price of each item is correct, and that each item is available in inventory. Then they transfer the order to the accounts receivable department. Accounts receivable prepares the hash $h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P)$, affixes the signature, and executes the rest of the protocol. Although all of these steps occur automatically within the merchant's information systems, they are implemented in a distributed way. The decryption and signature keys may be separately protected on different computer systems maintained by independent parts of the corporation.

The decision to include $N_{P.Q}$ within the encrypted unit, and the decision to hash $\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P$ rather than the encrypted component

$$\{\!| \ldots \mathbin{\hat{}} \mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P |\!\}_Q,$$

is thus motivated by a desire to accommodate separation of duty within enterprises, at least for the case $Q = M$. Thus, the portion of the protocol represented in Figure 3 ensures that the **Authentication, I** goal will be met in two separate ways.

4.2.2 Achieving Non-Repudiation

The behavior displayed in Figure 3 also achieves the non-repudiation goal.

Non-Repudiation Each participant P should be able to prove its **Authentication, I** guarantee to a third party.

If P wishes to hold Q responsible for the transaction, then P can disclose the plain-texts $N_{P.Q}$, $\mathsf{sec}_{P.Q}$ and shared_P, together with the signature

$$[\![\ldots \mathbin{\hat{}} N_{P.Q} \mathbin{\hat{}} h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P)]\!]_Q.$$

This certifies that Q received, processed, and approved the transaction. The certification depends only on the assumption that Q's signature key is uncompromised, as it relies on the unsolicited test that a message of this form can be produced only by Q. Because Q signs the decrypted values $\mathsf{sec}_{P.Q}$ and shared_P, the principal P must disclose the content of the transaction in order to hold Q responsible. This seems desirable from a business point of view.

4.2.3 Achieving Authentication, II

In order to achieve the second authentication goal, we must extend the protocol.

Authentication, II Each participant Q should receive a guarantee that data purportedly from a partner P in fact originated with P, freshly in a recent run of this protocol.

In particular, it originates at a 2-recent node (Definition 3.2).

We enrich the protocol exchange displayed in Figure 3 by having Q emit a uniquely originating value $N_{Q.P}$. P signs $N_{P.Q}$, $N_{Q.P}$, and the hash of the payloads in a recency certificate, taking the form

$$[\![\ldots \mathbin{\hat{}} N_{P.Q} \mathbin{\hat{}} N_{Q.P} \mathbin{\hat{}} h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P)]\!]_P.$$

This transforming edge completes an incoming test for Q, assuming P's signature key is uncompromised, as shown (right-to-left) in the lower rectangle in Figure 4. Q knows that this signature was generated after $N_{Q.P}$ was created. Moreover, if P is behaving properly, then this signature is emitted only in a run that also caused the origination of $N_{P.Q}$. Thus, m_2 is recent for n_2, and m_0 is recent for m_2. Therefore, m_0 is 2-recent for n_2.

$$\begin{tikzcd}
P & Q \\
m_0 \arrow[r, dashed] \arrow[d, Rightarrow] & n_0 \arrow[d, Rightarrow] \\
m_1 \arrow[d, Rightarrow] & n_1 \arrow[l, dashed, "N_{Q.P} \,\hat{}\, \llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \rrbracket_Q"'] \arrow[d, Rightarrow] \\
m_2 \arrow[r, dashed, "\llbracket \ldots \,\hat{}\, N_{P.Q} \,\hat{}\, N_{Q.P} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \rrbracket_P"] & n_2
\end{tikzcd}$$

Figure 4: Edges Achieving Authentication, II

Q can also use the signed component in the bottom line of Figure 4 as non-repudiation evidence, to establish the **Authentication, II** guarantee to a third party. In this case, Q must be willing to disclose the values $\mathsf{sec}_{P.Q}$ and shared_P.

4.3 Distinguishing the Subprotocols

The protocol as described in Figure 4 is a two party protocol between P and Q. We want a three party protocol involving C, M, and B, in which each successively plays the role of P and the role of Q with each of the other principals. We will want to interweave these protocols without undermining the guarantees that each of them would provide if executed purely in isolation.

By [13], it suffices that no encrypted unit emitted in one subprotocol could have been emitted in any other. One can achieve this by assigning each encrypted component an identifying tag showing to which subprotocol it belongs.

Since the behavior of Figure 4 occurs with any of the principals C, M, B as P and any of other principal as Q, we have six possibilities. We select, then, six distinct constants $c_1, \ldots, c_6$, which we refer to as $\mathsf{C.M}$, $\mathsf{C.B}$, etc. Here we do not intend C, M, and B as names for particular principals, but as constants referring to the three roles. We use the sans serif font to emphasize that they are constants, not variables referring to the identities of the participants. Even if the same participant plays multiple roles in a single protocol run, these constants distinguish which message component is sent as part of which role.

We will also include a constant distinguishing the messages; although this is strictly unnecessary, it may ease understanding. We will use S in message 1, indicating its role in achieving secrecy; we will use A in message 2, indicating its role in achieving the first authentication goal; and we will use R in message 3, indicating its role in achieving the recency guarantee.

