Trust Management in Strand Spaces: A Rely-Guarantee Method*

Joshua D. Guttman, F. Javier Thayer, Jay A. Carlson, Jonathan C. Herzog, John D. Ramsdell, Brian T. Sniffen

The MITRE Corporation
guttman, jt, nop, jherzog, ramsdell, bsniffen@mitre.org

Abstract. We show how to combine trust management theories with nonce-based cryptographic protocols. The strand space framework for protocol analysis is extended by associating formulas from a trust management logic with the transmit and receive actions of the protocol principals. The formula on a transmission is a guarantee; the sender must ensure that this formula is true before sending the message. The formula on a receive event is an assumption that the recipient may rely on in deducing future guarantee formulas. The strand space framework allows us to prove that a protocol is sound, in the sense that when a principal relies on a formula, another principal has previously guaranteed it. We explain the ideas in reference to a simple new electronic commerce protocol, in which a customer obtains a money order from a bank to pay a merchant to ship some goods.

Cryptographic protocol analysis has aimed primarily to determine what messages another principal must have sent or received, when one principal is known to have sent or received certain messages. However, other questions are also important: what does a principal commit herself to when she executes a protocol? What assumptions must she accept, on the basis of her peers' assertions, to be willing to execute a protocol to the end? Answers to these questions spell out the *trust assumptions* of a protocol. We introduce here a method for reasoning about trust assumptions. The method clarifies the goals and consequences of engaging in a protocol, and the trust required to complete a protocol run.

Trust management allows principals to make access control decisions using a local policy to combine assertions made by their peers [18, 5]. The same local access control policy also controls the action of making assertions (including requests) to other principals. Cryptographic methods such as digital signatures are used to determine which principal uttered each assertion. A central advantage of trust management is that it handles naturally different principals who trust each other for some kinds of assertions, but not all. A major subtrend is *logical trust management* [18, 2, 19]. Here the local policy is a logical theory held by the principal, so that access control is decided by logical derivation.

* Supported by the MITRE-Sponsored Research Program. This paper appears in the European Symposium on Programming, April 2004.

Despite sophisticated academic work, trust management has seen limited uptake. It imposes a substantial security management burden on organizations that would use it. This burden is exacerbated by problems with key management and revocation. If the cryptographic secrets on which the method depends are poorly protected, then the likelihood of achieving benefits appears too low to offset the effort. From this point of view, Trusted Platform Modules (TPMs) [4], create an opportunity. These inexpensive cryptographic chips, now available in commercial personal computers, provide secure storage, on-chip cryptographic operations, and facilities to report securely on the system's software state. The TPM is organized around nonce-based protocols,[1] so remote principals receive freshness guarantees with the information they retrieve from TPM-equipped devices. Thus, the TPM is a promising platform for trust management [16], assuming trust management can effectively exploit nonce-based protocols.

Goal of this paper Here we aim to resolve one underlying theoretical question needed to provide a rigorous basis for using the TPM as a platform for trust management. That is, what forms of reasoning can soundly combine information from nonce-based protocols and trust management theories?

Our answer uses the well-developed strand space theory. Strand spaces allow us to determine what security goals a cryptographic protocol achieves [12, 24]; to decide when different cryptographic protocols may safely be combined [11]; to study interactions between protocols and the cryptography or message formatting used to implement them [13, 15]; and to guide protocol design [10, 23].

We now augment strand spaces with a rely-guarantee method [17]. The formulas are borrowed from a trust management logic, the choice of which is not tightly constrained by our method. The designer of a protocol annotates the behaviors of the principals with formulas. The formula associated with a message transmission must be *guaranteed* by the sender. Before sending the message, a principal obeying the protocol ensures the truth of the formula, presumably by combining reliable locally available data with guarantees offered earlier by other principals, using deduction in the local policy theory. The sender asserts the formula when sending the message. When another principal receives a message, that principal may *rely* on the fact that the sender has asserted the formula. The receiving principal can use the assertion in later deductions.

A protocol annotated with rely and guarantee formulas is *sound* if in every execution, whenever a principal receives a message and relies on a formula, there were corresponding message transmissions, guaranteeing assertions that are at least as strong. The existing methods of the strand space theory may be used to prove annotated protocols sound. They may also be used to prove that the guaranteeing message transmissions occurred *recently*, rather than involving time scales on which revocation or key compromise are realistic threats.

Section 1 introduces a simple illustrative protocol, based on money orders. Section 2 codifies the essential ingredients of logical trust management. The rely-

[1] A nonce is a randomly chosen bitstring, used in protocols to ensure freshness and avoid replay attacks.

guarantee method itself is in Section 3. Section 4 defines soundness and proves soundness for the example. Related work is in Section 5.

