Completeness of Discovery Protocols

Alwyn E. Goodloe
National Institute of Aerospace

Carl A. Gunter
University of Illinois at Urbana-Champaign

ABSTRACT

Tunnel-complex protocols construct topologies of security tunnels by directing tunnel-establishment protocols to set up pair-wise tunnels, where the resulting collection of tunnels achieves an overall security objective. Such protocols ease the burden on network managers, but their design exhibits subtleties relating to functional correctness that can benefit from formal analysis. A class of tunnel-complex protocols that are of special interest are discovery protocols that discover security gateways and set up tunnels to negotiate their traversal by delivering the requisite credentials to satisfy the policies at security gateways on the dataflow path. We present a case study of a discovery protocol that sets up a concatenated sequence of tunnels. We then propose the concept of a theorem for discovery protocols that expresses the completeness of the protocol's credential distribution mechanism. The theorem is parameterized for different protocols. We show how it is instantiated for the protocol in our case study and discuss how specific instances of the theorem characterize different classes of discovery protocols.

Categories and Subject Descriptors

C.2 [**Computer and Communication Networks**]: Security and Protection; F.3 [**Theory of Computation**]: Logics and Meanings of Programs; D.2.4 [**Software/Program Verification**]: Formal Methods.

General Terms

Security, Theory, Verification.

Keywords

IPsec, Formal Methods, Certificates, Security Gateways, Discovery Protocols.

1. INTRODUCTION

Security gateways guard administrative domains by enforcing polices that control the principals allowed to traverse the boundary defined by the gateway. Organizations often have two or three layers of nesting of administrative domains so there may be as many as six gateways to traverse before two hosts can communicate. Security tunnel protocols, such as IPsec [25], protect information traveling over an insecure communication path. In order to ensure that ingress and egress traffic is authenticated and authorized, gateways can require all such traffic arrive in a security tunnel. The resulting complex of tunnels forms a virtual private network (VPN).

The administrative tasks involved in configuring a VPN are considerable and the process is error prone. For instance, while tunnel establishment protocols such as the Internet Key Exchange protocol (IKE V2) [24] are employed to create the cryptographic information used by a security association, many parameters remain to be configured by the administrator. As the number of security gateways grows and the overlay topology becomes more complex, the burden on the administrator grows. It is especially burdensome if administrators need to react to the addition or removal of a gateway by updating configurations throughout the network. A common solution is to limit the complexity of the topology, but even simple topologies are not trivial to administer. This difficulty has led to the development of support protocols that coordinate the construction of a tunnel complex, which we designate as *tunnel-complex protocols*. Scalability is limited if all the gateways on a given dataflow path need to be known beforehand, since different organizations may not communicate each time they change their gateway topology. This situation is akin to the early days of the Internet when forwarding tables were manually configured. A solution lies in the form of *discovery protocols* that discover the gateways, negotiate their traversal, and configure tunnels dynamically to form a virtual topology where the traffic flow is governed by the gateway policies.

The tunnel-complex protocols that have been deployed to date are relatively simple, but given the need for more sophisticated topologies, support protocols to create them are likely to follow. Since tunnel-complex protocols automatically configure security gateways, an error in the protocol can either allow unauthorized principals to traverse a gateway or fail to deliver available credentials, consequently, denying service. Our aim in this paper is to consider one property that can be asserted about candidate discovery protocols, forming a starting point for their systematic analysis. The property, which we call 'completeness', says that the protocol assures that all of the necessary credentials are communicated as part of the the discovery protocol. That

SafeConfig'09, November 9, 2009, Chicago, Illinois, USA.

is, if there is a set of credentials that will enable a communication, then these are passed to the nodes that need to have them. We illustrate the idea by presenting the design of a discovery protocol that dynamically constructs a complex of concatenated tunnels and proving a completeness theorem for this protocol.

In the next section, we present a formal treatment of policies, credentials, and tunnels. An overview of discovery protocols follows. We then give an example of a discovery protocol that constructs a collection of concatenated tunnels. Next, we formulate a general theory of completeness for discovery protocols and prove our concatenated discovery protocol does indeed satisfy this criteria. Finally, we discuss related works and conclude.

2. POLICIES AND TUNNELS

Security gateways enforce polices governing what principals are allowed to traverse the gateway. The policies are satisfied only if some requisite collection of credentials are presented to the gateway. If a principal can produce the necessary credentials, then a tunnel can be setup with the gateway to ensure that all traffic entering or exiting the gateway's administrative domain is both authenticated and authorized. To formulate our notions of discovery and completeness we must be more precise when speaking about policies, credentials, and tunnels. Our intent is not to model a specific standard or product, such as SPKI/SDSI, but to formally capture essential concepts needed later in our presentation.

2.1 Policies and Credentials

We assume gateway polices specify those principals that are allowed specific traffic flows. For instance, a policy at a gateway may say that Alice, represented as principal K_A, is allowed to communicate between the address a_1 and an address a_2. We formalize this as $\mathsf{Pol}\langle K_A : a_1 \leftrightarrow a_2\rangle$. Suppose a discovery protocol is invoked to enable communication between a_1 and a_2 and encounters a gateway enforcing this policy, then the protocol must deliver the credential K_A to the gateway; otherwise, the protocol will fail. Generally, gateway policies have the format $\theta = \mathsf{Pol}\langle K_1, \ldots, K_n : \eta\rangle$ specifying a list of (root) principals $K_1, \ldots, K_n$ that are authorized by a gateway to communicate between the addresses given in the selector η. We assume each gateway maintains a list of policies $\Theta = \mathsf{Pols}[\theta_1, \ldots, \theta_n]$ that it enforces. For simplicity, we assume that the policies at each gateway have disjoint traffic selectors. Hence, there is at most one policy at a gateway for a given traffic flow. In talking about the execution of a discovery protocol enabling communication between two specific nodes, we often need to reference the gateway policy that is enforced during this run. We write $\Theta_\eta \,@\, a$ to denote the policy at node a that matches the selector η.