Each subprotocol, involving roles P and Q, takes the form shown in Figure 5. We refer to an individual subprotocol as $\textsc{atspect}_{\mathsf{P.Q}}$, and we refer to the union of all strands containing behaviors according to any of the six subprotocols as $\textsc{atspect}^{\dagger}$. An *initiator strand* is one taking the form shown in the left column of Figure 5, and a *responder strand* takes the form shown in the right

$$\begin{tikzcd}
P & & Q \\
m_0 \arrow[rr, dashed, "\{\!| c_i \,\hat{}\, \mathsf{S} \,\hat{}\, N_{P.Q} \,\hat{}\, \mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P |\!\}_Q"] \arrow[d, Rightarrow] & & n_0 \arrow[d, Rightarrow] \\
m_1 \arrow[d, Rightarrow] & & n_1 \arrow[ll, dashed, "N_{Q.P} \,\hat{}\, \llbracket\, c_i \,\hat{}\, \mathsf{A} \,\hat{}\, N_{P.Q} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \,\rrbracket_Q"'] \arrow[d, Rightarrow] \\
m_2 \arrow[rr, dashed, "\llbracket\, c_i \,\hat{}\, \mathsf{R} \,\hat{}\, N_{P.Q} \,\hat{}\, N_{Q.P} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \,\rrbracket_P"] & & n_2
\end{tikzcd}$$

Figure 5: Subprotocol $P.Q$

column of Figure 5. The parameters of an initiator or responder strand are the variables P, Q (representing the identities of the participants), $N_{P.Q}, N_{Q.P}$ (their respective nonces), and $\mathsf{sec}_{P.Q}$ and shared_P (the secret and shared payloads).

5 Correctness

We address the correctness of the individual subprotocols first, and then make sure that they remain correct even when all are executed by the same principals over the same network.

5.1 Correctness of the Subprotocols

Let us focus on subprotocol ATSPECT$_{\mathsf{P.Q}}$ as defined in Figure 5. We identified six goals, of which four are essentially two-party goals. We will now formulate each of those as a theorem about the protocol ATSPECT$_{\mathsf{P.Q}}$. We let $\mathcal{B}$ be a bundle in which the regular participants execute strands of ATSPECT$_{\mathsf{P.Q}}$. Recall from Section 3.1 that if a key K is *safe* in $\mathcal{B}$ (written $K \in \mathsf{S}$), then the penetrator can never use K for encryption or decryption. In this section, italics letters such as P and Q are variables over principals, while sans serif letters such as $\mathsf{P.Q}$ refer to a constant such as $\mathsf{C.M}$, which labels one particular subprotocol.

Proposition 5.1 (Confidentiality, I) Suppose that $\mathcal{B}$ is a ATSPECT$_{\mathsf{P.Q}}$-bundle in which Q's private decryption key is safe, and suppose $\mathcal{B}$ has an Init-strand $\mathrm{Init}[P, Q, N_{P.Q}, N_{Q.P}, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$.

If $\mathsf{sec}_{P.Q}$ is uniquely originating, then there is no node $n \in \mathcal{B}$ such that $\mathrm{term}(n) = \mathsf{sec}_{P.Q}$.

PROOF. Let κ be the set of inverses of unsafe keys, i.e. $(\mathsf{K} \setminus \mathsf{S})^{-1}$. Let τ be $\{\mathsf{sec}_{P.Q}\} \cup \mathsf{S}$. By the honest ideal theorem, [21, Corollary 6.12], if there is a node $m \in \mathcal{B}$ with $\mathrm{term}(m) \in I_\kappa[\tau]$, then there a regular node n that is an entry point for $I_\kappa[\tau]$. However, inspecting the positive regular nodes of ATSPECT$_{P.Q}$, we see that no value in τ is ever sent, unless protected by a key whose inverse is safe. ■

Assuming that $\mathsf{sec}_{P.Q}$ is uniquely originating is essentially a way of assuming that no one guesses this value, and the theorem states that no one can discover it without guessing correctly. If the values $\mathsf{sec}_{P.Q}$ and shared_P are predictable, then someone may guess correctly. This would be unfortunate, because the protocol transmits $h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P)$, and this confirms a correct guess. That is why we insert some unpredictable ingredient into either $\mathsf{sec}_{P.Q}$ or shared_P.

The **Confidentiality, II** goal of secrecy for shared_P is a property of the composite protocol, as it is transmitted in more than one subprotocol. We will prove this in Section 6.4. In the remaining propositions, we use the notion of the $\mathcal{B}$-height of a strand (Definition A.4); the $\mathcal{B}$-height of a strand s is the number of nodes of s contained in $\mathcal{B}$.

Proposition 5.2 (Authentication, I) Suppose that $\mathcal{B}$ is an ATSPECT$_{\mathsf{P.Q}}$-bundle in which Q's private signature key K is safe, and suppose $\mathcal{B}$ has an initiator strand

$$\mathrm{Init}[P, Q, N_{P.Q}, N_{Q.P}, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$$

of $\mathcal{B}$-height at least 2. If $N_{P.Q}$ is uniquely originating, then $\mathcal{B}$ has a matching responder strand

$$\mathrm{Resp}[P, Q, N_{P.Q}, X, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$$

of $\mathcal{B}$-height at least 2 (for some X).

Proof. Apply the inbound authentication test, given that $K \in \mathsf{S}$ and $N_{P.Q}$ is uniquely originating. The only transforming edge producing

$$[\![\, c_i \mathbin{\hat{}} A \mathbin{\hat{}} N_{P.Q} \mathbin{\hat{}} h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P) \,]\!]_Q$$

is the first edge of a responder strand $\mathrm{Resp}[P, Q, N_{P.Q}, X, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$.

Because P does not occur explicitly in the initiator's message, the claim that the first parameter to the responder strand is P relies on the assumption that $\mathsf{core}(\mathsf{shared}_P)$ determines that the initiator is P (Section 4.1). ■

This proposition depends only on Q's signature key being safe, and the non-repudiation guarantee derives from this. P need not establish that it has behaved honestly, nor that he generated $N_{P.Q}$ in such a way as to make it originate uniquely.

