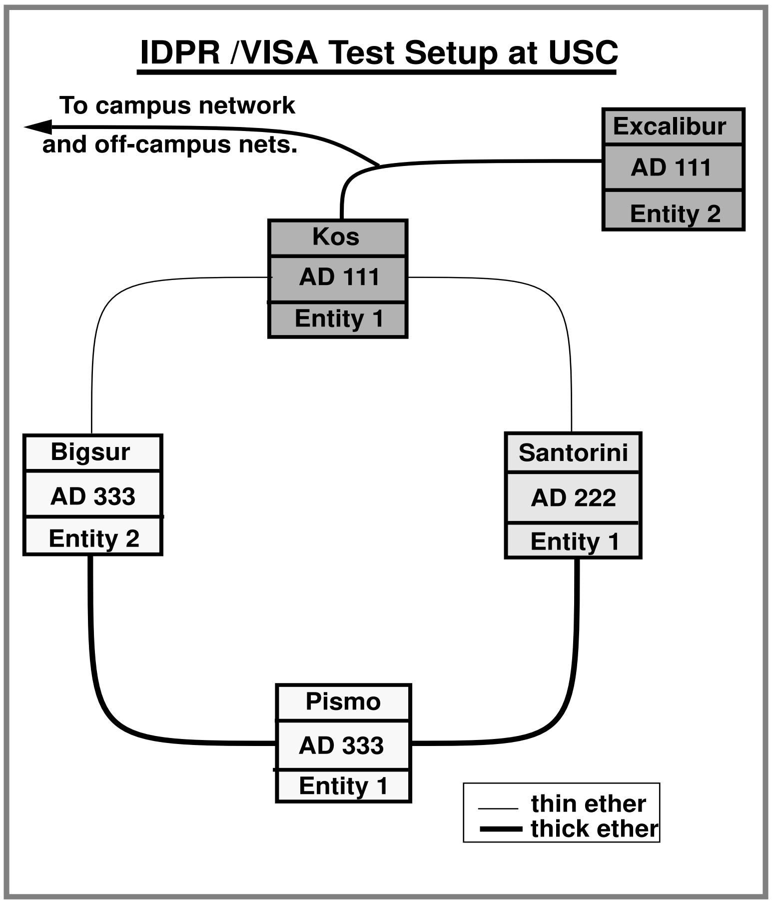

Figure 5.1: Laboratory Test Environment

contrast to its predecessor, the current version is implemented as a self-contained protocol entailing no modifications to other protocol software.

All visa control packets are processed inside the kernel. This speeds up connection setup (as compared to out-of-kernel implementations [61]). Recall that connection setup is the most "overhead-intensive" phase of *Visa* protocol. It involves the exchange of at least two inter-AD packets: VISA-REQUEST from ACS_a to ACS_b and VISA-GRANT from ACS_b back to ACS_a. Moreover, at least four inter-AD packets are exchanged (one HOST-REQUEST, and three VISA-GRANTs).

For the purpose of measuring the overhead we define connection setup time as the interval between the arrival of a REDIRECT packet from a visa-router to the arrival of a corresponding VISA-GRANT from the ACS (as shown in Figure 5.2). We measured the setup time for directly connected ADs as well as for ADs separated by one transit AD hop. For the directly connected case, the timings ranged between 100 and $180ms$ (averaging around $130ms$). In the case of one transit AD hop, the timings ranged between 110 and $200ms$ (averaging around $150ms$). These results show that the overhead is still quite high. However, we expect that performance-tuning of the code should be able to cut these numbers down by about a third.

Encapsulation of data packets is performed by all protocol participants. Normally, all outbound data packets are passed from the transport protocol to IP where each packet is prepended with an IP header and handed over to the interface output function. *Visa* encapsulation is interposed between IP and the interface driver. (See Figure 5.3). This is done transparently so that neither IP nor any higher-layer protocol is aware of the encapsulation. Encapsulating a packet requires the following steps:

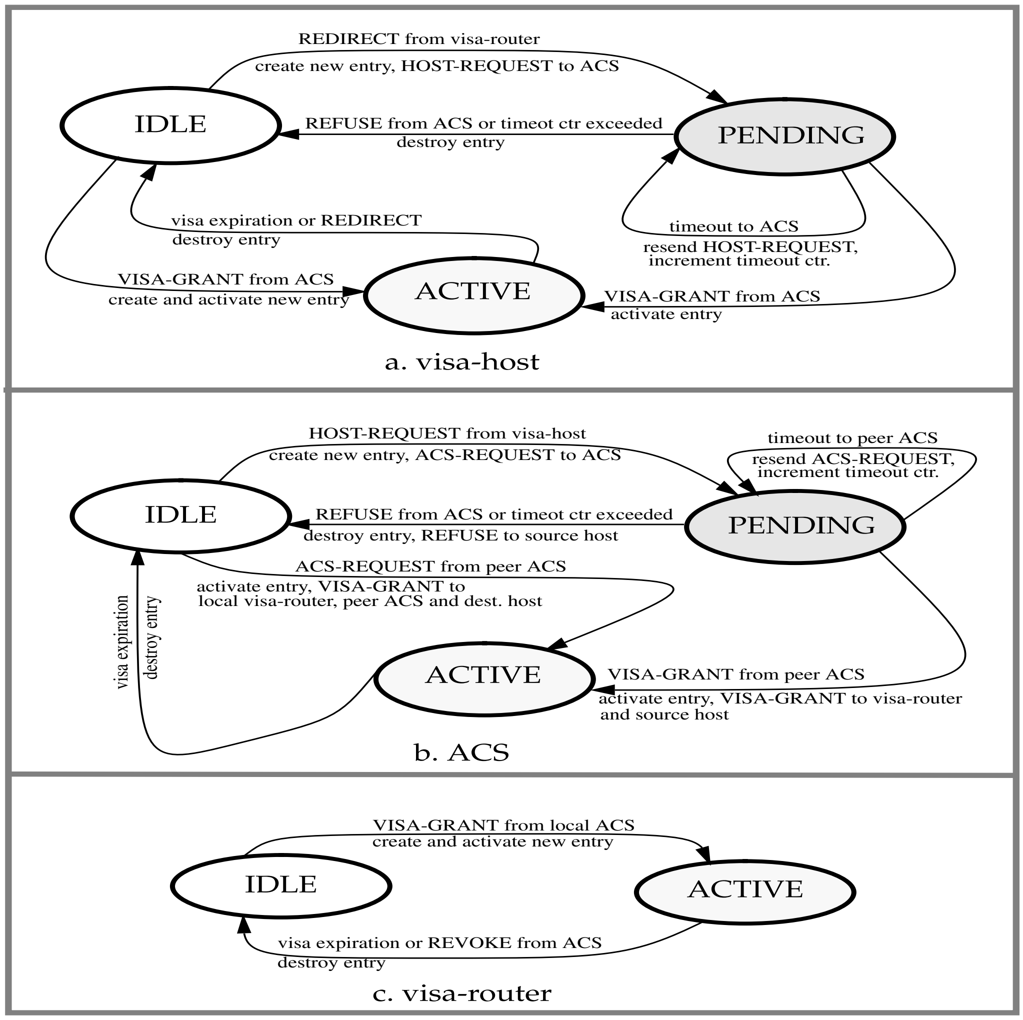

Figure 5.2: Visa State Diagrams

1. Visa-table lookup using $[SRC, DST, PROT, ToS]$ fields extracted from the IP header.[1]
2. Visa-stamp computation.

Packet switching at visa-routers involves the following operations:

1. Visa-table lookup with a visa identifier attached to each packet
2. Visa-stamp verification (i.e., integrity/authentication check)
3. *Light* bookkeeping, i.e., recording time of packet arrival, updating resource usage meters, etc.

We measured the overhead incurred by data packets flowing across previously established visa connections. The overhead measurements are derived using the ICMP Echo protocol [74]. In this protocol, a packet travels from the source to the destination end-system where it is immediately reflected back to the source. The results in Table 5.1 represent the the round-trip times (in milliseconds) for packets traveling between directly connected ADs. In other words, each packet travels across three ethernet segments: i) H_a to GW_a, ii) GW_a to GW_b, and, iii) GW_b to H_b where H_a refers to *pismo*, GW_a to *bigsur*, GW_b to *kos*, and, H_b to *excalibur* as depicted in Figure 5.1.

[1]Unfortunately, this has to be done for all packets, even those that may not require a visa.

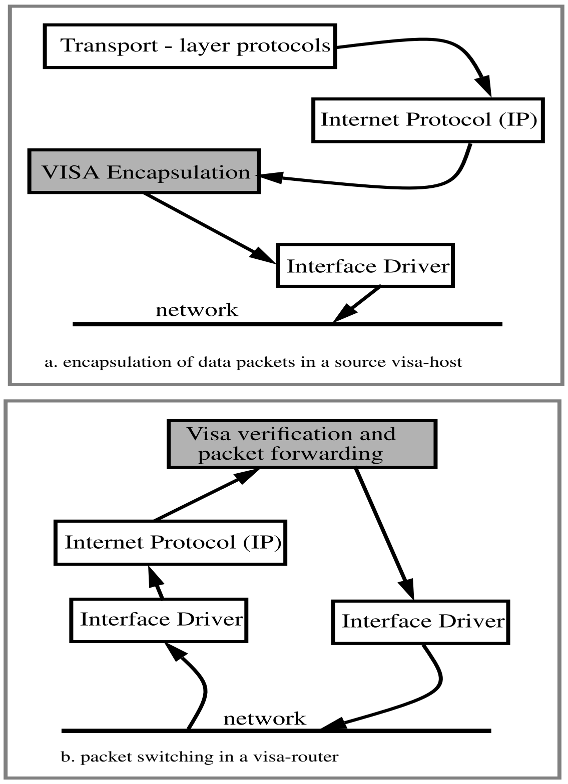

Figure 5.3: Visa encapsulation and packet switching

The first row shows the timings with *Visa* protocol inactive. The second row shows the timings for the same packet traffic, but, using *Visa* protocol without any integrity checking. These timings help us in identifying purely protocol-related overhead. The third row shows the results for *Visa* protocol with full integrity checking.

