via visa-routers. Each visa-router knows the ACSs in its AD, is willing to accept visas issued by some or all of these ACSs, and trusts their decisions about authorizing and terminating connections.

A visa-router allows any external party to communicate with any registered, internal ACS. Similarly, visa-routers allow all registered, local ACSs to communicate with any external party. Such trust is reasonable because ACSs are assumed to implement sufficient defense mechanisms and to enforce AD's policy.

However, this does not imply that visa-routers should let the ACS-originated outbound traffic go unchecked. In order to detect *bogus* ACS packets and prevent replay of pre-recorded ACS-sourced packets, a visa-router must verify packet signatures and validate packet timestamps of all packets purportedly originating at a local ACS.

It is more difficult to control traffic coming in from the outside destined for a local ACS. Because such traffic can originate anywhere in the internetwork its validation would require a visa-router to have means for verifying signatures and timestamps generated by a possibly large number of sources. This would necessitate a lot of state information in visa-routers which is clearly undesirable and impractical for reasons of performance. The alternative is not to scrutinize in-bound ACS traffic and let the ACSs filter out fraudulent packets. Visa-routers can still maintain some control by making sure that local ACSs do not get *flooded* by in-bound traffic. The disadvantage of this approach is that it may violate one of our fundamental goals of not allowing *unauthorized* externally-sourced traffic consume internal network resources. This subject is addressed further in Section 3.7.

Assuming that each AD employs visa-routers, each inter-AD packet travels through at least two such routers. A visa-router must scrutinize every packet it receives; packets without visas cannot be forwarded (except for those to or from trusted entities of the router's own AD). In section 3.6.1 we describe a mechanism for a visa-router to inform an end-system that a visa is required for inter-AD communication.

Packets originating at or destined for unequipped end-systems are discarded by the visa-routers since these end-systems are (by definition) not allowed any external access.

If the two stub ADs are not directly connected, packets will pass through the routers of transit networks. Visa-routers within a single transit AD are assumed to trust each other, and transfer transit packets via secure channels to prevent unauthorized entrance or exit. Non-visa routers in transit ADs treat visa-stamped packets as regular internet packets.

3.5.3 Participating End-systems

An end-system attempting communication outside of its AD must be able to obtain a visa allowing it exit from the local AD and entry to the destination AD. In order to obtain exit authorization it needs to contact one of the local ACSs which (upon authenticating the requesting end-system and checking its access control lists) then contacts an ACS in the destination AD and requests entry authorization on behalf of the original end-system. After establishing entry authorization, the remote ACS issues a visa which is subsequently delivered to the requesting end-system. Thereafter, a visa-stamp (computed with the corresponding visa-key) must be attached to every packet sent from the requesting end-system to the apparent destination.

An end-system, unlike a visa-router, does not have to have reliable knowledge of the local ACS's address; this may instead be supplied by a visa-router when an end-system first attempts to communicate across the AD boundary (see section 3.6.1). An end-system may still need to use an authentication protocol (e.g., Kerberos [85]) to make sure it is really talking to the ACS.

Since packet reception is a passive operation, the destination end-system is not required to initiate any actions. Of course, in most types of communication, packets flow in both directions, so each end-system is both a source and a destination. Therefore, to avoid additional overhead we assume that an AD may allow its ACS to issue *two-way* visas automatically if no authentication of the remote destination is required.

In summary, the requirements for a participating end-system are as follows:

- an authentication mechanism to allow for a dialog with a local ACS
- secure storage for active visas
- a means for generating visa-stamps

Lastly, a participating end-system must be identifiable, i.e., it must be assigned a unique key or key-pair (depending upon the encryption method).

It should be noted that *Visa* protocol, by itself, does not provide for multi-level security, nor does it eliminate a variety of covert channels. In the absence of additional non-discretionary controls, a participating end-system may still compromise access controls by serving as a conduit for communications between unauthorized end-systems.[2]

3.6 Protocol

Visa protocol consists of three phases. In the setup phase, an end-system obtains authorization for exiting its own AD and entering the destination AD. If successful, it culminates with the issuance and distribution of visas to all principals involved. In the packet forwarding phase, the visa-key is used to generate packet data signatures that are attached to all packets belonging to an authorized connection. Finally, the teardown phase involves the termination of a visa either because of normal expiration or by explicit revocation. In the remainder of this section, each protocol phase is discussed separately.

3.6.1 Setup Phase

The purpose of the setup phase is to i) authorize communication between two end-systems by the ACSs in the respective ADs, ii) issue a visa which embodies this authorization, and, iii) distribute it to all parties involved. The placement of the protocol participants is illustrated in Figure 3.1.

3.6.1.1 Exit Authorization

The protocol is put in motion when an end-system, H_a, in AD_a begins communication with another end-system, H_b, in a different AD, AD_b. H_a may already know that its intended destination is in a different AD, either because it has previously communicated with H_b or it may have discovered this through some external mechanism (e.g., a name server). If so, H_a may communicate directly with an ACS in its AD, ACS_a. Otherwise, it may discover that H_b is in a different AD when its packet reaches the exit visa-router. Since the packet carries no visa-stamp, the exit visa-router replies with a REDIRECT packet. REDIRECT is essentially a means of notifying H_a that the intended destination is non-local, and that it must *"apply"* for a visa with a local ACS.

$$REDIRECT\ =\ [\ H_a, H_b, ACS_a\ ] \tag{3.1}$$

The protocol begins by H_a requesting authorization from ACS_a. It does so by sending a HOST-REQUEST packet to ACS_a.

$$HOST-REQUEST\ =\ [\ H_a, H_b, TS_{H_a}\ ]^{K_{H_a}} \tag{3.2}$$

To maintain data integrity, HOST-REQUEST is signed with H_a's key, K_{H_a}. If conventional encryption is used, K_{H_a} is a key known only to H_a and ACS_a. With public key encryption, K_{H_a} is the private (secret) key known only to H_a. A timestamp, TS_{H_a}, is included to demonstrate the timeliness of the packet, i.e., to indicate its *freshness* to ACS_a as described in Section 2.2.3. (Fresheness is, of course, dependent upon the value of Δ_T.)

Next, ACS_a has to authorize communication between H_a and H_b. This step is dependent on the particular policy employed by AD_a. For example, it may involve a higher-level authentication dialog between AD_a and H_a. The details of this procedure are beyond the scope of this discussion.

If and when exit authorization is established, ACS_a composes a VISA-REQUEST packet:

$$VISA-REQUEST\ =\ [\ H_a, H_b, TS_a\ ]^{DK_{ACS_a}} \tag{3.3}$$

The packet is signed with ACS_a's secret key and timestamped to help verify both data integrity and timeliness. The timestamp, TS_a, is guaranteed to be unique; thereafter, it is used as a visa identifier. As with HOST-REQUEST, the signature is not needed if a higher-level authentication dialog is used between ACS_a and its counterpart in AD_b. Nevertheless, our purpose is to allow authentication to take place at the earliest possible time rather than relying on the presence of a higher-layer AD-dependent mechanism.

In order to deliver a VISA-REQUEST, ACS_a has to locate its counterpart in the destination AD, AD_b. It may know the address of ACS_b because of previous communication in which case VISA-REQUEST may be sent directly. It

[2] See section 3.7.

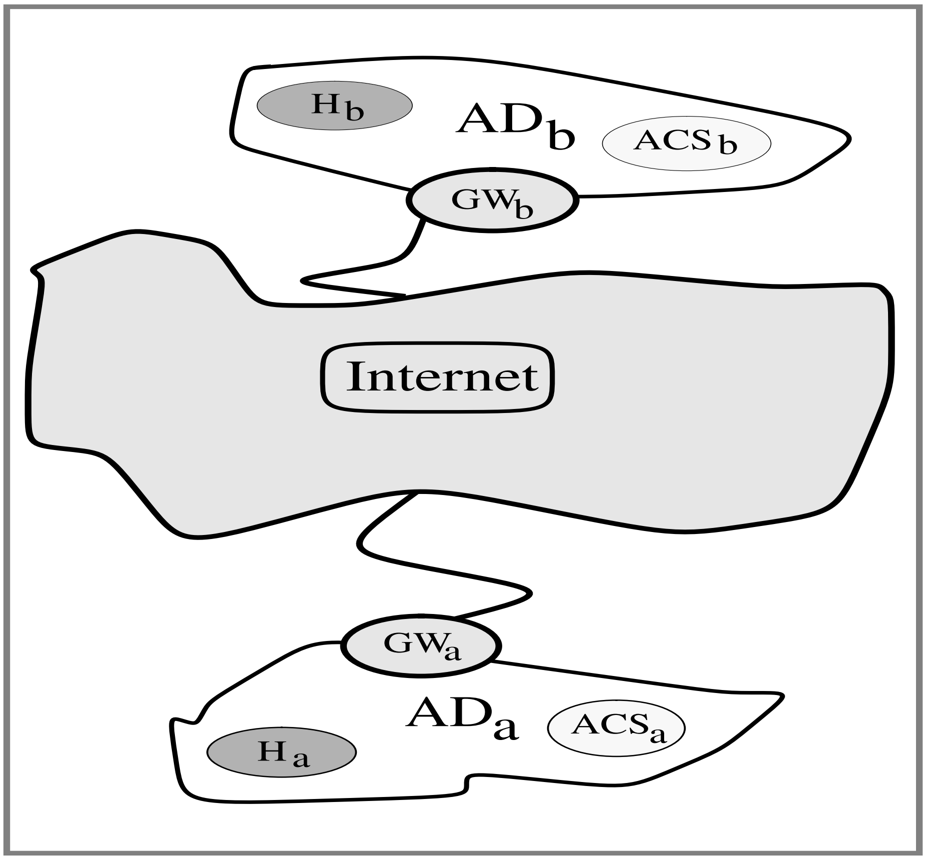

Figure 3.1: Two ADs participating in *Visa* protocol

could also obtain ACS_b's address via a name service query [60]. Alternatively, VISA-REQUEST may be sent addressed to H_b (the destination end-system). This is possible because visa-routers are required to reroute *all* VISA-REQUEST packets to one of the local ACSs.

Furthermore, in case of no previous communication with AD_b, ACS_a may choose to speed up the setup process a little by including its certificate, $CERT_{ACS_a}$ with the VISA-REQUEST packet. This would save ACS_b the need to request this certificate explicitly (from either ACS_a or some name server).

3.6.1.2 Entry Authorization

When a VISA-REQUEST reaches AD_b, the intervening visa-router, GW_b, forwards it to one of the local ACSs, ACS_b.

First, ACS_b verifies that H_b indicated in the VISA-REQUEST is in fact an end-system in AD_b. This is necessary in order to minimize time spent on potentially malformed VISA-REQUESTs. Next, before proceeding to authorize the connection ACS_b has to validate the VISA-REQUEST. In order to authenticate its contents (i.e., re-compute the signature), ACS_b needs the public key of ACS_a, EK_{ACS_a}. To obtain this key ACS_b has to know the AD identifier of ACS_a (i.e., AD_a), which is not included in the VISA-REQUEST. However, ACS_b can query its name service with H_a (which is part of VISA-REQUEST) and obtain AD_b, ACS_b and EK_{ACS_a}.

With the help of EK_{ACS_a} ACS_b can re-compute the signature of the VISA-REQUEST and verify both its origin and data integrity. Timeliness an uniqueness of the VISA-REQUEST is inferred from the enclosed timestamp, TS_a (as described in Section 2.2.3).

Next, ACS_b authorizes communication between H_a and H_b. AS before, this step is dependent on AD_b's policy. At last, when communication is authorized, ACS_b issues a fresh visa in a form of a VISA packet.

$$VISA - GRANT = [H_a, H_b, auth - type, TS_a, Expiration, Conditions, (S_b^a)^{EK_{ACS_a}}]^{DK_{ACS_b}} \tag{3.4}$$

Since the structure of a *VISA-GRANT* packet is crucial to the security of the protocol, we now consider the individual packet fields in more detail:

- H_a, H_b are the two end-systems that are authorized to use this visa. In order to associate packets with a particular connection, visa-routers GW_a and GW_b, need these end-system addresses to locate the appropriate visa-key.
- *auth-type* denotes the signature method to be used for packet signature computation.
- S_b^a is the so-called *visa-key*. It is subsequently used to compute packet signatures for traffic flowing between H_a and H_b. Depending on the signature method, S_b^a may be an encryption key (as in DES-based MAC) or a secret prefix/suffix for use in conjunction with MD4. (See Appendix A). Because a visa-key has to be kept secret to prevent interception by potential intruders, it is encrypted with ACS_a public key.
- TS_a is the timestamp assigned to the visa by ACS_a and *approved* by ACS_b. It is used primarily as a *nonce* [67], i.e., a unique, hereto unused, visa identifier.
- Expiration indicates the condition(s) for the termination of a visa. May be expressed as any combination of:

  maximum lifetime of a visa (e.g., in msec)

  maximum inactivity period

  maximum number of packets

  maximum amount of data transferred (e.g., in Kbytes)
- The Conditions field can be used to express the visa usage conditions, e.g.:

  Type of Service

  User Class

  Higher-level protocol, etc.

3.6.1.3 Visa Distribution

After entry authorization is obtained and the new visa is issued, ACS_b forwards the VISA-GRANT packet to the requesting ACS_a. A VISA-GRANT also has to be sent to GW_b so it can process entering packets accordingly. However, in the original VISA-GRANT packet, S_b^a is encrypted with EK_{ACS_a} to maintain its secrecy. Since DK_{ACS_a} is not known to GW_b, ACS_b has to create a different copy of a VISA-GRANT for delivery to GW_b where S_b^a is encrypted with EK_{ACS_b}. Upon receipt of the VISA-GRANT, GW_b creates a new entry in its visa-table.

When ACS_a receives a VISA-GRANT packet, it has to check several conditions:

- First, ACS_a verifies that H_a, H_b and TS_a are indeed the same values that were used in the VISA-REQUEST packet.
- Next, the integrity of the VISA-GRANT packet has to be verified. ACS_a can do so by re-computing the packet signature with EK_{ACS_b}.[3]
- Finally, the authentication type and the expiration conditions are checked for recognizability and consistency.