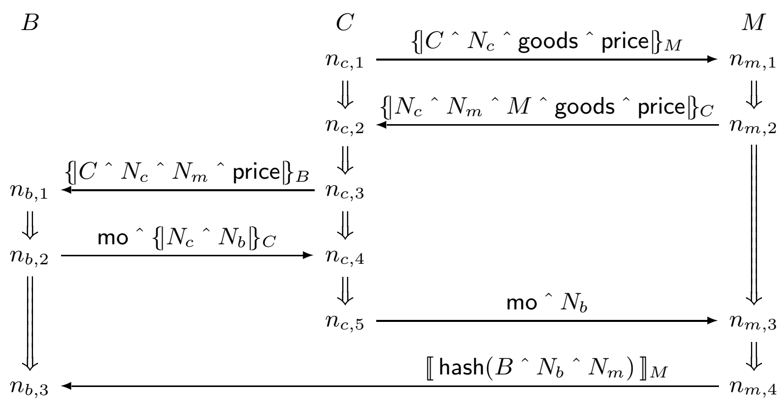

Fig. 1. EPMO with Money Order $\mathsf{mo} = [\![\, \mathsf{hash}(C \mathbin{\hat{}} N_c \mathbin{\hat{}} N_b \mathbin{\hat{}} N_m \mathbin{\hat{}} \mathsf{price}) \,]\!]_B$

1 Example: Electronic Purchase using Money Orders

The new protocol EPMO (Figure 1) borrows ideas from [22, 21], using the authentication tests as a design principle [10]. Variables N_p range over nonces; $[\![\, t \,]\!]_P$ is the message t signed by P; $\{\!|t|\!\}_P$ is t encrypted using P's public key; and $\mathsf{hash}(t)$ is a cryptographic hash of t.

A customer and a merchant want to agree on a purchase, transferring payment with the aid of a bank. Here goods is a description of the items requested; price is the proposed price. N_m serves as a transaction number. After obtaining a quote from the merchant, the customer obtains a "money order" containing N_b from the bank to cover the purchase, and delivers it to the merchant. The merchant "endorses" it by combining N_b with N_m, and delivers the endorsement to the bank. At the end of the protocol, the bank transfers the funds, and the merchant ships the goods; otherwise, the principals engage in a resolution protocol not specified here. B does not learn goods, and learns M only if the transaction completes. Although B does not transfer funds until M cashes the money order, B may put a "hold" on money in C's account. If the money order is not redeemed within an implementation-defined timeout period, then it expires and B releases the hold on C's account.

EPMO is designed not to disclose which principals are interacting, nor the goods or price. It does not protect against denial of service. An adversary can encrypt messages, using B's public key, to cause holds on all the money in C's account. Although a more complex protocol would prevent this attack, EPMO illustrates a more interesting interplay between protocols and trust.

2 Tenets of Logical Trust Management

A *trust management logic* consists of two ingredients, a language L and a consequence relation $\longrightarrow$. The language L is a set of formulas, some of which are assertions that we present in the form "P says ϕ," where $\phi \in L$. There may be other operators such as P $\mathsf{authorizes}$ ϕ. A formula is an *assertion* if it is of the form t says ϕ. Logical trust management is based on four underlying tenets.

1. Each principal P holds some set of statements as its theory Th_P. P derives conclusions within Th_P using other principals' utterances P' says ϕ as additional premises. P may utter P says ϕ after deriving ϕ.
2. Some assertions are self-certifying, namely P's statements about P's own utterances or desires. If P utters P says ϕ, then P has said something true. If P says P $\mathsf{authorizes}$ ϕ, then P has authorized it.
3. A principal P may have reliable knowledge of particular states of affairs. P draws only true conclusions about them, because P uses data available locally and sound inferences. For instance, a human resources department may reliably know the members of each department. It deduces that a person is an employee if there exists a department of which she is a member.
   Reliable knowledge is not a matter of logical form, however. Instead, a principal P_1 makes trust assumptions about other principals P_2. P_1 may believe that P_2 says ϕ implies ϕ. This implication expresses P_1's trust assumption that P_2 knows reliably about ϕ, and will speak truthfully.
   Th_P (from Tenet 1) includes both P's reliable knowledge and P's local policy for inferring conclusions.
4. A principal P_1 may control a resource r. P_1 takes an action $\phi(r, P_2)$ against r on behalf of another principal P_2, whenever P_1 derives a statement such as "P_2 requests $\phi(r, P_2)$ and P_2 should be permitted $\phi(r, P_2)$." Taking an action as a consequence of an inference is reminiscent of Aristotle's practical syllogism [3, Bekker 1144a31].

The derivations in Tenet 1 proceed according to the rules of inference of some underlying logic. In some recent work [19], these logics are simple, e.g. sublogics of Datalog. This has the advantage of a complete terminating deduction procedure, and some questions about trust between principals are efficiently decidable [20]. We adopt here the far stronger logic of [2], with syntactic sugar distinguishing "requests" and "authorizes" from "says".

3 A Rely-Guarantee Method for Strand Spaces

The rely-guarantee method was invented for reasoning about shared state parallel programs [17]. We adapt it here, with simplifications, for the case in which the "parallel program" or distributed system is a cryptographic protocol in which each regular (i.e. uncompromised) participant makes decisions according to a trust management theory. In this case, the shared state is the set of messages that have been sent and the set of formulas derived by each participant.

3.1 Annotated Protocols

A protocol designer must complete two successive steps, namely defining the protocol itself and then annotating the protocol.

Defining the Protocol A protocol is a finite set of parameterized strands [13]; see Appendix A for definitions. Each strand contains a number of nodes. Each node transmits or receives a message $\pm t$, where the message t may depend on the parameters. The forms of the strands are given by a set $\{S_j[\boldsymbol{X}]\}_{j \in J}$; the indices j are called the *roles* of the protocol. The behavior of the role j is like a template, containing parameters $\boldsymbol{X} = X_1, \ldots, X_k$. We regard this template as determining, for each assignment of values to the parameters, a set of strands; any two strands in this set transmit and receive the same values. We use the notation $S_j[X_1, \ldots, X_k]$ both to refer to the template itself, and also, when values are assigned to the parameters $\boldsymbol{X}$, to refer to the set of concrete strands whose behavior is to send and receive messages with these parameters.