Although the details of most standard credential systems are rather intricate, we simply view a credential as expressing a relationship between two principals or delegating authority from one principal to another. For instance, Alice may have a credential that says she belongs to Acme Inc. Formalizing this concept, we say a credential $\xi = \mathsf{Cred}\langle K_S, K_I\rangle$ defines a relation such as: K_I 'delegates' to K_S, or K_S 'is a member of' K_I, or K_S 'speaks for' K_I. Principal K_I is called the *issuer* and K_S the *subject*. Given a credential that says K_S speaks for K_I, we often write $K_S \Rightarrow K_I$. A collection of credentials Ξ is called a credential set. We shall later see that discovery protocols deliver a credential set to each gateway on the path in order to obtain permission to traverse that gateway.

To satisfy a gateway policy one must verify that a given credential set contains a key listed in the policy. For instance, if the policy at gateway G_3 says K_M is allowed to traverse the gateway and a discovery protocol initiated by Alice delivers the credential $K_A \Rightarrow K_M$ saying she belongs to principal M, then the policy is satisfied, but to ensure integrity we require that there be a delegation chain from G_3 to A such as

$$K_{G_3} \Rightarrow K_{G_2} \Rightarrow K_{G_1} \Rightarrow K_A \Rightarrow K_M.$$

It remains to formalize this concept. Let $\mathcal{T}(\Xi)$ be the forest of delegation trees formed from the credentials in Ξ. Given the policy Θ_η for the traffic flow η and a credential set Ξ, the satisfaction relation

$$\Xi \models_{K_a} \Theta_\eta$$

is defined to be true if there exists a chain in $\mathcal{T}(\Xi)$ rooted at K_a that contains one of the keys in Θ_η. The relation is defined to be false otherwise. We assume that there exists some mechanism that performs the task of computing $\Xi \models_{k_a} \Theta_\eta$, which returns $\uparrow \mathsf{GWPol}(u, \text{true})$ if successful, where u is a unique identifier that will be described in more detail later in the paper.

2.2 Security Tunnels and Establishment

From a high-level perspective, a security tunnel can be viewed 'type-theoretically' as follows. A node a communicates with a node g by wrapping each message m it sends to g within a constructor C. Node g holds a corresponding destructor C^-, which it applies to get the message m. The constructor C represents the bulk protocol between a and g. The behavior of the constructor and destructor pair constitute the essence of a security association. Node a may have a *security policy* that acts as a filter indicating all messages sent to g must be wrapped in C and node g may have a policy that messages it receives from a must be wrapped in C.

Tunnel establishment is the process of setting up a bidirectional tunnel between two nodes. In type-theoretical terms, a tunnel-establishment protocol causes the tunnel endpoints to obtain the constructor C and destructor C^- respectively in such a way that they authenticate each other, authorize the use of the constructor, and assure that they are the only parties that have these operators. Tunnel-establishment forms the core component of tunnel-complex protocols. We assume that any establishment protocol employed in a tunnel-complex protocol has the following three features. First, the two parties exchange credentials and a newly discovered gateway invokes a policy evaluation mechanism to verify that the credentials received satisfy the gateway's policies. We assume that if principal B is the newly discovered gateway participating in establishment with principal A, then the protocol passes a collection of credentials to A along with a delegation $K_B \Rightarrow K_A$. Secondly, we assume that the establishment protocol installs security policy entries. For instance, a tunnel being set up between nodes C and D may have a policy installed saying that all traffic flowing between nodes A and B should travel in the newly created tunnel. Third, the establishment of shared crypto-

graphic keys for the associations by way of a key exchange protocol [9, 28]. Depending on the specific tunnel-complex protocol, IKE V2's [24] traffic selection convention may not be sufficient to satisfy the second requirement. It may be possible to push the policy updates up to the tunnel-complex protocol, but the resulting designs are much messier. For further discussion of an establishment protocol intended for use in tunnel-complex protocols see [17].

3. OVERVIEW OF DISCOVERY

Having introduced the necessary background and definitions, we shall now focus on discovery protocols. Consider the situation depicted in Figure 1. Alice, who works at

Figure 1: Example Topology

ACME, wishes to communicate with Bob's server, located in the accounting department at Coyote Corporation. Alice is located in the administrative domain of ACME's corporate network gateway $GW1$ while Bob is located behind Coyote's corporate gateway $GW2$ and is also inside the administrative domain of the accounting department's gateway $GW3$. In order for Alice to communicate with Bob, she must traverse all three of these gateways.

Although one can conceive of many different discovery protocols, all of the discovery protocols considered here have the same basic skeleton. We shall assume each run of a discovery protocol has an associated unique identifier u called a *session identifier*. A node a initiates discovery session u to establish communication with a node b. The initiating host a sends the distinguished packet $\mathsf{P}(a, b, \mathrm{Dis}(a, u))$ toward b, containing the session identifier as well as the address a, which is the node with which the intercepting gateway will initiate establishment. The first gateway on the dataflow path, $GW1$, intercepts this packet and the discovery protocol invokes tunnel establishment to setup a tunnel with a. When this tunnel has been set up, $GW1$ releases the discovery packet. The address in the discovery packet will vary depending on the complex being created. For instance, if the gateway releases the packet $\mathsf{P}(a, b, \mathrm{Dis}(GW1, u))$, then the next intercepting gateway will set up a tunnel to $GW1$; on the other hand, if the packet released had been $\mathsf{P}(s, d, \mathrm{Dis}(a, u))$, then the next intercepting gateway will set up a tunnel to a that is nested inside of the tunnel between a and $GW1$. To prevent the tunnel complex from being too great a burden on network communication, the tunnels to the gateways only perform authorization and authentication. The process continues until the discovery process reaches the destination node b, at which point an end-to-end tunnel performing encryption is established to secure communication between a and b.

