Security Issues in Policy Routing

Deborah Estrin and Gene Tsudik[1]

Computer Science Department
University of Southern California

Abstract

Routing mechanisms for inter-autonomous region communication require distribution of policy-sensitive information as well as algorithms that operate on such information. Without such *Policy Routing* mechanisms, it is not possible for interconnected regions to retain their autonomy in setting and enforcing policy while still achieving desired connectivity. This problem of interconnecting and navigating across autonomous regions is of inherent interest to the security community because the policies in question concern control of resource access and usage. Moreover, the security of the Policy Routing protocols themselves must be considered if they are to be applicable in sensitive environments. On the other hand, as usual, the security mechanisms take a toll in overall system complexity and performance.

Most routing protocols, including proposed *Policy Routing* protocols [1], focus on environments where detection of an attack after it has taken place is sufficient. The purpose of this paper is to explore the design of Policy Routing mechanisms for sensitive environments where more aggressive *preventative* measures are mandated. In particular, we detail the design of four secure protocol versions that prevent abuse through cryptographic checks of data integrity. We analyze and compare these schemes in terms of their per-packet processing overhead. We conclude that preventative security is feasible, although the overhead cost is quite high. Consequently, it is critical that prevention-based schemes coexist with detection-based schemes.

1 Introduction

In order to realize autonomy and resource control in an increasingly open and heterogeneous environment, controls must be added to protocols at the expense of packet processing speed - a familiar tradeoff to the security community! This paper explores the design and performance issues of one important class of resource control protocols, Policy Routing.

In large internets that cross organizational boundaries, routes must be selected according to policy-related parameters such as cost and access rights, in addition to the traditional parameters of connectivity and congestion. In other words, Policy Routing (PR) is needed to navigate through the complex web of policy boundaries created by numerous interconnected Autonomous Regions. This paper is concerned with the *security* of policy routing mechanisms, i.e., their vulnerability to threats such as forgery, modification of data, and denial of service.

Policy Routing is fundamentally of interest to the security community because it addresses issues of resource control and accounting. Moreover, PR protocols themselves can be designed with varying levels of security. In some environments relatively vulnerable Policy Routing mechanisms may be used in conjunction with *post facto* detection mechanisms.

Most of the work in PR protocol development is being done with such environments in mind [1, 3]. However, this paper addresses those environments where post facto detection is not acceptable or possible in an adequate and timely manner. In particular, we address the costs of building more aggressive *preventative* security measures into PR schemes.

The paper is organized as follows. Section 2 provides the background for our discussion with a description of autonomous network interconnect problems, the role of Policy Routing, and an introduction to D. Clark's Policy Routing proposal[1] (which is used as an example throughout the remainder of the paper). Section 3 outlines the security issues that are particular to PR protocol design. We continue our discussion of security in the remaining sections on secure protocol design (Section 4), and cost assessment (Section 5). Section 6 concludes with analysis of our secure protocol proposals and discussion of outstanding research and implementation issues.

2 Policy Routing and Interconnection of Autonomous Networks

In an environment of interconnected Autonomous Regions (ARs) some ARs are interested only in controlling end-system communication with the outside world. Other ARs are interested in sharing communication resources as well by providing transit services to, and obtaining transit services from, other ARs. In previous work we addressed the need to control communication across the boundaries of autonomously developed, owned, and operated networks and systems, referred to as Inter-Organizational Networks or Interconnected Autonomous Regions [5, 6]. In particular, the authors developed a family of *Visa* protocols that allow individual ARs to control communication between internal and external entities[2, 4]. That design process revealed that from a performance perspective it is not feasible to use *Visa* protocols as a means for controlling transit traffic.

D. Clark [1] has since proposed a protocol for expressing policy constraints on transit traffic through the process of inter-AR route synthesis. In this protocol, summarized in the following section, information about policy constraints on transit traffic is distributed as a part of the inter-AR routing protocol and is thereby kept current by each AR in a Policy Routing table. When routing decisions are made, the source AR refers to its Policy Routing table. Since policies change relatively slowly, it is likely that Policy Routes are still valid at this time. Because authorization is established before the actual packet transfer, this scheme has lower overhead as compared with *Visa* protocols.

[1] Names are listed in alphabetical order

CH2703-7/89/0000/0183$01.00 © 1989 IEEE

However, Clark's PR protocol does not *prevent* the unintended use of communication channels. It is intended for environments where post facto *detection* of abuse is sufficient. Consequently, the integrity of the policy routing control information and the authenticity of the packet source and contents, are not checked regularly. In this paper, we demonstrate how Clark's PR paradigm can be adapted to environments where authentication and data integrity checking are necessary for more aggressive prevention of unintended resource use. We then analyze the costs of these adaptations.

After defining basic terms used throughout the paper, the remainder of this section summarizes Clark's PR protocol [1]. We use this PR protocol as a platform for evaluating security design and costs. Since any Policy Routing mechanism will have basic elements in common with Clark's, we intend our discussion to be relevant to Policy Routing in general.

2.1 Definitions

In this paper we make use of the following definitions and assumptions:

- Policy Terms (PTs)
  Policy Terms are the units of routing information exchanged by communicating ARs. Each PT represents a distinct policy of the AR that synthesized it. The format of a PT is:
  $[(H_{src}, AR_{src}, AR_{ent}), (H_{dst}, AR_{dst}, AR_{exit}), UCI, Cg]$
  The purpose of a PT is to specify that packets from some host, H_{src}, (or a group of hosts) in a source AR, AR_{src}, are allowed to enter the AR in question via some directly connected AR, AR_{ent}, and exit through another directly connected AR, AR_{exit}, on its way to a host, H_{dst}, (or a group of hosts) in some destination AR, AR_{dst}. User Class Identifier (UCI) allows for distinguishing between various user classes, e.g., Government, Research, Commercial, Contract. Global Conditions (Cg) represent billing and other variables.