Each parameterized strand $S[\boldsymbol{X}]$ has a distinguished parameter X_p which is the principal executing this strand. We write $\mathsf{prin}(S[X_1, \ldots, X_k]) = X_p$ and $\mathsf{prin}(n) = X_p$ if n is a node lying on this strand.

In the case of EPMO, the roles are *Bank*, *Customer*, and *Merchant*, each containing the send and receive events shown in the corresponding column of Figure 1. Letting p, g stand for the price and goods, the parameters to the roles and their distinguished parameters are:

$$\begin{array}{ll} s_b \in \mathsf{Bank}[B, C, M, p, N_m, N_b] & \mathsf{prin}(s_b) = B \\ s_c \in \mathsf{Cust}[B, C, M, p, g, N_c, N_m, N_b] & \mathsf{prin}(s_c) = C \\ s_m \in \mathsf{Merch}[B, C, M, p, g, N_c, N_m, N_b] & \mathsf{prin}(s_m) = M \end{array}$$

Each of these is a set of strands, i.e. all strands s such that $tr(s)$ is the sequence of sends and receives shown in one column of Figure 1, for the given values of the parameters.

Annotating the protocol Having defined a parametrized strand $S[\boldsymbol{X}]$ for each role, the protocol designer annotates them, attaching a formula to each node n.

If n is a positive (transmission) node, then the formula represents a guarantee that $\mathsf{prin}(n)$ asserts to its peers; we write these formulas as γ_n to emphasize their role as guarantees. If, for n positive, $\mathsf{prin}(n) = P$ and P holds theory Th_P, then the intended behavior of P at n is to attempt to derive γ_n within Th_P. Success means that P may continue the protocol and transmit term(n), in accord with Tenet 1; failure means that P must terminate executing this strand. Thus transmission of term(n) indicates that $\mathsf{prin}(n)$ says γ_n.

If n is a negative (reception) node, then $\mathsf{prin}(n)$ will rely on the formula associated with n, which we write ρ_n. Typically stating that other principals have made certain assertions, ρ_n is a premise that $\mathsf{prin}(n)$ can use in future deductions. In deriving γ_m for a later node m, P works from $\mathsf{Th}_P \cup \{\rho_n \colon n \Rightarrow^+ m\}$, i.e. the original theory augmented by rely statements ρ_n for earlier nodes on the strand.

The formulas γ_n and ρ_n may involve the parameters, so that the actual formula in a run involving values $\boldsymbol{v} = v_1, \ldots, v_j$ is $\gamma_i[\boldsymbol{v}/\boldsymbol{X}]$ or $\rho_i[\boldsymbol{v}/\boldsymbol{X}]$.

Definition 1. An *annotated protocol* Π consists of a set of parameterized regular strands $\{S_j[\boldsymbol{X}]\}_{j \in J}$ together with a pair of functions γ and ρ from nodes of these strands to formulas of L, such that γ is defined on positive nodes and ρ is defined on negative nodes.

The *strand space* Σ_Π *over* Π consists of all instances of the parametric strands $S_j[\boldsymbol{X}]$ together with all penetrator strands from Definition 8. The *bundles over* Σ_Π are determined by Definition 7.

Some earlier work [1, 18] defines protocols whose messages contain formulas like the γ_n. However, protocols can work properly without embedded formulas, and can fail even with them. The approach does not allow sharing different information with different peers, as C shares goods with M but not with B. We thus associate the guarantee with the sending node, not with the message itself.

3.2 EPMO Annotated

In the example of EPMO, there are four non-trivial guarantees, and four non-trivial rely formulas (Table 1). We write them in terms of the predicates:

$\mathsf{transfer}(B, p, M, N)$	B transfers p to M in reference to N
$\mathsf{ship}(M, g, C)$	M ships g to C

The guarantee or rely formula associated with node $n_{p,i}$ is $\gamma_{p,i}$ or $\rho_{p,i}$ respectively. Any node not shown in Table 1 has the trivially true formula True.

Bank:			
$\gamma_{b,2}$	$\forall P_M$	**if**	C authorizes $\mathsf{transfer}(B, \mathsf{price}, P_M, N_m)$, **and**
			P_M requests $\mathsf{transfer}(B, \mathsf{price}, P_M, N_m)$,
		then	$\mathsf{transfer}(B, \mathsf{price}, P_M, N_m)$.
$\rho_{b,3}$			C says $\gamma_{c,5}$, **and** M says M requests $\mathsf{transfer}(B, \mathsf{price}, M, N_m)$.
Customer:			
$\rho_{c,2}$			M says $\gamma_{m,2}$.
$\rho_{c,4}$			B says $\gamma_{b,2}$.
$\gamma_{c,5}$			C authorizes $\mathsf{transfer}(B, \mathsf{price}, M, N_m)$.
Merchant:			
$\gamma_{m,2}$	$\forall P_B$	**if**	$\mathsf{transfer}(P_B, \mathsf{price}, M, N_m)$, **then** $\mathsf{ship}(M, \mathsf{goods}, C)$.
$\rho_{m,3}$			B says $\gamma_{b,2}$, **and** C says $\gamma_{c,5}$.
$\gamma_{m,4}$			M requests $\mathsf{transfer}(B, \mathsf{price}, M, N_m)$, **and** $\mathsf{ship}(M, \mathsf{goods}, C)$.