When the VISA-GRANT packet is validated, ACS_a notifies its visa-router, GW_a via a VISA-GRANT packet, where S_b^a is encrypted with EK_{ACS_a} and the entire packet is signed with DK_{ACS_a} instead of DK_{ACS_b}.[4] On receipt of the VISA-GRANT, GW_a installs the new entry in its visa-table and becomes ready to pass packets between H_a and H_b. The VISA-GRANT packet sent to H_a is similar save for S_b^a which is encrypted with K_{H_a}.

[3] Freshness and uniqueness of the VISA-GRANT packet is evident from the presence in it of TS_a.

[4] The assumption is that GW_a and ACS_a share the same public/private key pair.

3.6.1.4 Setup Summary

In summary, the setup phase involves the following steps:[5]

1. $H_a \Longrightarrow ACS_a : HOST - REQUEST$
2. $ACS_a \Longrightarrow ACS_b : VISA - REQUEST$
3. $ACS_b \Longrightarrow GW_b : VISA - GRANT^{GW_b}$
4. $ACS_b \Longrightarrow ACS_a : VISA - GRANT^{ACS_a}$
5. $ACS_a \Longrightarrow GW_a : VISA - GRANT^{GW_a}$
6. $ACS_a \Longrightarrow H_a : VISA - GRANT^{H_a}$

In the description above, $VISA - GRANT^P$ denotes the *Visa* packet where S_b^a is encrypted for the principal P (e.g., GW_a) with a key which is either P's public key, or a key shared among the sender and P. All other fields are the same for all $VISA - GRANT^P$-s.

Of all the messages exchanged during the setup phase, those in steps (2) and (4) are of particular importance since they cross AD boundaries and are, therefore, subject to much greater exposure.

3.6.2 Packet Forwarding

When the setup phase is completed, H_a can begin communication. Each packet that H_a sends to H_b has to be accompanied by a visa-stamp. However, it is not enough to simply sign the packet. The reason for this is twofold. First, the protocol may need to support multiple visas for the same end-system pair. Therefore, a visa-router can not uniquely identify an entry in its table using only the two end-system addresses. For this reason, TS_a and TS_b are included with each packet. Second, since one of the protocol goals is the detection of *stale*, (or potentially replayed) packets, H_a needs to attach a timestamp to each packet. In this context, a regular (non-visa) packet is composed of a data segment and a network-layer header.

A visa-stamp is computed as:

$$VISA - STAMP = F([HEADER, DATA, TS_{H_a}], S_b^a) \tag{3.5}$$

where F is the signature function determined by the *auth-type* agreed to during the setup phase and TS_{H_a} is the timestamp assigned to the packet by H_a. TS_{H_a} must be unique, i.e., no two packets should carry the same timestamp (for a given connection).

The resulting data packet has the following form (see also Figure 3.2):

$$DATA - PACKET = [HEADER, DATA, VISA - STAMP, TS_a] \tag{3.6}$$

3.6.2.1 Exiting AD_a

As described in Section 3.3, when a packet with an attached visa-stamp arrives at GW_a it has to demonstrate authorization to leave AD_a as well as authenticity and freshness of its contents.

GW_a checks the first condition by indexing its visa-table with the $[TS_a, H_a, H_b]$ tuple. Successful look-up indicates the existence of exit authorization by ACS_a. Next, the freshness of the packet is verified by comparing the packet timestamp, TS_{H_a} with the stored timestamp of the last (previous) data packet that used the same visa (this value is referred to as $T_{last}^{(a,b)}$). Finally, GW_a re-computes the packet signature and compares it to the visa-stamp attached to the packet. If the two values match, GW_a can safely conclude that the packet contents are authentic.

The order in which these checks are performed is not arbitrary. In particular, the reason for verifying packet freshness before validating a packet signature has to do with the cost of the latter operation. Naturally, for *authentic* packets,

[5] The notation $A \Longrightarrow B : PACKET$ means "*A sends PACKET to B* ".

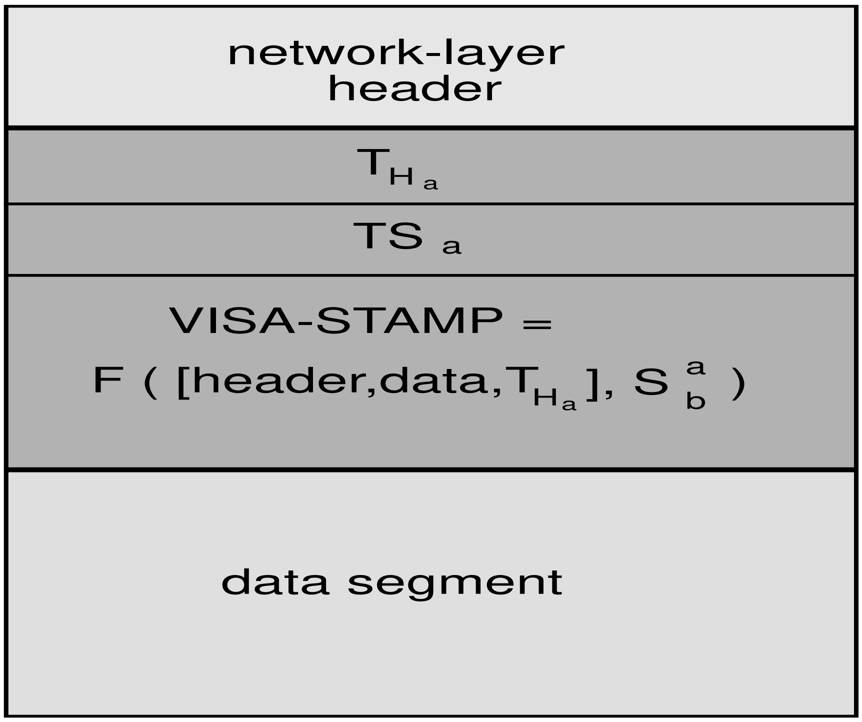

Figure 3.2: Data packet format

the order of these two operations is immaterial. However, because we would like to detect replayed (i.e., *stale*) packets at the earliest possible time, and because it is significantly cheaper to compare two timestamps than to re-compute a packet signature, the freshness test is performed first.

3.6.2.2 Entering AD_b

The goals of GW_b with respect to entering packet traffic are similar. Like GW_a, GW_b checks authorization by indexing its visa-table with the $[TS_b, H_a, H_b]$, verifies packet freshness by comparing the packet timestamp with the its $T_{last}^{(a,b)}$, and re-computes the signature to verify the authenticity of the packet contents.

3.6.3 Teardown

As mentioned previously, a visa may be terminated for one of two reasons:

- Normal expiration (time, packet or data ceiling reached)
- Explicit revocation (by explicit order of an ACS)

In the first case, a visa-router simply deletes the corresponding entry from its visa-table. If and when a packet bearing a visa-stamp computed using an expired visa arrives at the visa-router, it is promptly discarded since the table look-up fails.

In case of explicit revocation, an ACS may decide for some reason that a certain visa is no longer trusted and sends a REVOKE packet to the appropriate visa-router and a peer ACS:

$$REVOKE = [H_a, H_b, TS_a, TS_b, Reason]^{DK_{ACS}} \tag{3.7}$$

TS_a and TS_b in the REVOKE are the same visa identifiers assigned at the setup time. Since a visa can only be revoked once, there is no need to timestamp a REVOKE packet as its replay presents no danger.

3.7 Design Issues

In this section, we discuss several issues leading to the design of *Visa* protocol presented in this chapter. We also address several important features where the current protocol differs from its predecessors.

3.7.1 Visas

In the previous *Visa* protocol versions, each end-point AD issued a visa for authorized communication. Moreover, each visa had an associated visa-key, thereby necessitating the computation and subsequent verification of two distinct packet signatures. In other words, the end-system computed two packet signatures, $DSIG_{exit}$ and $DSIG_{entr}$. The former was verified by the exit visa-router in the source AD, and the latter, by the entry visa-router in the destination AD.

The setup phase of the protocol consisted of the following steps:

1. $H_a \Longrightarrow ACS_a : HOST - REQUEST$
2. $ACS_a \Longrightarrow ACS_b : VISA - REQUEST$
3. $ACS_b \Longrightarrow GW_b : VISA - GRANT_{entr}$
4. $ACS_b \Longrightarrow ACS_a : VISA - GRANT_{entr}$
5. $ACS_a \Longrightarrow GW_a : VISA - GRANT_{exit}$
6. $ACS_a \Longrightarrow H_a : VISA - GRANT_{entr}, VISA - GRANT_{exit}$

The primary reason for each ACS issuing its own visa (one entry and one exit) was the assumed lack of trust between the two ACSs as far as the issuance of *good* keys or visas. On the other hand, ACS_a still *trusts* its counterpart, ACS_b, enough not to *misuse* $VISA - GRANT_{entr}$ (and the included key). In other words, ACS_b issues a visa which it subsequently releases to ACS_a for distribution to H_a. This trust in ACS_a seems to contradict the reason for issuing two visas.

In the protocol described in Section 3.6.1, only one visa-key and a single visa is used for both exit of AD_a and entry of AD_b.

3.7.2 Replay Prevention

As motivated by our arguments in Chapter 1, one of the goals of an effective stub policy enforcement mechanism is the protection of stub AD network resources from unauthorized traffic. This includes traffic flowing to and from *equipped* end-systems. Previous *Visa* protocol versions [32, 27] dealt with this issue by ensuring data integrity and authenticity of the packets crossing AD boundaries. One deficiency in the previous design was the absence of any provisions for detecting replayed packets belonging to authorized connections. This has been remedied (as evidenced in Section 3.6) by requiring that end-systems attach a unique timestamp to each visa-stamped packet and visa-routers keep a per visa record of a timestamp for the last packet processed.

One important insight related to this requirement is the apparent impossibility of effective replay detection without state in visa-routers. With some degree of synchronization, a stateless visa-router may be able to detect *very old* packets. However, if an intruder duplicates each legitimate packet and injects duplicates into the packet stream shortly after the corresponding original packets, duplicates can not be detected.

3.7.3 Visa Expiration

The only method for visas to expire in the previous protocol versions is by way of timeouts, i.e., an explicit time limit is negotiated at setup time and a visa is invalidated when the time limit is exceeded. While this is adequate for some types of connections, provisions for other methods of visa expiration may be necessary. These include limits on inactivity periods, number of packets and bulk data transferred.

Unfortunately, expiration based on limits other than just simple timeouts is not possible without state in visa-routers. In order to expire visas according to any of the above criteria requires that a visa-router maintain running tally of packets, data bytes or the time of last packet arrival on a per visa basis.

3.7.4 Visa Revocation

Visa termination by explicit order from an ACS should be viewed as more of exception than the norm as, ordinarily, visas are terminated by exceeding some limit negotiated at the time of issuance. Nonetheless, in circumstances where there is suspicion of a visa's compromise an ACS may need to revoke a visa prematurely, i.e., before its resource limits are reached.

In order to revoke an active visa, an ACS contacts the appropriate visa-router and identifies the visa targeted for termination. Thereafter, the visa-router has to ensure that no more packet traffic belonging to the revoked visa connection passes through. In a stateful model, a visa-router can simply delete the entry from its table thereby barring any further traffic. In a completely stateless model where each packet carries the entire visa along with the derived visa-stamp, a visa-router has to keep **state** with respect to the revoked visa. (Otherwise, it can not distinguish *bona fide* visas from revoked ones). A direct consequence of this requirement is the inability of a completely stateless visa-router to support visa revocation.

3.7.5 Coverage of Packet Signatures

In the context of this section, a packet is composed of two portions: network-layer header and data. As discussed in Section 3.3, data authenticity is one of our primary goals. Hence, there is no question as to whether or not the data portion needs to covered by the packet signature. Network header is a different matter, however. A typical network-layer header contains addressing information such as source and destination end-system addresses, packet sequence number, packet length and other fields. (Figure 3.3 depicts the IP [73] datagram header, for example).

Any header field not covered by the packet signature leaves a potential covert channel, since an intruder could trap a valid packet, change the unchecked field, and forward the modified copy without raising suspicion. We could protect against this by including the entire network-layer header under the packet signature, but in most internetworking protocols there are some header fields that are modified by the intermediate routers, and hence cannot be included in the signature. (All routers may have to modify the header, not just visa-routers, and we assume that non-visa routers do no regenerate the signature. If a public-key method is used, not even visa-routers can do so.)

For example, there are two such variable fields in the IP protocol. One is the header checksum; this cannot be forged because it is a function of the other fields in the header, and is already recomputed by each IP router. The other is the 8-bit *Time-To-Live* (TTL) field, used to prevent packet from looping forever. The TTL must be decremented by each IP router, and must never be incremented. An intruder could communicate approximately 6 or 7 bits per datagram by manipulating the initial value of the TTL field in copies of otherwise validly-signed packets.

If this covert channel is a reason for concern, there are a number of steps that can be taken. The entry visa-router (GW_b) can use the knowledge of its AD topology to reduce the TTL value to the minimum necessary for the packet to safely arrive at H_b, thereby reducing the bandwidth of this covert channel. Alternatively, GW_b could always set the TTL to its maximum value. (However, this would violate the letter of the IP specification, and might confound protocols that use the TTL field to limit the lifetime of a packet).

Another issue has to do with the addressing information in the network header. Recall that visas are issued on the basis of the source and destination end-system addresses (sometimes, along with the transport-layer protocol number). As described in Section 3.6.2, each visa-related data packet carries a visa identifier (TS_a). This identifier is stored alongside the two end-system addresses in visa-tables of both GW_a and GW_b. Since a visa-router still has to consult its visa-table to look up the visa-key, it can (inexpensively) verify the H_a, H_b addresses as well. Therefore, it can be argued that end-system addresses and other information (e.g., type-of-service, transport protocol, etc.) that is stored in the visa-table entry does not need to be protected by the packet signature.

3.7.6 Fragmentation

In a number of internetworking protocols (e.g., IP) a router may have to fragment a packet if it cannot be transmitted in a single unit. Data signatures complicate the use of fragmentation since the fragments must appear to have been signed by H_{src}, but the signatures would have to be computed by the fragmenting router. With signatures based on conventional cryptography, fragmentation is a problem because only a visa-router can do it while preserving the data signatures. With public-key signatures, this is impossible, since only the originating end-system can compute the signature.