- Policy Route (PR)
  A Policy Route represents a sequence of Autonomous Regions (ARs), embedded in Policy Terms, that is traversed by packets between a unique source, destination host pair. Policy Routes are issued by a Policy Server (PS) in a source AR. A Policy Route is valid until either explicitly revoked by the issuing entity or revoked by one of the intermediate ARs.

- Policy Route Header (PH)
  In addition to the PR, the header contains user class identifier (UCI), Source AR, Charge Code (CC), Global Conditions (Cg), and a cryptographic seal, the contents of which are described below. As described in later sections, the Policy Route Header may be abbreviated in subsequent packets belonging to a stream. An abbreviated version of a Policy Route Header is referred to as a *handle.*

- Policy Server (PS)
  A Policy Server (PS) is an entity that collects Policy Routing information from remote ARs, distributes local policy information to remote ARs and synthesizes, as well as issues, PRs to local hosts or designated routers acting on hosts' behalf.

- Policy Gateway (PG)
  Policy Gateways are entities that in addition to the usual task of forwarding packets handle validation and verification of the PRs attached to the incoming packets. In other words, a PG is a reference monitor that enforces policy, and a PS is a policy generator.

To simplify our discussion we assume that multiple PSs and PGs within an AR are identical, i.e., share the same keys, as well as routing and charging information.

2.2 Summary of Clark's PR protocol

Following are the highlights of Clark's PR scheme:

- A Policy Route (PR) is a series of ARs, not a series of physical networks. In other words, there may be multiple physical realizations of a PR given multiple physical connections between ARs. However, Clark leaves the particular physical path selection between two neighboring ARs to the individual ARs to decide at packet forwarding time, rather than to the source AR to decide at route synthesis or selection time.

- Policies are expressed by source, destination, and all intervening, ARs. The source AR selects all constituent ARs in a PR, while transit and destination ARs control only which source and destination ARs can communicate via which directly connected ARs. In other words, transit and destination ARs do not exert control over the entire PR.

- ARs exchange Policy Terms (PTs) of the form described above.

- ARs run routing algorithms (similar to link state algorithms) to maintain their respective PR tables. There may be multiple PRs listed for a single destination, each with a different set of conditions associated with its use.

- A connection or a stream begins with a first packet carrying the full PR header which contains the ordered list of ARs to be traversed. Policy Gateways along the way validate that the PR listed agrees with the local PTs (through use of templates, for example). The result is cached so that a shorter handle can be used in the future to refer to the cached entry.

- Successive packets carry the handle to the cache entry, not the full PR header. PGs use handles in the packets to check for cache entries. PGs also relate return flow packets with forward flow. This aspect of Policy Routing exemplifies what Clark refers to as "soft state" in PGs.
- Given information about the next AR for a particular packet, each PG selects the next PG based on information exchanged in a more or less traditional up-down protocol.

Below is a simple example of a collection of interconnected ARs and their respective PTs. Note that the represented organizations may only be a subset of a larger internet, i.e., we assume that these policies exist in a larger context of numerous national networks, regional networks, universities and commercial organizations. In this example we represent the policies of two companies, A and B, two universities, C and D, a regional network E, and a national research network, G[2]. We make use of locally-defined categories to indicate groups of ARs in the policy terms: F refers to federal agency networks (or ARs), Re to regional, U to university, and Co to corporate. For simplicity's sake, we assume that the policies for each AR in this internet are symmetric [3]. The topology of this mini-internet is shown in Figure 1.

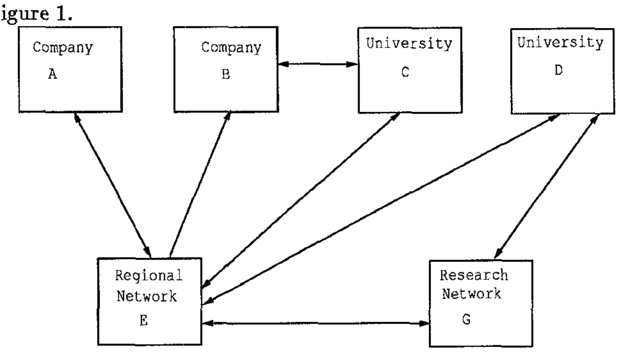

Figure 1: Topology of the example internet.

The desired policies and associated policy terms are detailed below (we make use of two conventions introduced by Clark: * is used as a wild card, and - indicates no transit when it appears in the ARent or ARexit field).