Packet size (bytes)	16	64	256	512	1024
No *Visa*	5	5	8	10	15
Visa w/o integrity	6	6	8.5	11	16
Visa with integrity	7.5	8	11	14	20

Table 5.1: Visa connection setup overhead (in ms)

First and foremost, these numbers clearly illustrate the relatively low cost of encapsulation and switching of visa packets. Another comforting aspect of our results is what appears to be insignificant additional overhead imposed by packet authentication and integrity checking. This may be due to the nature of MD4; since it is implemented entirely in software, data rates are constant independent of the packet size. (Hardware devices tend to offer much lower rates for small data units; mostly because of the fixed device initialization costs).

5.3 IDPR Experiments

In this section we describe the experiments with the IDPR architecture based on the initial prototype implementation. The IDPR architecture was originally put forward by the Internet Inter-Domain Policy Routing Working Group (IDPR-WG)[2] of the Internet Engineering Task Force [50, 84] The software development is a collaborative project between the University of Southern California (USC), Bolt Beranek and Newman (BBN), and Sparta, Inc.

Before proceeding with the discussion of experimental results, the following subsection provides additional background for the IDPR architecture.

5.3.1 Additional Background

In Chapter 4, we addressed the high-level design issues pertaining directly to policy enforcement in IDPR, i.e., installation of Policy Routes and subsequent packet forwarding. However, IDPR architecture includes much more than just policy enforcement mechanisms. Five protocols form the core of the IDPR architecture:

- Virtual Gateway Protocol (VGP)
  A *Virtual Gateway* (VG) is a set of PGs interconnecting a pair of adjacent ADs. A simple VG consists of two PGs, one in each of the neighboring ADs. All PGs within an AD use VGP to exchange status information with respect to their outside links, i.e., the status of their constituent VGs.
- Update Protocol (UPDATE)
  Each AD uses UPDATE to distribute both configured and dynamic status information about its VGs and the associated Policy Terms. The information is disseminated in a link state fashion, i.e., it is flooded throughout the internet.
- Route Synthesis (RSYNTH)
  Each AD has one or more Route Servers (each a distinct logical entity often co-located with a PG) that is responsible for maintaining the connectivity database of the entire internet and computing PRs for its constituent users. Strictly speaking, RSYNTH is not a protocol, but, rather, a route computation algorithm superimposed with a set of AD-dependent heuristics used in selecting optimal PRs. It embodies what we referred to in Chapter 1 as *route selection* policies.
- Path Setup (SETUP)
  As described earlier, SETUP is responsible for the installation and maintenance of PR-related state in PGs.
- Kernel Packet Switching (KERNEL)
  Switching of IDPR data packets at a PG involves mapping individual data packet into PRs and forwarding them according to the next PG hop established at route setup time. Because of the high throughput requirements, data packet switching is implemented within the kernel.

In addition, there is a high degree of inter-dependence among the various IDPR protocols. The particulars of this interplay are illustrated in Figure 5.4. IDPR architecture also provides a uniform mechanism for the reliable (i.e., timely, sequenced and error-free) delivery of IDPR control packets between PGs and between the different protocols within a PG. This function is implemented by the Control Message Transport Protocol (CMTP).

VGP, UPDATE and RSYNTH are not addressed below; for a complete description of IDPR architecture and protocol specifications the reader is referred to [50, 84, 7, 28].

5.3.1.1 Policy Routes

The foremost purpose of IDPR is the maintenance of PRs. Therefore, in order to understand the inner workings of IDPR, it is crucial to understand the PR life-cycle.

With respect to a given PR, there are three types of PGs: i) ORIGINATOR, ii) TARGET, and iii) TRANSIT. ORIGINATOR is the exit PG in the source AD that initiates the installation of the PR (i.e., originates the SETUP

[2] Of which the author is a member.

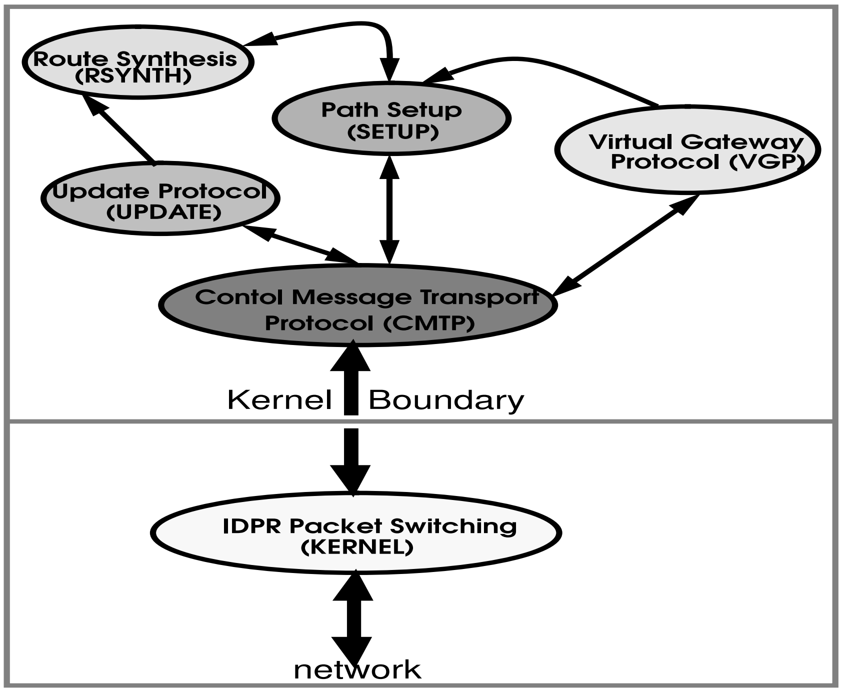

Figure 5.4: Interplay of IDPR protocols

packet). TARGET is the entry PG in the destination AD (i.e., the last PG hop in the PR). All other PGs involved are TRANSIT; for every TRANSIT PG both PG-NEXT and PG-PREV are defined.

A PR can be in any one of the seven states:

1. **PR-EMBRYONIC** (ORIGINATOR)
   SETUP-REQUEST from KERNEL is received and ROUTE-REQUEST is subsequently generated to RSYNTH.
2. **PR-INITIAL** (ORIGINATOR and TRANSITs)
   ROUTE-RESPONSE is received from RSYNTH and SETUP is forwarded to the next hop PG (PG-NEXT).
3. **PR-WAITING** (ORIGINATOR and TRANSITs)
   SETUP is ACK-ed by PG-NEXT, but ACCEPT has not yet been received.
4. **PR-ACTIVE** (TRANSITs and TARGET)
   ACCEPT is received and forwarded to PG-PREV; PR is now functional.
5. **PR-COMPLETE** (all PGs)
   ACCEPT is ACK-ed by PG-PREV (except for ORIGINATOR).
6. **PR-DYING** (all PGs)
   - TEARDOWN is generated or received from some PG in the path.
   - REFUSE is generated or received from some PG (via PG-NEXT) in the path.
   - CMTP TIMEOUT on ACCEPT to PG-PREV.
   - TIMEOUT (not CMTP) on receiving ACCEPT from upstream.
   - ERROR on processing SETUP.
7. **PR-FREE** (all PGs)
   PR entry is freed.

5.3.1.2 Protocol Description: ORIGINATOR PG

A PR is first *conceived* at the ORIGINATOR PG when the IDPR encapsulation module receives a packet and fails to locate a corresponding HOST-TABLE entry for it. It then composes a SETUP-REQUEST packet which consists of a short preamble followed by the IP packet itself. Once SETUP receives this request it first checks to see whether there is an existing PR which can be used for the new host-pair. If such a PR exists, SETUP fills in the *pi* field in the SETUP-REQUEST and installs a new HOST TABLE entry via a system call (IDPR-TABLE: Put-host-entry).

If no existing PR can be used, the ORIGINATOR generates a ROUTE-REQUEST to Route Synthesis, RSYNTH. It is assumed (for the time being) that RSYNTH is always co-located with the SETUP component, i.e., it is present at each PG. Therefore, there is (as of yet) no timeout on ROUTE-REQUESTs. The reply from RSYNTH comes in the form of a ROUTE-RESPONSE packet, whose format is identical to that of ROUTE-REQUEST. ROUTE-RESPONSE is physically trailed by the PR itself. It is composed of two or more segments where each segment corresponds to exactly one AD hop.

A special case of ROUTE-RESPONSE is ROUTE-RESPONSE-NONE which carries a value of zero in the *hops* field.[3] In this case, an ICMP HOST-UNREACHABLE packet is generated to the source host.

If RSYNTH computes a valid route, the ORIGINATOR proceeds to compose a SETUP packet. The format of a SETUP packet is very similar to that of ROUTE-RESPONSE. It also consists of a pre-amble followed by the sequence of route segments. The sequence of segments is copied directly from ROUTE-RESPONSE.