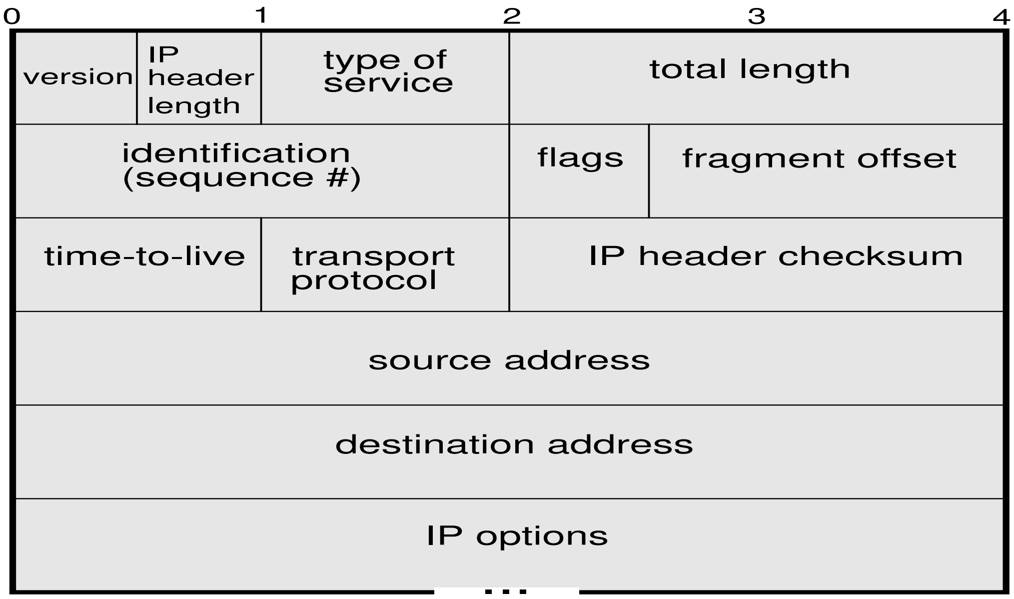

Figure 3.3: IP datagram header

Fragmentation is at best a necessary evil [47]; it is almost always better to set packet sizes at H_{src}, to make the best possible use of the available bandwidth and to provide acknowledgements for each transmission unit. Although methods have been proposed for accommodating fragmentation while preserving data signatures [88], we insist that the source end-system avoid sending packets that will have to be fragmented. A router should assist in this by returning an error packet when it is unable to transmit a data packet without fragmenting it; in fact, the IP protocol includes a mechanism for doing so through the use of ICMP *Destination Unreachable/Fragmentation Needed* message [74].

3.7.7 Loss of State

Reliable state in visa-routers (i.e., visa-tables) is one of the fundamental features and requirements of *Visa* protocol presented in this chapter. Nevertheless, it is unreasonable to expect visa-routers to be non-faulty at all times. The implication is that loss of state in visa-routers may occur infrequently and provisions must be made for the reinstatement of state after when a visa-router becomes operational. For this reason, an initializing visa-router sends a special ROUTER-UP packet to its local ACS. (Visa-routers are required to keep local ACS address(es) in stable storage). Upon receipt of a ROUTER-UP, ACS replies with one or more VISA-LIST packets which contain a set of all currently valid visas issued to the visa-router in question. The format of a VISA-LIST is similar to that of a *Visa* packet, except that it contains multiple visa records. ROUTER-UP only specifies the identity of the initializing visa-router.[6]

In spite of the mechanism described herein, not all state can be effectively recovered. In particular, visa expiration conditions such as inactivity timeouts, data and packet limits require maintaining state (i.e., usage meters) that is updated on a per packet basis. Unless it is kept in stable storage (an unreasonable requirement), this state is irrecoverably lost when a visa-router fails.

[6] An intruder masquerading as a legitimate visa-router may generate ROUTER-UP packets, however, visas contained in the subsequent VISA-LIST are only intelligible to genuine visa-routers.

3.7.8 Stateful Model

In this section, we summarize the reasons for selecting stateful over stateless visa-router model. There are several incentives for maintaining state in visa-routers:

- Replay prevention
  As discussed in Section 3.7.2, it is impossible to detect replayed packets in a visa-router without maintaining state. This is best illustrated by a situation whereby an intruder simply duplicates each valid packet and forwards it immediately after the original. Because the two packets are *back-to-back*, even if GW_a has some idea of the $H_a \rightarrow GW_a$ delay, it would accept both packets as valid.

- Visa expiration by means other than timeouts
  In order to expire visas by means of data, packet or idle-time limits, a visa-router must maintain running tallies. This is impossible to achieve with a stateless router model.

- Visa revocation
  If an ACS decides that it no longer trusts a previously approved connection, it may need to revoke a visa prematurely. As described in Section 3.7.4, ACS explicitly notifies the appropriate visa-router (via a REVOKE packet) that a certain visa is no longer valid. To uphold the revocation of a visa, i.e., to let no more packets bearing derived visa-stamps through, the visa-router has to maintain a record (state) with respect to the revoked visa.

- Interplay with transit policy enforcement
  A final reason has to do with the integration of stub and transit policy enforcement mechanisms. As described in the next chapter, transit policy enforcement in stub ADs also takes place at AD boundaries in entities referred to as *Policy Gateways* (PGs). As it turns out, each PG in a stub AD maintains a table of end-system pairs that are engaged in inter-AD communication. The purpose of this table is to map end-system pairs into so-called *Policy Routes*. Although a PG and a visa-router are logically distinct, for the most part, they share the same physical location.

3.8 Security Analysis

In this section we address the security of *Visa* protocol presented earlier in the chapter. As mentioned before, intra-AD messages and other intra-AD communication is assumed to be secure and each AD is assumed to employ an authentication mechanism of sufficient strength. What remains to be shown is the security of the inter-AD communication, i.e., the two messages exchanged among ACS_a and ACS_b as part of the setup phase and the subsequent data packets.

For each of these messages, two security issues are of interest: i) whether or not the message conveys the information necessary to establish its origin, authenticate its contents and assure freshness, and ii) whether it can be used maliciously (e.g., if it is intercepted by an intruder) to achieve unauthorized or otherwise compromised communication.

3.8.1 VISA-REQUEST

The first inter-AD message in the setup phase is the VISA-REQUEST:

$$ACS_a \Longrightarrow ACS_b : VISA - REQUEST \quad (3.8)$$

$$VISA - REQUEST \ = \ [\, H_a, H_b, TS_a \,]^{\,DK_{ACS_a}} \quad (3.9)$$

Upon its arrival, ACS_b has to verify that ACS_a originated **this exact** packet **recently**. Once ACS_b obtains $CERT_{ACS_a}$, extracts from it EK_{ACS_a} and re-computes the packet signature it is assured that the packet is originated by ACS_a and is authentic. Of course, this is very much dependent upon the strength of the signature mechanism.

Both timeliness and uniqueness of the VISA-REQUEST are established by examining the TS_a field. Recall that TS_a is the timestamp assigned by ACS_a. ACS_a is responsible for ensuring that it has never been used before. As described in Section 2.2.3, ACSs's clocks are not necessarily closely synchronized, i.e., a certain clock skew is expected. We assume, there exists an upper bound (referred to as Δ_T) on the clock skew between any two ACSs. By comparing its current clock reading to TS_a, ACS_b can establish the timeliness of the VISA-REQUEST. However, *timeliness* is relative to Δ_T since an intruder can still delay a VISA-REQUEST for, at most, the value of Δ_T.

Uniqueness is a different matter. In order to establish that a VISA-REQUEST has not been *seen* before, each ACS has to keep state in the form of a *peer-table*. Each entry in the peer-table corresponds to a peer ACS (in a different AD) which has previously communicated with the ACS in question. Among other information (such as public key certificates), each entry stores the timestamp of the last VISA-REQUEST by the corresponding ACS. (We refer to this value as TS_a^{last}). If $TS_a < TS_a^{last}$, ACS_b can suspect a replay attack. However, it can not be absolutely sure since VISA-REQUEST packets can arrive out of order. If $TS_a = TS_a^{last}$, the VISA-REQUEST packet is obviously a duplicate.

As far as the intruder is concerned, there is little value in a VISA-REQUEST packet. It contains no secret fields and it only reveals that a visa for communication between H_a and H_b has been requested (hence, a VISA-GRANT may flow in the opposite direction soon thereafter). Also, modification of packet data is *unproductive* as each VISA-REQUEST is protected by an unforgeable signature. Replay of a VISA-REQUEST is just as futile as can be seen from the above discussion.

Another security-related aspect is the issue of the implied beliefs carried in a VISA-REQUEST. Specifically, it conveys to ACS_b that i) ACS_a authorizes the $[H_a, H_b]$ communication, and ii) believes that the requesting end-system H_a is authentic. ACS_b has no reason to question or doubt the former (since it is subject to AD_a's local policy embodied in ACS_a), however, with regard to authentication of H_a, ACS_b has to *take ACS_a's word for it.* In other words, the end-result is that: *ACS_b believes that ACS_a believes that H_a is authenticated.* This conclusion may be too weak in some circumstances. In order to obtain a stronger conclusion such as *ACS_b believes that H_a is authenticated*, either:

1. ACS_b has to conduct a separate, possibly higher-level, authentication dialog with H_a, or
2. ACS_a has to include an unforgeable proof of identity for H_a as part of the VISA-REQUEST

The second choice is preferable because it does not involve relying on other protocols and introduces no additional communication overhead. However, it has two important drawbacks: i) the problem of scale with regard to issuing proof-of-identity certificates at the granularity of end-systems, and ii) the additional overhead incurred by the use of public-key encryption for subsequent communication. This subject is discussed further in Section 3.8.3 below.

3.8.2 VISA-GRANT

VISA-GRANT is the second inter-AD message exchanged as part of the setup phase.

(3.10) $$ACS_b \Longrightarrow ACS_a : VISA-GRANT^{ACS_a}$$

(3.11) $$VISA-GRANT = [H_a, H_b, TS_a, (S_b^a)^{EK_{ACS_a}}]^{DK_{ACS_b}}$$

When ACS_a receives a VISA-GRANT packet, it has to:

- match the TS_a found in the packet with the corresponding TS_a stored in its table of outstanding VISA-REQUESTs. A successful match, in itself, demonstrates both freshness and uniqueness of the VISA-GRANT packet.
- authenticate the contents of the packet. Since ACS_a already knows EK_{ACS_b}, it can re-compute the packet signature and establish that ACS_b originated this packet and its contents are authentic.

As with a VISA-REQUEST, an intruder can capture a valid VISA-GRANT packet and attempt to attack the protocol. However, since a VISA-GRANT is signed and timestamped, no modification or replay is possible without detection. A danger even graver than successful fabrication or replay of a VISA-GRANT packet is the possibility of an intruder discovering S_b^a. (This would permit the intruder to generate arbitrary "genuine" packets). Therefore, the effectiveness of *Visa* protocol depends to a great extent on the strength of the underlying public-key encryption (signature) function.

3.8.3 Data packets

Recall that data packets belonging to a visa connection have the following structure:

(3.12) $$DATA-PACKET = [HEADER, DATA, VISA-STAMP, TS_a]$$

A data packet arriving at GW_a or GW_b, has to show that:

- it originated at H_a and is addressed to H_b

- it was sent recently
- it was not modified in transit

We begin by observing that the method by which GW_a determines the origin of a data packet is the verification of the visa-stamp attached to the packet which is computed with a secret visa-key, S_b^a. In addition to H_a, S_b^a is known to four other principals: H_a, ACS_a, GW_a, ACS_b and GW_b. This implies that any of these principals can potentially generate *genuine* visa-stamps.

After verifying the visa-stamp, timestamp and addressing information of a data packet, the only conclusion that GW_a can make is: *the packet contents are authentic, the packet is not a replay, and its origin can be any of* H_a, GW_b *or* ACS_b. We assume that GW_a believes in its own *goodness*, thus it can be sure that it did not originate the said data packet, and, GW_a trusts ACS_a enough to believe that ACS_a did not originate the packet. Similarly, GW_b can establish that the origin of a data packet can be H_a, GW_a or ACS_a.[7]

The uncertainty about the exact origin of data packets is, potentially, a reason for concern. Nonetheless, we argue that the overall security and robustness of *Visa* protocol relies on the invulnerability of the ACSs and visa-routers. If the security of an ACS or a visa-router is somehow compromised, fraudulent visas may be issued and unauthorized communication can transpire. On the other hand, it is worth considering, if only for academic reasons, what it would take to remove the above uncertainty, i.e., allow visa-routers to determine the source of each data packet unambiguously.

To do so would require that each end-system be able to certify its identity, i.e., be uniquely identifiable by a some property that can be used to sign data leaving no doubt about the identity of the signature creator. The immediate consequence of this statement is that conventional cryptography is unable to solve the problem since it calls for (at least) two principals sharing a key. (If ACS_a and H_a possess the same key, K_{H_a}, any one of them can generate data signatures with that key).

The only remaining choice is the use of public-key cryptography. Suppose that each participating end-system is issued a public-key pair $[EK_{H_a}, DK_{H_a}]$ and a corresponding certificate, $CERT_{H_a}$, by a well-known authority.[8] Instead of issuing a secret, conventional visa-key, ACS_b would generate a public-key pair, $[EK_b^a, DK_b^a]$, and a VISA-GRANT packet would take the form of:

$$VISA - GRANT = [H_a, H_b, TS_a, , EK_b^a, (DK_b^a)^{EK_{H_a}}]^{DK_{ACS_b}} \tag{3.13}$$

Note that S_b^a is replaced by DK_b^a, the secret (signature) component of the public-key pair. Because it is encrypted with EK_{H_a}, DK_b^a can only be computed by H_a. However, anyone (ACS_a, GW_a, GW_b, etc.) can obtain the corresponding public key, EK_b^a. If H_a generates data packet visa-stamps with DK_b^a, both GW_a and GW_b can trace the packet source to H_a as no other entity can be in possession of DK_b^a.

While this approach results in somewhat increased protocol security, its benefits are outweighed by the cost of using public key encryption on a per packet basis. Another drawback is (as alluded to in Section 3.8.1) the issue of scale with respect to certification of the individual end-systems. Finally, one of the fundamental protocol assumptions is the security of the ACSs and visa-routers. Because ACSs are presumed to exercise extensive control over their constituent end-systems, it may be unreasonable to trust an end-system in an AD whose ACS or visa-router is not trusted.