Table 1. Guarantee and Rely formulas for EPMO

Bank Behavior The bank guarantees a quantified implication $\gamma_{b,2}$ on its second node $n_{b,2}$. The universally quantified payee makes the money order an instrument payable to the bearer. Its rely statement $\rho_{b,3}$ notes that C says C authorizes payment. Since "authorizes" is self-certifying (Tenet 2), this implies C does authorize payment. Formula $\rho_{b,3}$ also notes M's statement M says M requests price. B instantiates P_M with M, inferring that it should transfer price from C to M, as in Tenet 4. By Tenet 2, the bank does not need to trust either C or M.

Customer Behavior The customer makes no guarantee on $n_{c,1}$. It relies on M's offer $\gamma_{m,2}$ on $n_{c,2}$, and B's assertion $\gamma_{b,2}$ on $n_{c,4}$.

The self-certifying guarantee $\gamma_{c,5}$ is crucial for the protocol, as it authorizes the transfer. C utters $\gamma_{c,5}$ only if the transfer is acceptable. There are two trust decisions here (Tenet 3), whether to accept the implications

$$(B \ \mathsf{says}\ \gamma_{b,2}) \supset \gamma_{b,2} \quad \text{and} \quad (M \ \mathsf{says}\ \gamma_{m,2}) \supset \gamma_{m,2}.$$

C presumably accepts the former, having already decided to establish an account with B. C's decision whether to trust M for $\gamma_{m,2}$, uses previous experience or data from the Better Business Bureau, as encoded in C's trust management theory Th_C. If C accepts the implication and derives $\gamma_{c,5}$, and if M requests price, then by $\gamma_{b,2}$ and $\gamma_{m,2}$, M will ship the goods.

Merchant Behavior The merchant, upon receiving the first message, considers whether to sell goods for price to customer C. Deducing $\gamma_{m,2}$ from Th_M signals acceptance. In $\gamma_{m,2}$, the bank is universally quantified; the premise that the money will be transferred ensures M can reject an untrustworthy bank later.

On $n_{m,3}$, M relies on the self-certifying $\gamma_{c,5}$, as well as B says $\gamma_{b,2}$. M's criterion for trusting B for $\gamma_{b,2}$ is encoded in Th_M; possibly M checks a list of reputable banks. If M infers $\gamma_{b,2}$, it follows that B will transfer the funds upon request. If M makes the request, it must also ship the goods.

An affirmative decision is recorded in $\gamma_{m,4}$. The first conjunct is needed by B to justify the transfer. The second half, by Tenet 4, may have a side-effect, such as transmitting a shipping order to the shipping department.

4 Execution Semantics for Annotated Protocols

We now give a semantics for protocols by defining, given an assignment of theories to principals, what bundles can occur, namely the *permissible* bundles. We also define *soundness*, meaning that rely formulas always follow from previously asserted guarantees.

Definition 2. Let Π be an annotated protocol, let $\mathcal{B}$ be a bundle over Σ_Π, and let each principal P hold theory Th_P. $\mathcal{B}$ is *permissible* if, for each positive regular $n \in \mathcal{B}$ with $\mathsf{prin}(n) = P$, γ_n is derivable from $\{\rho_m : m \Rightarrow^+ n\}$ in Th_P.

Only permissible bundles can really occur, assuming that regular principals play by the rules and do not transmit a message without deducing the corresponding guarantee γ_n. There is no assumption that the penetrator deduces anything, as no formulas are associated with any penetrator node.

4.1 Sound Protocols

We next introduce a property of annotated protocols themselves, not concerned with individual bundles and principal theories. A protocol is sound if the formulas ρ_n on which principals rely are always true, in the sense that other principals have in fact made the assertions that ρ_n says they have made. A "soundness" attack on a protocol is a bundle $\mathcal{B}$ in which a regular principal relies on ρ_n, but without nodes $m \in \mathcal{B}$ whose guarantees γ_m would justify ρ_n. We give an example of unsoundness in Section 4.4.

Soundness is achievable only when certain nonces are unpredictable and certain keys are uncompromised. By a *security value* for a parameterized strand $S_j[\boldsymbol{X}]$, we mean a function $f(\boldsymbol{X})$. In practice f is always either a projection taking value X_i for some i or else an associated key such as $K_{X_i}^{-1}$. A *security value assignment* for a protocol Π is a pair of finite functions $\mathsf{unique}, \mathsf{non}$, each giving a set of security values for each role. An example appears in Table 2. If $s \in S_j[\boldsymbol{v}]$ is a strand and $\mathcal{B}$ is a bundle s intersects, then $\mathcal{B}$ *respects* $\mathsf{unique}, \mathsf{non}$ *for* s if $f \in \mathsf{unique}(S_j)$ implies $f(\boldsymbol{v})$ is uniquely originating in $\mathcal{B}$, and $f \in \mathsf{non}(S_j)$ implies $f(\boldsymbol{v})$ is non-originating in $\mathcal{B}$. $\mathcal{B}$ *respects* $\mathsf{unique}, \mathsf{non}$ if it respects it for every regular s such that $\mathcal{B}$ intersects s.

Definition 3. Soundness. Bundle $\mathcal{B}$ *supports* a negative node $n \in \mathcal{B}$ iff ρ_n is a logical consequence of the set of formulas $\{\mathsf{prin}(m) \text{ says } \gamma_m : m \prec_{\mathcal{B}} n\}$.

Let Π be an annotated protocol, and let $\mathsf{unique}, \mathsf{non}$ be a security value assignment for Π. *Π is sound for* $\mathsf{unique}, \mathsf{non}$ if, whenever $\mathcal{B}$ is a bundle over Π that respects $\mathsf{unique}, \mathsf{non}$, for every negative $n \in \mathcal{B}$, $\mathcal{B}$ supports n.

