Authentication and Confidentiality via IPsec*

Joshua D. Guttman, Amy L. Herzog, and F. Javier Thayer

The MITRE Corporation
{guttman, althomas, jt}@mitre.org

Abstract. The IP security protocols (IPSEC) may be used via security gateways that apply cryptographic operations to provide security services to datagrams, and this mode of use is supported by an increasing number of commercial products. In this paper, we formalize the types of authentication and confidentiality goal that IPSEC is capable of achieving, and we provide criteria that entail that a network with particular IPSEC processing achieves its security goals.
This requires us to formalize the structure of networks using IPSEC, and the state of packets relevant to IPSEC processing. We can then prove confidentiality goals as invariants of the formalized systems. Authentication goals are formalized in the manner of [9], and a simple proof method using "unwinding sets" is introduced. We end the paper by explaining the network threats that are prevented by correct IPSEC processing.

1 Introduction

The IP security protocols [7, 5, 6] (see also [8, 4]), collectively termed IPSEC, are an important set of security protocols currently making their way into the commercial world. The IPSEC standards include protocols for ensuring confidentiality, integrity, and authentication of data communications in an IP network. The standards are very flexible, and this flexibility has led to great commercial interest; many IPSEC products are now available. The same flexibility also means that the protocol set is complex [2]. Hence, naïvely configured IPSEC products will often be set up wrong, making it hard to know what security goals have actually been achieved. Our rigorous treatment suggests an approach to constructing IPSEC configuration tools, and suggests specific checks by which a system administrator can ensure his goals are met, even without a tool.

IPSEC can be used in two different ways. It can be used end-to-end, in which case the source and destination hosts for a datagram are responsible for all cryptographic processing. It can also be used via gateways, in which case a system near the source host is responsible for applying cryptographic operations on behalf of the source, while a system near the destination is responsible for checking and decryption. A flow of packets in which at least one endpoint is an IPSEC gateway is called a *tunnel*.

* This work was supported by the National Security Agency through US Army CECOM contract DAAB07-99-C-C201.

The IPsec protocols work by prefixing special IP headers (or infixing special header fields). These headers contain an index (the Security Protection Index) by which the system applying the cryptographic operations specifies algorithms and keys to be used. The cryptographic operations are intended to provide authentication and integrity[1] for packets, or else confidentiality for packets. A single IPsec header may provide both authentication and confidentiality, and indeed it is sound practice not to provide confidentiality without authentication [1]. Authentication is provided by means of keyed hashes, confidentiality by symmetric encryption. Hence, in both cases secrets must be shared between the systems applying and removing the cryptography. Manual key placement or cryptographic key exchange methods [4, 8] may be used to create these shared secrets.

We will always regard the action of an IPsec gateway as a matter of manipulating headers. To apply cryptography, a source host or a gateway wraps the datagram (including its non-IPsec header information) in a new header. If this header offers authentication then it can be applied only by a system sharing the symmetric key. Moreover, the payload is protected from alteration in the sense that alteration can be detected, and will prevent delivery of the (damaged) payload. If the new header offers confidentiality, then it can be removed only by a system sharing the symmetric key. Headers are applied at one end of a tunnel and checked and removed at the other. We abstract from operations on the payload itself, namely encrypting it or calculating a hash over it, although of course these operations are necessary for IPsec to be useful.

When IPsec is used via gateways, the hosts (or the organizations operating them) delegate a degree of *trust* to certain gateways. The purpose of this paper is to study the logic of that trust. We will formalize exactly what assumptions are needed for the two types of goal, authentication and confidentiality, and explain how the trust assumptions depend on network topology. We regard this paper as an extension of a research program, begun in [3], which aims to analyze the local processing required to enforce network-wide security policies. In [3], we studied the local packet filtering behavior routers must be trusted to perform, in order to enforce network-wide firewall-like security goals.

In the next section, we formalize the security goals one can achieve via IPsec, and introduce the notion of a *trust set*, which is central to our analysis. We then formalize the structure of networks using IPsec (Section 3.1), the state of packets relevant to IPsec processing (Section 3.2), and the properties of cryptographic operations (Section 3.3). We detail our behavior requirements, and prove that they are sufficient to ensure goal achievability (Section 4). We illustrate specific attacks that our approach prevents in Section 5. We end by summarizing, and discussing potential future work.

[1] For our purposes, integrity and authentication belong together. Jointly, they ensure that the packet has originated at a known system, and has remained unchanged since, except for such header manipulations required by IP routing and delivery mechanisms. We will speak henceforth only of authentication; it should be understood that integrity is also included.

2 Achievable Security Goals

We focus on authentication and confidentiality as security goals in our analysis. Concrete security goals select certain packets that should receive protection [7]; selection criteria may use source or destination addresses, protocol, and other header components such as the ports, in case the protocol is TCP or UDP.

2.1 Authentication Goals

The essence of authentication is that it allows the recipient to—so to speak—take a packet at face value. Thus, for a packet p selected for protection by a authentication goal,

> If A is the value in the source header field of p as received by B, then p actually originated at A in the past, and the payload has not been altered since.

We do not regard a packet as being (properly) received unless the cryptographic hash it contains matches the value computed from a shared secret and the packet contents. It will not be delivered up the stack otherwise, nor forwarded to another system after IPsec processing.