After a SETUP has been forwarded, the ORIGINATOR expects to receive an ACCEPT from the TARGET PG, or a REFUSE from any of the intervening PGs. If neither event takes place, ORIGINATOR promptly times out and attempts to set up the route again, however, not before selecting a new path id. This is done to prevent any confusion with regard to stale state in TRANSIT PGs. If the ORIGINATOR times out n (defined locally) times, it gives up and sends an ICMP HOST-UNREACHABLE packet to the source host.

If a REFUSE is received, the ORIGINATOR logs it and, once again, sends an ICMP HOST-UNREACHABLE packet to the source. This is clearly less than optimal; in the future, perhaps SETUP (depending on the reason for REFUSE) should ask RSYNTH for another route.

The arrival of an ACCEPT signals that the PR is fully operational. At this time the ORIGINATOR installs a new PR entry in the kernel PR-TABLE and also installs the corresponding HOST-TABLE entry for the host-pair that caused the setup originally.

Figure 5.5 illustrates the ORIGINATOR PG state diagram as discussed above.

5.3.1.3 Protocol Description: TRANSIT and TARGET PGs

From the point of view of a TRANSIT PG, a PR is *born* when a valid SETUP packet is received from a recognized PG neighbor. A SETUP packet is considered valid only when the following conditions are satisfied:[4]

1. The route must contain a segment corresponding to the local AD.
2. The local AD must appear only once in the route (i.e., no AD loops).
3. The PRid must not be currently in use.
4. There must be enough space in the PR-TABLE for a new entry.
5. The previous VG must be such that there is at least one PG neighbor that belongs to it.
6. The source address in the IP header of the SETUP packet must match one PG neighbor that is a member of a previous VG.
7. The next VG must be such that either: i) this PG is a member of it, or ii) there is a PG neighbor which is a member. In the former case, there must be a PG neighbor which is also a member of the said VG.
8. PTs referenced in the local AD segment must agree with the characteristics of the route.

[3] Actually, a route of length one is also treated as a negative response from RSYNTH.
[4] Timeliness, integrity and authenticity are verified by CMTP.

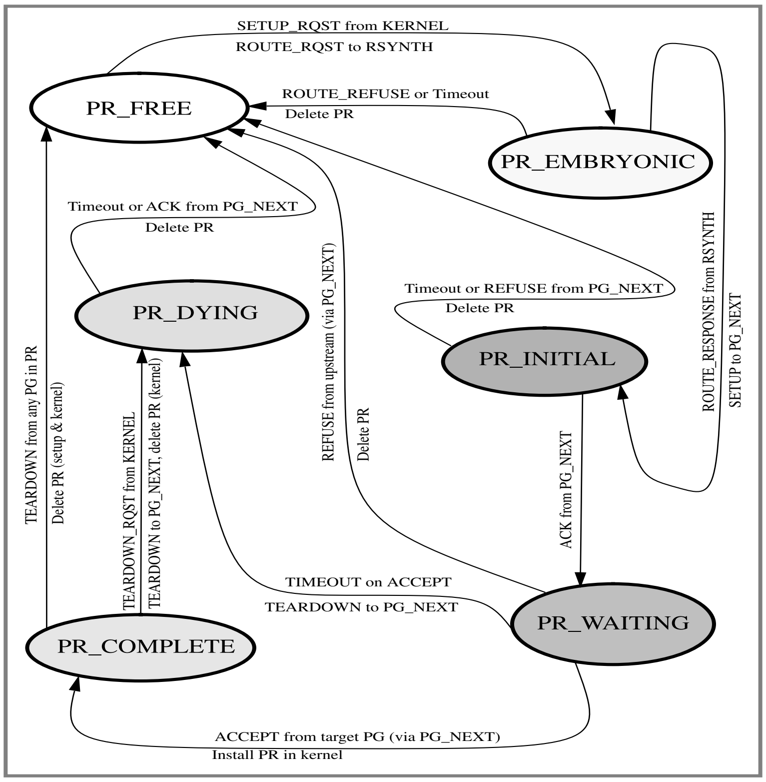

Figure 5.5: ORIGINATOR state diagram

9. Other PR usage conditions must be acceptable. This includes: UCI(s), ToS, authentication type and expiration conditions. Only TRANSIT PGs that are used for entering an AD need to be concerned with this check.

If a SETUP packet fails to meet any of the above conditions, a REFUSE with the appropriate reason code is generated to the PG-PREV. After the REFUSE is ack-ed (or times out), the PR entry is freed.

If the SETUP packet is valid, a new PR entry is created. If the last hop is reached (i.e., this PG is TARGET), ACCEPT is sent back to the ORIGINATOR and the PR is considered operational. Otherwise, the SETUP is forwarded to the PG-NEXT. If PG-NEXT fails to acknowledge the SETUP packet (timeout), TEARDOWN is sent back to the ORIGINATOR and the PR is destroyed.

After PG-NEXT acks the SETUP packet, the PR is put into a *dormant* state until the ACCEPT passes by on its way to the ORIGINATOR. The ACCEPT must satisfy the following conditions:

- It arrives via PG-NEXT.
- It is sourced by the TARGET PG.

If the ACCEPT fails to meet these conditions, it is discarded and not acknowledged. However, the PR is not destroyed. This is because a *valid* ACCEPT may still come through (if it does not, the PR is timed out; see below).

A valid ACCEPT is forwarded to PG-PREV and, once ack-ed, the PR is installed in the kernel PR-TABLE and considered fully operational. Failure to ack an ACCEPT is grounds for TEARDOWN (see below). State diagrams for TRANSIT and TARGET PGs are illustrated in Figures 5.6 and 5.7, respectively.

5.3.1.4 PR Teardown

A PR can be terminated for several reasons:

1. Lifetime exceeded

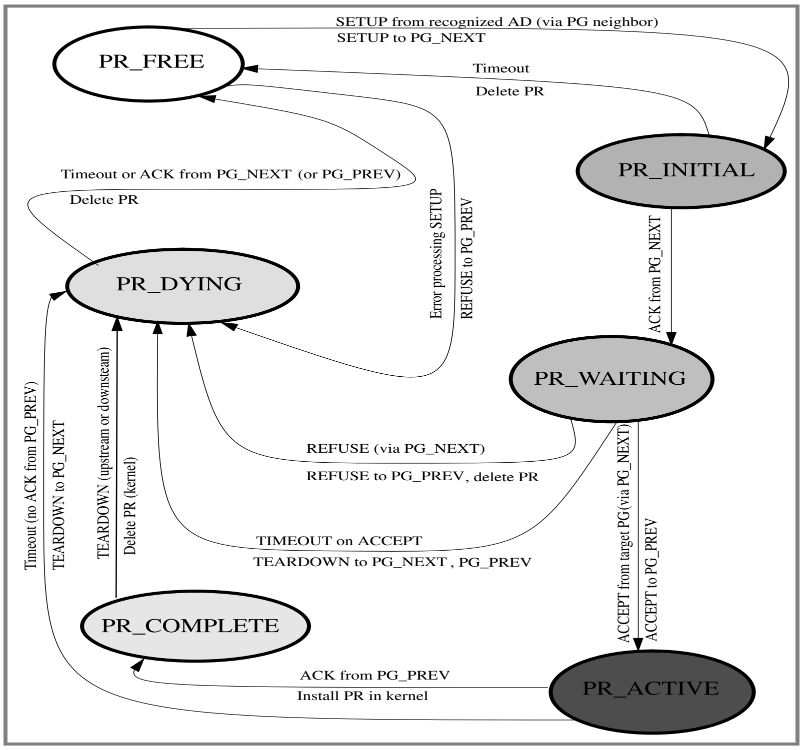

Figure 5.6: TRANSIT state diagram

2. Packet limit reached
3. Data limit reached
4. Maximum idle time exceeded
5. Previous-hop PG fails to ACK an ACCEPT packet
6. VGP reports that either previous- or next-hop PG is down

In the initial implementation, the first four types of teardown are initiated by KERNEL via a TEARDOWN-REQUEST packet that is delivered to SETUP (encapsulated with a dummy CMTP and IP header). Upon receiving TEARDOWN-REQUEST, SETUP sends a TEARDOWN packet to both PG-NEXT and PG-PREV (if defined). TEARDOWN-REQUEST and TEARDOWN have the same simple format consisting of the PRid and the termination code.

A TEARDOWN packet can arrive via PG-PREV or PG-NEXT. If it arrives via PG-NEXT, it must be forwarded to PG-PREV and vice-a-versa. A TEARDOWN must also be sourced at one of the intervening PGs, however, this is difficult to verify since PGs within a PR do not all necessarily have a means for mutual authentication.

5.3.2 Experiments

Experiments with the IDPR implementation were conducted in order to evaluate the overhead incurred by PR setup and IDPR data packet processing. This section summarizes our results.

We define PR setup overhead as the time interval between the arrival of a SETUP-REQUEST packet at the ORIGINATOR PG and the arrival of a corresponding ACCEPT packet at the same PG. Equivalently, it can be viewed as the transition time between the PR-FREE and PR-COMPLETE states as depicted in Figure 5.5.

We measured the PR setup overhead for routes ranging from 2 to 5 PG hops.[5] (The laboratory topology was re-configured for the case of 5-hops). We measured the overhead in relation to the number of PG, not AD, hops. This

[5] See Figure 5.1.