Proposition 5.3 (Non-Repudiation) Suppose that $\mathcal{B}$ is a ATSPECT$_{\mathsf{P.Q}}$-bundle in which Q's private signature key K is safe, and suppose there exists a node $n \in \mathcal{B}$ such that $[\![\, c_i \mathbin{\hat{}} A \mathbin{\hat{}} N_{P.Q} \mathbin{\hat{}} h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P) \,]\!]_Q \sqsubset \mathrm{term}(n)$. Then there is a responder strand

$$\mathrm{Resp}[P, Q, N_{P.Q}, X, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$$

of $\mathcal{B}$-height at least 2 (for some X).

PROOF. Immediate consequence of the unsolicited test principle, together with the observation that no other strand emits a term with any subterm of the form $[\![\, c_i \mathbin{\hat{}} A \mathbin{\hat{}} N_{P.Q} \mathbin{\hat{}} h(\mathsf{sec}_{P.Q} \mathbin{\hat{}} \mathsf{shared}_P)\,]\!]_Q$. ■

Proposition 5.4 (Authentication, II) Suppose that $\mathcal{B}$ is a ATSPECT$_{\mathsf{P.Q}}$ bundle in which P's private signature key K is safe, and

$$s \in \mathrm{Resp}[P, Q, N_{P.Q}, N_{Q.P}, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$$

has $\mathcal{B}$-height 3. Then there exists $s' \in \mathrm{Init}[P, Q, N_{P.Q}, N_{Q.P}, \mathsf{sec}_{P.Q}, \mathsf{shared}_P]$ with $\mathcal{B}$-height 3, and $\langle s', 1\rangle$ is 2-recent for $\langle s, 2\rangle$.

PROOF. The existence of the originator strand s' follows immediately from the inbound authentication test principle. Moreover, by the inbound authentication test recency guarantee, $\langle s, 2\rangle \prec \langle s', 2\rangle$. Thus, the node $\langle s', 1\rangle$, where $N_{P.Q}$, $\mathsf{sec}_{P.Q}$, and shared_P originate, is 2-recent for $\langle s, 2\rangle$. ■

We have now established the security goals of ATSPECT, as holding of the individual subprotocols ATSPECT$_{\mathsf{P.Q}}$, except the secrecy property for $\mathsf{shared}_{P.Q}$.

5.2 Independence of the Subprotocols

A primary protocol Σ_1 is *independent* of other protocols (jointly called the secondary protocol Σ_2) if the question whether the primary protocol achieves a security goal never depends on whether that secondary protocol is in use. In [13] we prove that the independence of Σ_1 from Σ_2 follows from "disjoint encryption." This condition has a somewhat technical definition to allow public key certificates or Kerberos-style tickets to be created in Σ_1 and consumed in Σ_2. However, a simple sufficient condition is "strongly disjoint encryption:"

> A primary protocol Σ_1 and secondary protocol Σ_2 have *strongly disjoint encryption* if, whenever n_1 is a node on some strand of Σ_1, n_2 is a node on some strand of Σ_2, and $\{\!|h|\!\}_K \sqsubset (n_1)$, then $\{\!|h|\!\}_K \not\sqsubset (n_2)$.

This is exactly why we included the constants $c_1, \ldots, c_6$, which we write as $\mathsf{C.M}$, etc. Let Σ_1 be ATSPECT$_{\mathsf{P.Q}}$, and letting Σ_2 be all strands of the protocols ATSPECT$_{\mathsf{P'.Q'}}$, where $\mathsf{P'} \neq \mathsf{P}$ or $\mathsf{Q'} \neq \mathsf{Q}$. If $\{\!|h|\!\}_K$ is sent or received on a strand of Σ_1, then h begins with the constant $\mathsf{P.Q}$. If $\{\!|h'|\!\}_{K'}$ is sent or received on a strand of Σ_1, then h begins with the constant $\mathsf{P'.Q'}$, which is different from $\mathsf{P.Q}$. Therefore $\{\!|h|\!\}_K \neq \{\!|h'|\!\}_{K'}$.

Thus, if ATSPECT$_{\mathsf{P.Q}}$ achieves a security goal in isolation, it achieves the same goal when run together with all of the protocols ATSPECT$_{\mathsf{P'.Q'}}$. We call the union of all these protocols ATSPECT†, so we have concluded that ATSPECT† achieves the goals of the individual protocols ATSPECT$_{\mathsf{P.Q}}$.

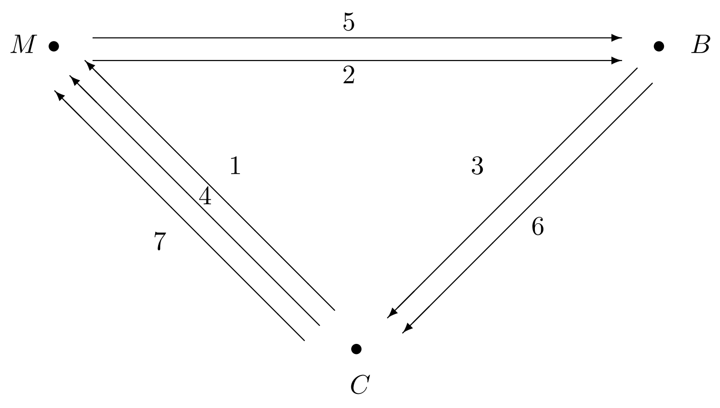

Figure 6: Message Flow for ATSPECT

6 A Three Party Protocol

At this stage, we need only design the message structure of the combined, three party protocol. There are numerous possibilities here. For instance, in theory the principals C, M, and B could simply asynchronously engage in ATSPECT†, i.e. in interleaved runs of the six subprotocols. This would not be incorrect, but it would be rather anarchic, and unlikely to complete transactions promptly.