3.9 Protocol Costs

In this section we address the impact of *Visa* protocol on its participants. In particular, we consider the costs incurred per visa as well as per data packet.

3.9.1 Setup and Distribution

The setup phase involves at least two packets: a HOST-REQUEST from H_a to ACS_a, and a VISA-REQUEST from ACS_a to ACS_b. The distribution phase involves at least four more packets: $VISA - GRANT^{ACS_a}$, $VISA -$

[7] Note that if the protocol operates correctly, none of GW_a, GW_b, ACS_a and ACS_b ever generate data packets signed with S_b^a.
[8] The authority can not be ACS_a or any other entity in AD_a, for obvious reasons.

$GRANT^{GW_b}$, $VISA - GRANT^{GW_a}$ and $VISA - GRANT^{H_a}$.[9] In total, a minimum of six packets are exchanged, yet only two of the six travel across AD boundaries.

3.9.2 State Overhead

State overhead, consisting of storage costs, is introduced in this protocol mainly by the need for all participants, but especially visa-routers, to keep visa-tables.

In order to facilitate fast packet switching, routers often use specialized hardware equipped with very fast memory. However, this memory is expensive and, hence its availability is limited. Therefore, storage overhead in visa-routers is of particular importance. Its major contributing factor is the maintenance of a visa-table where each entry corresponds to a currently valid visa. As illustrated in Figure 3.4, each entry contains essentially the same information as carried in the corresponding VISA-GRANT packet. The only exception is that visa-routers must constantly monitor bandwidth usage of individual connections. Also, the timestamp of the last data packet, $TS_{last}^{(a,b)}$ is kept in order to i) prevent replay of old packets, and ii) monitor the inactivity time.

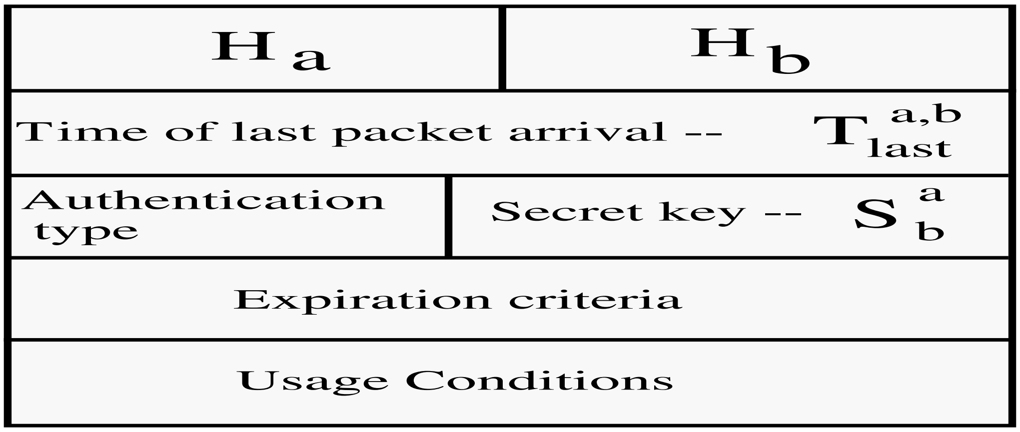

Figure 3.4: Visa table entry

3.9.3 Per packet costs

The per packet costs include the additional fields in data packets, table look-ups in visa-routers, and data signature computations.

Each data packet carries a visa header depicted in Figure 3.2 above. The visa identifier, TS_a is 64-bit quantity (sufficient width for a timestamp) and the end-system timestamp, TS_{H_a}, is also 64 bits wide. The size of the visa-stamp depends on the signature method, e.g., 128 bits for MD4 secret prefix/suffix or 64 bits for DES-based MAC.

The overhead due to data signature operations depends upon the particular signature scheme used. It also depends upon whether the operation involves passing over the entire or only part of, the packet. Furthermore, the cost of

[9] Additional exchanges may be required in order for H_a to authenticate itself to ACS_a and ACS_b.
[10] See Section 3.8.1.

signature computation can be significantly more expensive than the cost of signature verification as in some proposed variations of RSA [76]. In general, there are three operations involved: signature computation at H_a and two signature verifications at GW_a and GW_b, respectively.

3.10 Summary

In conclusion, *Visa* protocol presented in this chapter is a simple and powerful mechanism for controlling packet traffic at stub AD network boundaries. The primary goal of *Visa* protocol, i.e., protection of stub AD network and end-system resources from unauthorized access, is achieved by requiring that all communication be first authorized by the ACSs in end-point ADs. Authorization to communicate is then embodied in a visa which is issued to the requesting end-system and distributed to the intervening visa-routers. Subsequent packets carry unforgeable visa-stamps which are verified by the visa-routers. Visas are terminated when the underlying communication exceeds a pre-determined resource limit.

Visa protocol does not impact intra-AD communication and has no influence on the end-systems that do not partake in inter-AD communication. Although, as illustrated in Chapter 5, the cost of connection setup is somewhat high, per packet overhead due to the protocol is reasonable, especially, when non-cryptographic packet signatures are used.

Chapter 4

Transit Policy Enforcement: Control of Transit Internetwork Traffic

An important conclusion reached in Chapter 1 is that controlling the usage of network resources (such as routers and links) requires additional protocol support because of the need to coordinate routing decisions among all intervening networks. Thus, organizations (ADs) cannot simply enforce policy restrictions on a unilateral basis at packet forwarding time. Instead, internetwork *routing* decisions must be made according to policy-related parameters such as access rights and cost, in addition to the traditional parameters of connectivity and delay [13, 26, 53]. Consequently, policies pertaining to network resources must either be implicit in the topology of an internetwork, or advertised to the anticipated resource users. Only then can entities throughout the internetwork determine the logical, *policy-based* connectivity of an internetwork and compute valid routes.

This chapter addresses the design of protocols for **secure** control of transit traffic on an internetwork. Transit control protocols can be designed with varying levels of security. In some environments, relatively vulnerable protocols may be used in conjunction with *post facto* detection mechanisms. Most of the work in policy-based protocol development is being conducted with such environments in mind [13, 53, 34, 54]. We are interested in environments where post facto detection is not adequate or possible in a timely manner. In particular, we address the design and costs (i.e., performance and manageability) of incorporating more defensive, *preventative* security measures into the protocols for controlling internetwork traffic.

The rest of this chapter is organized as follows. Section 4.1 begins by considering the extension of network access control methods to control transit internetwork traffic. In response to some fundamental deficiencies of both network access controls and traditional, delay-based internetwork routing, the remainder of Section 4.1 describes the role of the so-called Policy Routing protocols in providing transit control. Next, Section 4.2 outlines the security concerns particular to the Policy Routing protocols. The discussion of security is continued in the remaining sections on secure protocol design (Section 4.3) and cost assessment (Section 4.5). Section 4.6 summarizes our discussion.

4.1 Controlling Transit Traffic

There are two basic approaches to controlling transit traffic. In the first part of this section, we discuss an approach based on the extension of traditional network access control mechanisms and identify its limitations. Subsequently, we consider alternative approaches based on integrating controls into internetwork routing.

4.1.1 Extending Network Access Controls

One potential method of enforcing transit policy enforcement is the extension of existing stub AD access control mechanisms to the generalized internetwork model. In this section, we discuss an extension of *Visa* protocol that incorporates support for transit policy enforcement. Other network access control schemes are discussed briefly at the end of this section.

4.1.1.1 Transit *Visa* Protocol

As described in the previous chapter, *Visa* protocol was originally designed to provide packet-level control at AD boundaries.[1] Recall that a secret visa-key is used by the communicating end-system to compute a visa-stamp which is attached to each data packet to assure appropriate border routers that the transmission across AD boundaries is authorized. A visa-stamp is analogous to a stamp on a passport that allows a traveler to cross international borders. A unique visa-stamp is bound to each packet in order to guarantee the authenticity and the integrity of the data.

The process of establishing authorization in visa-controlled transit ADs is essentially the same as for stub ADs in *Visa* protocol. The main difference is that now the source end-system must obtain authorization for each transit AD that it traverses, in addition to obtaining authorization from AD_{src} and AD_{dst}. In the worst case, each transit AD's ACS will conduct an authentication procedure before establishing authorization. Of course, transit ADs may choose to do so automatically, or not require any authentication at all where transit traffic is concerned. Furthermore, stub ADs could program their ACSs to obtain and issue transit visa-keys in advance of the actual communication. This would reduce the setup delay at connection establishment time. On the other hand, such mechanisms increase the problems associated with visas expiring before, or while, they are in use.[2]

4.1.1.2 Discussion

The use of *Visa* protocol for transit control may be appropriate if: (i) transit policies are as diverse as stub network policies, and (ii) policies change frequently. The former limits the practicality of expressing policies in a simple universal syntax. The second assumption also makes it impractical to distribute policies in the way that we distribute connectivity information, for the fear of using stale route information or incurring excessive overhead due to frequent information updates. These assumptions result in several interesting features of transit *Visa* protocol. First, organizational policies are embodied in ACSs and are not propagated outside; hence, a wide range of policy statements can be accommodated. Moreover, very little coordination among ADs (beyond that in *Visa* protocol for stub networks) is required to implement this protocol.

Although the extension of the *Visa* protocol concept to transit control is rather straightforward, the approach does not scale well to an internetwork where many ADs, both stub and transit, want to control traffic flows. For example, acquisition of visa-keys and route setup must be repeated (or adapted) each time an involved visa-router goes down. Moreover, a source AD has no way of determining if it will be issued a *Visa* without incurring the overhead of contacting the particular ACS in question.

This leads us to a (not unexpected) conclusion that transit traffic control is intimately related to internetwork routing. Therefore, controls for transit should be incorporated into the route calculation itself, not only into the packet forwarding function [13].

Other network access control schemes such as SP3 [81] and router packet filters [61] face the same limitation when it comes to controlling transit traffic, i.e., these schemes enforce controls on packet forwarding and do not provide information to the route computation. For example, SP3's access control policy is endpoint-oriented. It is concerned mainly with determining whether or not two end-systems may communicate and what type of information they may exchange.

In summary, *Visa* protocol and other network access control mechanisms are best suited for their original purpose, stub AD access control. Transit control for large internetworks is more efficiently achieved by integrating policy considerations into the route computation process. After discussing policy-based routing in the next section, we return to the problem of **secure** control of transit internetwork traffic.

4.1.2 Policy Routing

As described earlier, the central goal of transit traffic control is to allow ADs to independently express and enforce policies regarding transit traffic. The discussion in the previous section demonstrates that transit control is intimately

[1] In a connection-oriented network a similar approach can be applied to stamp packets. However, the establishment of a visa-key can be part of the connection setup and several of the datagram related design issues do not apply.

[2] More aggressive policies could also be implemented such as applying for group visas in advance of use to accommodate a collection of end-systems that have a need for communication. However, these may imply significantly more trust among the ADs and require more careful consideration.

related to Inter-AD Routing. We refer to inter-AD routing that incorporates policy constraints as policy routing (PR). Inter-AD routing constitutes the highest level of the OSI routing hierarchy as defined in [45].

In this section we concentrate one approach to policy routing, the Inter-Domain Policy Routing (IDPR) proposal [50, 84]. IDPR is designed to support more general transit policies than other policy routing methods discussed in Chapter 2. (For further discussion of inter-AD routing architectures see [7, 29].)

Policy routing operates at the network layer. Only border routers and associated route servers are directly affected by the presence of the inter-domain routing protocols. End-systems and interior routers continue employing internetworking protocols desired within their particular ADs. Border routers operate on behalf of the end-systems. For this reason, the term *source* hereafter refers to the border router in the AD of the source end-system.

4.1.2.1 IDPR Architecture

Inter-Domain Policy Routing architecture has been developed to support a wide range of policies. It incorporates mechanisms that represent a rather radical departure from existing routing techniques.[3] IDPR allows stub and transit ADs to express and exchange routing and packet forwarding policies. The most distinguishing feature of this approach is the use of AD-level source routing. A *Link State* algorithm [56] is used to compute source Policy Routes (PRs) at the granularity of ADs. Each AD expresses its policies in a *standard* form, called Policy Terms (PTs), and distributes them to other ADs. Each AD designates special Route Servers (RSs) to collect PTs and compute policy routes for constituent users. The basic assumptions of this model are:

1. Most policies can be classified and expressed in a standard notation.
2. Policies and inter-AD connectivity change relatively slowly.
3. End-point specific policies should be supported.

Two primary concepts in this proposal are *Policy Routes (PRs)* and *Policy Terms (PTs)*. A PR is an ordered sequence of ADs, i.e., an AD-level source route. In other words, there may be multiple physical realizations of a PR given multiple physical connections between ADs and multiple intra-AD routes. The actual selection of a particular physical path is done at packet forwarding time by each intervening AD, rather than by the source AD at route computation or route selection time. This *lazy evaluation* provides for a more adaptive protocol and unrestricted AD interconnection. Policy Terms (PTs) are the units of routing information exchanged by communicating ADs. Each PT represents a distinct policy of the AD that expressed it. The information distributed in a PT can be represented as:[4]

$$[(H_a, AD_a, AD_{ent}), (H_b, AD_b, AD_{exit}), UCI, Conditions] \tag{4.1}$$

The purpose of a PT is to specify that packets from some end-system (or a group thereof), H_a, in a stub AD, AD_a, are allowed to enter the AD in question via some directly connected AD, AD_{ent}, and exit through another directly connected AD, AD_{exit}, on its way to or from an end-system, H_b, in another stub AD, AD_b. User Class Identifier (UCI) allows policies to distinguish between various user classes, e.g., Government, Research, Commercial, Contract. Conditions represent quality of service, billing, and other variables, and can reflect the agreements between neighboring ADs. Examples of the Policy Terms can be found in [26, 13].

Policies are expressed by source, destination, and transit ADs. The source AD may select all transit ADs while transit and destination ADs control which source and destination ADs can communicate via which directly connected ADs. ADs run link state routing algorithms to compute their respective tables of PRs. There may be multiple PRs listed for the same destination AD, each with a different set of conditions associated with its use (e.g., QoS, time-of-day, or UCI).