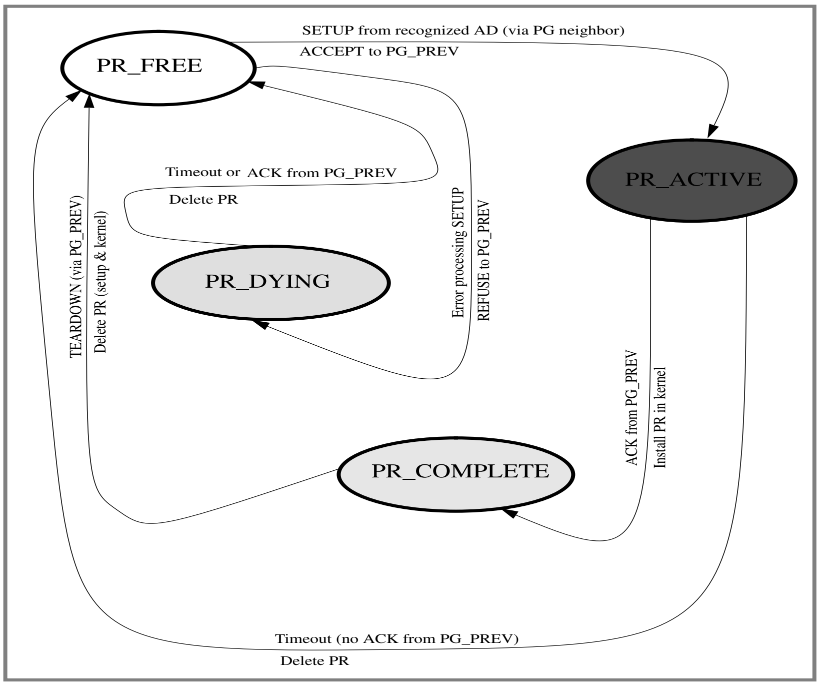

Figure 5.7: TARGET state diagram

is because traversing an AD may involve physically traversing one or more PGs. Therefore, it is more meaningful to measure per-PG, rather than per-AD setup overhead.

The results are illustrated in Table 5.2. They include MD4-based integrity check computation (for both SETUP and ACCEPT packets) at every PG hop. The first row shows the average per PG setup overhead which is based on the

# of PG hops	2	3	4	5
Per-hop overhead	50	57	54	58
Route computation	20	20	30	30
Setup overhead	50	170	215	290
Total overhead	70	190	245	320

Table 5.2: PR setup overhead (in ms)

total overhead in row 4. Because route computation is a relatively expensive operation incurred only at the first hop, we decouple the overhead due to route computation from the purely setup-related overhead (as shown in row 3 and 4).

These results show that the per PG setup cost averages about $55ms$. This number is quite low considering that it includes the following sequence of events at every transit PG:

1. Context switch from Kernel to User mode to receive SETUP
2. MD4 digest computation
3. Validity checking (including: timeliness, previous and next hop reachability and PT validation)
4. Context switch from User to Kernel mode to forward SETUP
5. Context switch from Kernel to User mode to receive ACCEPT[6]

[6] This does not occur immediately after the previous step.

6. MD4 digest computation
7. PR table lookup to locate the PR in question
8. Context switch from User to Kernel mode to forward ACCEPT

Although the results appear reassuring, a real-world, *deployable* implementation would include encryption-based signatures which will greatly increase the PR setup overhead. Also, since the number of ADs in our concocted internetwork is small and the topology is simple, overhead due to route computation and PT validation is kept artificially low. On the other hand, this implementation was not tuned for performance.

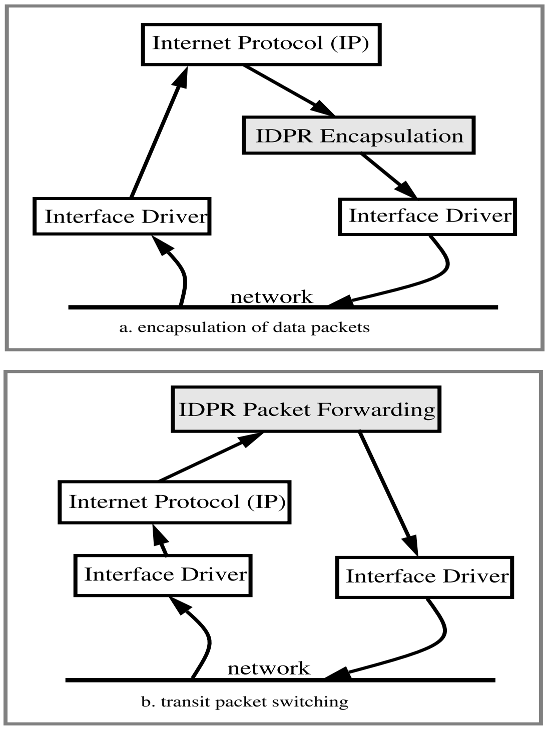

Table 5.3: IDPR encapsulation and transit packet switching

We also measured the overhead incurred by IDPR data packets in traversing previously established PRs. (Figure 5.3 depicts the placement of the encapsulation and packet switching functions). As before, we used the ICMP Echo protocol. The results in Table 5.4 represent the the round-trip times (in milliseconds) for traversing paths of variable length. For each path length we began by timing data packets without IDPR (subcolumn A). These numbers provide a basis for comparison. We then timed the same packet flows across IDPR-controlled paths (subcolumn B). This was done to isolate as much as possible the overhead due to additional packet length and IDPR-related processing in PGs. Finally, we *plugged in* MD4-based integrity checking to obtain the timings of full-blown IDPR packet switching (subcolumn C).

Unlike PR setup overhead, the numbers for data packet overhead are quite realistic since all protocol features have been implemented. The only exception is the PR table lookup. Recall that a PG uses the PRid field present in each data packet to locate an appropriate PR table entry. Because the volume of traffic on our expiremental *"internetwork"* was kept artificially low, PR table lookups were relatively fast. On the other hand, our table lookups were performed sequentially. Whereas, a more performance-tuned implementation able to support large volumes of inter-AD traffic would necessarily use some kind of a fast hashing technique for table lookups.

Judging by the above results, the overhead introduced by IDPR encapsulation and transit packet switching, while not negligible, appears to be sufficiently low to be of little hindrance to existing transport layer protocols. Even for

Packet size (in bytes)	2 a	2 b	2 c	3 a	3 b	3 c	4 a	4 b	4 c	5 a	5 b	5 c
16	4	4	4.5	5	6	7	8	7.5	9	10	10	12
64	4	5	5.5	5	7	8	8	9	10	10	12	16
256	6	7	8	8	9	10.5	13	12.5	16	14	17	18
512	8	9	10.5	10	11	13.5	17	17	22	19	20	26
1024	12	14	15	15	17	20	23	23.5	30	28	33	35

(Column groups: # of PG hops)

a. without IDPR
b. IDPR (no integrity checking)
c. IDPR with MD4 integrity checking

Table 5.4: Packet round-trip delay (in ms)

large packets (e.g., 1Kbyte) IDPR packet switching delay amounts to less than $1ms$ per PG hop. This is especially encouraging considering that the MD4 digest computation (performed at each hop) alone takes approximately $0.7ms$.

Chapter 6

Conclusions and Future Work

This final chapter summarizes the contributions of this thesis and concludes with a discussion of outstanding issues and future work.

6.1 Contributions of This Thesis

This thesis represents the first in-depth treatment of access control and policy enforcement in internetworks. Its main result is the design of an architecture geared towards enforcing a broad range of policies. The contributions of this research can be divided into three major parts:

- A framework for access control in internetworks
- Design of stub AD policy enforcement
- Design of transit AD policy enforcement

In addition, design choices are reinforced by experimental results obtained from functional implementations of the proposed mechanisms.

6.1.1 Framework

In Chapter 1, we introduced the problem of policy enforcement in the internetwork environment. It was shown that different types of controls are necessary for controlling access to three main objects of policy (resource types): i) end-systems, ii) network resources, and iii) internetwork routes. Owing to the nature of service provided and the types of potential threats, we made a further distinction between the protection of stub and transit AD network resources.

For each object of policy we considered a number of design parameters:

- Security services required
- Enforcement location
- Enforcement protocol
- Principal granularity
- Enforcement granularity and mode

In making design choices we applied the well-known *end-to-end* argument which essentially states that the placement of controls (policy, in this context) should be at the highest protocol layer at the end-points of communication. When the resource in question is an end-system that engages in inter-AD communication, the *end-to-end* argument suggests that this end-system should be able to protect itself. By the same token, other network resources, including unequipped or strictly-internal end-systems, must be protected at AD boundaries by border routers. Similarly, access to inter-AD routes must also be controlled before any traffic is allowed to cross the AD boundary. Therefore, to the extent that network resources require protection, the highest relevant endpoint is the border router and the associated packet-forwarding and routing protocols. In this sense, the *end-to-end* argument supports implementing controls at the network layer.

6.1.2 Stub AD Policy Enforcement

Chapter 3 addressed the design of policy enforcement for stub AD communication using the guidelines outlined in our framework. The goal of stub AD policy enforcement is to permit only authorized, authenticated and timely communication to take place across stub AD boundaries. Earlier, in Chapter 2 we discussed several existing approaches for controlling internetwork communication and concluded that they provide solutions that are either too restrictive, insecure, or not general enough.

Visa protocol presented in Chapter 3 is a simple, yet powerful mechanism for controlling packet traffic at stub AD network boundaries. The primary goal of *Visa* protocol is the protection of stub AD network and end-system resources from unauthorized access. This goal 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.