Instead, we will construct a more structured way of interweaving the protocols. We seek to achieve two goals in doing so. One is the confidentiality for the shared message ingredients shared_P, **Confidentiality, II**. The other is

Three-Party Agreement Suppose that P completes a run of ATSPECT with apparent interlocutors Q and R. Then Q and R have begun runs of ATSPECT with

$$\mathsf{core}(\mathsf{shared}_P) = \mathsf{core}(\mathsf{shared}_Q) = \mathsf{core}(\mathsf{shared}_R).$$

In some sense the collection of two-party protocols ATSPECT† contains the essence of our protocol; ATSPECT adds only a convenient temporal ordering for the subprotocols, with the added constraint that **Three-Party Agreement** holds of this ordering. Alternate orderings could also serve as well.

6.1 A Triangular Message Structure

The first ordering we will present has the message structure shown in Figure 6. The seven messages flow around a triangle. C, who initiates the exchange, sends three messages, and the other principals each send two. The sequence of events is determined by three principles:

1. C begins the exchange with $\mathsf{C.M}$ and $\mathsf{C.B}$. M and B begin their subprotocols on receiving messages from C and M respectively.
2. Each principal, on receiving a component intended for it in a subprotocol, constructs and transmits the next component in that subprotocol.
3. Each principal, receiving a component not intended for it, forwards it to the next principal.

Since some shorthand is useful, we will refer to the message components in the following way:

$S_{\mathsf{P.Q}}$ Payload-bearing units, taking the form

$$\{\!|\mathsf{P.Q} \,\hat{}\, \mathsf{S} \,\hat{}\, N_{P.Q} \,\hat{}\, \mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P|\!\}_Q$$

The subscript $\mathsf{P.Q}$ indicates that this component is prepared by P for Q's consumption.

$A_{\mathsf{P.Q}}$ Authenticators, taking the form

$$N_{Q.P} \,\hat{}\, [\![\, \mathsf{P.Q} \,\hat{}\, \mathsf{A} \,\hat{}\, N_{P.Q} \,\hat{}\, h(\mathsf{sec}_{P.Q} \,\hat{}\, \mathsf{shared}_P) \,]\!]_Q$$

where the subscript $\mathsf{P.Q}$ indicates that it authenticates Q's receipt of $S_{\mathsf{P.Q}}$.

$R_{\mathsf{P.Q}}$ Recency confirmations, taking the form

$$[\![\, \mathsf{P.Q} \,\hat{}\, \mathsf{R} \,\hat{}\, N_{P.Q} \,\hat{}\, N_{Q.P} \,\hat{}\, h(\mathsf{sec}_{Q.P} \,\hat{}\, \mathsf{shared}_Q) \,]\!]_P$$

where the subscript $\mathsf{P.Q}$ indicates that P vouches that it has freshly generated $N_{P.Q}$, and has received $S_{\mathsf{Q.P}}$ and $A_{\mathsf{P.Q}}$.

Using the three principles for ordering message components, we derive the message sequence shown in Figure 7. Each message consists of three portions, containing zero or more payload-bearing units, followed by zero or more authenticators and zero or more recency confirmations. In early messages, payloads predominate, while progressively authenticators and finally recency confirmations emerge. We require each principal to check that its shared data agrees with the shared data sent by the others. In M's case (e.g.), this means that shared_C, as extracted from $S_{\mathsf{C.M}}$, matches shared_B, as extracted from $S_{\mathsf{B.M}}$, both of which match the value shared_M as transmitted by M. B makes this check before sending message 3; C, before sending message 4; and M, before sending message 5. They refuse to continue the protocol by sending new authenticators or recency components if this check fails. Observe that P prepares components that will be received by both interlocutors after making this check; for instance, M transmits $R_{\mathsf{M.B}}$ and $R_{\mathsf{M.C}}$ in message 5. Thus, when B receives $R_{\mathsf{M.B}}$, it gives conclusive evidence that M's check succeeded.

This protocol requires the party playing a role P to generate four nonces, two within the secrecy units $S_{\mathsf{P.Q}}$ and $S_{\mathsf{P.R}}$ and two within the authenticators $A_{\mathsf{Q.P}}$ and $A_{\mathsf{R.P}}$. If we choose four distinct string constants $s_1, \ldots, s_4$, then we can generate all four nonces from a single random value N of reasonable length, using the four hashed values $h(N \,\hat{}\, s_i)$.

1. $C \rightarrow M$ $\quad S_{\mathsf{C.M}} \,\hat{}\, S_{\mathsf{C.B}}$
2. $M \rightarrow B$ $\quad S_{\mathsf{C.B}} \,\hat{}\, S_{\mathsf{M.B}} \,\hat{}\, S_{\mathsf{M.C}} \quad \hat{} \quad A_{\mathsf{C.M}}$
3. $B \rightarrow C$ $\quad S_{\mathsf{M.C}} \,\hat{}\, S_{\mathsf{B.C}} \,\hat{}\, S_{\mathsf{B.M}} \quad \hat{} \quad A_{\mathsf{C.M}} \,\hat{}\, A_{\mathsf{C.B}} \,\hat{}\, A_{\mathsf{M.B}}$
4. $C \rightarrow M$ $\quad S_{\mathsf{B.M}} \quad \hat{} \quad A_{\mathsf{M.B}} \,\hat{}\, A_{\mathsf{M.C}} \,\hat{}\, A_{\mathsf{B.C}} \quad \hat{} \quad R_{\mathsf{C.M}} \,\hat{}\, R_{\mathsf{C.B}}$
5. $M \rightarrow B$ $\quad A_{\mathsf{B.C}} \,\hat{}\, A_{\mathsf{B.M}} \quad \hat{} \quad R_{\mathsf{C.B}} \,\hat{}\, R_{\mathsf{M.B}} \,\hat{}\, R_{\mathsf{M.C}}$
6. $B \rightarrow C$ $\quad R_{\mathsf{M.C}} \,\hat{}\, R_{\mathsf{B.C}} \,\hat{}\, R_{\mathsf{B.M}}$
7. $C \rightarrow M$ $\quad R_{\mathsf{B.M}}$