Typically, each rely formula ρ_n is a conjunction of assertions P says ϕ. Each conjunct is of the form $\mathsf{prin}(m)$ says γ_m, or, if γ_m is itself a conjunction $\phi_1 \wedge \ldots \wedge \phi_k$, alternatively $\mathsf{prin}(m)$ says ϕ_i.[2] Thus, one need only show there is such an $m \prec_{\mathcal{B}} n$. The rely formulas of Table 1 were chosen in this way. For instance, $\rho_{c,2}$ will be true as long as there is a merchant strand whose second node occurred previously and agrees with $n_{c,2}$ on the variables occurring in $\rho_{c,2}$, namely $C, M, \mathsf{goods}, \mathsf{price}, N_m$. Parameters not occurring in $\rho_{c,2}$ are unconstrained in the corresponding node. For instance, N_c is relevant to the working of the protocol, but does not occur in $\rho_{c,2}$.

Soundness is a *horizontal* condition: it concerns the message communication arrows that tie different strands together. By contrast, permissibility is a *vertical* condition: it concerns the local behaviors on the individual regular strands in a bundle. All the reasoning on our framework occurs locally on individual strands, while soundness ensures that the protocol coordinates assumptions with earlier conclusions on other strands.

4.2 Recent Soundness

Assertions do not last forever. A money order becomes stale, and the issuing bank will no longer redeem it. Goods are no longer worth the price, or are no

[2] Following [2], P says $(\phi \wedge \psi)$ implies P says $\phi \wedge P$ says ψ.

longer be available at a quoted price. Principals want to ensure that they rely only on recently made statements. We write $\mathsf{first}(s)$ to refer to the first node on a strand s and $\mathsf{last}(s)$ to refer to the last, and we define [10]:

Definition 4. Recency A node m is *recent for* n in bundle $\mathcal{B}$ if there exists a strand s such that $\mathsf{first}(s) \preceq_{\mathcal{B}} m$ and $n \preceq_{\mathcal{B}} \mathsf{last}(s)$.

When m is recent for n, strand s that "measures" the delay between them. Suppose there is a timeout value b, and all regular strands are implemented to time out before the total elapsed time between first node and last node reaches b. Then recency bounds the total elapsed time between m and n by b. The incoming and outgoing authentication tests establish recency (Appendix A, Propositions 4–5).

Definition 5. Recent Soundness. Bundle $\mathcal{B}$ *recently supports* a negative node $n \in \mathcal{B}$ iff ρ_n is a logical consequence of the set of formulas

$$\{\mathsf{prin}(m) \text{ } \mathsf{says} \text{ } \gamma_m \colon m \prec_{\mathcal{B}} n \wedge m \text{ is recent for } n\}.$$

Π *is recent-sound for* $\mathsf{unique}, \mathsf{non}$ if, whenever $\mathcal{B}$ is a bundle over Π that respects $\mathsf{unique}, \mathsf{non}$, for every negative $n \in \mathcal{B}$, $\mathcal{B}$ recently supports n.

Organizations have various techniques to allow themselves to ensure assertions will remain true for the near future, the next b time units. In EPMO, B puts a hold on the amount price in C's account, so that when M redeems the money order, B will still be willing to transfer this amount of money.

4.3 Recent Soundness of EPMO

We now establish the soundness of EPMO, using the security value assignment in Table 2. We let K_P^{-1} be P's private decryption key, and let $K_{P,\mathsf{sig}}^{-1}$ be P's private signature key. The notation $S[v_1, *, v_3]$ means $\bigcup_{X_2} S[v_1, X_2, v_3]$. We assume tacitly that hashes are uninvertible, which could be formalized (Appendix A) as a non-origination assumption.

Role	non	unique
Bank	$\{K_C^{-1}, K_{B,\mathsf{sig}}^{-1}, K_{M,\mathsf{sig}}^{-1}\}$	$\{N_m, N_b, N_c\}$
Cust	$\{K_C^{-1}, K_M^{-1}, K_B^{-1}\}$	$\{N_c\}$
Merch	$\{K_C^{-1}, K_{B,\mathsf{sig}}^{-1}\}$	$\{N_m, N_b, N_c\}$

Table 2. Security Value Assignment for EPMO

Because the M's rely statement $\rho_{m,3}$ is a conjunction, M must be assured that there are matching bank and customer nodes $n_{b,2}$ and $n_{c,4}$. They must agree with $n_{m,3}$ at least for the variables B, C, p, N_m and B, C, M, p, N_m, respectively.