- A will accept traffic from/to directly connected federal or regional (F/Re) networks if it is for research. No transit is allowed.
  A1: (*,{F/Re},{F/Re}) (*,A,-) {research} {}
  A will accept traffic from/to universities, via Gnet, to/from hosts in A so long as the user is registered. No transit via A.
  A2: (*,U,G) (*,A,-) {registered} {}
- B will accept traffic from/to itself to/from directly connected networks. No transit is allowed.
  B1: (*,*,*) (*,B,-) {} {}
- University C will carry traffic to/from itself from/to any destination via the regional network so long as it is for research or support. No transit via C.
  C1: (*,C,-) (*,*,E) {research,support} {}
  University C will carry all traffic to/from G so long as it is for research/support. In other words, C will carry *transit* traffic to/from G only.
  C2: (*,G,G) (*,*,*) {research,support} {}
- University D will carry traffic to/from itself from/to any destination via any directly connected network, so long as it is for research/support. Transit is OK.
  D1: (*,D,*) (*,*,*) {research,support} {}
- Regional Network E will carry traffic from/to any directly connected network to any network via a commercial carrier regardless of its UCI. In this case the packets accrue charges since the commercial carrier charges per kilopacket.
  E1: (*,*,*) (*,*,*) {} {unauthenticated UCI,charge/Kpkt}
- G (a Federal agency) will carry traffic for any host connected to a F/R network talking to any other host connected to a F/R via any F/R entry and exit network, so long as there is it is being used for research or support. There is no authentication of the UCI and no per packet charging.
  G1: (*,{F/Re},{F/Re}) (*,{F/Re},{F/Re}) {rsrch,suprt} {unauthenticated UCI,no charge}

3 Security Issues in Policy Routing

This section addresses some of the potential security concerns in the presence of a Policy Routing mechanism.

3.1 Potential threats

Three basic threats to the security of PR protocols are: falsification of Policy Routes and related information, misuse of valid PRs, and denial of service [4].

1. Policy Routing information may be falsified in order to obtain unauthorized resource usage. In particular, an intruder may:

[2] The organizations and policies in this example are not real. However, they do represent the types of policies that such organizations express as desirable.

[3] Clark's syntax can be used to express asymmetric policies as well.

[4] Clark groups the threats into three types, falsification of PR header content, misuse of a PS generated PR, and misuse of some component of a PS generated PR header. We consider the third type to be a special case of the second.

   (a) Create its own PR.
   Since policy terms (PTs) are distributed widely, an intruder can collect PTs and use them to create its own Policy Route Header information [5]. This attack results in the accrual of charges to an inappropriate charge code or unauthorized access to a particular host pair. In order for this attack to be effective and to avoid immediate detection the intruder must also re-route return packets.

   (b) Substitute a Charge Code
   Alternatively, an intruder can steal a PR header created by an authorized PS and substitute an invalid Charge Code. This second attack is basically a variation on the first one listed. In both cases, some part of the PR header is modified.

2. The second class of attack is misuse of Policy Routing information. In this case an intruder may:

   (a) Make use of a valid PS-generated PR header and append its own data to packets in an authorized stream. In this case as well, return packets must be re-routed by the intruder.

   (b) Replay previously-recorded legitimate packets with the PR header intact.
   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.

   (c) Misuse a PS-generated Policy Route and/or related charging information without modifications of either PR header or data.
   In this case the intruder is a legitimate user. Misuse includes: unauthorized release of information, overcharging a (valid) charge code and using "stale" Policy Routes. As is the case with most "internal" threats, this issue is the most difficult to address within the technical design. One containment measure is to associate an expiration time and/or include a maximum packet (or byte) count with each Policy Route issued. All other mechanisms are of a post facto nature.

3. Denial of service.
   This class of attack overlaps with both preceding classes. While the intruder's goals are different, the means of attack are the same as in the case of misuse and/or falsification threats. In addition, an intruder can masquerade as a PS and distribute bogus PTs. This may cause other ARs to synthesize PRs that will be rejected, resulting in denial of service.

In the remainder of this paper we describe and analyze mechanisms to prevent 1a, 1b, 2a, 2b, and 3. As might be expected, cryptographic sealing is the building block for our protocols and PSs play the primary role in sealing PRs. Our approach is not inconsistent with Clark's original proposals. However, we propose applying *preventative* data sealing to both aspects of attack. In addition, we propose sealing handles as well as PR headers, providing per-packet data integrity checking on an optional basis, and accommodating suspicious PSs that do not all share the same key.

3.2 Elements of a Secure PR Protocol

In order to combat the security threats described in Section 3.1, each transit AR must have the means to:

1. Verify that the entire PR is issued by a valid, recognized PS in a known AR. When a PR is computed, the issuing PS must distribute it in a secure fashion to all transit ARs in the route. For each transit AR a packet containing the PR option must be sent, sealed with a key known only to the issuing PS and the target PS in a corresponding AR. All N copies (N is the number of transit ARs in the PR) of the PR may be included in the packet header. However, this represents wasted bandwidth. An alternative is to compute a signature of a PR for each of the intervening PSs, keep the PR itself in the clear, and send
   $$[PRoption, SIG_1, .., SIG_N]$$
   in the PR header. This would save bandwidth (only one such packet is needed) but would leave the PR option exposed. To minimize both bandwidth overhead and PR vulnerability, routes may be validated infrequently, i.e., during the initial setup and whenever the cache in one of the transit PSs overflows and the PR info is lost. This reconfirms the necessity for "soft state" in policy gateways, suggested by Clark [1] (or "flow state", also suggested by Clark in [8]).

2. Verify the authenticity of the sender by checking certain field(s) encrypted with a key known only to the sender and PSs in the route. This session key can be a part of the original PR option distributed to all intervening PSs and the source host.

3. Verify that the intended destination is in fact the same as the one sealed by the issuing PS (this may also be a part of the originally issued PR).

4. Check that the data has not been modified or substituted, i.e., check data integrity. For some environments this may be optional.