Figure 7: Full Message Flow

6.2 A Straightened Version

The triangular message flow has a disadvantage from the implementer's point of view: it does not match smoothly with the normal conventions of programming with TCP/IP and the standard socket library. To solve this problem, we can revise the message flow, adapting it to use eight messages:

$$\begin{array}{lllll}
C & \xrightarrow{1} & M & \xrightarrow{2} & B \\
B & \xrightarrow{3} & M & \xrightarrow{4} & C \\
C & \xrightarrow{5} & M & \xrightarrow{6} & B \\
B & \xrightarrow{7} & M & \xrightarrow{8} & C
\end{array}$$

This has the advantage that it may be implemented using a pair of socket connections, one between C and M, and one between M and B. There are two disadvantages to this alternative, first, the extra message, and second, that M controls all communication between C and B, which occurs only when M forwards components. We regard the triangular protocol of Section 6.1 as the authoritative version of ATSPECT, although the straightened eight-message version achieves the same protocol goals.

In practice, it may be unnecessary to use all six subprotocols. For instance, the subprotocols C.M, C.B, and M.B may suffice. In this case, we may want to augment the authenticator with some additional payload of information to be communicated back from responder to initiator. Truncated message flows may be based either on the triangular scheme or the straightened scheme. A truncated message flow based on the triangular scheme is displayed in Figure 8.

6.3 ATSPECT Protocol Usage

ATSPECT can be used in various ways, with differing interpretations of the individual messages and differing actions accompanying the protocol steps. We will illustrate the workings of the protocol in a specific scenario, focusing on

1. $C \to M \quad S_{\mathsf{C.M}} \,\hat{}\, S_{\mathsf{C.B}}$
2. $M \to B \quad S_{\mathsf{C.B}} \,\hat{}\, S_{\mathsf{M.B}} \quad \hat{} \quad A_{\mathsf{C.M}}$
3. $B \to C \quad A_{\mathsf{C.M}} \,\hat{}\, A_{\mathsf{C.B}} \,\hat{}\, A_{\mathsf{M.B}}$
4. $C \to M \quad A_{\mathsf{M.B}} \quad \hat{} \quad R_{\mathsf{C.M}} \,\hat{}\, R_{\mathsf{C.B}}$
5. $M \to B \quad R_{\mathsf{C.B}} \,\hat{}\, R_{\mathsf{M.B}}$

Figure 8: Truncated Message Flow

the truncated version of Figure 8. We aim to explain the practical value of the guarantees offered by the protocol.

In this scenario, we interpret $S_{\mathsf{C.M}}$ as placing an order with the merchant, so $A_{\mathsf{C.M}}$ accepts the order and undertakes to ship the goods, conditional on B delivering payment. $S_{\mathsf{C.B}}$ requests payment to the merchant's account, and $A_{\mathsf{C.B}}$ gives B's certification that there are sufficient funds and that B has accepted any relevant conditions on the transfer. The recency certificates $R_{\mathsf{C.M}}$ and $R_{\mathsf{C.B}}$ certify to M and B that the order and payment are fresh, rather than replays. $S_{\mathsf{M.B}}$ proposes payment terms, including B's fee for handling the transaction. $A_{\mathsf{M.B}}$ accepts those conditions, and $R_{\mathsf{M.B}}$ ensures that the proposal was fresh.

The crucial extra-protocol action is for B to transfer funds into M's account, debiting C. In a normal run, this occurs after B receives Message 5. B's **Authentication, II** guarantees then allow it to justify the transfer if challenged by C or by a regulatory authority.

What should happen if one of the participants interrupts execution part-way through a run? As we mentioned at the beginning of Section 2, this protocol is not intended to achieve fairness, so fully acceptable answers may not exist if execution is interrupted at certain stages. However, if C presents B with the authenticators $A_{\mathsf{C.M}}$ and $A_{\mathsf{C.B}}$, confirming that both other participants have accepted C's proposal, then B should effect the funds transfer and require M to provide evidence that the goods have been shipped (possibly after some delay to restock inventory). If M presents B with a recency certificate $R_{\mathsf{C.M}}$, and evidence that the goods have been shipped, then B should effect the transfer to M. A careful treatment of this supplementary protocol would define a message format, signed by C or M respectively, to deliver these payloads.

Why are these resolution procedures reasonable? If C possesses the authenticators $A_{\mathsf{C.M}}$ and $A_{\mathsf{C.B}}$, then the **Authentication, I** guarantees for the $\mathsf{C.M}$ and $\mathsf{C.B}$ subprotocols imply that M and B have executed the first two nodes of runs matching C's. Thus, it is reasonable to require M to make the shipment when the funds are delivered. Likewise, if M possesses a recency certificate $R_{\mathsf{C.M}}$, then his **Authentication, II** guarantee implies that C has completed a strand requesting this merchandise. Moreover, C's **Authentication, I** guarantee and M's **Authentication, II** guarantee together ensure that if either of them choose nonces with a high degree of randomness, then these two strands are at worst 2-recent for each other. Evidence that the goods have been shipped should entitle the merchant to be paid.

Turning to the full message flow defined in Figure 7, we have more opportunities for the participants to negotiate terms within the framework of the protocol. For instance, $\text{ATSPECT}_{\mathsf{M.C}}$ can be used to propose and accept a shipping schedule, and $\text{ATSPECT}_{\mathsf{M.B}}$ can be used to propose and accept a transaction fee between M and B. Again, in this version of the protocol, the crucial event is Message 5. When B receives this message, it possesses **Authentication, II** guarantees for runs of $\text{ATSPECT}_{\mathsf{C.B}}$ and $\text{ATSPECT}_{\mathsf{M.B}}$, so that recent executions by C and M are occurring. Thus, B is now justified in transferring the funds.