Suppose the gateways depicted in Figure 1 are assumed to have the following polices:

Gateway	Policy
GW1	$\mathsf{Pol}\langle K_{\mathrm{ACME}}, \mathrm{dom}(GW1) \leftrightarrow \mathrm{dom}(GW2)\rangle$
GW2	$\mathsf{Pol}\langle K_{\mathrm{Coyote}}, K_{\mathrm{CoyoteSub}},$ $\mathrm{dom}(GW1) \leftrightarrow \mathrm{dom}(GW2)\rangle$
GW3	$\mathsf{Pol}\langle K_{\mathrm{Alice}}, \mathrm{dom}(GW1) \leftrightarrow \mathrm{dom}(GW3)\rangle$

where $\mathrm{dom}(GW)$ denotes all the addresses in the domain of GW. Assume that Alice initiates communication with Bob. The above policies say that in order to traverse $GW1$, the gateway must be presented credentials from ACME. In order to traverse $GW2$, this gateway must be presented credentials from Coyote directly or proof that it is a subcontractor. Gateway $GW3$ requires credentials from Alice in order to gain entry to its administrative domain.

Each node c executing a discovery protocol is assumed to have a credential set Ξ^c that defines a relation with at least one other entity. For instance, hosts and gateways will have credentials defining the administrative domains to which they belong. In the situation illustrated in Figure 1, Alice and $GW1$ both belong to ACME, but gateway $GW1$ also contains a credential saying that ACME is a subcontractor of Coyote corporation. Gateways $GW2, GW3$, and Bob belong to Coyote Corp. In addition, both $GW3$ and Bob also belong to Coyote's accounting department. The credentials defining this relationship are given in the following table:

Node	Credential	Contents
Alice	Ξ^A	$K_A \Rightarrow K_{ACME}$
GW1	Ξ^{GW1}	$K_{GW1} \Rightarrow K_{ACME} \Rightarrow K_{CoyoteSub}$
GW2	Ξ^{GW2}	$K_{GW2} \Rightarrow K_{Coyote}$
GW3	Ξ^{GW3}	$K_{GW3} \Rightarrow K_{Acct} \Rightarrow K_{Coyote}$
Bob	Ξ^B	$K_B \Rightarrow K_{Acct} \Rightarrow K_{Coyote}$

We make no assumptions as to how credentials get initialized at the nodes.

The host that initiates discovery may not have the requisite credentials to traverse all the gateways on the path. On the other hand, some of the gateways on the path may possess the needed credentials. In the case of our example, Alice possess the necessary credentials to traverse $GW1$ and $GW3$, but not $GW2$. Yet $GW1$ does possess the requisite credentials. Suitably designed discovery protocols can collect credentials at the gateways on the path. Different protocol designs will deliver different credential sets. In the next section, we shall focus on one such protocol and illustrate how it delivers credentials to enable Alice to communicate with Bob.

4. CONCATENATED DISCOVERY

Given the topology depicted in Figure 1 with the aforementioned gateway policies and credentials, there are a number of different tunnel complexes that would enable Alice to communicate with Bob. Perhaps the simplest topology conceptually is a complex composed of a collection of *concatenated* tunnels connecting the gateways with an end-to-end tunnel such as the one depicted in Figure 2.

A skeleton for the structure of our concatenated discovery protocol is given as follows. A host sends out a discovery packet that gets intercepted by the next gateway on the

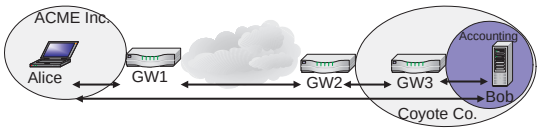

Figure 2: Concatenated Tunnel Complex

dataflow path. Establishment is invoked at the newly discovered gateway to set up a pair of associations between itself and the node that last released the discovery packet. When the establishment initiator indicates that it has terminated, this node sends the establishment responder an acknowledgment ACK1 and receives an ACK2 acknowledgment in turn. If the newly discovered node is not the destination node, then the discovery packet is released and the establishment responder is invoked to set up a tunnel to the next node on the path. If the most recently discovered node is the destination, then upon the termination of establishment and the receipt of the ACK2 message, establishment is invoked to set up an end-to-end tunnel. When this instance of establishment completes, a FIN message is sent to the host that initiated the discovery protocol.

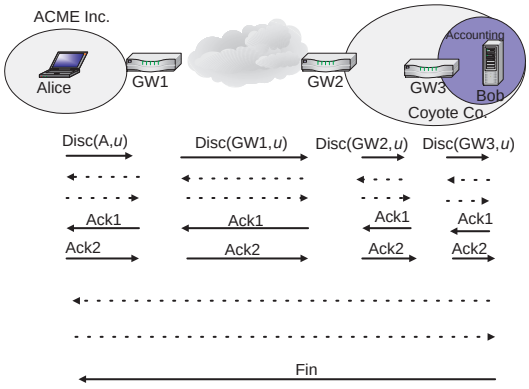

Figure 3: Concatenated Discovery Execution

Figure 3 illustrates the execution of session u of our concatenated discovery protocol on the Alice-Bob example in Figure 2. The protocol messages are displayed as black lines and the establishment messages are displayed as dotted black lines. Alice releases a discovery packet $\mathsf{P}(A, B, \mathrm{Dis}(A, u))$ destined for Bob. The packet is intercepted by gateway $GW1$, which invokes establishment with Alice. The gateway then sends Alice an acknowledgment message $\mathsf{P}(GW1, A, \mathrm{ACK1})$ in the newly created association flowing from $GW1$ to Alice. Alice responds by sending an acknowledgment $\mathsf{P}(A, GW1, \mathrm{ACK2})$ in the association flowing from Alice to gateway $GW1$. Upon receiving this message, $GW1$ releases the discovery packet $\mathsf{P}(A, B, \mathrm{Dis}(GW1, u))$, where the address A is replaced by $GW1$. The process repeats until Bob has set up a tunnel with $GW3$, after which, Bob initiates establishment with Alice to set up the end-to-end tunnel. When this tunnel has been set up, Bob sends Alice a Fin message.