2.2 Confidentiality Goals

We assume that confidentiality headers (as in the Encapsulating Security Payload (ESP) protocol [6]) provide authentication, and add encryption. We have two reasons for doing so. First, the IPsec specification allows both authentication and confidentiality to be used with the ESP header; it is inadvisable to request only confidentiality when authentication can also be had at the same time, and at modest additional processing cost. Second, it seems hard to state precisely what data is kept confidential, if that data might change as the packet traverses the network. It is thus hard to say what protection has been achieved [1, 2]. When using confidentiality headers, we are therefore attempting to achieve an authentication goal as well as a confidentiality goal.

A confidentiality goal for a packet with source field A, requiring protection from disclosure in some network location C, stipulates:

> If a packet originates at A, and later reaches the location C, then while it is at C it has a header providing confidentiality.

The cryptographic protection may refer to the ESP header more specifically, stipulating certain parameters (key length, algorithm, etc). The proviso that the packet was once at A is necessary, because in most cases we cannot prevent someone at C from creating a spoofed packet with given header fields. However, a spoofed packet cannot compromise the confidentiality of A's data if it has no causal connection to A.

2.3 Example Goals

Consider the network in Figure 1. Given this example network, a potential authentication goal could be that packets traveling from *EngineeringA* to *EngineeringB* should be authenticated, meaning that any packet with source field claiming to be from *EngineeringA* that reaches *EngineeringB* should in fact have originated in *EngineeringA*. An example confidentiality goal is that packets traveling from *FinanceA* to *FinanceB* should be encrypted whenever outside those areas. This means that if a packet has source field in *FinanceA*, and actually originated there, then if it reaches any other area R, it has an ESP header providing encryption while at R.

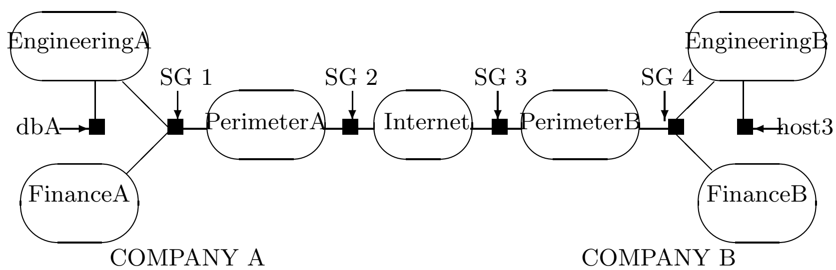

Fig. 1. Sample IPSEC Network Representation

One advantage to this form of expression is that it is semantically precise. Another is that policies expressed in this form appear to be intrinsically composable, in the sense that separate goals can always be satisfied together. Moreover, this form of expression often suggests placement of *trust sets*, in a sense we will now introduce.

2.4 Trust Sets

Once a packet enters an appropriate cryptographic tunnel, achieving a security goal does not depend on what happens until it exits. Thus, the set of locations in the network topology that are accessible to the packet from the source (before entering the tunnel) or accessible from the exit of the tunnel (before reaching the destination) are the only ones of real importance. We will call these locations a *trust set* for a particular security goal. A trust set is goal-specific; different goals may have different trust sets. For instance, an engineering group working on a sensitive project could easily have much more restrictive security goals than its parent corporation (in terms of trust).

Typically, a trust set is not a *connected* portion of the network. In many of the examples we will describe later, the trust set consists of two large connected portions, with a large public 'Internet' network between them. In some cases the trust set may consist of several islands, and the tunnels may not connect all of them directly. In this case, a packet may need to traverse several tunnels successively in order to get from one island of the trust set to a distant one.

The choice of trust set for a particular security goal is a matter of balance. Clearly, the source must belong to the same island of the trust set as the tunnel entrance, and the tunnel exit must belong to the same island as the destination (or the entrance to the next tunnel). This encourages creating trust sets as large as possible, since then a few tunnels may serve for many endpoints. However, the scope of a trust set must generally be limited to a set of networks on which it is possible to monitor traffic and check configurations. This encourages making the trust sets as small as possible. The art of using IPsec effectively consists partly in balancing these two contrasting tendencies.

Boundaries Of special importance are those systems inside a trust set with a direct connection to systems outside the trust set. We term these systems the *boundary* of the trust set. We assume that every device on the boundary of a trust set is capable of filtering packets. This may be a portion of its IPsec functionality [7]. Alternatively, the device may not be IPsec-enabled, but instead be a filtering router or packet-filtering firewall. We regard such devices as a degenerate case of an IPsec-enabled device, one which happens never to be configured to apply any cryptographic operations.

3 Network Modeling

We begin talking about systems by viewing them as composed of networks and devices capable of IPsec operations or packet filtering. A device has interfaces on one or more networks. Any machine (such as a switch or host) that performs no IPsec operations or filtering we may simply ignore. We may also ignore a machine that can perform IPsec operations, if it is not a member of the trust set for any security goal we would like to enforce. For instance, IPsec-enabled machines elsewhere on the Internet can be ignored.

We regard a system as a graph with two kinds of nodes, representing the networks and the devices respectively. In the example shown in Figure 1, networks appear as ovals and devices appear as black squares. An edge represents the interfaces between a device and the network to which it is connected. We will never connect two networks directly via an edge; this would not give a security enforcement point to control the flow of packets between them. Instead, we will coagulate any two networks that are connected by a device that provides no security enforcement, representing them by the same oval. Figure 1 is a simple picture: for any two nodes, deletion of a single edge causes them to become disconnected. In other cases, there may be many disjoint paths between a pair of locations.