6.4 ATSPECT's Three-Party Goals

We turn now to the last correctness concerns, whether ATSPECT achieves confidentiality for shared_P and the **Three-Party Agreement** goal.

Proposition 6.1 (Confidentiality, II) Suppose $\mathcal{B}$ is a bundle in which P completes a run of ATSPECT with interlocutors Q and R, using shared component shared_P, and all three principals have safe private decryption keys.

If shared_P is uniquely originating, then there is no node $n \in \mathcal{B}$ such that $\text{term}(n) = \mathsf{shared}_P$.

Proof. Apply the honest ideal theorem to $\kappa = (\mathsf{K} \setminus \mathsf{S})^{-1}$ and $\tau = \{\mathsf{shared}_P\} \cup \mathsf{S}$, to infer that $I_\kappa[\tau]$ has only regular entry points. But all regular nodes transmit shared_P encrypted with a key whose inverse is safe. ■

Proposition 6.2 (Three-Party Agreement) Suppose $\mathcal{B}$ is a bundle in which P completes a run of ATSPECT with interlocutors Q and R, using shared component shared_P. Then if Q's signature key is safe, Q has begun a run of ATSPECT with P and R, with shared components shared_Q and shared_R, and

$$\mathsf{core}(\mathsf{shared}_P) = \mathsf{core}(\mathsf{shared}_Q) = \mathsf{core}(\mathsf{shared}_R).$$

Proof. Q transmits either $A_{\mathsf{P.Q}}$ or $R_{\mathsf{P.Q}}$ after receiving both $S_{\mathsf{P.Q}}$ and $S_{\mathsf{R.Q}}$; it therefore guarantees to P that the shared values in these components match (Section 6.1). P does not transmit its last message until after P has received this guarantee from Q.

Moreover, P has received $S_{\mathsf{R.Q}}$ and has the shared value matches shared_Q as contained in $S_{\mathsf{Q.P}}$ and shared_P as P transmitted it in $S_{\mathsf{P.Q}}$ and $S_{\mathsf{P.R}}$. shared_Q as transmitted in $S_{\mathsf{Q.R}}$ matches because Q is assumed uncompromised. Thus, all six values match. ■

7 Related Work

Woo and Lam's 1994 paper on protocol design [23] diagnosed the faulty design process leading to a protocol in an earlier paper [22]. They focused on how to safely remove information from a "full information" but inefficient version of a protocol to a less cluttered version. There are two limitations to their approach. First, no guidance is given about how to construct a full information protocol to achieve given goals, especially if these goals are complex, as in ATSPECT. Second, the criteria for safely removing information seem fragile. One might well wonder whether they are always valid, or whether there are ambiguities in how to apply them.

Buttyan et al. [4] describe a BAN-style logic that they say motivates a design method, but it seems hard to abstract the method from the example they give.

Perrig and Song's automated protocol generator APG [18] uses heuristics related to ours to generate plausible candidate protocols. APG then calls Athena [20] to use the strand space model to filter protocols, retaining those proved to meet their specifications. APG does not, however, capitalize on protocol independence to decompose the design process and to synthesize protocols from two-party subprotocols.

The bulk of work on protocol design seems to rely on the skill and ingenuity of the designer. Notable here is Abadi and Needham [1], which contains a wealth of information about cryptographic protocols, what makes them correct, and how to design them so that they will be. However, they make no claim to be systematic, nor do they base their advice on a theory of protocol goals and correctness.

8 Conclusion

In this paper, we have illustrated a protocol design methodology, based on the authentication tests. The method has led to a protocol, ATSPECT, that demonstrably meets precisely stated security goals. The ATSPECT design process required less than three weeks of labor, by contrast with the major effort invested in SET. ATSPECT appears to provide security guarantees similar to those of SET.

The design process has the following steps:

1. Formulate a number of precise goals that the protocol is intended to meet, such as those of Section 2.2. Goals that concern a subset of the principals may be achieved using subprotocols involving only those principals.

2. For each goal, select an authentication test pattern to use to achieve it, and design a transforming edge that will satisfy this authentication goal but no other, as in Section 4.2.1. Verify the subprotocols achieve the individual goals (Section 5.1). Use disjoint encryption to ensure that subprotocols are independent (Section 5.2).

3. Piece the subprotocols together to construct a single protocol as illustrated in Sections 6.1–6.2, and justified in Section 6.4. There is freedom in choosing the combination, allowing trade-offs in number of messages and in communication pattern.

More refined methods may improve the last step, in which the subprotocols are combined, by indicating encrypted components that can be merged or simplified (cp. [17] for transformations on concatenated submessages).

Our protocol design method shows how to construct special-purpose protocols for specific situations in secure communication or electronic commerce. It allows us to meet varied trust objectives with a conceptual toolkit justified by strand spaces and the authentication tests.

Acknowledgments I am grateful to Sylvan Pinsky for encouragement, support, and technical discussions. David Basin challenged me to use the authentication test heuristics for the design of a better, simpler electronic commerce protocol. Andy Gordon and Alan Jeffrey commented on an earlier version and applied their type system for authentication [7, 8]. Gavin Lowe pointed out the importance of having an unpredictable ingredient in the payload.

References

[1] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. In *Proceedings, 1994 IEEE Symposium on Research in Security and Privacy*, pages 122–136. IEEE, IEEE Computer Society Press, 1994.

[2] G. Bella, F. Massacci, and L. C. Paulson. Verifying the SET purchase protocols. Technical report, Cambridge University Computer Laboratory, 2001. Short version appeared in International Joint Conference on Automated Reasoning, June, 2001. Available at http://www.cl.cam.ac.uk/users/lcp/papers/protocols.html.