5. Verify that the packet is not a replay of a previously recorded valid packet using an existing Policy Route (for some environments this may be optional).

[5] This includes UCI, AR_{src} and AR_{dst} addresses, CC, Cg and the cryptographic seal.

3.3 Policy Servers and Key Management

An important issue in designing a secure Policy Routing scheme is the assumption of global trust among all Policy Servers (hence, all ARs). If all ARs do trust one another then we can avoid a key explosion by having each PS uniquely identified to all other PSs by a single key. On the other hand, this assumption implies that *all* participating ARs must agree whenever a new AR is admitted into the internet because of the degree of global trust attributed to each participating PS (e.g., access to *the* key that all PSs use to authenticate themselves). The environment where PSs enjoy global trust will be referred to as "trusted PS", and that where no such trust exists, as "mutually suspicious PS".

If PSs are not globally trusted, then for each pair of communicating PSs (PSs that exchange Policy Routing information), a unique key will be necessary. In general, if a unique key is required for every pair of communicating ARs, the total number of keys is at most *(n-1)*n/2* (where n is the total number of ARs). After the initial expensive setup stage (key generation and assignment) the number of keys that are kept by each PS in the mutually suspicious PS environment is actually quiet small (n-1 for each PS). Moreover, not all ARs will engage in communications with all other ARs. If the policy terms of AR_a exclude handling traffic for AR_b, then there is no reason for PSs in AR_a to have means for secure communication with PSs in AR_b. Thus, in reality, the total number of keys is bound to be somewhat lower than the worst case of *(n-1)*n/2 keys.* Nevertheless, one benefit of the "trusted PS" environment is clear: the total number of keys is low and manageable.

However, of equal or even greater importance is that Policy Route authentication is facilitated in the "trusted PS" environment. A PS that issues a Policy Route can sign and send one copy of the PR to all intermediate ARs contained in the route, whereas in the "mutually suspicious PS" environment a PR (or a portion thereof) must be sealed individually for each transit AR in the route.

The "trusted PS" approach, however, has a critical drawback. Upon gaining control of just one participating PS, an intruder can impersonate all (or a large subset of) PSs in the Internet by using their corresponding keys. In the "mutually suspicious PS" environment, on the other hand, gaining control of a PS gives the intruder the ability to impersonate only the PS in question.

Another parameter in the design of our protocol and the operation of PSs, is the use of Public vs. Private key schemes for sealing PRs. Unfortunately, available Public Key hardware is significantly slower than that of Private Key [6]. Therefore, the rest of the paper will concentrate on applications of Private Key cryptosystems such as DES[7][7].

Assuming the use of private keys, and given the need to send uniquely sealed PR data to each intervening AR, there is no longer any reason to send the entire PR to each transit AR. Clark's policy terms do not allow anyone other than the source AR to control the selection of the entire PR. In other words, each transit AR can be sent its own packet listing only the relevant

$$[AR_{src}, AR_{dst}, AR_{entr}, AR_{exit}]$$

sealed in the appropriate key for that AR. Alternatively the relevantly sealed information for each AR can be concatenated into one large packet with the sealed PR information back to back in the header.

4 Secure PR Protocol

In this section we detail the steps needed in a secure protocol and discuss issues dealing with data integrity checking. We assume the more general case of a mutually suspicious PS environment. An optional replay prevention scheme is also presented.

4.1 General scheme

In the following protocol description we assume that the PTs necessary to synthesize a PR are available to the issuing PS. The information derived from PTs includes, but is not limited to: data authentication parameters (if required, one of the methods described in Section 4.2 must be identified) and terms of usage (e.g., time-of-day restrictions, maximum packet count and billing conditions).

All variations of the secure PR protocol include the following steps:

1. Upon issuance of a valid PR, the PS_{src} in the source AR, AR_{src}, generates a session key, K_i.

2. For all AR_j in the PR, let Control Packet of AR_j, $CP_j = [AR_{ent(j)}, AR_{exit(j)}, K_i, H_{src}, H_{dst}, AR_{src}, AR_{dst}, Cg]$ where $AR_{ent(j)}$ and $AR_{exit(j)}$ represent ARs through which packets from H_{src} to H_{dst} will enter and exit AR_j, respectively.

3. For all AR_j in the PR, PS_{src} computes $[CP_j]^{KEY^j_{src}}$ and sends it to AR_j. Since the size of each CP_j is constant and quite small (perhaps, around 24 bytes), PS_{src} can concatenate all CP_j-s and send them in one packet thus reducing overhead slightly[8].

4. In each transit AR_j some PS_j will receive a packet containing CP_j. It will decrypt the packet using the same key, KEY^j_{src}, and determine whether or not the route is

[6] See Section 5.5

[7] Although currently inadequate in terms of performance, Public Key encryption schemes have a number of well-known advantages over their Private Key counterparts. Therefore, the design presented here may be adapted to Public Key or hybrid encryption schemes in the future.

[8] KEY^j_{src} is used to denote a key used for communication between AR_{src} and AR_j.

valid, charging information is correct, global conditions are acceptable, etc. Thereafter, CP_j is stored in a cache of each transit AR's PS. If the route is invalid the packet is dropped. PS_j may optionally notify the sender by returning the packet and specifying the reason.