3.1 System Model

While the simple representation we have just described is useful for understanding one's network in terms of security policy, it is inconvenient for more rigorous examination. IPsec processing depends heavily on which interface a packet is traversing, as well as the direction in which the packet is traversing that interface. Therefore it is convenient to have a system model in which there are two nodes corresponding to each interface. They represent the conceptual location of a packet when IPsec processing is occurring, either as it traverses the interface inbound into the device or as it traverses the interface outbound from the device. We call these conceptual locations *directed interfaces*.

We introduce a model consisting of a directed graph containing three kinds of nodes. These represent networks, devices, and directed interfaces. To construct a model from a system representation in the style of Figure 1, for each edge between a device g and a network r we two directed interface nodes, which we will call $i_g[r]$ and $o_g[r]$. These represent inbound processing for a packet traveling from r to g and outbound processing for a packet traveling from g to r respectively. We add four directed arcs:

1. $r \to i_g[r]$ and $i_g[r] \to g$, the inbound arcs, and
2. $g \to o_g[r]$ and $o_g[r] \to r$, the outbound arcs.

For instance, the result of applying this process to the system representation shown in Figure 2 produces the enriched model shown in Figure 3.

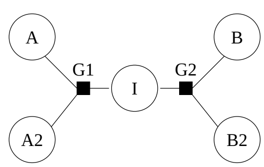

Fig. 2. Unenriched System Representation

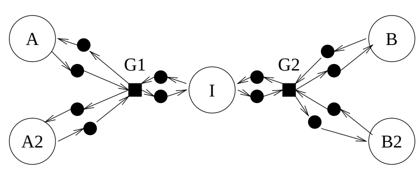

Fig. 3. Enriched System Representation

We will assume an enriched system representation $G = (V, E)$ throughout the remainder of this section. A location ℓ is a node, i.e. a member of V.

3.2 Packet States

Let P be a set of values we call protocol data. We may think of its values as the elements of IP headers other than source and destination. For instance, an IP header may specify that the protocol is TCP, and the embedded TCP header may specify a particular source port and destination port; this combination of protocol and port information may be taken as a typical member of P.

Let $A \subset P$ be a set we call authenticated protocol data; it represents those headers that provide IPsec authentication services. Let $C \subset A$ be a set we call confidentiality protocol data; it represents those headers that provide IPsec confidentiality services. The assumption $C \subset A$ codifies our decision not to consider ESP headers that provide only confidentiality (cf. Section 2.2).

A *header* is a member of the set $H = V \times V \times P$, consisting of a source location, a destination location, and a protocol data value. Packet states are members of H^*, that is, possibly empty sequences $\langle h_1, \dots, h_n \rangle$. We use $\cdot$ as prefixing operator: $h \cdot \langle h_1, \dots, h_n \rangle = \langle h, h_1, \dots, h_n \rangle$.

Let K be a set of "processing states," with a distinguished element $\mathsf{ready} \in K$. Intuitively, when an interface has taken all of the processing steps in the Security Association (SA, see [7]) for a packet p, then it enters the processing state ready, indicating that the packet is now ready to move across the arc from that interface. If this is an outbound interface, it means that the packet may now go out onto the attached network; if it is an inbound interface it means that the packet may now enter the device, typically to be routed to some outbound interface or for local delivery. Other members of K are used to keep track of complex IPsec processing, when several header layers must be added or removed before processing is complete at a particular interface. These clusters of behavior represent IPsec Security Association bundles.

We regard the travels of a packet through a system as the evolution of a state machine. The packet may not yet have started to travel; this is the start state. The packet may no longer be travelling; this is the finished state. Every other state is a triple of a node $\ell \in V$, indicating where the packet currently is situated; a processing state $\kappa \in K$, indicating whether the packet is ready to move, or how much additional processing remains; and a packet state $\theta \in H^*$, indicating the sequence of headers nested around the payload of the packet.

Definition 1. $\Omega(G, K, P, A, C)$ is the set of network states over the graph $G = (V, E)$, the processing states K, and the protocol data P with $C \subset A \subset P$. $\Omega(G, K, P, A, C)$ is the disjoint union of

1. start,
2. stop, and
3. the triples (ℓ, κ, θ), for $\ell \in V$, $\kappa \in K$, and $\theta \in H^*$.

The transition relation of a network state machine is a union of the following parameterized partial functions. We define what the resulting state is, assuming that the function is defined for the state given. We also constrain when some of these functions may be defined; different IPsec security postures are determined by different choices of domain for each of these partial functions (subject to the constraints given).

Definition 2. A network operation is any partial function of one of the following forms:

1. Packet creation operators $\mathsf{create}_{\ell,h}(\mathsf{start}) = (\ell, \mathsf{ready}, \langle h \rangle)$, when defined, for $(\ell, h) \in V \times H$. $\mathsf{create}_{\ell,h}$ is not defined unless its argument is the state start.
2. The packet discard operator $\mathsf{discard}(\ell, \kappa, \theta) = \mathsf{stop}$, when defined. $\mathsf{discard}$ is undefined for start.
3. Packet movement operators $\mathsf{move}_{e,\kappa}(\ell, \mathsf{ready}, \theta) = (\ell', \kappa, \theta)$, when $e \in E$, $\ell \xrightarrow{e} \ell'$ and $\kappa \neq \mathsf{ready}$. $\mathsf{move}_{e,\kappa}$ is undefined for all other network states.
4. Header prefixing operators $\mathsf{prefix}_{h,\kappa}(\ell, \kappa', \theta) = (\ell, \kappa, h \cdot \theta)$ when defined. The function $\mathsf{prefix}_{h,\kappa}$ is nowhere defined when $h \notin A$.
5. Header pop operators $\mathsf{pop}_{\kappa}(\ell, \kappa', h \cdot \theta) = (\ell, \kappa, \theta)$ when defined.
6. Null operators $\mathsf{null}_{\kappa}(\ell, \kappa', \theta) = (\ell, \kappa, \theta)$ when defined.