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93 Proceedings*. Springer-Verlag, 1993. Full version available at http://www-cse.ucsd.edu/users/mihir/papers/eakd.ps.

[4] L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *11th IEEE Computer Security Foundations Workshop*, pages 153–162, 1998.

[5] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *Proceedings, 8th ACM Conference on Computer and Communications Security*, pages 176–185, New York, November 2001. ACM Press.

[6] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[7] A. Gordon and A. Jeffrey. Authentication by typing. In *Proceedings, 14th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2001.

[8] A. Gordon and A. Jeffrey. Types and effects for asymmetric cryptographic protocols. In *Proceedings, 15th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2002.

[9] J. D. Guttman. Key compromise and the authentication tests. *Electronic Notes in Theoretical Computer Science*, 47, 2001. Editor, M. Mislove. URL `http://www.elsevier.nl/locate/entcs/volume47.html`, 21 pages.

[10] J. D. Guttman. Security goals: Packet trajectories and strand spaces. In R. Gorrieri and R. Focardi, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 197–261. Springer Verlag, 2001.

[11] J. D. Guttman, F. J. Thayer, and L. D. Zuck. The faithfulness of abstract protocol analysis: Message authentication. *Journal of Computer Security*, 2003. Forthcoming.

[12] J. D. Guttman and F. J. Thayer Fábrega. Authentication tests. In *Proceedings, 2000 IEEE Symposium on Security and Privacy.* May, IEEE Computer Society Press, 2000.

[13] J. D. Guttman and F. J. Thayer Fábrega. Protocol independence through disjoint encryption. In *Proceedings, 13th Computer Security Foundations Workshop.* IEEE Computer Society Press, July 2000.

[14] J. D. Guttman and F. J. Thayer Fábrega. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.

[15] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Proceedings, 15th Computer Security Foundations Workshop.* IEEE Computer Society Press, June 2002.

[16] G. Lowe. Analyzing protocols subject to guessing attacks. *Journal of Computer Security*, 2003. Forthcoming.

[17] S.-L. Ng. Posets and protocols: Picking the right three-party protocol. *IEEE Journal on Selected Areas in Communication*, pages 55–61, January 2003.

[18] A. Perrig and D. X. Song. Looking for diamonds in the desert: Extending automatic protocol generation to three-party authentication and key agreement protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, July 2000.

[19] SET secure electronic transaction specification, May 1997. Available at http://www.setco.org/download.html.

[20] D. X. Song. Athena: a new efficient automated checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop.* IEEE Computer Society Press, June 1999.

[21] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[22] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, January 1992.

[23] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, pages 24–37, 1994.

A Strand Space Definitions

This appendix, derived from [10, 14, 21], defines the basic strand space notions.

A.1 Strands, Strand Spaces, and Origination

Consider a set A, the elements of which, called terms, are the possible messages to be exchanged between principals in a protocol. A *subterm* relation $\sqsubset$ is defined on A.

In a protocol, principals send and receive terms. We represent transmission of a term with a positive sign, and reception of a term with a negative sign.

Definition A.1 *A* signed term *is a pair* $\langle \sigma, a \rangle$ *with* $a \in \mathsf{A}$ *and* σ *one of the symbols* $+, -$. *We will write a signed term as* $+t$ *or* $-t$. $(\pm\mathsf{A})^*$ *is the set of finite sequences of signed terms. We will denote a typical element of* $(\pm\mathsf{A})^*$ *by* $\langle \langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle \rangle$.

A strand space *over* A *is a set* Σ *with a trace mapping* $\mathrm{tr} : \Sigma \to (\pm\mathsf{A})^*$.

By abuse of language, we often treat signed terms as ordinary terms. We represent strand spaces by their underlying set of strands Σ.

Definition A.2 *Fix a strand space Σ.*

1. *A* node *is a pair $\langle s, i\rangle$, with $s \in \Sigma$ and i an integer satisfying $1 \leq i \leq$ length$(tr(s))$. The set of nodes is denoted by $\mathcal{N}$.*

2. *If $n = \langle s, i\rangle \in \mathcal{N}$ then index$(n) = i$ and strand$(n) = s$. Define term(n) to be $(tr(s))_i$, i.e. the ith signed term in the trace of s.*

3. *There is an edge $n_1 \rightarrow n_2$ if and only if term$(n_1) = +a$ and term$(n_2) = -a$ for some $a \in \mathsf{A}$. Intuitively, the edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.*

4. *When $n_1 = \langle s, i\rangle$ and $n_2 = \langle s, i+1\rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Rightarrow n_2$. Intuitively, the edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand s. We write $n' \Rightarrow^+ n$ to mean that n' precedes n on the same strand.*

5. *An unsigned term t* occurs in *$n \in \mathcal{N}$ iff $t \sqsubset$ term(n).*

6. *Suppose I is a set of unsigned terms. The node $n \in \mathcal{N}$ is an* entry point *for I iff term$(n) = +t$ for some $t \in I$, and whenever $n' \Rightarrow^+ n$, term$(n') \notin I$.*

7. *An unsigned term t* originates *on $n \in \mathcal{N}$ iff n is an entry point for the set $I = \{t' : t \sqsubset t'\}$.*

8. *An unsigned term t is* uniquely originating in *a set of nodes $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n. The term t is* non-originating in *$S \subset \mathcal{N}$ iff there is no $n \in S$ such that t originates on n.*

$\mathcal{N}$ together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$ is a directed graph $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow)\rangle$.

A.2 Bundles and Causal Precedence

A *bundle* is a finite subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow)\rangle$, for which we can regard the edges as expressing the causal dependencies of the nodes.