Proposition 1. *Suppose:* $m \in \mathsf{Merch}[B, C, M, p, g, N_c, N_m, N_b]$*;* $n_{m,3} \in \mathcal{B}$*;* $\mathcal{B}$ *respects* unique*,* non *for* m*; and* $N_c \neq N_m$*. Then* $n_{b,2}, n_{c,5} \in \mathcal{B}$ *for some* $b \in \mathsf{Bank}[B, C, *, p, N_c, N_m, N_b]$ *and* $c \in \mathsf{Cust}[B, C, M, p, g, N_c, N_m, N_b]$*; moreover* $n_{b,2}, n_{c,5}$ *are recent for* $n_{m,3}$*.*

Proof. Node $n_{b,2}$ follows by an incoming test on test value N_m. The message sent on $n_{b,2}$ takes the form $[\![\, \mathsf{hash}(C \ \hat{}\ N_b \ \hat{}\ N_m \ \hat{}\ \mathsf{price}) \,]\!]_B \ \hat{}\ \{\!|N \ \hat{}\ N_b|\!\}_C$ for some N. To establish $n_{c,5}$, observe that N_b is uniquely originating on $n_{b,2}$; $K_C^{-1} \in \mathsf{safe}$, and $\mathsf{hash}(\_)$ is uninvertible. Hence nodes $n_{b,2}$ and $n_{m,3}$ form an outgoing test. Thus there is a customer strand transforming $t \ \hat{}\ \{\!|N' \ \hat{}\ N_b|\!\}_{C'} \Rightarrow^+ t \ \hat{}\ N_b$. By the protocol definition, t is a money order containing bank nonce N_b. Because N_b is uniquely originating, and a money order is always a point of origination for its bank nonce, $t = [\![\, \mathsf{hash}(C \ \hat{}\ N_c \ \hat{}\ N_b \ \hat{}\ N_m \ \hat{}\ \mathsf{price}) \,]\!]_B$. Finally, to see that c agrees on p, observe that $n_{m,2} \Rightarrow n_{m,3}$ is also an outgoing test. Thus, there is a customer strand c' with parameters C, M, p, g, N_c, N_m; $c' = c$ because N_c originates uniquely, but originates on both customer strands.

For recency, observe that $N_m \sqsubset \mathrm{term}(n_{b,2})$ and $N_m \sqsubset \mathrm{term}(n_{c,3})$, while N_m originates uniquely at $n_{m,2}$; thus, m measures the delay since $n_{b,2}$ and $n_{c,3}$. ■

We show next that $\mathcal{B}$ gives guarantees for the rely formula $\rho_{b,3}$, i.e. it contains nodes $n_{c,5}$ and $n_{m,4}$. Nodes $n_{c,5}$ and $n_{m,4}$ must agree with $\rho_{b,3}$ on the parameters $B, C, M, \mathsf{price}, N_m$ occurring in $\gamma_{c,5}$ and in the first conjunct of $\gamma_{m,4}$.

Proposition 2. *Suppose that* $b \in \mathsf{Bank}[B, C, M, p, N_c, N_m, N_b]$*, where* $N_c \neq N_b \neq N_m$*; suppose* $\mathcal{B}$ *respects* unique, non *for* b*; and suppose* $n_{b,3} \in \mathcal{B}$*. Then* $n_{m,4}, n_{c,5} \in \mathcal{B}$ *for some*

$$m \in \mathsf{Merch}[B, C, M, p, *, N_m, N_b] \quad \textit{and} \quad c \in \mathsf{Cust}[B, C, M, p, *, *, N_m, N_b].$$

Nodes $n_{m,4}, n_{c,5}$ *are recent for* $n_{b,3}$*.*

Proof. By the incoming test principle (Proposition 5), there is a regular node between $n_{b,2}$ and $n_{b,3}$ emitting $[\![\, \mathsf{hash}(B \ \hat{}\ N_b \ \hat{}\ N_m) \,]\!]_M$. By pattern matching, this is node $n_{m,4}$ of a merchant strand with parameters B, M, N_b, N_m. Hence, it was preceded by a node receiving $\mathsf{mo}' = [\![\, \mathsf{hash}(C' \ \hat{}\ N_b \ \hat{}\ N_m \ \hat{}\ p') \,]\!]_B$; by unique origination of N_b and because the protocol emits mo' only at a point of origination for N_b, $\mathsf{mo}' = \mathsf{mo}$. Thus, $C' = C$ and $p' = p$. Hence, by Proposition 1, the required customer strand exists. ■

Customer rely formulas may be justified similarly.

4.4 An Unsound Protocol

An unsound variant of EPMO, in the spirit of the original Needham-Schroeder protocol [22], replaces message 2 from M to C by $\{\!|N_c \ \hat{}\ N_m \ \hat{}\ \mathsf{price}|\!\}_C$. As M's name is absent, we can then mount a Lowe-like attack [21]. When, in $\rho_{m,3}$, M relies on C says C authorizes $\mathsf{transfer}(B, \mathsf{price}, M, N_m)$, there is no matching guarantee. Instead, C has authorized $\mathsf{transfer}(B, \mathsf{price}, M', N_m)$ for some different M'. Therefore, soundness is false.

M' can use the attack, e.g. to arbitrage for his own advantage some discount that C can obtain from M. Suppose that C contacts M', when the latter knows C could have got a discounted price from M. M' contacts M purporting to be C and requesting a larger amount of goods to make up the expected price. The remainder of the exchange occurs as expected, except that M' skims off the additional goods and delivers the expected shipment to C.

5 Conclusion

This paper appears to be the first offering a rigorous method for reasoning about trust management in the context of nonce-based cryptographic protocols, rather than using certificates with expiration dates, as in [18] and much other work. We have illustrated our method with an electronic commerce protocol. We are also using it to make TPMs a basis for access control via trust management [16].

Related Work A large literature applies logics to reasoning about cryptographic protocols, stretching from [6] to [8]. However, the logics determine what authentication goals a protocol achieves. By contrast, we use the non-logical strand space framework to determine authentication. We cash in those authentication results using soundness (Definition 3), obtaining rely formulas that coordinate the local per-strand trust management reasoning of the distributed principals.