A transition relation $\rightarrow \subset (\Omega(G, K, P, A, C) \times \Omega(G, K, P, A, C))$ is a union of operators create, $\mathsf{discard}$, move, prefix, pop, and null.

The assumption that $\mathsf{prefix}_{h,\kappa}$ is nowhere defined when $h \notin A$ means that the only nested headers we consider are IPSEC headers.

We call the assumption that $\mathsf{move}_{e,\kappa}(\ell, \kappa', \theta) = (\ell', \kappa, \theta)$ is not defined when $\kappa' \neq \mathsf{ready}$ the **motion restriction**. We call the assumption that it is not defined when $\kappa = \mathsf{ready}$ the **inbound motion restriction**. The motion restriction codifies the assumption that a device will not move a packet until it is ready. The inbound motion restriction codifies the assumption that there will always be a chance to process a packet when it arrives at a location, if needed, before it is declared ready to move to the next location.

Given a packet p, we call the address in the source header field of its topmost header $\mathsf{src}(p)$. We call the address in the destination header field of the topmost header $\mathsf{dst}(p)$. We also call a packet p an IPSEC packet if its outermost header is an AH or ESP header; an ESP packet if its outermost header is an ESP header; and an AH packet if its outermost header is an AH header.

Definition 3. A trust set S for $G = (V, E)$ consists of a set $R \subset V$ of networks, together with all devices g adjacent to networks in R and all interfaces $i_g[*]$ and $o_g[*]$.

The inbound boundary of S, written $\partial^{in} S$ is the set of all interfaces $i_g[r]$ or $i_g[g']$ where $g \in S$ and $r, g' \notin S$.

The outbound boundary of S, written $\partial^{out} S$ is the set of all interfaces $o_g[r]$ or $o_g[g']$ where $g \in S$ and $r, g' \notin S$.

In the remainder of this section, fix a trust set S and locations $a, b \in S$. We will use the following notation: Suppose $x = (\ell, \kappa, \theta)$ is a network state where $x \neq \mathsf{start}, \mathsf{stop}$.

- $\ell(x) = \ell$,

- $\kappa(x) = \kappa$,
- $\theta(x) = \theta$.

A transition $x \to y$ is *header non-augmenting* iff it is of the form $(\ell, \kappa, \theta^\frown\theta') \to (\ell, \kappa', \theta')$, where θ' is a final segment of the concatenation $\theta^\frown\theta'$.

3.3 Cryptographic Assumptions

We will make two assumptions about the IPsec cryptographic headers. First, we assume that cryptographic headers cannot be spoofed; in other words, that if we receive a message with an authenticating header from a source "known to us,"[2] then the entity named in the source field of the header is the entity that applied the header, and the payload cannot have been changed without detection. Second, confidentiality headers have the property that packets protected with them can be decrypted only by the intended recipient, i.e. the device named in the ESP header destination field. More formally, using a dash for any field that may take any value, we stipulate that for any transition:

$$\begin{array}{c} (\ell, \kappa, \langle [s', d', -], \ldots \rangle) \\ \big\downarrow \\ (\ell, \kappa', \langle [s, d, \alpha], [s', d', -], \ldots \rangle) \end{array}$$

$\alpha \in A$ and $s \in S$ implies $\ell = s$. Moreover, for any transition:

$$\begin{array}{c} (\ell, \kappa', \langle [s, d, \gamma], [s', d', -], \ldots \rangle) \\ \big\downarrow \\ (\ell, \kappa, \langle [s', d', -], \ldots \rangle) \end{array}$$

$\gamma \in C$ and $s \in S$ implies $\ell = d$.

These properties axiomatize what is relevant to our analysis in the assumption that key material is secret. If keys are compromised then security goals dependent on them are unenforceable.

4 Security Goal Enforcement

Given an environment in which one can rigorously reason about packet states, and precise specifications of security goals, how does one ensure the goals are enforced? This section focuses on answering that question, by detailing formal behavior requirements for systems and then proving they guarantee enforceability.

In our reasoning, we will assume that security goals are stated in the form given in Section 2.3.

[2] Presumably as certified by some public key infrastructure, and certainly assumed to include those devices which are shown as nodes in the system model.

4.1 Authentication

Our authentication problem can be stated in the following way. Suppose a and b are network nodes. What processing conditions can we impose such that an authentication goal holds?

To make this precise, let us say an *authenticated state* is one having the form $(a, \kappa, \langle [a, -, -] \rangle)$ and an *acceptor state* is one of the form $(b, \mathsf{ready}, \langle [a, -, -] \rangle)$. The symbol $\mathsf{authentic}$ denotes the set of authenticated states, accept denotes the set of acceptor states. Our question can be stated thus: exhibit a set of processing restrictions which ensure the following:

> For any path $\mathsf{start} \longrightarrow^* \omega$ where $\omega \in \mathsf{accept}$ there is an intermediate state ω' so
>
> $$\mathsf{start} \longrightarrow^* \omega' \longrightarrow^* \omega$$
>
> where $\omega' \in \mathsf{authentic}$.