We now show how the protocol delivers credentials to the gateways using the example given in Figure 3. Assume that the gateway polices and credentials at nodes are as given in the tables above. Each session maintains its own credential set Ξ^u, which is initialized to Ξ^A at the start of the protocol. During the execution of establishment between Alice and $GW1$, Alice sends the credential set

$$\{K_{GW1} \Rightarrow K_A\} \cup \Xi^u$$

to $GW1$, which invokes a mechanism to determine if these credentials satisfy its policies. The the credential set delivered to $GW1$ says $K_{GW1} \Rightarrow K_A \Rightarrow K_{\mathrm{Acme}}$, which satisfies the policy at that gateway, whence the authorization mechanism returns $\uparrow \mathsf{GWPol}(\mathrm{true})$. At this point $GW1$ adds it's credentials Ξ^{GW1} to Ξ^u. In setting up the tunnel between $GW1$ and $GW2$, the establishment protocol delivers to $GW2$ the credential set Ξ^u, composed of Ξ^A and Ξ^{GW1} as well as credentials saying that $GW1$ speaks for Alice and that $GW2$ speaks for $GW1$. These credentials form the chain

$$K_{GW2} \Rightarrow K_{GW1} \Rightarrow K_A \Rightarrow K_{\mathrm{Acme}} \Rightarrow K_{\mathrm{CoyoteSub}}.$$

Since this satisfies the policy at $GW2$, the protocol adds Ξ^{GW2} to Ξ^u and continues executing. During the exchange between $GW2$ and $GW3$ the protocol delivers Ξ^u to $GW3$, which is composed of Ξ^A, Ξ^{GW1}, Ξ^{GW2}, and credentials saying $GW1$ speaks for *Alice*, $GW2$ speaks for $GW1$, $GW3$ speaks for $GW2$, thus it possible to form the chain

$$K_{GW3} \Rightarrow K_{GW2} \Rightarrow K_{GW1} \Rightarrow K_A \Rightarrow$$
$$K_{\mathrm{Acme}} \Rightarrow K_{\mathrm{CoyoteSub}},$$

which satisfies the applicable policy at $GW3$ and the protocol adds Ξ^{GW3} to Ξ^u and continues executing. Notice that the protocol collected credentials at each gateway as it executes and presents them to the next gateway on the path.

We shall now present the concatenated discovery protocol in more detail by describing the actions of each of the principals. We first present the the actions of the initiating host s, we then present the actions performed by a gateway a on the path, finally we present the actions at the destination host d.

Initiating Host

Initiate Protocol The protocol is invoked at initiating host s to communicate with principal d. Principal s generates a unique protocol session identifier u and initializes the credential set Ξ^u to Ξ^s.

Send Discovery Message The initiator sends the message

$$\mathsf{P}(s, d, \mathrm{Dis}(s, u)).$$

The initiator then awaits the intercepting node to initiate establishment.

Exchange Acknowledgments Upon termination of the establishment protocol with the first gateway on the path a, the host awaits the receipt of the message $\mathsf{P}(a, s, \mathsf{ACK1})$. This acknowledgment indicates that the intercepting gateway a has installed security polices directing all traffic flowing between s and d to travel in the newly established tunnel between s and a. The host sends back a similar acknowledgment $\mathsf{P}(s, a, \mathsf{ACK2})$ indicating that establishment has terminated and the corresponding security policies have been installed.

The host then awaits establishment to be initiated by d.

Await Fin Upon termination of establishment of the end-to-end tunnel between s and d, the host awaits receipt of a FIN acknowledgment indicating that the policy database at d were written and when this is received, the protocol terminates.

Gateway on Path We assume the gateway under consideration is designated as a and that it intercepts a discovery packet sent from node b. In the case of the first intercepting gateway on the path, $b = s$.

Intercept Discovery Message If gateway a receives a message of the form $\mathsf{P}(s, d, \mathrm{Dis}(b, u))$, it initiates establishment with node b.

Exchange Acks When establishment has successfully terminated, node a sends node b the message $\mathsf{P}(a, b, \mathsf{ACK1})$. The node then awaits the arrival of an acknowledgment from b.

Release Discovery Packet Upon receipt of a message of the form $\mathsf{P}(b, a, \mathsf{ACK2})$, the gateway releases the discovery packet $\mathsf{P}(s, d, \mathrm{Dis}(a, u))$, where node a has replaced b in the discovery message. The gateway awaits the next node e on the path to initiate establishment.

Exchange Acks When establishment has successfully terminated and a message of the form $\mathsf{P}(e, a, \mathsf{ACK1})$ has been received gateway a replies with a message of the form $\mathsf{P}(a, e, \mathsf{ACK2})$.

Destination Host

Intercept Discovery Message If the destination d receives a message of the form $\mathsf{P}(s, d, \mathrm{Dis}(b, u))$, it initiates establishment with node b.

Exchange Acks When establishment has successfully terminated, node d sends the message $\mathsf{P}(d, b, \mathsf{ACK1})$. Node d then awaits the receipt of an acknowledgment from node b.