One of the main advantages of *Visa* is that visa-routers do not bear the brunt of the responsibility for making access control decisions. By issuing a visa, an ACS has pre-computed an authorization decision. The task of a visa-router is thus reduced to ensuring that a visa is valid and is being used appropriately; the expensive part of policy enforcement is done once per connection, by the ACSs of the end-point ADs, rather than once per packet, by the border routers. In addition, *Visa* does not impact intra-AD traffic and has no bearing upon end-systems that do not partake in inter-AD communication.

As part of the design process, we evaluated the security of the proposed *Visa* protocol using the motivation and some terminology (but not the methods) from the well-known BAN logic [9]. The protocol was shown to be resistant to hostile attacks (assuming a strong underlying cryptosystem.) The security analysis also demonstrated the limitations to the beliefs and assurances attained by protocol participants.

For the purpose of quantifying the costs of stub AD policy enforcement, we conducted experiments with a prototype *Visa* protocol implementation. As illustrated by the experimental results in Chapter 5, the cost of the initial setup phase is still somewhat high. However, data packet overhead was demonstrated to be quite insignificant for the case of non-cryptographic packet signatures.

6.1.3 Transit AD Policy Enforcement

Relative to stub AD access control, access control with respect to transit network resources is a new phenomenon. It has been brought about by (among other factors) the increasing commercialization of transit internetwork facilities.

The framework discussion in Chapter 1 suggested that controlling the usage of (i.e., access to) transit network resources (such as routers and links) requires a new generation of routing and packet-forwarding protocols because of the need to coordinate transit policies among all intervening ADs. This rules out unilateral policy enforcement 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.

Chapter 4 addressed the design of protocols for **secure** control of transit internetwork traffic using Inter Domain Policy Routing (IDPR) as the design foundation. We concentrated on PR setup and packet-forwarding protocols.

We began by identifying potential security threats facing IDPR and discussed a number of defense mechanisms. Special attention was paid to the costs of implementing security services. In particular, we considered (and evaluated) a range of data integrity scenarios varying in both strength and cost. The flexibility to select a desired tradeoff between security and performance is built into the architecture.

After specifying PR setup and packet-forwarding protocols we conducted a thorough analysis of their security. As in *Visa* protocol, this analysis identified the exact beliefs attained by the protocol participants, i.e., PGs. At the same time, it revealed that certain goals are ultimately unattainable, e.g., reliable verification of the exact path a packet traverses.

Experimental results obtained from actual protocol implementations appear very reassuring. The cost of PR setup was shown to be tolerable, at around $55ms$ per PG hop. The overhead cost of data packet switching in IDPR is of greater concern than the cost of PR setup. However, our measurements indicated that even for large packet sizes (e.g., 1Kbyte) packet-switching overhead per PG hop amounted to less than $0.5ms$.

6.1.4 Security .vs. Cost

Security and performance are always at odds. This is neither surprising nor easily remedied. Throughout this thesis, we attempted to reduce as much as possible the adverse effects of our protocols (namely, added costs) without sacrificing security.

Certain types of overhead are difficult, if not impossible, to avoid. For example, reducing the number of packets exchanged during connection setup in *Visa* protocol or PR setup in IDPR is not feasible without sacrificing security. On the other hand, using *light-weight* data authentication methods can greatly influence packet-switching overhead in visa-routers and PGs.

An important contribution of this thesis is the development of encryption-free methods for verifying data authenticity. In Appendix A, we describe two methods for computing MD4 message digests based on a secret quantity. We believe the strength of these methods to be at the very least equal to the strength of the underlying one-way hash function.

6.2 Future Work

This thesis represents a first attempt at consolidating policy enforcement mechanisms. There remain numerous opportunities for future research; a few of them are discussed below.

6.2.1 Multicasting

Multicasting is a useful and convenient technique for communication within node groups [86]. Multicasting support for the internetwork environment have been proposed in [18]. Two types of multicast predominate:

1. Multicasting within small, static and short-term groups, e.g., for the purpose of tele- or video-conferencing. This type of communication typically has high bandwidth requirements.

2. Multicasting within large, dynamic and long-term groups, e.g., name or time servers. Typically, communication is infrequent and not bandwidth-intensive, e.g., global time server updates can take place once every hour.

Visa protocol, as specified, provides no special support for either type of multicasting. Consequently, n separate visas would be required for a multicast group member to communicate with n peers assuming *type 1* multicast. Things get much worse when n is not known in advance (*type 2* multicast) and the exact locations of the peers are not known. We believe this to be a problem not unique to *Visa* protocol but inherent to any strong network access control mechanism.

IDPR is also not equipped with multicast support. In order to achieve *type 1* multicast using the protocols specified in Chapter 4, every group member must set up n distinct PRs, one to each peer. Moreover, subsequent traffic would have to be replicated individually for every such PR. Ideally, we would like to avoid unnecessary replication and excessive state in PGs. This can be made possible if IDPR is augmented to allow setup of multi-destination PR "trees" in addition to regular, single-destination PRs. Furthermore, subsequent data traffic must be multiplexed along every tree branch in transit PGs.

Supporting *type 2* multicasting is more complicated because of large group sizes and volatile group membership. One possible solution is to treat such multicast traffic as being "exempt" from certain controls. For example, we could restrict uncontrolled multicast to well-known groups such as time or name servers. The main drawback is that an intruder can use this *back-door* as an opportunity for abuse by flooding an AD with fraudulent traffic. Alternatively, multicasting can be restricted to destinations (peers) that have existing PRs.

Full support of multicasting in the presence of policy enforcement may prove to be impossible, however, it remains subject to future work to determine exactly to what extent multicasting can coexist with policy enforcement.

6.2.2 Fault Tolerance

Fault tolerance in *Visa* and IDPR has not been extensively considered in this thesis. This does not imply that the subject is unimportant, quite the contrary. In this section, we consider the issues of state recovery and connection repair.

6.2.2.1 State Recovery

State recovery is the problem of reestablishing lost state whenever a visa-router or a PG crashes and reappears shortly thereafter.

When a visa-router crashes, its visa-table is lost. *Visa* protocol description incorporates a means for an ACS to reestablish a visa-table in a "re-incarnated" visa-router. However, the important drawback of this method is that ACSs, unlike visa-routers, do not keep up-to-date visa-tables, i.e., a visa-table entry in an ACS does not reflect any resource usage since a corresponding visa was originally issued. This is because, all records of resource usage are irrecoverably lost when a visa-router crashes. Therefore, simply refreshing a visa-router may not be an optimal decision.

IDPR does not currently specify any mechanism for the recovery from PG crashes. Below, we outline one possible solution.

If a PG crashes, it loses its entire PR table. However, we assume that all PGs have reliable (perhaps, stored in stable storage) knowledge of their neighbor PGs. When a PG recovers from a crash it first announces itself to the neighbors. Each neighbor then proceeds to search its PR table for all entries that specify the said PG as either previous or next PG hop. Having collected all such entries, each neighbor composes a special REFRESH packet which it forwards to the PG in question. Upon receiving REFRESH packets from all of its neighbors, a PG should be able to reconstruct its PR table with high accuracy, including even such details as packet and data meters.

This approach works best for recovering PRs for which the crashed PG is TRANSIT. This is because both previous and next hop PGs are defined for TRANSITs. Therefore, in order to fully recover a *lost* PR, a PG must receive two REFRESH packets (from two of its neighbors) such that each containing a matching record for the said PR. By combining the information in the records, a PG can recreate an entry in its PR table. Resource usage meters can be recovered by taking the smaller set of values from the two REFRESH packets.

It is slightly more problematic to recover PRs that specify the failed PG as either ORIGINATOR or TARGET. Since there is but a single source of information, only one REFRESH can be expected. Hence, there is no longer the assurance from two independent sources. Moreover, in both cases, a REFRESH always comes from a neighbor in a different (though directly-connected) AD. This raises the issue of trust, i.e., what kind of assurances are necessary for a PG to reestablish its PR table reliably? Perhaps every PG should store copies of signed SETUP packets for each active PR just in case a neighbor crashes and subsequently demands strong evidence of PRs' existence.

6.2.2.2 Connection and Route Repair

An issue related to state recovery is connection and route repair. When a visa-router or a PG crashes and then fails to recover, previously authorized traffic will be unable to propagate as expected.

In *Visa*, unless an alternate visa-router is available, connection repair will not succeed. To allow for *on-the-fly* adaptation to crashes, all potential "backup" visa-routers must replicate the necessary state.

In IDPR, when a PG crashes, it may still be possible to utilize the original PR. This is because a PR specifies ADs, not physical PGs. Hence, if there is an alternate PG, *parallel* to the one that failed (i.e., a PG in the same AD and the same AD), route repair may succeed, especially if PR state is replicated in parallel PGs at setup time. Barring replication, the previous hop PG may attempt to repair the PR by setting up appropriate state in the alternate PG; however, it must be able to produce a copy of the original SETUP packet.

If a PG crashes and no parallel PGs are available, the issue of route repair becomes more complicated. Suffice it to say that, because of policy considerations, a transit PG can not attempt route repair unilaterally. In other words, a PR can not be modified without prior authorization from its original source.

One solution is to tear-down the PR and let the source recompute an alternate route. This approach is simple and requires no special protocol support. However, it involves a significant amount of overhead.

Alternatively, the source can pre-compute a number of different PRs for the same destination and *bundle* them together in a single SETUP packet. All of the enclosed PRs can then be set up simultaneously. When a a transit PG becomes unavailable, the previous hop PG can switch to a "sibling" PR transparently. This technique is known as *Adaptive Multi-Path Source Routing* and is subject of current and future research [8].