There is a well-entrenched tendency to regard protocol actions as assertions, reflecting the principal's beliefs and goals, and its appraisals of assertions by others. Generally, however, one has regarded the *messages* as statements, which may lead to confusion about who is drawing conclusions from the statements, and which masks whether the statement was made by the expected peer. Hence, we have attached the formulas to purely local transmission and reception events. The key notion of protocol soundness allows us to decide whether statements were uttered by the expected peer. A sound protocol is a reliable coordination mechanism for deduction carried out by separate principals.

Taking protocol actions as assertions animates Abadi and Needham's advice for protocol design [1]. Our interpretation of the assertions as associated with actions rather than with messages is compatible with much of what they say, and could lead to a more precise treatment. For instance, they write:

> Principle 1. Every message should say what it means: the interpretation of the message should depend only on its contents.

If the "interpretation" is the guarantee offered by the sender, then this principle recommends that for each message t, there should be a single guarantee γ asserted on every regular node on which t is sent. All parameters in γ should occur in t. Similarly:

> Principle 2. The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.

To act on a message means to continue the protocol and send the next message. Our trust management framework "clearly sets out" the condition for this in the guarantee and rely statements. To decide whether to act on a message received, P uses its rely statement, trying to deduce the next guarantee.

Future Work An important question suggested by our work is how to determine the possible shapes that the bundles of a protocol may have. By a *shape*, we mean a set of parametric regular strands with a partial ordering $\preceq$, and a maximum node n_f that all other nodes precede. To be a shape, there must be a bundle, containing precisely the given regular strands, which is normal and efficient in the sense of [12]. In some protocols there is only one possible shape; a bundle for EPMO that contains $n_f = n_{b,3}$ has the shape shown in Figure 1. Other protocols (Otway-Rees, for instance) have more than one shape for a given n_f.

Shape analysis allows the protocol designer to explain, when P_1 draws some conclusion, exactly which other principals may have made guarantees contributing to P_1's conclusion. For instance, suppose that P_1 trusts P_2's local information, so P_1 would like to accept P_2 says $\phi \supset \phi$. However, P_1 may want to know that there is no execution in which P_2, when deducing ϕ, relied on another principal's assertions. A shape analysis tells us whether a strand of some P_3 may have preceded P_2's message transmission, in which case it may have contributed a guarantee on which P_2 relied. This is a novel protocol design criterion.

The ideas here also suggest an appealing implementation strategy, in which protocol principals are constructed using a trust management engine [19], together with tabular information describing the structure of the possible strands.

We would also like to know how this method relates to well-developed frameworks for reasoning about knowledge [9, 14]. Can the elements of a *trust structure* [7] replace the formulas used here? Computational aspects of logic must also eventually be considered; our framework suggests new properties for which tractability would be desirable [19, 20].

Conclusion In this paper, we have developed a rely-guarantee method that can be used to combine trust management logics with nonce-based protocols. The key technical idea was *soundness*, which provides a criterion for whether the protocol is adequate to support the trust reasoning of the principals. We believe that the resulting theory can be useful as a mechanism for cross-organization access control, particularly when supported by hardware such as the TPM that provides reliable security services.

Acknowledgments Boris Balacheff, Joe Pato, David Plaquin, and Martin Sadler of HP Labs helped orient this work in relation to the TPM environment. The informal Protocol Exchange seminar provided an excellent forum for discussion, at which Dusko Pavlovic and John Mitchell in particular planted seeds. Dale Johnson ironed out the Aristotle connection.

References

1. Martín Abadi and Roger 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. Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *6th ACM Conference on Computer and Communications Security*, November 1999.
3. Aristotle. *Nicomachean Ethics*. Oxford University Press, 1953.
4. Boris Balacheff, Liqun Chen, Siani Pearson (ed.), David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, 2003.
5. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Distributed trust management. In *Proceedings, 1996 IEEE Symposium on Security and Privacy*, pages 164–173, 1997.
6. Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, Series A, 426(1871):233–271, December 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.
7. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A formal model for trust in dynamic networks. In Antonio Cerone, editor, *International Conference on Software Engineering and Formal Methods*. IEEE CS Press, September 2003.
8. Nancy Durgin, John Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–721, 2003.
9. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, MA, 1995.
10. Joshua D. Guttman. Authentication tests and disjoint encryption: a method for security protocol design. *Journal of Computer Security*, 2004. Forthcoming.
11. Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Proceedings, 13th Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.
12. Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.
13. Joshua D. Guttman, F. Javier Thayer, and Lenore D. Zuck. The faithfulness of abstract protocol analysis: Message authentication. *Journal of Computer Security*, 2004. Forthcoming.
14. Joseph Y. Halpern and Riccardo Pucella. On the relationship between strand spaces and multi-agent systems. *ACM Transactions on Information and System Security*, 6(1):43–70, February 2003.
15. James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings, 13th Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.
16. Jonathan Herzog, Brian Sniffen, Jay Carlson, Joshua D. Guttman, and John D. Ramsdell. Trust management with cryptographic hardware assistance. MTR 03B0082, The MITRE Corp., Bedford, MA, September 2003.
17. Cliff B. Jones. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 1983.
18. Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

19. Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings, 2002 IEEE Symposium on Security and Privacy*, pages 114–130. May, IEEE CS Press, 2002.
20. Ninghui Li, William H. Winsborough, and John C. Mitchell. Beyond proof-of-compliance: Safety and availability analysis on trust management. In *Proceedings, 2003 IEEE Symposium on Security and Privacy*. May, IEEE CS Press, 2003.
21. Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.
22. Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.
23. Adrian Perrig and Dawn Xiaodong 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.
24. F. Javier THAYER Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

A Strand Spaces

A set A contains the messages ("terms") to be exchanged. They are freely generated from a set of atoms including texts and keys by concatenation and encryption, in which the second argument is a key. We formalize hashing as encryption with an asymmetric key of which no principal knows the inverse. Message transmission has positive sign, and reception has a negative sign.