Thus, whenever an acceptor state is reached, an authenticated state must have occurred earlier in the state history. In this sense, the prior occurrence of an authenticated state is guaranteed when an acceptor state is observed. This use of "authenticated" for the states $\omega' \in \mathsf{authentic}$ follows Schneider [9].

Achieving authentication requires two types of behavior restrictions on trusted nodes, depending on whether the system in question is in the boundary or not. We list behavior restrictions for each.

First we list a constraint that is required for the proofs, but is vacuous in IPsec [7], where inbound processing can only remove packet headers (but never add them).

Prefix Ready Rule.

$$\begin{array}{c}(\ell, \kappa, \theta)\\ \downarrow\\ (\ell, \kappa', h \cdot \theta)\end{array}$$

If $\ell \in \partial^{in} S$ then $\kappa = \mathsf{ready}$.

Authentication Tunnel Constraints In order to achieve authentication, there are two rules that must be observed by every IPsec-enabled device in the trust set. The first of these is that nodes in S must not spoof packets with sources in S.

Creation Rule. For any transition

$$\begin{array}{c}\mathsf{start}\\ \downarrow\\ (\ell, \kappa, \langle [s, -, -] \rangle)\end{array}$$

if $\ell \in S$ then $\ell = s$.

For the second rule, fix a trust set S. Whenever an IPSEC-enabled device in S processes an IPSEC packet p with $\mathsf{src}(p) \notin S$, and removing this header leads to a packet p' with $\mathsf{src}(p') \in S$, p' must be discarded. It codifies the idea that only nodes in S should be trusted to certify a packet as coming from S.

Pop Rule. For any transition

$$\begin{array}{c}(\ell, \kappa, \langle [s, d, A], [a, -, -] \rangle) \\ \downarrow \\ (\ell, \kappa', \langle [a, -, -] \rangle)\end{array}$$

If $\ell \in S$ then $s \in S$.

Authentication Boundary Constraints Given the authentication goal above, boundary systems must only abide by one extra processing constraint: they must not pass an inbound packet that did not present any authentication headers.

Inbound Ready Rule.

$$\begin{array}{c}(\ell, \kappa, \theta) \\ \downarrow \\ (\ell, \mathsf{ready}, \langle [a, -, -] \rangle)\end{array}$$

If $\kappa \neq \mathsf{ready}$ and $\ell \in \partial^{in} S$, then $\theta = \langle [s, d, A], [a, -, -] \rangle$.

Unwinding We prove that the processing restrictions formulated above are sufficient to ensure the authentication goal. To do so, we exhibit an *unwinding set* G.

Definition 4. An *unwinding set* G is a set such that

1. $\mathsf{start} \notin G$,
2. $\mathsf{accept} \subseteq G$,
3. $\mathsf{authentic} \subseteq G$,
4. For any transition $x \to y$ with $x \notin G$ and $y \in G$ then $y \in \mathsf{authentic}$.

Proposition 1. A sufficient condition for the authentication condition to hold is the existence of an unwinding set.

PROOF. Any path $\mathsf{start} \longrightarrow^* \omega$ with $\omega \in \mathsf{accept}$, must have the form

$$\mathsf{start} \longrightarrow^* x \to y \longrightarrow^* \omega$$

with $x \notin G, y \in G$. By the unwinding condition 4, $y \in \mathsf{authentic}$. ■

We now exhibit an unwinding set G.

$$G = \mathsf{accept} \cup \mathsf{authentic} \cup \mathsf{continue}$$

where $\mathsf{continue}$ is defined:

Definition 5. (Continuing States) A state is a continuing state if it belongs to one of the three disjoint classes below:

C1 $(\ell, \mathsf{ready}, \langle[a, -, -]\rangle)$ for $\ell \in \partial^{in} S$;
C2 $(\ell, \kappa, \langle[a, -, -]\rangle)$ for $\ell \in S \setminus \partial^{in} S$,
i.e. for locations in the portion of S other than the inbound boundary;
C3 $(\ell, \kappa, \langle \cdots [s, d, A], [a, -, -]\rangle)$ for $s \in S$ and any ℓ.

Proposition 2. G is an unwinding set.

PROOF. Suppose $x \to y$ with $x \notin G, y \in G$. The proof is a completely mechanical enumeration of cases. In each case, we show either that either it cannot really occur or that $y \in \mathsf{authentic}$.

Case I: $y \in \mathsf{accept}$. By definition of accept, y is of the form $(b, \mathsf{ready}, \langle[a, -, -]\rangle)$.

1. $b \in \partial^{in} S$.
   (a) $x \to y$ is a motion. The **inbound motion restriction** excludes this case.
   (b) $x \to y$ is non-augmenting. By the **inbound ready rule**, x is of the form
   $$(b, \kappa, \langle \ldots, [s, d, A], [a, -, -]\rangle)$$
   with $s \in S$. This implies $x \in \mathbf{C3} \subseteq \mathsf{continue} \subseteq G$.
   (c) $x = \mathsf{start}$. In this case, by the **creation rule** $b = a$. Thus $y \in \mathsf{authentic}$.
2. $b \in S \setminus \partial^{in} S$.
   (a) If $x \to y$ is a motion, then x must be of the form $(\ell, \mathsf{ready}, \langle[a, -, -]\rangle)$ By definition of network boundary of S, $\ell \in S$. This implies $x \in \mathbf{C1} \cup \mathbf{C2} \subseteq G$. This case is thus excluded.
   (b) Otherwise x must be of one the forms
      i. $(b, \kappa, \langle[s, d, A], [a, -, -]\rangle)$ with $s \in S$, so $x \in \mathbf{C2} \subseteq G$, which excludes this case also.
      ii. start. In this case, by the **creation rule** y is of the form $(b, \kappa, \langle[s, -, -]\rangle)$ with $b = s = a$. Thus $y \in \mathsf{authentic}$.