Set up End-to-End Tunnel Upon receipt of a message of the form $\mathsf{P}(b, d, \mathsf{ACK2})$, node d initiates establishment with s. When establishment has successfully terminated, it sends the message $\mathsf{P}(d, s, \mathsf{FIN})$ to indicate that the tunnel complex enabling communication between s and d has been setup.

5. FORMALIZING DISCOVERY

Although space limitations have prevented us from being too formal in our presentation, in order to formulate our completeness theorem in the next section, we need to assume a greater degree of formalization when talking about the protocols. Following the presentation in [17], we assume that a discovery protocol is expressed as a set of rewrite rules, but limit our presentation to only those aspects needed in this paper. Our rewrite rules take the form

$$t_1 \mathbin{@} a_1, \ldots, t_n \mathbin{@} a_n \longrightarrow t'_1 \mathbin{@} a'_1, \ldots t'_m \mathbin{@} a'_m,$$

where t_i are terms and the a_i represent the address at which the term is located. The application of a rule $L \longrightarrow R$ to the multiset M rewrites to the multiset $M' = M - L' \cup R'$, where L' is a multiset of terms in M matching L and R' is a multiset matching the pattern R. The sequential application of a collection of rules yields a sequence of multisets $M_1, M_2, \ldots, M_{n+1}$ called a *trace* and provides a view of the multiset representing the network state as the protocol executes. Each change to the network state results in a new multiset being added to the trace sequence. For instance, if the policy evaluation mechanism returns true, then we assume that the trace records a $\uparrow \mathsf{GWPol}(u, \mathrm{true})$ term having been written.

In this presentation, we are primarily concerned with analyzing how credentials migrate during the execution of discovery protocols and whether these credentials satisfy the polices at the gateways. Since credentials get moved in messages we need to consider how message transmission is recorded in a trace. Consider a packet $\mathsf{P}(b, c, \mathrm{msg}) \mathbin{@} a$, with source address b, destination address c, and payload msg, located at node a. We model the movement of the packet from one node to another using the rewrite rule

$$\mathsf{P}(b, c, \mathrm{msg}) \mathbin{@} a \longrightarrow \mathsf{P}(b, c, \mathrm{msg}) \mathbin{@} f(a),$$

where the function f represents a forwarding table lookup. The execution of this rule rewrites the term $\mathsf{P}(b, c, \mathrm{msg}) \mathbin{@} a$ in M to $\mathsf{P}(b, c, \mathrm{msg}) \mathbin{@} f(c)$ in M'. As we have seen, discovery protocols move credentials from one node to another as they execute by including them in tunnel-establishment protocol messages. Hence $\Xi^u \mathbin{@} a$ becomes $\Xi^u \mathbin{@} b$ as the protocol moves the credentials from node a to node b.

Consider a trace $T = M_1, \ldots, M_n$, of the execution of a discovery protocol, we say $M_i \in T$ if M_i is identical to M_i in T and we say that for a term t, $t \in T$ if there exists a multiset $M_i \in T$ such that $t \in M_i$. We denote the set of elements appearing in a multiset M as $\mathfrak{L}(M)$.

If the gateway policies or the credentials defining the entities to which gateways belong change during execution of the discovery, then our notion of completeness is problematic. Consequently, we need to qualify the theorem with an assertion that the theorem holds for a fixed set of gateway polices and gateway credentials. This is formalized as follows. Suppose $T = M_1, \ldots, M_n$ is a trace recording the execution of a discovery protocol, where $a_1, \ldots, a_m$ are the nodes on the dataflow path. We say that the *gateway polices and node credentials are fixed in* T if for all i $(1 \leq i \leq m)$ and for all j, k $(1 \leq j, k \leq n)$,

$$\text{if } \Xi^{a_i} \in M_j \text{ and } (\Xi')^{a_i} \in M_k, \text{ then } \Xi^{a_i} = (\Xi')^{a_i}$$
$$\text{if } \Theta \mathbin{@} a_i \in M_j \text{ and } \Theta' \mathbin{@} a_i \in M_k, \text{ then } \Theta = \Theta'.$$

Let $\mathfrak{G}(M)$ denote the multiset where the credentials at each node in M are distributed to each node in the network of M. This is formalized as follows. Denote the nodes in the network of M as $a_1, \ldots, a_n$. The multiset $\mathfrak{G}(M)$ is defined to be the same as M, but with each Ξ^{a_i} is replaced by.

$$\bigcup_{1 \leq k \leq n} \Xi^{a_k}.$$

It is assumed that when a discovery protocol is run in $\mathfrak{G}(M)$ that the mechanism that determines whether credentials satisfy a gateway's policies uses the credentials that reside at the node where authorization is invoked. It is assumed that one protocol session cannot interfere with another. Suppose a discovery protocol always executes to completion if invoked in $\mathfrak{G}(M)$. This means that the protocol will always run to completion if the requisite credentials are delivered to the proper gateways on the dataflow path.

6. COMPLETENESS THEOREM

The case study in Section 4 illustrates how one particular protocol delivers credentials to gateways on the path in order to enable the traversal of the said gateways. A simplistic view of discovery protocol correctness would be to say that if the credentials necessary to traverse a gateway on the path are available either at the initiating host or at previous gateways on the path, then the protocol will deliver them. Yet this is somewhat presumptuous given that there may be valid reasons to restrict access to a gateway's credential set. Rather than exclude such protocols, we formulate a parameterized theorem that accommodates a variety of discovery protocols. The gist of our notion of completeness is that if the credentials that are supposed to be delivered to gateways on the path do indeed satisfy the gateway's policies, then the protocol delivers those credentials.