Definition 6. 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. A *strand space* over A is a set Σ with a trace mapping $tr : \Sigma \to (\pm\mathsf{A})^*$. Fix a strand space Σ:

1. 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.
2. Suppose I is a set of terms. The node $n \in \mathcal{N}$ is an *entry point* for I iff $\mathrm{term}(n) = +t$ for some $t \in I$, and whenever $n' \Rightarrow^+ n$, $\mathrm{term}(n') \notin I$.
3. An term t *originates* on $n \in \mathcal{N}$ iff n is an entry point for $I = \{t' : t \sqsubset t'\}$.
4. An term t is *uniquely originating in* $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n, and *non-originating* if there is no such $n \in S$.

If a term t originates uniquely in a suitable set of nodes, then it can play the role of a nonce or session key. If it is non-originating, it can serve as a long-term secret, such as a shared symmetric key or a private asymmetric key. $\mathcal{N}$ together with both sets of edges $n_1 \to n_2$ (message transmission) and $n_1 \Rightarrow n_2$ (succession on the same strand) is a directed graph $\langle \mathcal{N}, (\to \cup \Rightarrow)\rangle$. A *bundle* is a subgraph of $\langle \mathcal{N}, (\to \cup \Rightarrow)\rangle$ for which the edges express causal dependencies of the nodes.

Definition 7. Suppose $\rightarrow_{\mathcal{B}} \subset \rightarrow$; suppose $\Rightarrow_{\mathcal{B}} \subset \Rightarrow$; and let $\mathcal{B} = \langle \mathcal{N}_{\mathcal{B}}, (\rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}}) \rangle$ be a finite acyclic subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$. $\mathcal{B}$ is a bundle if:

1. If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $\mathrm{term}(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_{\mathcal{B}} n_2$.
2. If $n_2 \in \mathcal{N}_{\mathcal{B}}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_{\mathcal{B}} n_2$.

A node n is in a bundle $\mathcal{B} = \langle \mathcal{N}_{\mathcal{B}}, \rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}} \rangle$, written $n \in \mathcal{B}$, if $n \in \mathcal{N}_{\mathcal{B}}$. The *$\mathcal{B}$-height* of a strand s is the largest i such that $\langle s, i \rangle \in \mathcal{B}$. 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 3. *If $\mathcal{B}$ is a bundle, $\preceq_{\mathcal{B}}$ is a partial order. Every non-empty subset of the nodes in $\mathcal{B}$ has $\preceq_{\mathcal{B}}$-minimal members.*

Definition 8. A *penetrator trace* is one of the following:

M_t: $\langle +t \rangle$ where $t \in$text	K_K: $\langle +K \rangle$
$\mathsf{C}_{g,h}$: $\langle -g,\ -h,\ +g \mathbin{\hat{}} h \rangle$	$\mathsf{S}_{g,h}$: $\langle -g \mathbin{\hat{}} h,\ +g,\ +h \rangle$
$\mathsf{E}_{h,K}$: $\langle -K,\ -h,\ +\{\!|h|\!\}_K \rangle$	$\mathsf{D}_{h,K}$: $\langle -K^{-1},\ -\{\!|h|\!\}_K,\ +h \rangle$.

We write safe for *safe* keys, i.e. keys that the penetrator can never learn or use [12]. Since long term shared keys and private asymmetric keys are never transmitted in reasonable protocols, these keys are safe unless compromised before execution of the protocol. Session keys are safe if transmitted only protected by keys K with $K^{-1} \in \mathsf{safe}$.

When S is a set of terms, t_0 *occurs only within* S in t if, regarding t as an abstract syntax tree, every branch from the root to an occurrence of t_0 traverses some occurrence of a $t_1 \in S$ before reaching t_0. It *occurs outside* S in t if $t_0 \sqsubset t$ but t_0 does not occur only within S in t. A term t_0 *occurs safely* in t if it occurs only within $S = \{\{\!|h|\!\}_K \colon K^{-1} \in \mathsf{safe}\}$ in t.

Proposition 4 (Outgoing Authentication Test). *Suppose $\mathcal{B}$ is a bundle in which a originates uniquely at n_0; a occurs only within S in term(n_0) and a occurs safely in S; and $n_1 \in \mathcal{B}$ is negative and a occurs outside S in term(n_1).*

There are regular $m_0, m_1 \in \mathcal{B}$ such that $m_0 \Rightarrow^+ m_1$, where m_1 is positive, a occurs only within S in term(m_0), and a occurs outside S in term(m_1). Moreover, $n_0 \preceq m_0 \prec m_1 \prec n_1$.

Proposition 5 (Unsolicited, Incoming Test Principles). *Suppose $n_1 \in \mathcal{B}$ is negative, $\{\!|h|\!\}_K \sqsubset$ term(n_1), and $K \in \mathsf{safe}$. (Unsolicited test:) There exists a regular $m_1 \prec n_1$ such that $\{\!|h|\!\}_K$ originates at m_1. (Incoming test:) If in addition $a \sqsubset h$ originates uniquely on $n_0 \neq m_1$, then $n_0 \prec m_0 \Rightarrow^+ m_1 \prec n_1$.*