Case II: $y \in \mathbf{C1}$. Thus $y = (\ell, \mathsf{ready}, \langle[a, -, -]\rangle)$ for $\ell \in \partial^{in} S$.

1. $x = (\ell', \kappa, \theta)$. The **inbound motion rule** excludes this case.
2. $x = (\ell, \kappa, \theta)$. By the **inbound ready rule**, the transition $x \to y$ must be a pop. In this case, by the **pop rule** x must be of the form $(\ell, \kappa', \langle[s, d, A], [a, -, -]\rangle)$ for $s \in S$, so $x \in \mathbf{C3} \subseteq G$, which excludes this case also.
3. $x = \mathsf{start}$. In this case, the **creation rule** implies $\ell = a$, so $y \in \mathsf{authentic}$.

Case III: $y \in \mathbf{C2}$. In this case y is of the form $(\ell, \kappa, \langle[a, -, -]\rangle)$ for $\ell \in S \setminus \partial^{in} S$.

1. $x = (\ell', \kappa', \theta)$ with $\ell' \neq \ell$. In this case, the transition $x \to y$ must be a location change. By definition of border, $\ell' \in S$ and by the motion ready restriction, $\kappa' = \mathsf{ready}$. In this case $x \in \mathbf{C1}$ or $x \in \mathbf{C2}$ depending on wehether $\ell' \in \partial^{in} S$ or $\ell' \in S \setminus \partial^{in} S$. Thus this case is excluded.
2. $x = (\ell, \kappa', \theta)$. In this case, the transition $x \to y$ must be a pop. By the **pop rule** x must be of the form $(\ell, \kappa', \langle[s, d, A], [a, -, -]\rangle)$ for $s \in S$, so $x \in \mathbf{C3} \subseteq G$, which excludes this case also.
3. $x = \mathsf{start}$. In this case, the **creation rule** implies $\ell = a$, so $y \in \mathsf{authentic}$.

Case IV: $y \in \mathbf{C3}$. y is of the form $(\ell, \kappa, \langle \cdots [s, d, A], [a, -, -] \rangle)$ for $s \in S$.

1. If $x \to y$ is a motion, then $x \in \mathbf{C3}$
2. If $x \to y$ is a non-augmenting header transition, then x must also be of the form $\mathbf{C3}$.
3. If $x \to y$ is a push, then either $x \in \mathbf{C3}$ or x is of the form $(\ell, \kappa', \langle [a, -, -] \rangle)$. By cryptographic restriction, $\ell = s \in S$. In this case $x \in \mathbf{C1}$ or $x \in \mathbf{C2}$ depending on wehether $\ell \in \partial^{in} S$ or $\ell \in S \setminus \partial^{in} S$. Thus this case is excluded. ■

4.2 Confidentiality

We will consider the following confidentiality problem: Suppose a and b are network nodes. What conditions can we impose on the enclave nodes' processing to ensure that packets travelling from a to b are encrypted whenever they are not in the trust set S? More formally,given some set of processing restrictions,

> If we start with a packet of the form $(a, \mathsf{ready}, \langle [a, b, p] \rangle)$, where $a, b \in S$, then it will never be the case that $(\ell, \kappa, \langle [a, b, p] \rangle)$ if $\ell \notin S$.

Achieving confidentiality is even simpler than authentication. There are two simple constraints, one on all devices in the trust set, and an additional constraint for boundary members.

Confidentiality Tunnel Constraints Fix a trust set S. The constraint on all trust set members requires them not to "tunnel" packets requiring protection out to a dangerous area. Our constraint will ensure that whenever a system inside S adds a confidentiality header to a packet which would require protection, the source and destination of the added header are also in S.

Destination Prefix Rule. For any transition

$$\begin{array}{c}(\ell, \kappa, \langle [s_1, d_1, p_1] \cdots [a, b, p] \rangle) \\ \downarrow \\ (\ell, \kappa', \langle [s_2, d_2, p_2][s_1, d_1, p_1] \cdots [a, b, p] \rangle)\end{array}$$

if $\ell \in S$, $s_1, d_1 \in S$, and $p_1 \notin C$, then $s_2, d_2 \in S$. We include the case where $\langle [s_1, d_1, p_1] \cdots [a, b, p] \rangle = \langle [a, b, p] \rangle$.

Confidentiality Boundary Constraints As with authentication, we impose one constraint on boundary members. If a packet p is traversing an outbound interface on the boundary of S, and p could contain a packet $p_0 \in P$ with no confidentiality header, discard p.

One way to safely implement this is to pass a packet p only if its topmost layer is a confidentiality header, or else it has no IPsec headers and $p \notin P$.

Outbound Ready Rule. For any transition

$$\begin{array}{c}(\ell, \kappa, \theta)\\ \downarrow \\ (\ell, \mathsf{ready}, \theta')\end{array}$$

if $\ell \in \partial^{out} S$, then either $\theta' = \langle [s, d, C], \ldots [a, b, p] \rangle$ for $s, d \in S$, or else $\theta' = \langle [s', d', -], \ldots \rangle$ where either s' or d' not in S.