Definition A.3 *Suppose $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, (\rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C})\rangle$ is a graph, where $\mathcal{N}_\mathcal{C} \subset \mathcal{N}$; $\rightarrow_\mathcal{C} \subset \rightarrow$; $\Rightarrow_\mathcal{C} \subset \Rightarrow$. $\mathcal{C}$ is a bundle if:*

1. *$\mathcal{N}_\mathcal{C}$ and $\rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C}$ are finite.*

2. *If $n_2 \in \mathcal{N}_\mathcal{C}$ and term(n_2) is negative, then there is a unique n_1 such that $n_1 \rightarrow_\mathcal{C} n_2$.*

3. *If $n_2 \in \mathcal{N}_\mathcal{C}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_\mathcal{C} n_2$.*

4. $\mathcal{C}$ is acyclic.

In conditions 2 and 3, it follows that $n_1 \in \mathcal{N}_\mathcal{C}$, because $\mathcal{C}$ is a graph.

Definition A.4 *A node n is in a bundle $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, \rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C} \rangle$, written $n \in \mathcal{C}$, if $n \in \mathcal{N}_\mathcal{C}$; a strand s is in $\mathcal{C}$ if all of its nodes are in $\mathcal{N}_\mathcal{C}$. The $\mathcal{C}$-*height *of a strand s is the largest i such that $\langle s, i \rangle \in \mathcal{C}$.*

Definition A.5 *If $\mathcal{S}$ is a set of edges, i.e. $\mathcal{S} \subset \rightarrow \cup \Rightarrow$, then $\prec_\mathcal{S}$ is the transitive closure of $\mathcal{S}$, and $\preceq_\mathcal{S}$ is the reflexive, transitive closure of $\mathcal{S}$.*

Proposition A.6 *Suppose $\mathcal{C}$ is a bundle. Then $\preceq_\mathcal{C}$ is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in $\mathcal{C}$ has $\preceq_\mathcal{C}$-minimal members.*

We regard $\preceq_\mathcal{C}$ as expressing causal precedence, because $n \prec_\mathcal{S} n'$ holds only when n's occurrence causally contributes to the occurrence of n'. When a bundle $\mathcal{C}$ is understood, we will simply write $\preceq$. Similarly, "minimal" will mean $\preceq_\mathcal{C}$-minimal.

A.3 Terms, Encryption, and Freeness

We specialize the set of terms A, assuming given:

- A set $\mathsf{T} \subseteq \mathsf{A}$ of texts (i.e. atomic messages).
- A set $\mathsf{K} \subseteq \mathsf{A}$ of cryptographic keys disjoint from T, equipped with a unary operator **inv** : $\mathsf{K} \rightarrow \mathsf{K}$. We assume that **inv** is an inverse mapping each member of a key pair for an asymmetric cryptosystem to the other, and each symmetric key to itself.
- Two binary operators **encr** : $\mathsf{K} \times \mathsf{A} \rightarrow \mathsf{A}$ and **join** : $\mathsf{A} \times \mathsf{A} \rightarrow \mathsf{A}$.

We follow custom and write **inv**(K) as K^{-1}, **encr**(K, m) as $\{|m|\}_K$, and **join**(a, b) as $a \,\hat{}\, b$.

We assume that A is freely generated.

Axiom 1 A *is freely generated from* T *and* K *by* **encr** *and* **join**.

Definition A.7 *The subterm relation $\sqsubset$ is defined inductively, as the smallest relation such that $a \sqsubset a$; $a \sqsubset \{|g|\}_K$ if $a \sqsubset g$; and $a \sqsubset g \,\hat{}\, h$ if $a \sqsubset g$ or $a \sqsubset h$.*

By this definition, for $K \in \mathsf{K}$, we have $K \sqsubset \{|g|\}_K$ only if $K \sqsubset g$ already.

A.4 Penetrator Strands

The atomic actions available to the penetrator are encoded in a set of *penetrator traces.* They summarize his ability to discard messages, generate well known messages, piece messages together, and apply cryptographic operations using keys that become available to him. A protocol attack typically requires hooking together several of these atomic actions.

The actions available to the penetrator are relative to the set of keys that the penetrator knows initially. We encode this in a parameter, the set of penetrator keys $\mathsf{K}_{\mathcal{P}}$.

Definition A.8 *A* penetrator trace *relative to* $\mathsf{K}_{\mathcal{P}}$ *is one of the following:*

$\mathbf{M}_t$ *Text message:* $\langle +t \rangle$ *where* $t \in \mathsf{T}$.

$\mathbf{K}_K$ *Key:* $\langle +K \rangle$ *where* $K \in \mathsf{K}_{\mathcal{P}}$.

$\mathbf{C}_{g,h}$ *Concatenation:* $\langle -g,\ -h,\ +g \,\hat{}\, h \rangle$

$\mathbf{S}_{g,h}$ *Separation:* $\langle -g \,\hat{}\, h,\ +g,\ +h \rangle$

$\mathbf{E}_{h,K}$ *Encryption:* $\langle -K,\ -h,\ +\{\!|h|\!\}_K \rangle$.

$\mathbf{D}_{h,K}$ *Decryption:* $\langle -K^{-1},\ -\{\!|h|\!\}_K,\ +h \rangle$.

$\mathcal{P}_\Sigma$ *is the set of all strands* $s \in \Sigma$ *such that* $\mathrm{tr}(s)$ *is a penetrator trace.*

A strand $s \in \Sigma$ is a *penetrator strand* if it belongs to $\mathcal{P}_\Sigma$, and a node is a *penetrator node* if the strand it lies on is a penetrator strand. Otherwise we will call it a *non-penetrator* or *regular* strand or node. A node n is M, C, etc. node if n lies on a penetrator strand with a trace of kind M, C, etc.

Contents