Note that ADs (with the exception of the source AD) do not exert control over the entire Policy Route. Referring back to our travel analogy, it is difficult to enforce policies that are based on information that is not verifiable at the point of reference. For example, it is difficult to enforce a policy that dictates non-admittance to anyone who has *ever* passed through country X, since it is very much dependent on X stamping passports reliably. In the environment of interconnected ADs, a transit AD can verify the previous and next hops because of its direct connections and the

[3] This approach was first described by D. Clark in [13].

[4] This is a simplified version of the actual PT format used in the protocol. However, the differences are not relevant to this discussion.

feasibility of employing pairwise authentication with the relatively small number of neighbors. Verifying other transit components of the PR is difficult, if not impossible.

If the end-point authentication described in later sections is *not* employed, spoofing of end-point addresses en route will result in violation of some end-point specific transit policies and possibly inappropriate billing of the endpoints for transit services. However, the functionality available to the interlopers is limited– they can send packets in between the indicated endpoint ADs. In contrast, if the spoofer modifies path information other than the end-points and previous and next hop in order to be permitted by a transit AD's path specific policies, the spoofer is not constrained in the manner in which it can exploit these policies.

Each AD has one or more Route Server (RS), an entity that collects Policy Routing information from other ADs, distributes local policy information to other ADs and computes, as well as issues, PRs to local end-systems. Actual policy enforcement is done at a Policy Gateway (PG) which, in addition to the usual task of forwarding packets, handles validation and verification of the PRs attached to the incoming packets.

A path is established with the first packet carrying the full PR, i.e., the complete sequence of ADs in the route and applicable PT identifiers. PGs along the route make sure that the PR agrees with the local PTs (through use of templates, for example). The result is cached so that a specified *PR handle* can be used in the future to refer to the cached entry. Successive packets carry a short PR handle, not a full PR. Many transport level sessions, and even pairs of end-systems, may share a single PR if the policies enabling it are not end-system-specific; this reduces the average latency and state overhead for intervening PGs. PGs use PR handles in the packets to check for cached entries. PGs also may relate return flow packets with forward flow. Given information about the next AD for a particular packet, each PG selects the next PG based on the information exchanged in a traditional up-down protocol.

4.1.2.2 Discussion

Policy routing allows ADs to interconnect to the global internet while still protecting network resources from general, unconstrained use. (We described earlier why such a function can not be left to end-systems.) However, policy routing mechanisms do not preclude the need for network access controls in the border routers of ADs that wish to control access to individual end-systems. This subject has been extensively discussed in Chapter 1.

One essential difference between *Visa* protocol and policy routing approaches is the per session setup overhead. Transit *Visa* protocol requires that a dialog transpire between the source and each transit ADs' ACS, and that corresponding keys be distributed. Consequently, the initial setup delay grows in proportion to the number of transit ADs in a PR. For short transactions such overhead is not acceptable. A policy-sensitive approach such as IDPR avoids this setup dialog through background distribution of policy information that is used in route computation. The work that the Policy Routing protocols do to distribute policy terms and compute authorized routes must be done at the time of the session setup in *Visa* protocol.

In particular, with transit *Visa* protocol this translates into a dialog with an ACS for each AD in the path. Moreover, this assumes that the source attempts communication over a path for which it has authorization. If there is a conflict with even a single transit AD's policy, the process must begin again. Policy Routing incorporates policy into the route computation process in advance of the actual communication, thereby avoiding this problem.

On the other hand, policy-based routing, as described thus far, relies on *post-facto* detection of abuse, and, is in that sense less secure than network access control schemes, such as *Visa* protocol. The remainder of this chapter addresses the integration of preventative mechanisms into policy routing to achieve secure control of transit traffic for those ADs that desire it.

4.2 Security Issues in Transit Control

In this section we identify potential security threats faced by policy routing and detail the steps needed in a secure protocol.

4.2.1 Specific Threats

The two basic threats to the security of a PR protocol are falsification of routing information and falsification of data packets.

1. An intruder may distribute false routing information in order to (i) disrupt communication, e.g., create routing loops, or (ii) eavesdrop on communication, e.g., re-route traffic to a specific location. This can take the form of distributing falsified or prerecorded PG UP/DOWN information which may cause stub ADs to compute PRs that are desired by the intruder. Also, falsified Policy Terms can be similarly distributed which may result in invalid PRs being computed.

2. If routing protocols protect themselves by prescribing an authentication mechanism for validating routing information, the intruder can turn to falsifying control and/or data packets. This kind of attack can lead to unauthorized resource usage, unauthorized communication, and inappropriate accrual of charges. We identify three sub-threats:

   (a) Falsification of control information
   In addition to the usual network layer information (source and destination addresses, data size, etc.), control information includes the route setup and packet forwarding parameters. An intruder can also *steal* a PR handle created by an authorized AD and substitute an invalid Charge Code.

   (b) Falsification of data
   Data portion of a packet can be modified or replaced by an intruder.

   (c) Replay
   Previously-recorded legitimate packets can be replayed by an intruder. Two sources of replay are of equal concern: accidental replay due to the *stuttering* of a misbehaving machine, and malicious replay due to a misbehaving person attempting a denial of service attack.

In the remainder of this chapter we describe and analyze mechanisms to resist these threats. Our approach is not inconsistent with the original proposal in IDPR. The *prevention*-based (as opposed to detection-based) measures proposed here can be included as optional features within the protocol.

4.2.2 Terminology

The following terminology is used throughout this chapter:

- $||$ is the concatenation operator, e.g., $X||Y$.
- N denotes the length of a PR, i.e., the number of ADs in a PR.
- AD_i $(0 < i < N)$ denotes the i-th AD in a PR, $AD_1 = AD_{src}$ and $AD_N = AD_{dst}$.
- K_i^j denotes a secret key shared by AD_i and AD_j.
- K_{dsig} denotes a secret key used for computing data signatures.
- EK_i, DK_i denote public (encryption) and private (decryption) keys of AD_i.
- $E(Data)^K$ and $D(Data)^K$ denote encryption and decryption, respectively, of $Data$ with key K.
- $F_{hash}(Data)$ is a one-way hash function digest of $Data$.

In the next three sections we consider separately security issues pertaining to different phases of Policy Routing: distribution of PTs, PR setup and packet forwarding. In each of these phases we are concerned primarily with data integrity and source authenticity. Confidentiality of user data is left to end-to-end mechanisms [90]. Confidentiality of routing control information is a less general requirement than integrity and authenticity, and is discussed briefly. Traffic analysis is not addressed here (per our discussion in Chapter 1).

4.2.3 Distribution of Policy Terms

In order to provide for secure distribution of Policy Term updates, each AD must be able to sign its own, and authenticate incoming PTs. Because of the Link State nature of the protocol, PT updates must be flooded to ADs throughout the internetwork so that all participant ADs can use them in their PR computation. Before using a new PT, each AD needs to verify the authenticity and integrity of its contents. This is difficult to achieve in a conventional encryption environment, as the number of potential recipients of a PT update can be quite large.[5] In general, conventional encryption is not well-suited for an environment such as Link State routing where routing updates are broadcasted to a large number of recipients.

The alternative is to use public key encryption for the distribution of PTs. At first glance, this might appear problematic because current public key technology is still inferior in terms of performance. However, we are not interested in the privacy, but only the integrity and authenticity of routing information. Therefore, signature methods described in Appendix A can be used with little performance impact. Moreover, one of the central assumptions in the IDPR proposal is that policies change relatively slowly. Any added processing time associated with public key encryption is counter-balanced by the ubiquity and efficiency of being able to generate a single unforgeable packet signature which can be authenticated by any recipient. An example of a Link State routing protocol which uses public key encryption for routing update distribution is presented in [71].

Routing information distribution is one area of policy routing where confidentiality measures may be considered useful. However, it implies that the entire routing update must be encrypted **separately** for each anticipated destination AD. In general, this is impractical regardless of the type of encryption used. Since a link state update is flooded throughout the internetwork (in this case, to all ADs), the number of potential destinations can be quite large. In order to achieve confidentiality in this environment, the source of a link state update needs to encrypt the update N times (where N is the total number of ADs). Furthermore, the traffic due to update propagation will increase N-fold.

For the remainder of this chapter we consider environments in which only integrity and authenticity of routing data is required and where public key encryption and the certifying authority mechanisms described in Chapter 2 are used for this particular function.

Although the distribution of PTs raises a number of interesting issues, the underlying concept of broadcasting signed messages is not unique to Policy Routing.[6] Other aspects of secure Policy Routing, such as PR setup and packet forwarding, require more careful protocol design because of the associated costs and threats.

4.2.4 Route Setup

PR setup phase requires that each intervening AD have means to forward subsequent data packets along a specific Policy Route. As described in Section 4.1.2.1, each AD along the route must be supplied with the next hop AD at PR setup time. The purpose of PR setup is to establish authorization in all transit ADs and create state in all intervening PGs. Thus, Policy Routing decisions can be made in advance of the actual communication, and, subsequent data packets can carry a minimum of PR-related information thereby reducing both overhead and latency.

In order to recognize a *bona fide* PR, each AD must authenticate and authorize a PR. Authentication means verifying that a PR was issued recently by a recognized entity and was not tampered with. Authorization entails making sure that a PR conforms to local policy.

More specifically, secure PR setup needs to address two types of threats:

Type 1. *Creation of fraudulent PRs or tampering with existing PRs.*
In order to defend against this threat, each PR must be traceable to the issuing AD. In other words, it must be signed with an unforgeable signature. For all intervening ADs, this would provide for non-repudiation of issuance and sender authenticity.

Type 2. *Replay of previously issued setup packets.*
This can be prevented if we include a timestamp within each PR. The signature of a setup packet becomes dependent on this timestamp, and, makes replay detection possible within the granularity of the timestamp and clock synchronizations. As the number of setups is relatively small (in relation to the number of data packets), relatively coarse timestamp granularity (e.g., $1ms$) should be adequate and is preferable to the management required to keep track of unique sequence numbers. We note that, detecting replay of PR setup packets is very similar to that of VISA-REQUEST packets in *Visa* protocol (see Chapter 3).

As with PT update distribution, we are concerned with the data integrity and authenticity of the setup packets. However, unlike PT updates, PRs are set up relatively frequently and increased latency is experienced directly by the end-users. Thus, we are far more concerned with the per-signature overhead for PR setup than we are for PT distribution. Consequently we will investigate the use of both conventional and public key encryption signature mechanisms.

[5] Sharing a single common key among all nodes affords little protection, while distributing pairwise keys to each AD-pair is impractical.
[6] With the exception that routing updates are infrequent, thus, favoring the use of public key encryption.

As discussed earlier, conventional encryption, implies a significant key management burden, since an AD has to share a secret key with every other AD that it ever communicates with. Moreover, it entails computing a PR signature for every AD involved, whereas a single PR signature verifiable by all intervening ADs is sufficient in public key encryption. On the other hand, a typical PR traverses a relatively small number of ADs (N is usually much less than the diameter of the internetwork). This makes conventional encryption a viable choice since a PR would only have to be signed N times.

The above discussion is a standard public key *verses* conventional encryption debate. Both methods have beneficial as well as burdensome features. In Section 4.3 we demonstrate a PR setup protocol based on public key encryption, a conventional encryption variant is similar.

4.2.5 Packet Forwarding

After a PR has been set up, subsequent data packets can take advantage of the PR state in intermediate ADs. First, instead of a full PR, each data packet carries only an abbreviated version referred to as the *PR handle*. Second, the state in all intervening PGs allows them to bypass expensive authorization checks on a per packet basis. A PR handle only needs to contain the information necessary to identify the appropriate state in intervening PGs. Its exact contents are described in the next section.

Assuming appropriate security measures to prevent PR setup threats above, there remains the possibility of attacks at packet forwarding time:

Type 3. *An intruder can copy a valid PR handle from a legitimate packet, attach its own data and send it along a PR, thus, obtaining service fraudulently.*

This attack can be fully remedied if each data packet is signed by the source and the signature is verified at each transit hop. However, even more so than in the case of PR setup, processing delay is a critical concern with respect to forwarding of data packets. For this reason, many ADs are likely to forego per-packet signature and verification of most traffic. We now discuss a spectrum of possibilities whereby ADs can trade the level of protection for the amount of overhead incurred.

There are three factors that directly influence the cost of data authentication (we discuss them below):

1. Signature computation method
2. Signature coverage
3. Signature verification method

4.2.5.1 Signature computation

The particular method used in the signature computation influences not only the overhead but also the security of the packet forwarding protocol. The choice of a signature mechanism is mainly dependent upon the severity of the anticipated hostile attacks.

If hostile attacks are not anticipated, the signature needs only afford protection against occasional non-malicious modification of data packets. This can be accomplished by using a simple, relatively weak CRC function (e.g., an IP-style CRC [73]).

To protect against tampering with legitimate data packets, the signature function must have the property that modification of data results in unpredictable changes in the signature. A strong one-way function such as MD4 or MD2 is a good candidate for this type of protection.

Protection against message substitution is more complicated. In addition to strong data integrity protection, the signature must be *unforgeable*, i.e., only a legitimate entity must be able to compute a signature. This is not the case with one-way hash functions since anyone can produce an arbitrary message and compute its signature. In other words, protection against message substitution must incorporate some notion of signature origin authenticity. Consequently, a secret quantity (i.e., a key) must be used in signature computation such that only the true originator can produce genuine signatures and all recipients can easily verify them.

One simple solution is for the source to sign data packets in the same manner as PR setup packets, i.e., using public key signatures. This satisfies our goals in that there is but one entity able to generate signatures with a given key. Moreover, since transit ADs are already in possession of the source's public key (after processing setup packets), signature verification requires little additional work. Alas, all the benefits of this approach are overshadowed by the high cost of per packet public key encryption.