We view a defining characteristic of a discovery protocol to be the credential set it delivers to each node on the path. If setting up the complex of nested tunnels depicted

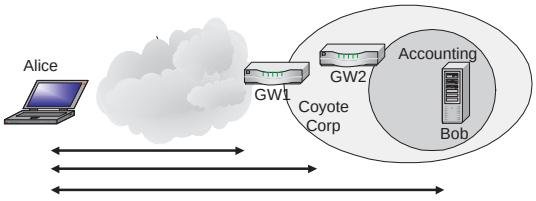

Figure 4: Nested Tunnel Complex

in Figure 4, the most obvious protocol would release a discovery packet that gets intercepted by $GW1$. The security gateway invokes tunnel establishment with Alice who sends her credentials to $GW1$. Upon termination of the establishment protocol, the discovery protocol releases the discovery packet, where Alice is still the given tunnel endpoint. Gateway $GW2$ intercepts this message and initiates establishment with Alice. In this simple protocol, the credentials from $GW1$ are not propagated back to Alice so unlike the concatenated protocol presented in Section 4, $GW2$ receives Alice's credentials, but not those from $GW1$. This models a common situation where road warriors must provide passwords and other credentials to set up VPNs to each of gateway on the path in order to establish communication with a corporate server.

Having seen that different discovery protocols have different credential delivery patterns, we define a *protocol's characterizing credential set* $\mathcal{C}$ as the credentials delivered to each node by a specific protocol. In the case of the nested protocol given above, the characterizing credential set is $\mathcal{C}_{a_i} = \{K_{a_i} \Rightarrow K_a\} \cup \Xi^a$, where a is the initiating host and node a_i was the last discovered node. In the case of the concatenated protocol presented in Section 4, the characterizing credential set is more subtle, the protocol delivers to node a_i the credentials from previously discovered gateways $a_1, \ldots, a_{i-1}$. This is formalized as

$$\mathcal{C}_{a_i} = \bigcup_{1 \le l < i} (\{K_{l+1} \Rightarrow K_l\} \cup \Xi^{a_l}). \tag{1}$$

It is possible for different protocols to possess the same characterizing credential set. For instance, one can construct a protocol that sets up the nested tunnels in Figure 4 where the credentials at gateways are propagated back to the initializing host and used to set up the next tunnel. This would have the same characterizing credential set as our concatenated protocol.

We can now formulate a general completeness theorem defining correctness in terms of credential delivery. The theorem is stated in terms of the execution trace of a specific session u of the protocol in question. Recall that the session identifier is assumed to be unique. To ensure that the trace records the initialization of session u we insist that u not appear in first multiset of the trace. In order to keep the focus on credential delivery, we assume that the protocol would execute successfully if every node had all the credentials needed. The theorem states that if the credential set that is supposed to be delivered to a node on the path satisfies that node's policies, then the trace records that the credential evaluation mechanism returns an affirmative judgment for that node. The theorem is formalized as follows.

THEOREM 1. *Given a discovery protocol P with characterizing credential set $\mathcal{C}_{a_i}$, let $T = M_1, \ldots, M_n$ be a trace of the execution of session u of P initiated at node a_1 $(= s)$ to communicate with node a_n $(= d)$ and the gateway policies and node credentials are assumed fixed in T, where u is a unique identifier. Assume $u \notin \mathfrak{L}(M_1)$ and that the forwarding tables indicate that the nodes on the dataflow path are $a_2, \ldots, a_{n-1}$. Assume that a protocol session u always runs successfully when initiated in $\mathfrak{G}(M_1)$. For each i $(2 \le i \le n)$ there exists a term $\uparrow \mathsf{GWPol}(u, true) @ a_i$ such that the following statement holds: if*

$$\mathcal{C}_{a_i} \models_{a_i} \Theta_{a_1 \leftrightarrow a_n} @ a_i$$

then

$$\uparrow \mathsf{GWPol}(u, true) @ a_i \in T.$$

To apply the theorem to a specific protocol, one must formalize the characterizing credential set and then prove the theorem for that protocol. We illustrate this for the concatenated protocol described in Section 4, using the characterizing credential is given in Equation 1.

PROOF. Proof is by induction on the number of gateways.

For the base case there are no intermediate gateways only source a_1 and destination a_2. Consider the execution of the concatenated discovery protocol. Node a_1 releases the discovery packet and invokes the establishment responder. Upon receiving the discovery packet, node a_2 invokes the establishment protocol initiator to set up a pair of associations between a_1 and a_2. The establishment protocol responder sends $\Xi^u = \Xi^{a_1} \cup \{K_{a_2} \Rightarrow K_{a_1}\}$ to a_2 in the establishment response message. Upon receiving this message, the protocol invokes the policy evaluation mechanism with the credentials received . Since it was assumed that $(\Xi @ a_1 \cup \{K_{a_2} \Rightarrow K_{a_1}\}) \models_{a_2} \Theta_{a_1 \leftrightarrow a_2} @ a_2$, we can conclude that the policy evaluation mechanism returns true so the trace records a term $\uparrow \mathsf{GWPol}(u, \text{true}) @ a_2$ and thus the base case is satisfied.

Suppose the statement is true for i gateways, we show it is true for $i+1$ gateways. It follows from the induction hypothesis that the discovery protocol ran to the point that it had successfully set up tunnels between a_j and a_{j+1}, where $1 \le j < i$. It remains to show that the protocol successfully sets up the pair of associations between a_i and a_{i+1}. Gateway i is not the final destination, hence it releases the discovery packet. Upon receiving this message, gateway a_{i+1} executes invokes the establishment protocol to set up a pair of associations between a_i and a_{i+1}. The establishment protocol responder sends $\Xi^u = \Xi^u \cup \Xi^{a_i} \cup \{K_{a_{i+1}} \Rightarrow K_{a_i}\}$ to a_{i+1}. It follows from the induction hypothesis, that authorization succeeded at nodes $a_1, \ldots, a_i$ and consequently the credential sets from these nodes are passed to the next node on the path as the protocol executes; so the credential set sent from a_i to a_{i+1} is

$$\bigcup_{1 \le l \le i} (\Xi @ a_l \cup \{K_{a_{l+1}} \Rightarrow K_{a_l}\}).$$