6.2.3 Accounting and Billing

Internetwork communication has been, until now, treated mostly as a free (or fixed-cost) good [13]. This is likely to change with the advent of internetwork policy enforcement and commercially-owned and operated transit facilities.

If communication is to be charged for on a usage-sensitive basis, accounting must take place in transit ADs. Considerations for usage accounting are extensively discussed in [33, 16]. IDPR, by virtue of maintaining state in policy gateways, can easily support accounting on the basis of packets, or PR duration[1].

Cost recovery is an entirely different matter. As Clark points out in [13], it is a very complicated problem with many unresolved issues. One of the more difficult ones is the problem of reliable billing. Looking to the real-world for an analogy, credit card billing statements often include itemized charges that help the consumer in identifying and verifying individual charges. Furthermore, most credit card companies are willing to provide copies of signed receipts to customers who have *trouble* remembering certain purchases. In other words, a customer can choose to maintain a sort of a "stateless" model with respect to the purchases made and still be assured of the validity of individual items billed at the end of the month.

Ideally, a transit AD billing for service would provide an equivalent of a signed statement to its stub AD subscribers. Whether or not this billing integrity can be accomplished remains to be seen.

6.2.4 Integration

In Chapter 1, we argued for dividing the discussion of security according to the type of resource being protected. However, from a technical and policy perspective, it is essential for these two types of policy enforcement to be coordinated and consistent.[2] To this end, we discuss an integrated view of policy enforcement support. We focus on two opportunities for integration – border routers and specialized servers.

6.2.4.1 Border Routers

The heaviest burden of policy enforcement falls on the border routers. This is particularly true of border routers in stub ADs which need to implement controls at the granularity of both ADs and end-systems, i.e., *Visa* and IDPR.

For locally originated traffic, the border router needs to control: i) source end-system access to the outside, ii) destination AD and end-system, and iii) inter-AD route access. Traffic coming in from the outside has to be controlled on the same basis, with the exception of route selection (if routes are composed at the source). However, it is possible for a border router in a destination AD to refuse incoming traffic based on the route it followed.

No attempt has been made in this thesis to integrate and streamline the two types of enforcement mechanisms. This does not imply that integration is not possible or desirable. On the contrary, significant performance benefits can be gained through integration.

Visa and IDPR can benefit substantially from sharing many types of information. For example, a visa-router maintains a visa-table where every entry refers to a connection between two end-systems. Similarly, a PG in a stub AD maintains a host-table where every entry points to a PR. If a router implements both *Visa* and IDPR, it is more than likely that the same information is duplicated in the two tables. Merging these two tables into one can save a lot of storage space. Moreover, processing overhead can be reduced if only a single table lookup is performed for both protocols.

6.2.4.2 Specialized Servers

Network administrators need to create consistent policies across all three types of controls addressed. This may involve more than one server function. In a stub AD, network administrators will define which end-systems will be reachable from the outside. These policies are then enforced by the particular Access Control Server (ACS) used. The ACS controls the inclusion of particular principals and objects (e.g., end-system addresses) in router access control lists. For those end-systems that are made accessible, network administrators will want to monitor the end-system controls employed to make sure that unreachable end-systems are not vulnerable to external access.

[1] Actually, IDPR already supports packet, data and time counters on a per PR basis.

[2] This is similar to the integrated layer processing argument presented in [14].

These same network administrators will create policies to be used in selecting routes. The same inter-AD relationships and trust levels that determine the end-system and network resource access policies will determine route selection criteria. Once routes are selected for possible use by the AD, route servers are responsible for handing out routes to end-systems (or border routers) within the AD. Therefore, network administrators must also specify access control lists to control distribution of the route servers' routes. A route server, therefore, is involved in all three types of policy – network resource control policies are considered in route computation, followed by route selection policies, and finally by end-system policies that control distribution of routes. Finally, for hybrid ADs that support some transit traffic as well as end-system access, transit routing policies, advertised via the inter-AD routing protocol, must be coordinated with stub-AD network resource controls.

In the case of transit ADs, there is less of a concern with coordinating several types of policy. However, network administrators will need tools to assist them in composing useful transit policies. Such policies must on the one hand realize the AD's control requirements, and on the other hand, be as compatible as possible with the policies of all other ADs. If no attention is paid to compatibility, then the transit service may be of little use to anyone in the internetwork.

6.2.5 Other Policy Routing Approaches

The IDPR architecture described in this thesis is not the only possible paradigm for Policy Routing. Other existing proposals reviewed in Chapter 2 are no match for IDPR insofar as the range of policies supported. Nevertheless, it is conceivable that other, radically different approaches may be more suitable to Policy Routing. (Alternative Policy Routing approaches are discussed in [7].) For example, it has been suggested that IDRP (not IDPR) be augmented to include full AD-level paths coupled with associated policies along the distance vector tables it already exchanges[3]. It is unclear at the time of this writing whether the resultant architecture can contend with IDPR.

[3] D. Estrin, private communication, March 1990.

Reference List

[1] Advanced Micro Devices, Inc., *AMD MOS Microprocessors and Peripherals Data Book*, **Advanced Micro Devices, Inc.**, Sunnyvale, CA. 1987

[2] ANSI, *Intermediate System to Intermediate System Inter-Domain Routeing Information Exchange Protocol*, **ANSI Document X3S3.3/90-132**, June 1990.

[3] E. Biham, A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, **Proceedings of CRYPTO'90**, August 1990.

[4] M. Bishop, *A Security Analysis of Version 2 of the Network Time Protocol (NTP)*, **Dartmouth College Technical Report PCS-TR91-154**, January 1991.

[5] *Blacker Front End Interface Control Document* **DDN Protocol Handbook, Vol. 1**, DDN Network Information Center, SRI International, 1985.

[6] A. Birrell, B. Lampson, R. Needham, M. Schroeder, *A Global Authentication Service Without Global Trust*, **Proceedings of 1986 IEEE Symposium on Security and Privacy**, May 1986.

[7] L. Breslau and D. Estrin, *Design of Inter-Administrative Domain Routing Protocols*, **Proceedings of 1990 ACM Sigcomm Conference**, September 1990.

[8] L. Breslau, D. Estrin, and L. Zhang, *Adaptive Multi-Path Source Routing*, **DRAFT, in submission to 1991 ACM Sigcomm Conference**, April 1991.

[9] M. Burrows, M. Abadi, and R. Needham, *A Logic of Authentication*, **Proceedings of ACM Symposium on Operating System Principles**, December 1989.

[10] CCITT, *The Directory, Part 1: Overview of Concepts, Models, and Services*, **CCITT Recommendation X.500**, December 1988.

[11] CCITT, *The Directory Authentication Framework*, **CCITT Recommendation X.509**, 1988.

[12] D. Clark, *Design Philosophy of the DARPA Internet Protocols*, **Proceedings of 1988 ACM SIGCOMM Symposium**, August 1988.

[13] D. Clark, *Policy Routing in Internet Protocols*, **Journal of Internetworking: Research and Experience**, Vol. 1, No. 1, October 1990.

[14] D. Clark and D. Tennenhouse, *Architectural Considerations for a New Generation of Protocols*, **Proceedings of 1990 ACM SIGCOMM Symposium**, September 1990.

[15] D. Clark, D. Wilson, *A Comparison of Commercial and Military Computer Security Policies*, **Proceedings of 1987 IEEE Symposium on Security and Privacy**, May 1987.

[16] R. Cocchi, D. Estrin, S. Shenker, L. Zhang, *A Study of Priority Pricing in Multiple Service Class Networks*, **DRAFT, in submission to 1991 ACM SIGCOMM Symposium**, April 1991.

[17] D. Davies and W. Price, *Security For Computer Networks*, **New York, NY:Wiley**, 1984.

[18] S. Deering, *Multicast Routing in Internetworks and Extended LANs*, **Proceedings of 1988 ACM SIGCOMM Symposium**, August 1988.

[19] W. Diffie, *The First Ten Years of Public-Key Cryptography*, **Proceedings of the IEEE**, Vol. 76, No. 5, May 1988.

[20] W. Diffie and M. Hellman, *New Directions in Cryptography*, **IEEE Transactions on Information Theory**, Vol. 22, No. 11, November, 1976.

[21] Digital Equipment Corporation, *Intermediate System to Intermediate System Intra-Domain Routeing Exchange Protocol*, October 1989.

[22] E. Dijkstra, *Self-Stabilization in Spite of Distributed Control*, **Communications of the ACM**, November 1974.

[23] T. El Gamal, *A Public Key Cryptosystem and a Signature Based on Discrete Logarithms*, **IEEE Transaction on Information Theory**, Vol. 31, July 1985.

[24] D. Estrin, *Controls for Inter-Organization Networks*, **IEEE Transactions on Software Engineering**, February 1987.

[25] D. Estrin, *Interconnection Protocols for Interorganization Networks*, **IEEE Journal on Selected Areas in Communications**, December 1987.

[26] D. Estrin, *Policy Requirements for Inter Administrative Domain Routing*, **Computer Networks and ISDN Systems**, To appear in 1991.

[27] D. Estrin, J. Mogul, G. Tsudik, *Visa Protocols for Controlling Inter-Organizational Datagram Flow*, **IEEE Journal on Selected Areas in Communications**, May 1989.

[28] D. Estrin and M. Steenstrup, *Inter-Domain Policy Routing: Overview of Architecture and Protocols*, **ACM Computer Communications Review**, January 1991.

[29] D. Estrin and G. Tsudik, *Secure Control of Transit Internetwork Traffic*, **Computer Networks and ISDN Systems**, To appear in 1991.