5. Next, $CP = [H_{src}, H_{dst}, K_i]$ is released to H_{src} or a designated router acting on H_{src}'s behalf.

6. When H_{src} wants to send a packet to H_{dst}, it (or a designated router) attaches the following header to the packet: $PH = [H_{src}, H_{dst}]^{K_i}$. A header may also optionally contain the packet data signature computed with K_i and/or "secret" sequence number for replay prevention (see Section 4.3). This block of data is intelligible only to:

   (a) The issuing PS

   (b) All intervening PSs in transit ARs in the PR

   (c) H_{src} or its designated router

7. When a packet purporting to be from H_{src} to H_{dst} arrives into AR_j, the corresponding PS looks up an entry in its cache according to the $[H_{src}, H_{dst}]$ pair found in the IP header.

   (a) If an entry is found:
      - PS decrypts PH using K_i found in the cache.
      - Makes sure H_{src}, H_{dst} in the cache entry match those found in the decrypted PH.
      - Optionally, verifies data integrity of the packet. This depends upon which variation is implemented (see Section 4.2).
      - Optionally, verifies that the packet is not a replay (see Section 4.3 for explanation).

   (b) If no entry is found then:
      - an entry was purged because of a cache overflow, or
      - a route within an AR has changed, or
      - the packet is bogus

   To avoid unnecessary delays, PS may do one of three things:
      - drop the packet
      - drop the packet but send back a request for a copy of a PR
      - send on the packet and send back a request for a copy of a PR and mark a "pending" entry in the cache. In this case, it will refuse to accept any more packets from H_{src} until a copy of PR is received.

4.2 Data Integrity

A critical dimension of secure PR protocol design is the degree and granularity of data integrity checking. In the protocol described below, we provide for four variations: endpoint-AR, transit, source-patterned and round-robin. In Section 5 the performance and cost of each scheme is analyzed.

- Transit-AR
  In network environments where data integrity and security concerns outweigh the overhead of extra processing, the data portion of every packet is subject to forgery and must be authenticated at each hop on its way to the destination. The protocol for this class of environment has the highest overhead, commensurate with security requirements.

- Endpoint-AR
  If authenticating data in each packet at every hop is prohibitively expensive, end-to-end data integrity similar to that in *Visa* protocols may be appropriate. This 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 AR. This can result in inappropriate billing of the source. On the other hand, this approach benefits from lower per packet overhead which is independent of the PR's length.

- Source-Patterned
  While we would like to reduce per packet overhead due to encryption, the issue of data security and timely detection of forgeries needs to be addressed. Instead of each transit AR having to authenticate each packet, it may suffice to authenticate every m-th packet. In the simplest version of this patterned authentication scheme, AR_{src} would choose m at random from a locally defined range of values and then specify m during route setup. Transit ARs would either accept or reject the proposed m. If all ARs accept the proposed value for m, then every AR will check data integrity of every m-th packet. If any AR does not accept m (if it is considered too large or too small) then the source and all other ARs 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.

- Round-Robin
  This scheme achieves constant per packet overhead by using "round-robin" data authentication. Transit ARs take turn authenticating packets. In general, packet number K is authenticated by a PS in $AR_{[KmodM]}$ where M is the number of ARs in the PR. The overhead is reduced to just

two encryptions per packet (regardless of the route). End-to-end checking can be added for extra assurance at the cost of a single additional decryption by the destination. On the other hand, inter-AR independence is sacrificed due to the coordination required to set up the round-robin arrangement. However, unlike the previous scheme, lost and out-of-order packets can be accommodated easily. While this approach benefits from fair sharing of encryption costs among transit ARs, it is only worth considering in cases when the number of transit ARs is large, i.e., the PR is long.

4.3 Preventing Replay

Replay is of concern in this environment because it can lead to unjustified charges and denial of service. Two sources of replay are of equal concern. 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 generating replay packets in order to deny resources (or inflate costs) to the rightful owner. Neither kind of replay can be handled on a strictly end-to-end basis because by the time the duplicate packet is discovered, the resources may have been used and associated charges incurred, e.g., the bill reflects the replayed packets 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. In more sensitive environments, more aggressive prevention is required, albeit at significant cost. Since this paper is analyzing the implications of PR in sensitive environments, we discuss the design of a protocol for preventing replay.

Two separate issues must be addressed: replay of PS-to-PS packets and replay of data packets. Replayed PS-to-PS packets can reestablish a previously used flow resulting in denial of service. Because PS-to-PS packets are fewer in number and much less frequent than data packets, we can use timestamps to distinguish between different packets [9]. This requires that the clock skew between any two communicating PSs be either known or negligible. Alternatively, nonce identifiers similar to those proposed by Needham and Schroeder[10] can be utilized. However, this second approach requires that PSs keep a cache of recently used identifiers for integrity checking, while timestamps can be validated via comparison with a clock value.

Replay of data packets can result in inappropriate charges and denial of service. Data packets, due to their density, can not be easily accommodated by the timestamping protocol. Therefore, a nonce identifier must be present in each packet. A secret sequence number can be thought of as an example of a nonce. Consider the following protocol steps:

1. When a PR is issued, the issuing PS generates a random number, a nonce. This number, hereafter referred to as a seed, will serve as the initial sequence number for the PR.
2. The seed is then distributed in a secure fashion (as a part of PR) to all intervening PSs in transit ARs. If the seed is not available to a potential intruder, chosen plaintext attacks can be avoided.
3. When a host, H_{src}, has a packet to send, it (or a designated router) modifies the seed and includes it in the (encrypted) PR option in each packet.
4. When a packet reaches a policy gateway, the PR option is decrypted and the sequence number found in it is compared to the expected sequence number found in the gateway's cache. Three outcomes are possible:
   (a) Sequence numbers match.
   The packet is forwarded and expected sequence number is incremented.
   (b) The sequence number is greater than expected.
   The difference between the two numbers must be examined. If the difference is small, i.e., less than some threshold, T, the packet can be forwarded. Otherwise the gateway suspects foul play and requests an updated sequence number from the issuing PS.
   (c) Sequence number is less than the threshold T.
   The packet is dropped.