Upon receiving the establishment response message, the protocol invokes the policy evaluation mechanism. Since it was assumed that

$$\bigcup_{1 \le l \le i} (\Xi @ a_l \cup \{K_{a_{l+1}} \Rightarrow K_{a_l}\}) \models_{a_{i+1}} \Theta_{a_1 \leftrightarrow a_n} @ a_{i+1},$$

we can conclude that the policy evaluation mechanism returns true so the trace records a term $\uparrow \mathsf{GWPol}(u, \text{true}) @ a_{i+1}$. The theorem follows from induction. □

The concatenated protocol sets up a fixed topology, consequently the formulation of the characterization credentials as well as the completeness proof is straightforward, but more dynamic protocols can be subtle. For instance, a protocol that dynamically sets up a less uniform tunnel complex can create a tunnel that in effect hides gateways so that their credentials are not available to present to the latest discovered gateways [16]. Formalization and proof of the completeness theorem for such a protocol is nontrivial.

7. RELATED WORK

A popular VPN configuration in use today is the hub and spoke model where a collection of branch offices connect to a central 'hub' office. Cisco's Dynamic Multipoint VPN (DMVPN) [11] protocol can be viewed as a basic tunnel-complex protocol aimed at this topology. DMVPN alleviates the burden on the hub by setting up direct spoke-to-spoke IPsec tunnels. The spokes typically have dynamic addresses, but maintain a tunnel to the hub. The hub acts as a next hop resolution protocol server (NHRP) [27] with a static IP address. A spoke router learns the address of another spoke from the NHRP hub. If Alice is a host located in spoke A and she wishes to communicate with host Bob located in spoke B, Alice will notify the DMVPN spoke router, which in turn sends a NHRP resolution request to the hub. The hub then sends a resolution request on to spoke B. Spoke B then initiates IKE tunnel establishment with spoke A. When a DMVPN process terminates, an IPsec tunnel is set up directly between spokes A and B through which Alice and Bob communicate.

The Layer 3 Accounting (L3A) protocol [18] is a tunnel-complex protocol that sets up a complex of tunnels to protect a network's accounting infrastructure from DoS attacks.

Cisco's Tunnel Endpoint Discovery (TED) [15] is a discovery protocol that assumes only a single gateway in each organization. The protocol works as follows. Host A sends a packet to Host B. A router acting as a gateway to the domain where host A dwells intercepts the packet and checks to see if there is an existing association with a matching policy. If so, the packet is sent to its destination. Otherwise, a discovery probe is sent to the packet's destination. A gateway protecting Host B intercepts the probe and sends back a reply. The tunnel-establishment protocol IKE is then invoked to set up an IPsec tunnel between the two gateways. More complex discovery protocols are needed in order to traverse nested gateway structures.

Cisco's Group Encrypted Transport VPN (GET VPN) [10] takes a different approach to the one advocated in this document in that it employs a centralized key server holding all polices relating to tunnels. Gateways download security polices and security association configuration information from the central server. All members in the GET VPN use the same shared key. The GODI group key management protocol [3] is employed to manage the shared keys. Although this approach may work well within an organization, it does not easily scale to facilitate VPNs that cross enterprise boundaries.

The concatenated protocol defined in Section 4 can be viewed as a generalization of TED to more than two gateways and the protocol sketched in Section 6 to set up the nested tunnel complex illustrated in Figure 4 generalizes and automates a common configuration used by road warriors today. Our models and implementations [18] have assumed a simple network infrastructure. In practice, discovery protocols may encounter problems interacting with existing infrastructure such as Multiprotocol Label Switching Architecture (MPLS) [30]. Network route changes may break a tunnel complex requiring the protocol to be rerun. Further research is required to model such interactions.

In order to traverse a security gateway, a principal must be authenticated and authorized to perform the requested action. In the case of discovery protocols, we also require gateways to authenticate themselves. Within our framework, authorization is performed using distributed credentials and assumes the existence of some public key infrastructure. Tunnel-complex protocols are responsible for delivering the credentials that satisfy a particular node's policy. There is a large body of work on distributed credentials that we took inspiration from. Rivest and Lampson proposed a public-key infrastructure called the Simple Distributed Security Infrastructure (SDSI) [29]. SDSI principles are identified with public keys and only things signed by the corresponding private key are recognized. Ellison and Franz, et al. developed the Simple Public Key Infrastructure (SPKI) that provides a mechanism for authorization. SPKI uses authorization certificates to delegate a specific authority from an issuer to a subject. The two efforts have been merged and there is currently a SPKI/SDSI working group in the IETF that has produced several RFCs [12–14]. The tunnel calculus authorization layer can be viewed as similar to, but simpler, than SPKI/SDSI. There is a substantial body of research dedicated to giving a formal semantics to SPKI/SDSI [1, 20, 21, 26], and [23]. The trust management approach [5] to authorization seeks to create an application independent component to verify if a request is authorized to take some action. Given the local security policy, a set of credentials, and a request, the trust engine returns a decision as to the whether or not the request complies with the policy. PolicyMaker [6, 7] and Keynote [8] are notable examples of this approach. The Query Certificate Manager (QCM) [19] is closely related to completeness for discovery protocols because it combines verification with protocols for transporting certificates to the needed verifier. Our model for credentials can be viewed as an abstraction inspired by SPKI and our model of policies are admittedly simple compared to what some others have proposed, but both are sufficient for our purposes. Rather than model the, necessarily complex, machinery used in a specific technology for verifying that a credential set satisfies a given policy, we simply define a relation that acts as a specification that could be satisfied by many of aforementioned proposals.