[30] D. Estrin and G. Tsudik, End-to-End *Argument for Network-Layer Access Controls*, **Journal of Internetworking: Research and Experience**, Vol. 2, No. 2, May 1991.

[31] D. Estrin and G. Tsudik, *Security Issues in Policy Routing*, **Proceedings of 1989 IEEE Symposium on Security and Privacy**, May 1989.

[32] D. Estrin and G. Tsudik, *Visa Scheme for Inter-Organization Network Security*, **Proceedings of 1987 IEEE Symposium on Security and Privacy**, April 1987.

[33] D. Estrin and L. Zhang, *Design Considerations for Usage Accounting and Feedback in Internetworks*, **SIGCOMM Computer Communications Review**, October 1990.

[34] European Computer Manufacturers Association, *Inter-Domain Intermediate Systems Routing*, **Technical Report ECMA/TC32-TG10/89/56**, May 1989.

[35] L. Ford and D. Fulkerson, *Flows in Networks*, **Princeton, NJ:Princeton University Press**, 1962.

[36] J. Galvin, K. McCloghrie, J. Davin, *Secure Management of SNMP Networks*, **Proceedings of 1991 Integrated Network Management Symposium**, April 1991.

[37] M. Gasser, A. Goldstein, C. Kaufman, B. Lampson, *The Digital System Security Architecture*, **Proceedings of 1989 National Computer Security Conference**, Noveber 1989.

[38] D. Gomberg, *A Model of Inter-administration Network User Authentication and Access Control*, **MITRE Corp. Technical Report MTR-87W00003**, December 1987.

[39] S. Hares, D. Katz *Administrative Domains and Routing Domains, a Model for Routing in the Internet*, **RFC 1136, SRI Network Information Center**, December 1989.

[40] C. Hedrick, *An Introduction to IGRP*, **Rutgers University Technical Report**, October 1989.

[41] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, **Rockville, MD:Computer Science Press**, 1978.

[42] IEEE 802.10, Standard for Interoperable LAN Security, **LAN Security Working Group, P802.10/D6**, September 1989.

[43] International Standards Organization, *Information Processing Systems – Open Systems Interconnection – Basic Reference Model*, **ISO 7498**, 1977

[44] International Standards Organization, *Security Architecture*, **ISO 7498-2-1988(E)**, 1988

[45] International Standards Organization, *OSI Routeing Framework*, **ISO/TF 9575**, January 1989.

[46] P. Karger, *Authentication and Discretionary Control in Computer Networks*, **Computer Networks and ISDN Systems**, January 1986.

[47] C. Kent and J. Mogul, *Fragmentation Considered Harmful*, **Proceedings of 1987 ACM SIGCOMM Symposium**, August 1987.

[48] S. Kent, *Protocols and Techniques for Data Communication Networks*, **Englewood Cliffs, NJ:Prentice Hall**, 1980.

[49] A. Konheim, Cryptography*, a Primer* **New York, NY:Wiley**, 1981.

[50] M. Lepp and M. Steenstrup, *An Architecture for Inter-Domain Policy Routing*, **BBN Report Number 7345**, July 1990, BBN Corporation, Cambridge, MA 02138.

[51] J. Linn, *Privacy Enhancement for Internet Electronic Mail. Part I: Message Encipherment and Authentication Procedures*, **RFC 1113, SRI Network Information Center**, January 1989.

[52] M. Little, *Dissimilar Gateway Protocol (DGP)*, **Working Paper**, 1988.

[53] K. Lougheed and Y. Rekhter, *Border Gateway Protocol*, **RFC 1163, SRI Network Information Center**, June 1990.

[54] K. Lougheed and Y. Rekhter, *Border Gateway Protocol* **RFC 1105, SRI Network Information Center**, June 1989.

[55] J. McQuillan, *Adaptive Routing Algorithms for Distributed Computer Networks*, **BBN Inc, BBN TR 2831**, May 1974.

[56] J. McQuillan, I. Richer, E. Rosen, *The New Routing Algorithm for the ARPANET*, **IEEE Transactions on Communications**, May 1980.

[57] R. Merkle, *One-way Hash functions and DES*, **Proceedings of CRYPTO 89**, August 1989.

[58] D. Mills, *On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System*, **ACM Computer Communications Review**, January 1990.

[59] D. Mills, *Internet Time Synchronization: the Network Time Protocol*, **RFC 1129, SRI Network Information Center**, October 1989.

[60] P. Mockapetris, *Domain Names - Implementation and Specification*, **RFC 1035, SRI Network Information Center**, November 1987.

[61] J. Mogul, *Simple and Flexible Datagram Access Controls for Unix-based Gateways*, **Proceedings of Summer 1989 USENIX Technical Conference**, August 1989.

[62] J. Moy, *The Open Shortest Path First (OSPF) Specification*, **RFC 1131, SRI Network Information Center**, October 1989.

[63] J. Mracek, *Network Access Control in Multi-Net Internet Transport*, **S.B. Thesis, MIT Department of Electrical Engineering and Computer Science**, June 1983.

[64] S. Mulender et al, *Amoeba: A Distributed Operating System for the 1990s*, **IEEE Computer**, Vol. 23, No. 5, May 1990.

[65] T. Narten, *Internet Routing*, **Proceedings of 1988 ACM SIGCOMM Symposium**, August 1989.

[66] National Bureau of Standards, *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.

[67] R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, **Communications of the ACM**, Vol. 21, No. 12, December 1978.

[68] R. Needham and M. Schroeder, *Authentication Revisited*, **ACM Operating Systems Review**, Vol. 21, No. 7, January 1987.

[69] D. Nessett, *Factors Affecting Distributed System Security*, **IEEE Transaction on Software Engineering**, Vol. SE-13, 1987.

[70] M. Padlipsky, *The Elements of Networking Style*, **Englewood Cliffs, NJ:Prentice Hall**, 1985.

[71] R. Perlman, *Network Layer Protocols with Byzantine Robustness*, **Ph.D. Dissertation, MIT LCS TR-429**, October 1988.

[72] R. Perlman, *Fault-Tolerant Broadcast of Routing Information*, **Computer Networks and ISDN Systems**, December 1983.

[73] J. Postel, *Internet Protocol*, **RFC 791, SRI Network Information Center**, September 1981.

[74] J. Postel, *Internet Control Message Protocol*, **RFC 792, SRI Network Information Center**, September 1981.

[75] R. Rivest, *The MD4 Message Digest Algorithm*, **Proceedings of CRYPTO'90**, August 1990.

[76] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, **Communications of the ACM**, February 1978.

[77] E. Rosen, *Exterior Gateway Protocol (EGP)*, **RFC 827, SRI Network Information Center**, October 1982.

[78] J. Saltzer, D. Reed, and D. Clark, *End-to-End Arguments in System Design*, **ACM Transactions on Computer Systems**, June 1984.

[79] J. Saltzer and M. Schroeder, *Protection of Information in Computer Systems*, **Proceedings of the IEEE**, September 1975.

[80] M. Satyanarayan, *Scalable, Secure, and Highly Available File Access*, **IEEE Computer**, Vol. 23, No. 5, May 1990.

[81] SDNS Protocol and Signaling Working Group, SP3 Subgroup, *SDNS Secure Data Network System Security Protocol 3 (SP3)*, **SDN.301 Revision 1.5 1989-05-15**

[82] A. Shimizu, S. Miyaguchi, *Fast Data Encryption Algorithm (FEAL)*, **Proceedings of EUROCRYPT'87**, April 1987

[83] K. Sollings, *Cascaded Authentication*, **Proceedings of 1987 IEEE Symposium on Security and Privacy**, April 1987.

[84] M. Steenstrup and Open Routing Working Group, *Inter-Domain Policy Routing Specification*, **BBN Report Number 7346**, December 1990, BBN Corporation, Cambridge MA 02138.

[85] J. Steiner, *The Kerberos Network Authentication Service Overview*, **MIT Project Athena RFC, Draft 1**, April 1989.

[86] A. Tanenbaum, *Computer Networks*, **Englewood Cliffs, NJ:Prentice Hall**, 1981.

[87] P. Tsuchiya, *Efficient and Robust Policy Routing using Multiple Hierarchical Addresses*, **DRAFT, in submission to 1991 ACM SIGCOMM Symposium**, April 1991.

[88] G. Tsudik, *Datagram Authentication in Internet Gateways*, **IEEE Journal on Selected Areas in Communications**, May 1989.

[89] U.S. Department of Defense, *Trusted Computer System Evaluation Criteria*, **Standard 5200.28-STD**, December 1985.

[90] V. Voydock and S. Kent, *Security Mechanisms in High-level Network Protocols*, **ACM Computing Surveys**, June 1983.

[91] S. Walker, *Network Security: the Parts of the Sum*, **Proceedings of 1989 IEEE Symposium on Security and Privacy**, May 1989.

Appendix A

Message Authentication with One-Way Hash Functions

In this appendix, we discuss some potential applications of fast one-way hash functions to the problem of message authentication. Several methods for protection against message substitution are presented. The security of the proposed methods is conjectured to be equivalent to the strength of the underlying one-way hash function.

A.1 Introduction

In recent years, some important advances were made in the area of fast one-way hash functions for message integrity. One of these functions, Rivest's Message Digest algorithm (MD4)[75], is considered by many in the security community to be the most promising (and hereto unbroken) method. Due to its structure, MD4 lends itself to software implementations that can perform at speeds significantly higher than earlier, encryption-based integrity methods, e.g., DES-based MAC[66, 17].