Invariant. We will prove that the processing restrictions formulated above are sufficient to ensure the confidentiality goal using an invariant of our state machine. We will first show that the invariant holds, then prove that given the invariant, our confidentiality goal holds as well.

Proposition 3. Suppose that Σ is a state machine satisfying the outbound ready rule and the destination prefix rule, and suppose that (ℓ, κ, θ) is the state resulting from a sequence of actions beginning with $\mathsf{create}_{a,[a,b,p]}$, where $a, b \in S$.

1. If $\ell \in S$, then either
   (a) whenever $[s_1, d_1, p_1]$ is any layer of θ, then $s_1, d_1 \in S$ and $p_1 \notin C$, or
   (b) there is a final segment of θ of the form $\langle [s_k, d_k, C] \cdots [s_i, d_i, p_i] \cdots \rangle$ where $s_k, d_k \in S$ and for each $i < k$, $s_i, d_i \in S$ and $p_i \notin C$.
2. If $\ell \notin S$, then there is a final segment of θ of the form $\langle [s_k, d_k, C] \cdots [s_i, d_i, p_i] \cdots \rangle$ where $s_k, d_k \in S$ and for each $i < k$, $s_i, d_i \in S$ and $p_i \notin C$.

Proof. We will examine each of the possible state transitions in turn, showing for each that they cannot violate the invariant.

Case 1: create and $\mathsf{discard}$ In the case of the create operator, we know that the first transition in our state machine is the following (which does not violate the invariant):

$$\begin{array}{c}\mathsf{start}\\ \downarrow \\ (a, \mathsf{ready}, \langle [a, b, p] \rangle)\end{array}$$

The invariant imposes no constraints on the finish state, thus the $\mathsf{discard}$ transition is irrelevant.

Case 2: pop Assume that we are at location ℓ. The state transition we are interested in is $\mathsf{pop}_\kappa(\ell, \kappa', h \cdot \theta) = (\ell, \kappa, \theta)$. Our cryptographic assumptions prevent any location from removing an encryption layer not destined for them. Thus, no location can remove the necessary confidentiality protection (provided it was applied), and the invariant is not violated.

Case 3: prefix Assume once again we are at location ℓ. The transition is $\mathsf{prefix}_{h,\kappa}(\ell, \kappa', \theta) = (\ell, \kappa, h \cdot \theta)$. The only case which has bearing on the invariant is that where $\ell \in S$, and there is no encryption layer in θ. By the Destination Prefix rule, $src(h), dst(h) \in S$ as well. If h is a confidentiality header, the packet now satisfies the second invariant condition for locations in S. If h is not a confidentiality header, the packet satisfies the first invariant condition for locations in S.

Case 4: null The invariant imposes no constraints on κ.

Case 5: move Again, assume we are at location ℓ. The transition is

$$\mathsf{move}_{e,\kappa}(\ell, \mathsf{ready}, \theta) = (\ell', \kappa, \theta)$$

Since this involves no change of state, the only case which could violate the invariant is that where $\ell \in \partial^o S$ and $\ell' \notin S$. The Outbound Ready rule ensures that the top layer of θ is either $[s, d, C]$ with $s, d \in S$ or $[s', d', -]$ with $s', d' \notin S$. The Destination Prefix rule ensures that below the bottom-most confidentiality layer, all layers have source and destination in S. So, regardless of which portion of the Outbound Ready rule is appropriate, the invariant is not violated.

Thus, the given invariant holds for our state machine. We now must show it implies enforecement of the confidentiality goal.

The confidentiality goal is ensured if it is never the case that $(\ell, \kappa, \langle [a, b, p] \rangle)$ if $\ell \notin S$. Condition 2 of the invariant provides this: suppose that we're at $\ell \notin S$. Then there is at least one layer $[s, d, C]$ with $s, d \in S$, and no layers with external sources 'beneath' that layer.

5 What Can Go Wrong

We have seen that the restrictions of Sections 4.1 and 4.2 suffice to ensure that authentication and confidentiality goals are achieved. In this section, we argue less formally that they are necessary. We illustrate the problems that can otherwise arise, by presenting example attacks, focusing first on authentication. In all examples, we use the example network described in Figure 1.

5.1 Failures of Authentication

Corresponding to our three authentication constraints, there are three ways that IPsec (incorrectly configured) may fail to achieve authentication goals.

Spoofing near source or destination The first of these problems arise if locations within a trust set spoof packet addresses.

Consider the following security goal: packets traveling from *EngineeringA* to *EngineeringB* should be authenticated. If, in this case, one host in *EngineeringA* creates a packet claiming to come from another, the security gateway applying the cryptographic headers (either *SG1* or *SG2*, here) would be unable to determine that the packet was not legitimate. A similar situation could arise if a host in *EngineeringB* created packets with source header field of an *EngineeringA* system.

Tunneling packets to a point near source or destination The second attack becomes possible if machines inside the source region do not ensure that removed layers correspond to an acceptable authentication tunnel.

Suppose again that the security goal is to authenticate packets traveling from *EngineeringA* to *EngineeringB*. Further suppose that the device which performs outbound cryptographic processing for Company A is *SG1*.

Consider, then, the following case: a machine in the region *Internet* creates a packet with an forged source of a host in *EngineeringA*, and a destination in *EngineeringB*, and then adds a tunnel mode authentication header destined for the host *dbA*, which performs IPsec processing.