The π-calculus based ProVerif tool [4] provides sophisticated automated assistance to reason about secrecy properties and has been applied to the analysis of the JFK [2] key exchange protocol. State machines have long been used to represent protocols. Guttman, Herzog, and Thayer [22] model IPsec tunnels using state machines and formalize authentication and confidentiality properties.

8. CONCLUSION

The design of discovery protocols has been a difficult process. Despite the clear benefits of a more dynamic way to

configure tunnels and gateways and the existence of flexible base protocols that should be good building blocks for tunnel complex construction and a number of attempts to advance standards for them, discovery protocols still seem more a dream for the future than a current-day reality. Developing an appropriate theoretical foundation can play a valuable role in exploring the design space. In this paper we advance one important type of result: the completeness of a discovery protocol. We have shown how at least one major type of discovery protocol, one based on concatenated tunnels, can be shown complete. This step may show the way to further theoretical advances that lead to a scientific foundations for the design of practical discovery protocols.

Acknowledgments

This work was supported in part by NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, ONR N00014-08-1-0248, NSF CNS 05-24516, NSF CNS 05-24695, DHS 2006-CS-001-000001, and grants from the MacArthur Foundation and Boeing Corporation. The views expressed are those of the authors only.

9. REFERENCES

[1] M. Abadi. On SDSI's Linked Local Name Spaces. *Journal of Computer Security*, 6(1-2):3–21, 1998.

[2] M. Abadi, B. Blanchet, and C. Fournet. Just Fast Keying in the Pi calculus. In D. Schmidt, editor, *European Symposium on Programming (ESOP)*, Lecture Notes in Computer Science 2618. Springer-Verlag, 2004.

[3] M. Baugher, B. Weis, T. Hardjono, and H. Harney. The Group Domain of Interpretation (GDOI). RFC 3547, IETF, 2003.

[4] B. Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *Proceedings of the 25th Annual IEEE Symposium on Security and Privacy (Oakland 04)*, pages 86–100. IEEE, 2004.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The Role of Trust Management in Distributed Systems Security. In *Secure Internet Programming*, Lecture Notes in Computer Science 1603. Springer-Verlag, 1999.

[6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proceedings of Symposium on Security and Trust Management*, pages 164–173, 1996.

[7] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in Policy Maker. In *Proceedings of Financial Cryptography*, Lecture Notes in Computer Science 1465, pages 254–274. Springer-Verlag, 1998.

[8] M. Blaze, J. Ioannidis, and A. Keromytis. Trust Management in IPsec. *ACM Transactions on Information and System Security*, 32:1–24, 2002.

[9] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[10] Cisco. Group Encrypted Transport VPN (GET VPN): Design and Implementation Guide. Document V1.0, August 2008.

[11] Dynamic Multipoint VPN (DM VPN). Cisco White Paper.

[12] C. Ellison. SPKI Requirements. RFC 2692, IETF, 1999.

[13] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Simple Public Key Certificate. Technical report, IETF, 1999.

[14] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, IETF, 1999.

[15] S. Fluhrer. Tunnel endpoint discovery. Internet Draft draft-fluhrer-ted-00.txt, IETF, 2001.

[16] A. Goodloe, M. McDougall, M.-O. Stehr, and C. A. Gunter. Design and Analysis of Sectrace: A Protocol to Set up Security Associations and Policies in IPsec Networks. September 2004.

[17] A. E. Goodloe and C. A. Gunter. Reasoning About Concurrency for Security Tunnels. In *Proceedings of 20th IEEE Computer Security Foundations (CSF 07)*, pages 64–78. IEEE, 2007.

[18] A. E. Goodloe, M. Jacobs, G. Shah, and C. A. Gunter. Proceedings of the L3A: A protocol for layer three accounting. In *Proceedings of Secure Network Protocols (NPSec '05)*, Boston, MA, November 2005. IEEE.

[19] C. A. Gunter and T. Jim. Policy-Directed Policy Certificate Retrieval. *Software: Practice and Experience*, 30(15):1609–1640, September 2000.

[20] J. Halpren and R. van de Meyden. A Logic for SDSI's Linked Local Name Space. *Journal of Computer Security*, 9(1):47–74, 2001.

[21] J. Halpren and R. van de Meyden. A Logical Reconstruction of SPKI. *Journal of Computer Security*, 11(4):581–614, 2003.

[22] J. Guttman and A. Herzog and F. Javier Thayer. Authentication and Confidentiality via IPsec. In F. Cuppens and Y. Deswarte and D. Gollmann and M. Waidner , editor, *Proceedings of the 6th Annual European Symposium on Research in Computer Security (ESORICS)*, Lecture Notes in Computer Science 1895, pages 255–272. Springer-Verlag, 2000.

[23] S. Jha and T. Reps. Analysis of SPKI/SDSI Certificated Using Model Checking. In *Proceedings of 15th IEEE Computer Security Foundations Workshop (CSF 02)*, pages 129–144. IEEE, 2002.

[24] C. Kaufman. Internet Key Exchange (IKE V2) protocol. RFC 4306, IETF, 2005. Obsoletes: 2407, 2408, 2409.

[25] S. Kent and K. Seo. Security architecture for the internet protocol. RFC 4301, IETF, 2005. Obsoletes: 2401.

[26] N. Li and J. Mitchell. Understanding SPKI/SDSI Using First Order Logic. In *IEEE Computer Security Workshop*, pages 89–103, 2003.

[27] J. Luciani, D. Katz, D. Piscitello, B. Cole, and N. Doraswamy. Next Hop Routing Protocol (NHRP). Technical report, IETF, 1998. RFC.

[28] A. J. Menezs, P. C. van Oorchot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[29] R. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure, 1996.

[30] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, IETF, 2001.