Another possibility is for the source to maintain pairwise secret keys with the signature verifiers, i.e., transit ADs. Then, every data packet would need to be signed N times (where N is the PR length) by the source using a different key each time. This implies significant overhead for long PRs in terms of encryption and added length.[7] On the other hand, the security of this method is almost as strong as that of public key signatures.[8]

Alternatively, the source can distribute a secret key during PR setup to be shared by all intervening ADs. (Thereby establishing a simple group channel). Then, only a single signature would be computed by the source and subsequently verified by all transit ADs. This method is efficient in terms of computation and added packet length. However, because the key is shared, it becomes possible for any AD along the route to masquerade as the source for the duration of the PR. Moreover, the key must be distributed without undue disclosure, i.e., only the intended recipients must be able to obtain it. Therefore, the key has to be encrypted N times, once for each transit AD. However, this expensive operation takes place only once per PR, at setup time.

This subject is discussed further in Section 4.3.

4.2.5.2 Signature Coverage

Independent of the signature computation method, is the issue of signature coverage, i.e., what part of a packet has to be signed. Recall that an IDPR-encapsulated data packet consists of: i) network-layer header, ii) PR header, and iii) data segment.

Maximum protection against tampering can be attained only if the entire packet is *covered* by the signature. Depending on the signature method used, this may prove to be prohibitively expensive. Consequently, transit ADs may not care about the authenticity of the *entire* packet.

One inexpensive alternative is to compute signatures over the network-layer header only. However, since a PR may be utilized by multiple end-system pairs, assigning signature keys per end-system pair complicates signature verification in transit PGs. This is because transit ADs maintain state on a per PR,‘not per end-system pair, basis. To remedy this scaling issue, a single signature key must be shared for all end-system pairs utilizing a given PR. (Actually, since signatures are always generated by the source PG, little additional security is afforded by using signature keys on a per end-system pair basis, over per PR signature keys). If the source PG signs the network-layer header, an intruder can still tamper with the data segment. The utility of this type of attack is limited since the intruder is not able to alter end-system addressing information.

Another possibility is to sign only the PR header. This leaves the network-layer header and the data segment unprotected, but we can guarantee that packets will travel along established, authorized PRs. The exact contents of a PR header are important in assessing the vulnerabilities of this method.

First, we suppose that a PR header contains only a PR identifier that helps each transit AD locate the appropriate entry in its PR table. Then, all packets belonging to a given PR carry the same PR header and the same PR header signature. An intruder can attack in two ways: i) modify data segments of existing data packets, and ii) inject its own data packets complete with PR headers copied from valid data packets. The first attack is impossible to counter with this method since data segments are not covered by the signature. We can, however, address the second attack by making each PR header unique. The originator PG can include a unique timestamp in the PR header of each data packet, which makes the signature dependent on the packet timestamp. Hence, each packet carries a distinct signature. Furthermore, if we require transit PGs to record and keep track of most recent packet timestamps on a per PR basis, duplicate and reordered packets can be easily detected [72] as described in Section 4.2.5.4 below.

The accomplishment of this method amounts to restricting the bandwidth available to a potential intruder to, at most, **the bandwidth utilized by the legitimate traffic source**. In other words, an intruder can still intercept every legitimate data packet, substitute (or otherwise mangle) its data segment and inject it back into the traffic stream. Such fraudulent packets will not be detected in transit. However, an intruder is *unable* to: i) create its own PR headers,

[7] For long PRs, the overhead may actually equal or exceed that of using a single public key-based signature!

[8] Strictly speaking, the difference between them is twofold: i) public key encryption is generally considered more resistant to attacks, and ii) public key signatures provide **non-repuditation** of origin (signatures generated using a shared key can not be unambiguously attributed to any member of the group that shares the key).

since it has no means of generating signatures, and ii) use any of the captured PR headers more than once, since transit PGs can detect duplicate and *stale* PR headers.

Finally, both network-layer and PR headers can be protected. In this case, the destination stub AD is assured of the addressing authenticity, while transit ADs are satisfied that packets flow along authorized PRs. We note that, insofar as transit ADs, the security of this approach gains little protection over signatures covering only the PR header because network-layer headers are "transparent" to PGs.

The above discussion leads to an important observation that a typical transit AD may only be concerned with traffic not violating its policies, i.e., it may not care about the authenticity of the data therein as much as it cares about the traffic flowing to, from or through appropriate locations. In other words, as long as the packet indexes a valid PR, it is authorized to travel to its intended destination.

To summarize our discussion, Table 4.2.5.2 illustrates the resilience of the various signature coverage methods to a range of hostile attacks.

	Replay	Data Segment modification	Addressing modification	PR header modification
None	N	N	N	N*
Network header	N	N	Y	N*
PR header	N	N	N	Y
PR header (timestamped)	Y	N	N	Y
Network and PR headers	N	N	Y	Y
Network and PR headers (timestamped)	Y	N	Y	Y
Entire packet	N	Y	Y	Y
Entire packet (timestamped)	Y	Y	Y	Y

Table 4.1: Resilience of signature coverage methods

One curious detail in the above table is the similarity between the cases of no signature protection and PR header signatures (without timestamps). These two methods differ only insofar as PR header modification is concerned. However, when a PR header does not include a timestamp, i.e., it is constant for the lifetime of a given PR, its contents are reduced to a PR handle alluded to earlier. A PR handle is, essentially, an identifier that indexes an appropriate PR table entry in each transit PG.[9] If the intruder modifies the unprotected PR handle of a data packet, when this packet reaches its next-hop PG, two outcomes are possible:

1. The modified PR handle can not be mapped into a valid PR entry and the PG is forced to discard the packet. In this case, we conclude that the attack did not succeed.
2. The modified PR handle is mapped into a valid PR entry and the PG switches the "fraudulent" packet to the next hop (specified in the entry).

The intruder's goal in this attack is to generate fraudulent packet traffic, specifically, to send packets along PRs that it has no authority to use. But, even if data packets carry signed PR headers (without timestamps), the intruder can achieve the same result by recording PR headers (along with the corresponding signatures) of genuine packets and composing any number of fraudulent packets by concatenating any recorded PR header with an arbitrary data segment. Consequently, as long as data packets do not carry *unique* PR headers, PR header signatures do not contribute to the security of the protocol.[10]

4.2.5.3 Signature Verification

Another factor contributing to both the security and the overhead is the question of who checks the signature and how often.

Endpoint If authenticating data in transit is prohibitively expensive, end-to-end data integrity similar to that in *Visa* protocol may be appropriate. Every data packet is signed at the source but is checked only at the destination. This

[9] See Section 4.3.4 below.

[10] The only benefit afforded by header signatures is the ability to detect non-malicious header modifications, e.g., those due to noisy lines.

approach has limitations, most notably the fact that an intruder located at some point along the route can modify data in each packet and the forgery will not be detected until the packet reaches the destination AD. This can result in unauthorized use of transit resources and inappropriate billing of the source. On the other hand, per packet latency is low and independent of the PR's length. This approach provides preventative control for stub ADs, but only detection-based control for transit ADs.

Full Transit In environments where security concerns outweigh the overhead of extra processing, the data portion of every packet is subject to forgery and must be checked (for authenticity and integrity) at each hop on its way to the destination. Every data packet is signed at the source and checked at each AD hop en route. The protocol for this class of environment has the highest overhead, commensurate with security requirements. The per-packet processing in this variant is similar to transit *Visa* protocol.[11]

Designated Transit If Endpoint exposes transit resources to excessive misuse, yet Full Transit is too expensive, the source AD can designate at PR setup time a specific transit AD to perform data integrity checks. Every data packet is still signed at the source, but only one transit and the destination check the signature. The positioning of the designated AD in the PR is important: having it too close to AD_{src} is almost equivalent to no checking at all, whereas having it too close to AD_{dst} is equivalent to the Endpoint variant. The designated AD can be reassigned from time to time in order to reduce the chance of its exploitation by an intruder.

Patterned Instead of each transit AD having to authenticate each packet, it may suffice to authenticate every m-th packet. In the simplest version of this patterned authentication scheme, AD_{src} would choose m at random from a locally defined range of values and then specify m during route setup. In this scheme only $1/m$ data packets are signed at the source and the same $1/m$ packets are checked.

Transit ADs would either accept or reject the proposed m. If all ADs accept the proposed value for m, then every AD will check data integrity of every m-th packet. If any AD does not accept m (if it is considered too large or too small) then the source and all other ADs must choose a different m. In return for reduced overhead, if the value for m is discovered by an intruder then *(m-1)/m* of the PR's bandwidth can be abused. Moreover, the synchronization inherent to this protocol implies that care must be taken to recover from lost and out-of-order packets.

Alternatively, instead of the source choosing m , every transit AD_i can choose its own m_i and elect not to disclose it. Or, a transit AD_i could choose to authenticate each packet with probability p_i. In this scheme all data packets are signed at the source, but only $1/m$-th (or $p\%$) of the packets are checked per AD hop. This method has the advantage of being flexible and robust in that coordination among transit ADs is not required.

An important disadvantage common to all *patterned* variants stems from the potentially large variance in delay between "checked" and "unchecked" packets. Applications expecting little variance in delay (e.g., packetized voice) may therefore suffer.

Round Robin This scheme achieves constant per packet overhead by using *round-robin* data authentication. Transit ADs take turn authenticating packets. In general, packet number K is authenticated by a PG in $AD_{[KmodM]}$ where M is the number of ADs in the PR. All data packets are signed at the source but only one check is done en route to the destination. Destination checking can be added for extra assurance at the cost of a single additional decryption by the destination. Moreover, unlike Source Patterned variant, lost and out-of-order packets can be accommodated easily. On the other hand, AD independence must be sacrificed due to the coordination required to set up the round-robin arrangement. While this approach benefits from fair sharing of encryption costs among transit ADs, it is only worth considering in cases when the number of transit ADs is large, i.e., the PR is long.

4.2.5.4 Preventing Replay of Data Packets

The final type of attack considered is the replay of data packets:

Type 4. *An intruder can replay previously recorded data packets, which can lead to unjustified charging and/or denial of service.*

[11] However, the Full Transit approach avoids the per-session setup dialog associated with visa acquisition.

There are other, more serious threats posed by malicious replay. However, we are concerned primarily with protecting network-layer resources; other replay attacks are assumed to be handled by the end-points. Also, we only need to protect against replayed packets within the life-span of the associated PR. After a PR expires or is closed all packets carrying the expired PR identifier will not be processed.

Two sources of replay deserve equal consideration. The first is accidental replay due to a misbehaving machine stuttering and generating replayed packets. The second is malicious replay due to an intruder intentionally replaying prerecorded packets in order to deny resources (or inflate costs) to the rightful owner. Neither kind of replay can be handled on a purely end-to-end basis because by the time a duplicate packet is discovered, the resources are consumed and associated charges are incurred, e.g., the bill reflects the replayed packet and the rightful user of the afflicted charge code can no longer obtain service due to an overdrawn account.

In some circumstances, the post facto approach of replay detection and cost recovery may be adequate. This includes auditing packet counts, setting a limit on the number of packets that can use a PR and other ad hoc methods. However, in sensitive environments, more aggressive prevention is required, albeit at some cost. Since our goal is to analyze the implications of secure control of transit traffic, we present a method for preventing replay.

There are two basic approaches for countering replay attacks: i) nonce identifiers, and ii) timestamps.[12] The main disadvantage of using nonces is the difficulty in their verification. In particular, each relevant entity (each PG, in our case) needs to keep a complete history of past nonces which makes the verification inefficient. Timestamps are much better suited for this application. First, clocks need not be continuously synchronized between the source and the transit PGs. This is because a PR setup packet is timestamped; its timestamp can be used as a *lower-bound* for subsequent data packets in all intervening PGs. (It can also serve as an indication of the clock skew). Furthermore, if intervening PGs maintain a more *current* lower-bound timestamp (t_{lower}), opportunities for replay can be reduced further.

Consider the following protocol:

1. When a PR is issued, the originator PG timestamps the PR setup packet, and distributes the timestamp, t_{setup}, in a secure fashion to all intervening PGs in transit ADs. All transit PGs initialize their t_{lower} values for this PR to t_{setup}.
2. When a data packet is sent, the originating (first-hop) PG, timestamps its PR header. (Let t_{data} denote this value).
3. When this data packet reaches a transit PG, its PR header is examined and the t_{data} is compared to t_{lower}. Three outcomes are possible:
   (a) $t_{data} < t_{lower}$. The difference between the two values is examined. If it is small, i.e., less than some (locally defined) threshold, $delta_t$, the packet can be forwarded. Otherwise, the packet is discarded.
   (b) $t_{data} > t_{lower}$. In that case, the packet is forwarded and t_{lower} is set to t_{data}. Of course, a PG may get suspicious if the difference is too large.
   (c) $t_{data} = t_{lower}$. This can occur when two successive data packets belonging to the same PR stream carry the same timestamp. To distinguish between such packets, it would be necessary to keep additional information, e.g., a packet signature, for the last data packet processed. However, it is desirable for the clock rate to be at least as fast as the maximum packet rate. This would preclude duplicate timestamps on data packets.

This protocol prevents most, but not all, replay attacks. In order to prevent all replay attacks, $delta_t$ values must be set to zero in all transit PGs, which would essentially disallow any out-of-order data packets. This is a choice that will not be practical for internetwork environments where out-of-order packets are a frequent occurrence.

4.3 Protocol Description

Based on the above discussion of security issue, this section describes the secure policy routing protocol. The following design choices are made:

- All PR setup control packets are protected by signatures computed with the private key of the originating entity.

[12] It can be argued that timestamps of sufficient width and granularity are nonces as well. However, a *true* nonce is a randomly chosen number that is hard to predict.

- A secret key is distributed to all intervening PGs as part of PR setup. (This key is subsequently used to compute data packet signatures). As described in Section 4.2.5.1, the key is enciphered N times, once for each intervening AD using that AD's public key.

- Data packet signatures can be computed a range of signature methods. The particular method, is negotiated at the time of PR setup.

- Signatures cover either the entire packet or only the (timestamped) PR header. The choice is made at the time of PR setup.

- A transit ADs is free to choose on a per PR basis whether or not to authenticate data packets.

- All packets (control and data) are timestamped at the origin. No two packet of the same origin and type can bear identical timestamps. Clocks are assumed to never run backwards.

After describing protocol participants and packet handling rules in the next two sections, protocol details are presented in Sections 4.3.3 and 4.3.4.