In brief, MD4 computes a 128-bit digest of an arbitrary length message. Its basic input unit is a 512-bit data block. It is conjectured that MD4 possesses the following properties:

1. The difficulty of computing two distinct messages, M_1 and M_2 such that $MD4(M_1) = MD4(M_2)$ is on the order of 2^{64} operations.

2. Given a message digest, MD, the difficulty of computing a message, M such that $MD4(M) = MD$ is on the order of 2^{128} operations.

The second property is of particular interest. It is based on the fact that MD4 maps 2^{512} 512-bit blocks into 2^{128} 128-bit digests. Consequently, on the average, $[2^{512}/2^{128} = 2^{384}]$ blocks are hashed into the same 128-bit digest. Therefore, given a message digest, it takes, on the average, 2^{128} trials before discovering one of the 2^{384} messages that maps into that digest.

A.2 Motivation

In the realm of secure communications, message integrity alone does not provide sufficient defense against some attacks. While a strong one-way hash function protects against data modification, protection against *message substitution* requires that the message integrity value be irreproducible. Encryption-based integrity methods provide this service since one of the inputs to the integrity function is a secret key (e.g., DES). This is not the case with MD4. Since MD4 implementations are readily available, anyone can produce an MD4 value of an arbitrary message and insert $[msg, MD4(msg)]$ into the stream. One simple solution is to require that the message source encrypt the MD4 value of each message. In a public key cryptosystem, this can be done with the sender's private key. In a conventional cryptosystem, the sender can use the secret key shared amongst the appropriate parties.

While this simple approach solves the message substitution problem, it has some drawbacks. Even though the use of expensive encryption is limited to signing a short fixed-length message digest, encryption still takes its toll. Even the fastest commercially available cryptographic devices operate at rates about an order of magnitude behind MD4, especially, for small data units [1].

Drawing from the discussion above, an ideal scenario for countering both message substitution and message modification attacks would: i) avoid encryption altogether, and ii) produce unforgeable integrity values. Initially, these two properties appear to be at odds, however, some innovative application of fast one-way hash functions may be able to reconcile them.

A.3 Protocol Description

We suppose that two principals, A and B, would like to communicate over an insecure channel. Furthermore, there exist secure means of principal authentication [67, 90, 85]. At the time of session initiation, after mutual authentication is achieved, one of principals, say, A, generates a random 512-bit value, S_{AB}. A then communicates S_{AB} to B in secret (using encryption, if necessary).

When A has a message M to send to B, it computes $MD_M = MD4(M||S_{AB})$[1]. It then sends $[M, MD_M]$ to B. Since B possesses S_{AB}, it can re-compute $MD4(M||S_{AB})$ and verify MD_M. We refer to this method as the *secret suffix* technique.

Alternatively, A and B can agree to compute MD_M as $MD4(S_{AB}||M)$ and use S_{AB} as a *secret prefix*. The secret prefix method was developed independently by the Internet Security and Privacy Working Group (SPWG)[2] for use in Simple Network Management Protocol (SNMP) [36].

A.4 Informal Analysis

Throughout the following discussion it is assumed that an intruder has the ability to record and analyze messages as well as insert fraudulent messages into the message stream. In order to mount a successful attack or effectively *break* the protocol, an intruder has to be able to produce a fraudulent message and an accompanying digest such that the $(message, digest)$ pair is accepted as genuine by any of the legitimate principals.

The following observations are based on the second property[3] of the MD4 algorithm:

- **Claim 1:** *Given a message digest, MD, and a message M_1, the difficulty of computing a message, M_2 such that $MD4(M_2||M_1) = MD$ is on the order of 2^{128} operations.*
- **Claim 2:** *Given a message digest, MD, and a message M_1, the difficulty of computing a message, M_2 such that $MD4(M_1||M_2) = MD$ is on the order of 2^{128} operations.*

Assuming these two claims hold, the only venue left to the intruder is the *brute force* attack. For most purposes, attacks that take on the order of 2^{128} are considered *futile*, i.e., the system being attacked is considered secure[4] Still, understanding the details of possible attacks is of interest. As described below, the two seemingly similar methods proposed differ in their ability to withstand brute force attacks.

A.4.1 Definitions

- Given two 512-bit blocks, M_1 and M_2, we say that $M_1 \equiv M_2$ if $MD4(M_1) = MD4(M_2)$. The '$\equiv$' symbol denotes that M_1 and M_2 are *prefix-equivalent*. For a given 512-bit block M, let PEC_M denote the *prefix-equivalence class* of M.
- Two 512-bit blocks, $M_1 \doteq M_2$ if for all blocks M, $MD4(M||M_1) = MD4(M||M_2)$. The '$\doteq$' symbol denotes that M_1 and M_2 are *suffix-equivalent*. Let SEC_M denote the *suffix-equivalence class* of a 512-bit block M.

---

[1] $||$ denotes concatenation.
[2] S. Crocker, S. Kent, Private Communication, December 1990.
[3] See Section A.1.
[4] Brute force attack on DES requires **only** on the order of 2^{56} operations.

A.4.2 Secret Prefix

A naive intruder wishing to discover the secret prefix would first record a message M accompanied by its integrity value, $MD4(S_{AB}||M)$. He would then need to try on the average 2^{128} possibilities before discovering a prefix string $\hat{S}$ where $\hat{S} \equiv S_{AB}$.

One important observation that can be made by a more *astute* intruder is that the secret prefix, S_{AB}, is not really what is needed to *break* the protocol. In other words, the secret prefix is sufficient, yet not necessary, for a successful attack. Instead, to attain the goal of composing *genuine* messages, the intruder needs only the intermediate MD4 value of S_{AB}, i.e., $MD4(S_{AB})$. The length of $MD4(S_{AB})$ is 128 bits regardless of the length of S_{AB}. Nevertheless, the number of operations required to find $MD4(S_{AB})$ remains a hefty 2^{128}.

The subtle difference between the *naive* and the more *educated* brute force attacks is that, in the latter, the intruder is guaranteed to discover $MD4(S_{AB})$ (hence, successfully attacking the protocol) in at most 2^{128} operations. This is in contrast to the former where on the **average** 2^{128} operations are required. In summary, the shortcut makes the attack *deterministic* as opposed to *probabilistic*.

A.4.3 Secret Suffix

A brute force attack on the secret suffix variant is similar in that the intruder would need to try on the average 2^{128} possibilities before discovering an $\hat{S}$ such that $MD4(M||\hat{S}) = MD4(M||S_{AB})$. However, the probability that $\hat{S} = S_{AB}$ is only $1/2^{384}$ (since there are, on the average, 2^{384} 512-bit blocks which are hashed into the same 128-bit MD4 value). On the other hand, finding an $\hat{S}$ such that $\hat{S} \doteq S_{AB}$ is, for all purposes, equivalent to finding S_{AB}.

At this point in time, because of MD4's relatively recent introduction, and, hence, limited analysis thereof, little can be said with certainty about the average size of suffix-equivalence classes. One conjecture that can be made with some degree of certainty is that the size of an average suffix-equivalence class is no greater than that of an average prefix-equivalence class.

This does not imply, however, that it takes at most 2^{128} operations to find $\hat{S} \doteq S_{AB}$. Note that the intruder has to record multiple *genuine* messages and compute their respective digests at each step of the attack. Thus, the number of operations required to find $\hat{S}$ is $N \times 2^{128}$ where N is the size of the intruder's *message test pool.*

We also note that, in the suffix variant, there appears to be no counterpart to the shortcut described in the previous section.

A.5 Cost

It is evident from the above discussion that, provided MD4's conjectured properties hold, both secret suffix and secret prefix variants provide strong protection against unauthorized message fabrication and tampering. Just as there are some subtle differences among the two variants with respect to their ability to withstand attacks, there are also differences in their respective cost.

The prefix variant costs close to nothing, since the intermediate MD4 value of the secret prefix can be pre-computed and used as a sort of a *initialization vector* for verifying message signatures. Furthermore, the concept of a *prefix* can be abandoned altogether in favor of a 128-bit *MD4 initialization vector.*

Secret suffix costs slightly more as no pre-computation can be done. The cost amounts to exactly one extra 512-bit block MD4 computation per message.

A.6 An Extension

At this time, any protocol requiring on the order of 2^{128} operations to defeat is considered strong, i.e., secure. Nevertheless, it is conceivable that a need for stronger protocols may arise in the future. For this reason, we consider a combination of the secret suffix/secret prefix variants. In this third variant, the MD4 value of a message is computed by using a secret initialization vector and by appending an (also secret) 512-bit suffix. The cost of this variant is equivalent to the cost of the suffix method alone[5], while the brute force attack would require 2^{256} operations. This is significantly stronger than either of the two methods discussed thus far.

A.7 Applications

The secret suffix/prefix methods are particularly applicable to environments where high bandwidth and low delay requirements have, until now, made the implementation of certain security services prohibitively expensive, e.g., packetized voice and video applications. Also, network layer protocols, e.g., Inter-Domain Policy Routing (IDPR) [84] and *Visa* protocol [27], can benefit substantially from encryption-free message authentication.

A.8 Summary

In conclusion, fast one-way hash functions such as MD4 can be used as a foundation for some novel implementations of security services. In particular, simple and inexpensive secret prefix and secret suffix methods provide protection against message substitution attacks when used in conjunction with a strong one-way hash function (which protects against message modification). The strength of the methods presented herein is largely dependent on the ability of the underlying hash function to withstand hostile attacks.

[5] As mentioned above, the secret prefix method adds no cost to digest computation.