The packet goes via *SG2*, which routes it through *PerimeterA*, and past *SG1*. When it reaches the destination *dbA*, that host removes the tunnel-mode AH header and successfully checks the cryptographic checksum against the payload and tunneled header. The result of removing the tunnel-mode header is a non-IPsec packet with source header field in *EngineeringA* and destination header field in *EngineeringB*. The normal routing mechanism causes it to pass back to *SG1*, which adds the "appropriate" authentication header. As a consequence it reaches Company B with a cryptographic header purporting to guarantee that it originated in *EngineeringA*.

The same effect is obtained if the machine in the region *Internet* creates the same packet initially, and then adds a tunnel mode authentication header destined for the host *host3*. Clearly, the IPsec-enabled devices *dbA* and *host3* must be configured to compare the source address on the IPsec header, namely the *Internet* machine, against the source address on the inner, non-IPsec header (which claims to be in *EngineeringA*). The **Pop Rule** ensures that they will discard the forged packet when it emerges from the authentication tunnel with source outside the trust set S.

Entry of packets near source or destination The third problem arises if systems on the *boundary* of the trust set S allow incoming packets to enter, even if the packets claim to have originated inside.

Again consider the security goal that all traffic between *EngineeringA* and *EngineeringB* must be authenticated. Suppose further that *SG1* and *SG4* are the endpoints of the authentication tunnel. Then, if *SG1* does not perform inbound filtering on packets arriving from *Perimeter A* to check that they do not have source header field from any system in *EngineeringA*, a machine in the region *Internet* could create a packet with source in *EngineeringA*, use source routing (or another routing attack) to send it into *EngineeringA*, where it would then be sent to *EngineeringB*.

5.2 Failures of Confidentiality

Again corresponding to our behavior constraints, there are two types of attacks that IPsec, configured incorrectly, cannot protect against. Assume that the corporations Company A and B wish to protect the confidentiality of traffic flowing from *EngineeringA* to *EngineeringB*, by ensuring that it is always encrypted while in the public Internet.

Tunneling to a point distant from source and destination The tunnel endpoints are *SG2* and *SG3* in this example. If *SG4* is misconfigured, it may insert the packets decrypted by *SG3* into another tunnel, intended for a different sort of traffic. The exit from that tunnel may be on the public Internet, say in Company C, with which Company B has some commercial relation, but which is a competitor of Company A. This would be a failure of confidentiality. Company C might even have been surprisingly helpful when the system administrators in Company B designed their IPsec configuration.

Our *Destination Prefix Rule* (Section 4.2) prevents this sort of occurrence.

Escape to a point distant from source and destination In this example, the tunnel endpoints are *SG1* and *SG4*. If *SG4* or *SG3* is not set up to prevent back-flow of packets with source header field in *EngineeringA* and destination header field in *EngineeringB*, then these packets could be re-routed out past *SG4* and *SG3* after having traversed *SG4*, the point at which the ESP header was removed. The risk that this may be a reasonable routing strategy (or a feasible attack on routing in *Engineering B*) increases if *Engineering B* consists of a large collection of networks, and if it is more richly connected to the outside world than in our sample diagram.

Hence, systems on the boundary of the trust set must inspect packets before passing them on; this sort of attack is not possible if the *Outbound Ready Rule* (Section 4.2) is in effect.

6 Conclusion

In this paper, we formalized the main security goals IPsec is suited to achieve, and we formalized IPsec-relevant aspects of networks. We then provided criteria entailing that a network with particular IPsec processing achieves its security goals. Achieving these security goals requires identification of a *trust set* for each goal.

Our approach has several benefits.

- It is rigorous: provided the behavior restrictions are enforced, the security goals are formally guaranteed.
- It explains clearly exactly which systems must be trusted, and in exactly what ways.

– Its security management discipline can largely be enforced by software that checks the configurations of nodes within the trust set.

Future work will include the development of such a software tool. In addition, it seems likely that other related security protocol sets (for example, PPTP) could be analyzed in the same way; additional security goal types—such as traffic flow confidentiality or firewall-like restrictions on types of traffic, in the manner of [3]—could also be added.

References

1. Steven Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth USENIX UNIX Security Symposium*, July 1996. Also at ftp://ftp.research.att.com/dist/smb/badesp.ps.
2. Niels Ferguson and Bruce Schneier. A cryptographic evaluation of ipsec. Counterpane Internet Security, Inc., available at http://www.counterpane.com/ipsec.html, 1999.
3. Joshua D. Guttman. Filtering postures: Local enforcement for global policies. In *Proceedings, 1997 IEEE Symposium on Security and Privacy*, pages 120–29. IEEE Computer Society Press, May 1997.
4. D. Harkins and D. Carrel. *The Internet Key Exchange (IKE).* IETF Network Working Group RFC 2409, November 1998.
5. S. Kent and R. Atkinson. *IP Authentication Header*. IETF Network Working Group RFC 2402, November 1998.
6. S. Kent and R. Atkinson. *IP Encapsulating Security Payload*. IETF Network Working Group RFC 2406, November 1998.
7. S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. IETF Network Working Group RFC 2401, November 1998.
8. D. Maughan, M. Schertler, M. Schneider, and J. Turner. *Internet Security Association and Key Management Protocol (ISAKMP).* IETF Network Working Group RFC 2408, November 1998.
9. Steve Schneider. Security properties and CSP. In *Proceedings, 1996 IEEE Symposium on Security and Privacy*, pages 174–87. IEEE Computer Society Press, May 1996.