4.3.1 Participants

Policy Gateways (PGs) are the principal participants in both PR setup and packet forwarding protocols. In PR setup, a Route Server (RS) is initially consulted for a route, however, the PG-RS interaction is strictly intra-AD, which makes it of little interest in the context of our discussion. Unlike *Visa* protocol, end-systems need not be involved in the protocol; PR setup and PR-based packet forwarding are completely transparent to them.

With respect to a single PR, there are three types of PGs:[13]

- **Originator** is the first PG in a PR. Its task is to associate outgoing packets with appropriate PRs. When the originator receives a data packet it first looks up the source and destination network addresses in its end-system table. Each entry in this table points to a PR entry in the PR table. Using the information from the PR table, the originator constructs a PR header and attaches it to the packet.

- **Transit** is any intermediate PG in a PR. A transit PG's main task is to route data packets to next-hop adjacent PGs. Each data packet carries a PR handle (see below) which a PG uses to lookup a corresponding entry in its PR table. Each entry contains (among other things) the next-hop PG address.

- **Target** is the last PG in a PR. Its task is complementary to that of the originator PG, i.e., it has to remove PR headers from data packets and forward them to the destination end-systems.

In summary, the requirements for a PG are as follows:

- maintenance of active PRs

- maintenance of local PTs (for validating SETUP packets)

- a means for computing and verifying public key signatures (for all control packets)

- support for a range of signature methods

The above is in addition to the more traditional routing-related requirements, e.g., monitoring the reachability status of adjacent PGs:[14]

[13] See also Figure 4.1.

[14] Only PGs in stub ADs are expected to compute routes. Also, since local policy serves as one of the inputs to route computation, ADs are expected to use different procedures for route selection. (See [84, 28] for more information).

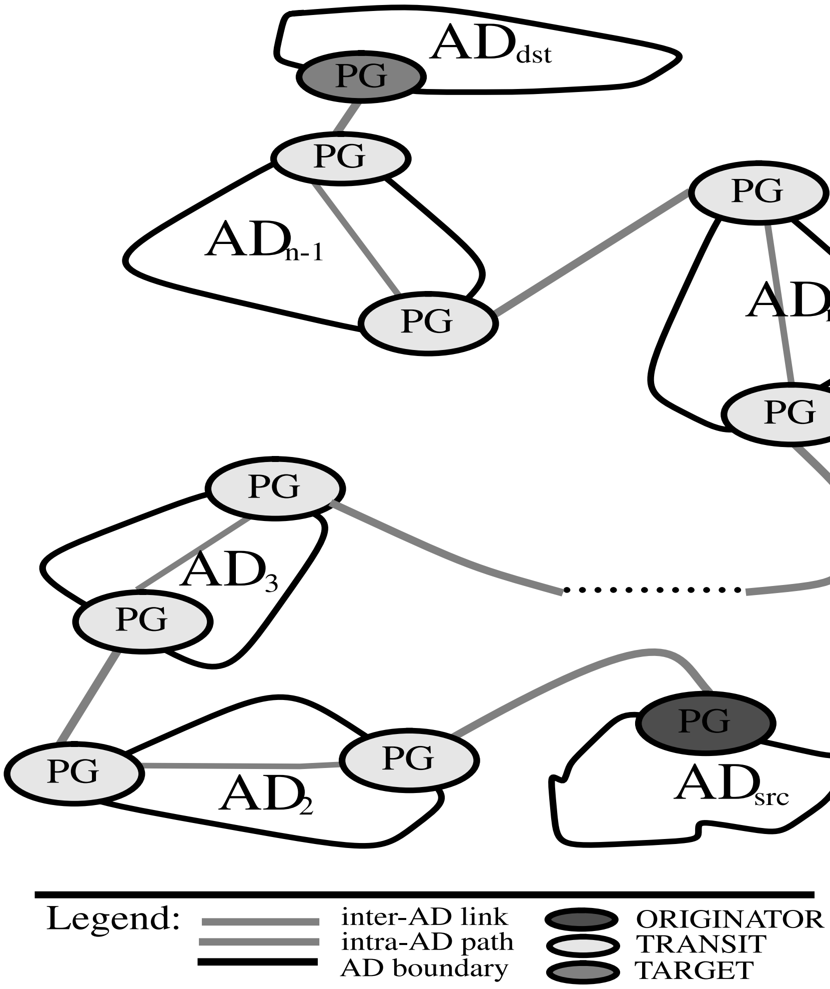

Figure 4.1: Policy Gateways in IDPR

4.3.2 Packet Handling

Protocol participants distinguish between two types of packets: control and data. Control packets are used for route setup and maintenance, dissemination of link-state PT updates and PG status messages , i.e., all packets intrinsic to IDPR. Of these, only control packets pertaining to route setup and maintenance are relevant to our discussion.[15] The balance of the internetwork traffic consists of data packets.

Control and data packets differ in the manner they are processed by the PGs. Control packets are transmitted reliably. Each control packet is acknowledged on a hop-by-bop basis (i.e., between adjacent PGs). If a control packet is not ACK-ed within a predetermined interval, the sender retransmits. After attempting n (n is a locally defined value) successive retransmissions, the sender PG *gives up* and informs the control packet's source of the failure to deliver. All control packets are uniquely identified by the combination of: i) packet type (SETUP, REFUSE, ACCEPT, etc.), and ii) PR handle which refers to a particular PR. This combination is *guaranteed* to be unique, i.e., no two control packets of the same type carry the same PR handle. Every control packet is signed with the secret key of its originator, so as to

[15]For a detailed description, see IDPR Architecture and IDPR Specification documents [50, 84].

allow all interested parties to authenticate its contents. After receiving and authenticating a control packet a PG replies to the previous hop PG with an ACK packet containing the identifier of the control packet being acknowledged. An ACK is used only between adjacent PGs. It is protected by a signature of its sender and contains the packet type and the PR handle of the control packet being acknowledged (see Figure 4.3.2). It does not, however, include the address or any other identifier of the sender. This is because the intended recipient of an ACK can unambiguously associate a PR handle with an adjacent PG which, in turn, determines the key to be used for signature verification.

Packet type (=ACK)
Ack-ed packet type
PR handle
SIG_{sender}

Figure 4.2: ACK packet

Unlike control packets, data packets are not acknowledged at every PG hop. Their transmission is inherently unreliable, since a PR may traverse a number of lossy datagram subnets. One similarity between control and data packets is that both carry PR handles. PR handles allow transit PGs to associate data packets with PR table entries and route data packets according to the information stored therein (i.e., the next hop PG).

4.3.3 PR Setup

The protocol begins with a data packet arriving at a PG in AD_{src}. When this PG discovers that it has no active PR for the source-destination end-system pair in that packet, it contacts a local RS and asks for a new PR to the destination AD. (Hereafter, the requesting PG is known as the originator.) RS replies with the full path:

$$PR = [SEGMENT_1, SEGMENT_2, ..., SEGMENT_N] \tag{4.2}$$

Each segment contains: $SEGMENT_i = [AD_i, VALID_i]$ for $(0 < i <= N)$

where $VALID_i$ is the information that AD_i can use to validate the PR (e.g., a list of applicable Policy Terms).[16]

Now, the originator composes a new PR SETUP packet. Included in the packet is the unique timestamp, TS_{src}. SETUP also contains the *Authentication-type* and *Expiration* fields which serve the same purposes as their respective counterparts in *Visa* protocol, i.e., authentication method for subsequent data packets, and, PR expiration conditions, respectively.

A SETUP packet is protected by the signature of its originator:

$$SIG_{src} = [F_{hash}(P_{setup})]^{DK_{src}} \tag{4.3}$$

where DK_{src} is the secret (signature or decryption) key of AD_{src} and one-way hash function (such as MD4) and a strong encryption function (such as RSA), the signature is sufficient to maintain the integrity of the PR SETUP packet. Freshness is attained by TS_{src} within the SETUP packet.

In order to allow for the authentication of subsequent data packet traveling along this PR, the originator PG needs to generate a new data signature key, K_{dsig}, for use in whatever per-packet variation is used.[17] K_{dsig} must be communicated in secret to each AD_i. This requires that AD_{src} encrypt K_{dsig} N times, i.e., compute $E(K_{dsig})^{EK_i}$ for all AD_i. The resulting PR SETUP packet is depicted in Figure 4.3.

As the PR setup packet propagates along the route, each transit PG in the PR performs the following checks:

1. Validates the timestamp, TS_{src}.

[16]Note that a PR carries no information regarding individual end-system pairs. This is because all PRs are initially validated on the basis of AD_{src}, AD_{dst} and usage conditions. If a transit AD's policy terms are end-system specific, that AD can verify end-system addresses at the time of packet forwarding.

[17]Actually, K_{dsig} is not necessarily a key, per se. It is a secret quantity to be used in data signature computation.

Packet type (=SETUP)	
AD_{src}	TS_{src}
Authentication-type	
Expiration	
PR	
$E(K_{dsig})^{EK_1}$	
$E(K_{dsig})^{EK_2}$	
...	
$E(K_{dsig})^{EK_N}$	
SIG_{src}	

Figure 4.3: SETUP packet

2. Recomputes and verifies SIG_{src}.
3. Obtains K_{dsig} by decrypting $E(K_{dsig})^{DK_i}$.
4. Validates the PR by checking the $VALID_i$ field in the corresponding route segment field.

At this point, if the setup packet passes all checks, the transit PG is assured that: (i) the PR is valid, i.e., does not violate local policy, (ii) the PR is authentic, i.e., issued by a recognized entity; and, (iii) the PR is *fresh*, i.e., issued recently. In other words, each AD_i is protected against attacks of **Type 1** and **Type 2**.

Finally, the PG creates a new table entry for the new PR and computes the next hop PG using the next hop AD specified in the next PR segment.

If the setup packet fails one of the above checks, the transit (or target) PG in question informs the originator via a REFUSE packet. (See Figure 4.4). Unlike a SETUP packet, a REFUSE bears no timestamp of its own, since there can be at most one REFUSE packet for a single SETUP packet. Instead, a REFUSE packet contains the timestamp and the AD_{src} field of the corresponding SETUP packet which helps the originator in identifying the exact SETUP packet being rejected. A REFUSE packet is protected by a signature of its creator, so that all interested parties can verify that it has been generated by a recognized entity, i.e., by one of the PGs in the route. (SIG_i in Figure 4.4 refers to the signature generated by the i-th hop transit AD). Upon receipt of a REFUSE all PGs tear down the state information pertaining to the PR mentioned therein and forward the REFUSE to the next PG in the direction of the originator PG.

Packet type (=REFUSE)	
AD_{src}	TS_{src}
AD_i	Reason
SIG_i	

Figure 4.4: REFUSE packet

After a SETUP packet successfully passes through all transit PGs, it finally reaches the last-hop, target PG. After validating the PR, the target PG informs the originator of the setup completion by means of an ACCEPT packet (Figure 4.5). An ACCEPT packet carries only the PR handle. Unlike a REFUSE, it does not need to include the identity of the originating AD. This is because an ACCEPT can be generated only by a target PG in the destination AD.

As an ACCEPT propagates through each transit PG on its way to the originator, it enables the (hereto dormant) corresponding PR table entry. In other words, an ACCEPT signifies that the PR has been fully authorized by all parties involved. When the ACCEPT finally reaches the originator PG, data packets can, at last, start flowing along the newly established PR.

Failure to receive an ACCEPT within a pre-determined period of time, causes the originator to generate and re-send a new SETUP packet for the same PR. The new SETUP is essentially the same as the original one except for the TS_{src}

Packet type (=ACCEPT)	
AD_{src}	TS_{src}
SIG_{dst}	

Figure 4.5: ACCEPT packet

field which must reflect the current clock reading. This is done so that all parties involved can distinguish between successive setup attempts for the same route. For example, if the originator times out prematurely and generates a new SETUP while the ACCEPT for the original SETUP is still en route, it will be able to decide unambiguously that the ACCEPT corresponds to the original SETUP packet, not the retransmission.

4.3.3.1 PR Setup Summary

In summary, PR setup involves the following steps:[18]

1. $AD_{src} \Longrightarrow AD_2 : SETUP$
2. $AD_2 \Longrightarrow AD_{src} : ACK^2_{src}$

   ...

3. $AD_i \Longrightarrow AD_{i+1} : SETUP$
4. $AD_{i+1} \Longrightarrow AD_i : ACK^{i+1}_i$

   ...

5. $AD_{n-1} \Longrightarrow AD_{dst} : SETUP$
6. $AD_{dst} \Longrightarrow AD_{n-1} : ACK^{dst}_{n-1}$
7. $AD_{dst} \Longrightarrow AD_{n-1} : ACCEPT$
8. $AD_{n-1} \Longrightarrow AD_{dst} : ACK^{n-1}_{dst}$

   ...

9. $AD_i \Longrightarrow AD_{i-1} : ACCEPT$
10. $AD_{i-1} \Longrightarrow AD_i : ACK^{i-1}_i$

    ...

11. $AD_2 \Longrightarrow AD_{src} : ACCEPT$
12. $AD_{src} \Longrightarrow AD_2 : ACK^{src}_2$

4.3.4 Packet Forwarding

In order to make use of an existing PR, the originator PG must be able to supply information necessary to associate each data packet with a specific PR. This information is placed in the PR header:

$$PR_{hdr} = [AD_{src}, TS_{src}, TS_{packet}, DSIG] \tag{4.4}$$

The combination of AD_{src} and TS_{src} forms the *PR handle* mentioned above. Transit PGs are able to look up the appropriate PR in their tables using the PR handle, $[AD_{src}, TS_{src}]$, as a look-up key. TS_{packet} is an optional packet-level timestamp used for detecting old and out-of-order data packets. Finally, $DSIG$ is the (also optional) packet signature used to protect against **Type 3** attack. $DSIG$'s verification is subject to the authentication method agreed to upon PR setup.

[18] The notation $A \Longrightarrow B : PACKET$ means "*A sends PACKET to B* ".

4.4 Security Analysis

We now discuss the security of the proposed protocol considering separately the route setup and packet forwarding phases.

4.4.1 PR Setup

Successful PR setup involves the transmission of a SETUP packet from the originator PG along the PR and the subsequent transmission of an ACCEPT packet from the target PG along the inverted PR.