   Case (c) is quite "unforgiving" if T is small. However, mechanisms to support out-of-order packets are complicated and costly. If out-of-order arrivals are to be supported, the mechanism of choice would be a sliding-window scheme. The biggest problem is selecting an optimal window size that is acceptable to all intervening ARs. In this situation, robustness of end-to-end connectivity across ARs is traded off for security and performance of individual ARs.

An intruder's chances for subverting the protocol are further reduced if, in addition, packet data integrity is verified at each intervening PS in the PR.

One alternative to (secret) sequence numbers is the use of "electronic tickets", a concept described in [11]. In this model, pre-paid tickets are included in every transaction and serve as proof of resource usage. Electronic tickets are self-verifying and limited in number, thus accommodating out-of-order packets without compromising security. However, this concept, while similar in nature to sequence numbers, requires that all interested parties (intervening PSs) be able to distinguish between fake and genuine tickets. This can be achieved easily for PS-to-PS packets. However, keeping track of valid ticket "stubs" for data packets is much more difficult due too their greater numbers.

Authentication Method	Cost in # of encryptions
None (Clark's PR)	NONE
Endpoint-AR	2
Transit-AR	N
Source-Patterned	N/m
Round-Robin	2

Table 1: Encryption Operations.

5 Assessment and Cost of Security

Our purpose is to investigate the bounds on achievable data rates with the security schemes described above. Previous work in the area of performance and cost evaluation of secure protocols (e.g., *Visa* Protocols [2]) suggests that the following four factors comprise most of the cost of secure mechanisms:

1. Per Packet Encryption
2. Increased Packet Length
3. Additional Packets Generated
4. Other Additional Per Packet Processing, e.g., cache lookups

Results (see Table 3) show that that overhead due to encryption constitutes the majority of the total overhead. In the remainder of this Section we analyze each of the above contributing factors in several variations of the general scheme.

5.1 Per Packet Encryption Costs

For both trusted and mutually suspicious PS environments, per packet encryption costs are summarized in Table 1. N refers to the PR length including AR_{src} and AR_{dst}, and m refers to the index agreed to in the source-patterned protocol[9]. Replay prevention can be used independent of the data authentication method. The cost of replay prevention amounts to one small (PR-header) encryption for the source host and all intervening ARs that choose to implement it.

5.2 Costs Due to Increased Packet Length

Increased packet length is incurred solely due to the PR-handle carried in every data packet. It is anticipated that the length of this handle will be on the order of 20-30 bytes. Previous measurements of *Visa* Protocols[2] show that this overhead ranges from 20% for small (e.g., 16 byte) packets to less than 4% for larger (e.g., 1K byte) packets; the length of *Visa* handles is roughly the same as that of PR handles.

5.3 Additional Packets and Initial Setup Costs

The additional packets generated by the PR protocol include packets generated during initial PR setup and those due to cache misses in intermediate ARs. Their number depends on the following:

- Environment (trusted or mutually suspicious PS)
  In a trusting environment one packet containing the entire PR is signed and sent along the route chosen. In a mutually suspicious PS environment either one packet containing individually signed portions (CP_i-s) of the PR or N (N is the length of the PR) small packets are sent. The difference in length is minimal. The encryption costs are as follows: in trusting PS, the entire PR must be signed and decrypted by all intervening ARs, whereas in mutually suspicious PS, the PR is signed individually for each AR in the PR and each AR only decrypts a small constant portion of the route, CP_i.
- Cache sizes and cache replacement algorithms used by intervening Policy Gateways.
  When a cache miss occurs at an intermediate PG, that PG must be supplied with a fresh copy of a relevant PR portion, i.e., CP. This requires that a fixed-length packet (around 20-30 bytes) be sent to the said PG by the issuing PS. The overhead is expected to be minimal, unless misses take place often enough to overwhelm the issuing PS. If the number of active PRs in a PS measures in hundreds, cache sizes and lookup costs can be kept small. If an LRU cache replacement algorithm is used, a hit ratio in the 90% range can be achieved.

5.4 Per Packet Processing Costs

Additional (other than encryption) processing costs are incurred mainly by the added logic in gateways for processing of PR-based packets, in particular, cache lookups. As discussed above, cache sizes are expected to measure in hundreds of entries. Similar experiments show that time spent on cache lookups is overshadowed by encryption costs[2].

5.5 Cost example

In order to demonstrate the magnitude of potential overhead due to security mechanisms we present both theoretical and experimental results obtained from analyzing encryption and PR-header length overhead for all data authentication methods described in Section 4.2. All calculations and experiments were conducted using a "mock" internet with a PR of length three – $[AR_{src}, AR_{trns}, AR_{dst}]$. The length of this PR is not arbitrary as many PRs are expected to traverse only one transit AR.

[9] Note that when no data authentication or replay prevention is done, PR-header authentication involves only a cache lookup at each intervening PS. This is because encrypted, as well as decrypted, PR-header values can be cached.

In Table 3, we present data for five data integrity variations: a) Clark's original proposal without per-packet data integrity checking, b) endpoint-AR and round-robin, c) transit-AR, and d) source-patterned. The first sub-table in each of (a-d) represents theoretical calculations based on the encryption rate of 1.0 Mbyte/sec – a readily attainable speed for several commercially available DES hardware devices. The second sub-table in each of (a-d) is based upon measurements in a laboratory implementation with (significantly) slower prototype hardware. Both sets of measurements represent very conservative estimates since much faster customized encryption hardware is available. The experimental setup, depicted in Figure 2, consists of additions to IP running under 4.3 BSD on IBM RTs[10]. Measurements were taken for packets traveling from AR_a to AR_c via AR_b that traversed the route: $[H_a, PS_a, PS_b, PS_c, H_c]$. All three ARs are physically located on a single ethernet but are implemented as logically separate networks. DES encryption is performed in hardware, using prototype cards from the Information Technology Center of Carnegie-Mellon University (CMU-ITC). Although the AMD AMZ8068 chip used on the card is specified to encrypt up to 1.7 Mbyte/sec[12], the board itself encrypts large data blocks at approximately 0.2 Mbyte/sec due to slow I/O on the prototype card[11]. The numbers in Table 2 are constant for all data authentication schemes analyzed.

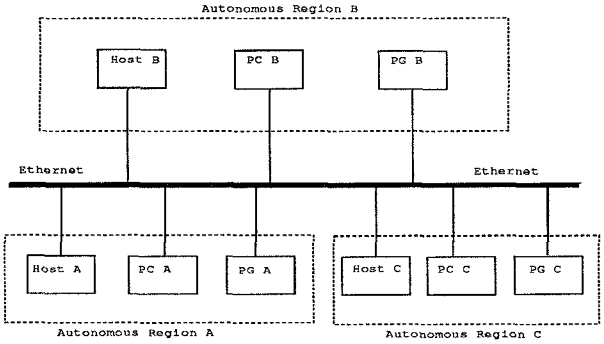

Figure 2: Topology of the laboratory internet.

Looking at the theoretical calculations, the overhead for Clark's scheme (between 6% and 25%) represents the cost of the PR header processing alone, with no data integrity checks. It is presented here as a basis for comparison with the other four schemes. The endpoint-AR scheme adds cryptographic data integrity checks at the source and destination ARs. The total overhead of this PR protocol ranges from 12% to 27%. The same overhead is incurred by the Round-Robin scheme where each packet is checked by only one of the ARs. The transit-AR scheme implements cryptographic data integrity checks at all transit ARs and, therefore, incurs a higher overhead of 18% to 28%. Finally, the source-patterned scheme only checks data integrity of every m-th packet (every m-th packet is treated as in the transit-AR), thus, achieving lower overhead of 9% to 26% for m=4.

	Packet Size		
	64	500	1000
Travel time clean(ms)	8.0	20.0	33.0
PR hdr overhead(ms)	2.0	2.0	2.0

Table 2: Packet travel times and PR header overhead.

	Packet Size				Packet Size		
	64	500	1000		64	500	1000
# of encryptions	0	0	0		2	2	2
			Theoretical				
Encr. overhead(ms)	0	0	0		0.13	1.00	2.00
Total overhead(ms)	2	2	2		2.13	3.00	4.00
% overhead	25.00	10.00	6.00		26.63	15.0	12.12
			Experimental				
Encr. overhead(ms)	0	0	0		2.5	7.75	13.25
Total overhead(ms)	2	2	2		4.50	9.75	7.00
% overhead	25.00	10.00	6.00		56.25	48.75	46.00

(a) Clark's

(b) Endpoint-AR and Round-Robin

	Packet Size				Packet Size		
	64	500	1000		64	500	1000
# of encryptions	4	4	4		1	1	1
			Theoretical				
Encr. overhead(ms)	0.26	2.00	4.00		0.06	0.50	1.00
Total overhead(ms)	2.26	4.00	6.00		2.06	2.50	3.00
% overhead	28.22	20.00	18.18		25.80	12.50	9.09
			Experimental				
Encr. overhead(ms)	5	15.5	26.5		1.25	3.90	6.65
Total overhead(ms)	7.00	17.50	28.50		3.25	5.90	8.65
% overhead	87.50	87.50	86.00		40.60	29.50	26.00

(c) Transit-AR

(d) Source-Patterned with m=4

Table 3: Expiremental/Theoretical Overhead Measurements.

The experimental laboratory results represent at least a 60% (and up to 370%) increase in overhead over theoretical calculations for each scheme that implements cryptographic data integrity checks. The reason for this discrepancy is the actual encryption speed of the prototype card available for the RT. In particular, the 1.7 Mbyte/sec rate advertised for the chip was only achieved for very large blocks of data. The CMU-ITC encryption hardware is a prototype design not intended for real-time applications. State-of-the-art DES hardware speeds are significantly higher[12].

[10] The IBM RT scores 2690 on the "Dhrystone Benchmark", compared with 2993 for SUN 3/50 and 1577 for Digital Equipment Microvax II.

[11] Our thanks to Paul Crumley of the CMU ITC for generously supplying DES cards and providing information and support.

[12] For example, the AMD chip number AMZ8068 is specified to encrypt up to 1.7 Mbyte/sec[12]. In contrast, available RSA speeds are still quite low (the fastest commercially available hardware, the Cylink Corporation chip number CY1024, is specified to encrypt only up to 2 Kb/sec [13]). Faster hardware for both DES and RSA is anticipated in the near future...of course.