4.4.1.1 SETUP Processing

A SETUP packet is signed with the secret key of the originator and timestamped with a unique timestamp. This allows any transit PG to establish that the SETUP packet i) was sourced in AD_{src},[19] ii) was not modified in transit, and iii) was generated *recently.*

In addition, a transit PG can attempt to verify that the PR being installed does not violate local policy by checking the SETUP packet for compliance with the PTs referenced therein. A PT can restrict any one of the following (see expression 4.1):

1. Source AD
2. Source end-system
3. Destination AD
4. Destination end-system
5. Entry AD (i.e., previous AD hop)
6. Exit AD (i.e., next AD hop)
7. User Class, Type of Service and other characteristics [84]

Of the above, only items (1) and (6) can be effectively verified. Source AD is established as part of validating the SETUP packet's signature. Exit AD is established by the combination of: i) checking the next AD segment within the PR and ii) physically forwarding the SETUP packet to that next hop AD and receiving ACK in response.

Although, in theory, PTs can be used to restrict source or destination end-systems, such restrictions are generally impractical. The reason for this is twofold. First, it is impractical for transit ADs to be concerned with large numbers of objects at the granularity of end-systems. Second, (also because of scale) it is difficult to authenticate traffic on the basis of signatures generated by individual end-systems, since many end-system pairs can be associated with a single PR. If a transit AD still decides to include end-system restrictions in its PTs, it has to **trust** the end-point PGs to issue PRs only to appropriate end-systems. Nevertheless, an AD might restrict access for a small number of select end-systems using this method.

Verification of the previous AD hop is complicated by the fact that a SETUP packet does not reflect the actual traversed route (which may be different from the one it carries). In other words, a PG in a transit AD receives a SETUP packet which specifies some adjacent AD as the previous hop. However, this gives no assurance of the SETUP packet having actually passed through the adjacent AD in question. For example, a link connecting two adjacent ADs may also be shared by other entities, any one of which could have forwarded the SETUP packet.

Even the destination AD is not verifiable at SETUP processing time. A transit AD can merely conclude that the last AD hop in the PR is the *apparent* destination. Stronger conclusion can be made only upon receiving a corresponding ACCEPT packet (see below). For transit ADs, this uncertainty may serve as grounds for not accepting any data packets for a PR before receiving ACCEPT.

Finally, verification of User Class, Type of Service and other restrictions based on traffic characterists is not possible at PR setup time. For example, if a PT contains User Class restrictions, the originator can lie about the User Class

[19] For simplicity, we assume that all PGs in the same AD share the same public-key pair.

of the intended traffic source. Unfortunately, a transit AD has no means of detecting such abuse. A sole exception is the Date/Time restriction which specifies the time intervals when a certain PT can be used. It is trivial to establish compliance with a Date/Time restriction at SETUP processing time.

4.4.1.2 ACCEPT Processing

An ACCEPT packet is signed by the target PG (in the destination AD) and bears the PR handle of the corresponding SETUP packet. This allows all transit PGs and the originator PG to verify its origin and contents. Timeliness is derived from the presence of the PR handle, i.e., since the SETUP bearing the same PR handle was observed recently and the PR handle is unique, the ACCEPT packet is also unique, hence, it must have been recently generated.

With respect to the overall *validity* of a PR, a *bona fide* ACCEPT packet adds the following:

- By virtue of checking the packet signature, the destination AD is finally verified.
- The ACCEPT packet itself implies that the destination AD validated the PR.

Little else can be derived from an ACCEPT. We observe that the beliefs derived from a *bona fide* ACCEPT packet fall short of assuring the originator that *ALL* transit ADs (PGs) have validated the PR and established the necessary state. Since the ACCEPT is signed by the target PG (and carries within it the PR handle of a recently sent SETUP packet), the originator PG is assured that the SETUP packet *somehow* reached the AD_{dst} and that AD_{dst} validated the PR. Nonetheless, there is no direct evidence to support the claim that the SETUP passed through (and was approved by) all transit ADs. In other words, either (or both) SETUP or ACCEPT packets may have traveled along a route *different* from that specified in the PR. This type of anomaly appears unlikely as it requires active interference on the part of one (or more) transit ADs.

In order to derive stronger beliefs about setup completion, the originator must be assured by each transit hop individually. This can be accomplished if we require each transit PG to sign the ACCEPT as it travels towards the originator. Then, upon verifying all of the trailing signatures, the originator can be satisfied that all transit AD have, in fact, authorized the PR and established the necessary state. The main drawback of this mechanism is, of course, the additional overhead due to signature computations and verifications (as many as there are ADs in the PR). Moreover, the added complexity of this mechism is only worthwhile if subsequent data packets are treated similarly.

4.4.2 Packet Forwarding

As discussed in Section 4.2.5, unless a strong authentication method is used in conjunction with replay prevention, a transit PG can derive very little insofar as the *goodness* of a data packet. If a packet's PR handle indexes a valid PR in the PR table, the only conclusion that a transit PG can make is that the traffic flow which this packet purports to be a part of, has been previously authorized. Stronger beliefs are unattainable without a stronger proof of the relationship between the packet and the PR that it indexes.

Without any authentication, only packets bearing invalid PRs can be detected. If only PR header is authenticated (without replay prevention), no additional security is gained. Since all packet belonging to the same PR carry the same *"authenticated"* PR header, an intruder can duplicate this header indefinitely (for the lifetime of the PR) attach its own data and inject the resulting traffic into the stream unnoticed!

If replay prevention is used, each PR header is distinct by virtue of carrying a unique timestamp (hence, a unique PR header signature as well). A transit PG can establish the origin and timeliness of each data packet,[20] however, the authenticity of the data segment is still suspect. Since the intruder is no longer able to duplicate PR headers, the only venue of attack left is to mangle existing data packets (e.g., substitute data segments). Still, this level of protection is relatively inexpensive and quite effective considering that it makes it impossible for the intruder to i) replay old packets and ii) create more packets than generated by the true source. The latter protects against inflated charges and packet storms.

Finally, akin to SETUP and ACCEPT packets, data packets convey no information about the actual route taken. Moreover, because data packets are not acknowledged on hop-by-hop basis, neither previous nor next AD hop can be verified.

[20] As described in Section 2.2.3, timeliness depends on the particular Δ_T used.

4.5 Assessment and Cost

Our purpose in this section is to investigate bounds on achievable data rates with the security schemes described above. Previous work in the area of performance and cost evaluation of secure protocols [27] identifies four important overhead contributors (listed in the order of magnitude):

- Per Packet Signature
- Increased Packet Length
- Setup Overhead
- Other Additional Per Packet Processing

In the remainder of this section we analyze each of the above contributing factors in several variations of the general scheme.

4.5.1 Packet Signatures

As discussed earlier in this chapter, per packet signature costs are largely dependent upon the particular variation of packet signature checking, signature computation method, and signature coverage. Previous results in measuring *Visa* protocol overhead [32, 27] suggest that, whenever encryption or encryption-based signatures are used, the overhead becomes quite noticeable. In Chapter 5, we illustrate experimental results using non-cryptographic packet authentication techniques.

Replay prevention can be used independent of the data authentication method. The cost of replay prevention amounts to one additional PR header field (32-64 bits depending on the timestamp granularity) and several instructions for implementing the protocol described in Section 4.2.5.4.

4.5.2 Costs Due to Increased Packet Length

Increased packet length is incurred by the PR header carried in every data packet. Recall (see 4.4) the PR header carries the AD identifier of the source and the timestamp of the original setup packet. Assuming 32 bits for AD_{src} and (at most) 64 bits for TS_{src}, these two fields add up to 96 bits. Depending on the requirements for data authentication and replay prevention, PR header may include a packet timestamp and a data signature. The former is a 64-bit quantity. The length of the data signature depends on the particular signature method and can vary between 64 bits (e.g., DES MAC) to 128 bits (e.g., MD4). Thus, we can anticipate that the length of the PR header will range between 96 and 288 bits. Previous measurements of *Visa* protocol implementations [27] show that this overhead ranges from 20% for small (e.g., 16 bytes of user data) packets to less than 4% for larger, e.g., 1Kbyte, packets. (The length of a visa header is roughly the same as that of a PR header).

4.5.3 Setup Overhead

PR setup is accomplished by composing and sending a packet containing the entire PR as described in Section 4.3. The costs include:

- N conventional encryption operations by AD_{src} to encrypt K_{dsig} for all intervening AD_is, if data packet authentication is desired.
- A hash function computation over the entire PR setup packet followed by a single public key signature computation over the hash function value.
- N hash function computations and N public key signature verifications for verifying the setup packet signature *en route* (one at each AD_i).
- N decryption operations by each AD_i to decrypt K_{dsig}. However, this can be done after the setup packet is forwarded to the next hop and so does not contribute directly to the setup latency.

The above is the worst case scenario. Some ADs may not care to authenticate PRs upon setup. Furthermore, if PR authentication and integrity requirements (or lack thereof) are expressed in a transit AD's PTs, the source AD can avoid unnecessary signature computation and reduce the setup overhead.

Measurements of PR setup overhead are presented in Chapter 5.

4.5.4 Other Per Packet Processing Costs

Additional (other than encryption) processing costs are incurred mainly by the added logic in routers for processing of PR-based packets, in particular, table lookups.

Each PG maintains a PR table where each entry corresponds to an active PR. Not all PGs look up this table directly. A PG in a stub AD acts mainly as an *originator* and does not search its PR table; instead it searches the end-system table with source-destination end-system addresses. Each entry in the end-system table points directly to the corresponding PR table entry. Transit PGs, on the other hand, have to perform direct PR table lookups. PR tables in transit PGs are anticipated to be several orders of magnitude larger than those in stub PGs. Therefore, lookup costs are of concern. However, if entries are hashed on PR handles using a *uniform* hash function [41], the cost of a PR table lookup can be expected to be negligible (on the average).

In addition to authentication and lookup costs, there is also the cost of PR maintenance. For each data packet switched, a corresponding PR table entry is updated. This includes incrementing several counters and refreshing two timestamp values (see Chapter 5). The cost of these operations is greatly overshadowed by other overhead-contributing factors.

4.6 Conclusions

Transit control mechanisms are needed by interconnected ADs to retain their autonomy in setting and enforcing policy while still achieving desired connectivity. The problem of interconnecting Administrative Domains and navigating across them is important because the policies in question concern control of resource access and usage. Moreover, the security of the transit policy enforcement is important, especially, in sensitive environments. On the other hand, the security measures, as can be expected, take a toll in overall system complexity and performance.

The purpose of this chapter was to investigate the design of transit policy enforcement mechanisms for sensitive environments, to analyze their security and to evaluate the performance overhead. We began by attempting to extend existing network access control methods to the more general transit environment. Subsequent evaluation identified several basic deficiences of the resulting extension. These deficiencies motivate the integration of policy enforcement into the internetwork routing, route computation, and packet forwarding protocols. We used Inter Domain Policy Routing (IDPR) as the foundation for the protocol design. After identifying potential security threats we presented the secure Policy Routing protocol and analyzed its security and performance impact.

Security of the transit control mechanisms should be approached from an integrated perspective. It exists in the context of end-system and network access control, hence, the division of labor deserves careful consideration. In this chapter we proposed that Policy Routing be equipped to prevent unauthorized use of network resources, and exert control over routing across AD boundaries. Network access controls are responsible for finer grain control (e.g., on an end-system, as opposed to AD basis) and end-systems for protection of non-network resources. The proposed mechanisms were designed to support inter-operability across ADs with heterogeneous policies to the extent that their combined policies allow.

Chapter 5

Experimental Results

In Chapters 3 and 4 we presented the design of stub and transit policy enforcement mechanisms, respectively. In order to support our design choices and to demonstrate the feasibility of deployment, this chapter describes the experiments conducted with prototype implementations of the proposed protocols. The purpose of the experiments was twofold: i) to verify the functionality of the implementation, and ii) to measure the overhead imposed by the respective protocols.

This chapter is organized as follows. We begin with a brief description of the experimental environment in the next section. In Sections 5.2 and 5.3 we describe our experiments with *Visa* and IDPR protocols, respectively.

5.1 Experimental Platform

For the purposes of functionality and performance testing, both *Visa* and IDPR protocols were implemented within the SunOS 4.1 environment which is substantially similar to the Berkeley 4.3 UNIX. Sun *SparcStation 1+* workstations were used as the hardware platform throughout the experiments. (*SparcStation 1+* scores 15.8 MIPs on the well-known *Dhrystone benchmark*). All experimental workstations were equipped with 8 to 12 Mbytes of main memory.

The workstations directly participating in the experiments were interconnected with Ethernet segments to form a topology depicted in Figure 5.1. However, a number of other workstations were used as *secondary*, non-essential nodes (i.e., end-systems in IDPR testing and visa-hosts in *Visa* protocol experiments).

In all experiments, MD4 was used as the integrity mechanism for all packet traffic. Data packet authentication in both *Visa* and IDPR protocols was performed using the secret prefix method in conjunction with MD4 as described in Appendix A.

Signatures based on public-key encryption were not implemented. Also, we did not implement secure key distribution which is part of both connection setup in *Visa* and PR setup in IDPR. The main reason for this is the apparent lack of fast public key encryption implementation. Unfortunately, existing state-of-the-art public-key software implementations are excruciatingly slow. Even the fastest RSA hardware can only attain a meager 2-4 Kbits/second throughput [19]. Faster, scaled-down variants of RSA have been proposed [51]. For example, Privacy-Enhanced Electronic Mail specifies an RSA variant where signature verification costs significantly less than signature computation. This is achieved by using a large (e.g., 512-bit) secret exponent in conjunction with a very small (e.g., 8-bit) public exponent. Although signature verification becomes much cheaper (on the order of milliseconds), the cost of signature computation remains quite high (e.g., 4-5 seconds for a 512-bit modulus) [51, 36]. Future versions of *Visa* and IDPR protocols will take advantage of such methods as they become available.

5.2 Visa Experiments

In its previous incarnation [27, 61], *Visa* protocol was implemented within the network layer which required modifications to IP. All visa-related information was carried within IP option fields. This had several unpleasant side-effects: i) the visa option fields exceeded 40 bytes which is the maximum IP option length, and ii) IP options were not transparent to IP routers (even those that do not implement *Visa* protocol). The former limits the length of visa-related data and control messages, while the latter contributes to increased processing delay at all (including non-visa) IP routers. In