In parts a, b and d (Table 3), the percentage overhead *decreases* rapidly as the packet size *increases*, i.e., per-packet overhead is relatively stable. In contrast, table c, shows that the percentage overhead for the transit-AR scheme decreases very slowly as the packet length increases. This is because: 1) a significantly larger number of cryptographic operations must be implemented in this scheme (one per transit AR), and 2) the time to execute DES cryptographic operations increases along with packet size.

In summary, based upon assumptions of attainable encryption speeds for available DES hardware, we calculate overhead to range from 9% for source-patterned method to around 28% for transit-AR. Experimental measurements confirmed the basic pattern of overhead predicted by our calculations (comparative costs and cost trends), and, therefore, validated the correctness of our cost model. However, the overhead was at least double that of the theoretical calculations due to the lack of available and affordable state-of-the-art DES hardware.

These results show that it is possible to integrate preventative security measures into PR protocols, although the overhead is still quite high. Nevertheless, the cost is tolerable and secure PR protocols should be deployable in sensitive environments if faster DES hardware becomes more readily available, and if prevention schemes can coexist with lower overhead detection-based schemes. Section 6 concludes with a final summary of our proposals and analysis.

6 Summary

Routing mechanisms for inter-autonomous region communication require distribution of policy-sensitive information as well as algorithms that operate on such information. Without such Policy Routing mechanisms, it is not possible for interconnected regions to retain their autonomy in setting and enforcing policy while still achieving desired connectivity. This problem of interconnecting and navigating across autonomous regions is of inherent interest to the security community because the policies in question concern control of resource access and usage. Moreover, the security of the Policy Routing protocols themselves must be considered if they are to be applicable in sensitive environments. On the other hand, as usual, the security mechanisms take a toll in overall system complexity and performance. The purpose of this paper was to explore the design of Policy Routing mechanisms for sensitive environments and to investigate the performance overhead of the proposed mechanisms. There remains much work to be done in several areas: simulating and experimenting with implementations of Policy Routing protocols, speeding up available hardware encryption rates, simulating the design alternatives proposed here for more detailed understanding of behavior and tradeoffs, integrating Policy Routing with end-to-end mechanisms, (e.g., *Visa*), and developing formal tools to verify the Policy Terms and routes synthesized by ARs[14].

This paper represents a first attempt at designing and analyzing policy routing protocols that *prevent* unintended use of resources. We concluded that, given todays encryption technology, data integrity mechanisms produce overheads ranging from 9% to 28% (for a sample 3-hop PR) depending upon the complexity and assurance provided by the protocol. These results show that while it is possible to implement preventative security measures in PR protocols, the cost is quite high. The cost will be tolerable, and therefore PR protocols will be deployable in sensitive environments, if prevention and detection based schemes can coexist and if faster DES hardware becomes more readily available.

Our proposed mechanisms were designed to support interoperability across ARs with heterogeneous policies to the extent that their combined policies allow. Moreover, these *preventative* PR schemes can inter-operate with *detection*-based PR schemes. In order to accomplish this, PR headers must encode the flavor of data integrity, replay prevention mechanisms must be employed for packets belonging to a stream, and each AR's Policy Term global conditions (Cg) must contain details as to data integrity checking and replay prevention requirements.

In summary, we designed preventative security mechanisms and analyzed their performance implications in the context of a particular PR protocol in order to provide a concrete basis for the evaluation of performance and implementation overhead. Any Policy Routing mechanism will have basic elements in common with Clark's. This includes the need to distribute policy information, select Policy Routes, represent identifying information in packet headers, and verify the authenticity of the PR's creator and the integrity of the packet data attached to the PR header. Consequently, the discussion provided here should shed light upon alternative proposals as well.

7 Acknowledgments

We are grateful to Dave Clark for providing early access to his ideas on Policy routing, and to Kim Korner, Tom Hinke and Sharon Anderson for their detailed comments on a previous draft of this paper. Also, many thanks to the anonymous reviewers for their comments. This research was supported by The National Science Foundation Presidential Young Investigator award and matching funds from GTE and NCR.

References

[1] D. Clark, *Policy Routing ın Internet Protocols, Version 1.1*, **Unpublished paper**, To be published as an Internet RFC. Available from the author at the MIT LCS, 545 Technology Sq, Cambridge MA 02139.

[2] D. Estrin, J. Mogul and G. Tsudik, *Visa Protocols for Controlling Inter-Organizational Datagram Flow*, **IEEE Journal on Selected Areas in Communications, Special Issue on Secure Communications**, Spring 1989.

[3] A. Nakassis, *Routing Algorithm for Open Routing*, **Unpublished paper**, Available from the author at the National Institute of Standards and Technology, Washington D.C.

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

[5] D. Estrin, *Controls for Inter-Organization Networks*, **IEEE Transactions on Software Engineering**, Vol. SE-13, No. 2, February, 1987.

[6] D. Estrin, *Interconnection Protocols for Inter-Organization Networks*, **IEEE Journal on Selected Areas in Communications**, Vol. SAC-5, No. 9, December, 1987.

[7] *Federal Information Processing Standards*, NBS, Publication 46, 1977.

[8] D. Clark, *Design Philosophy of the DARPA Internet Protocols*, **in Proceedings of ACM Sigcomm 1988**, pp. 106-114, August 1988, Palo Alto, CA.

[9] D. Estrin, J. Mogul, G. Tsudik, K. Anand, *Visa Protocols for Controlling Inter-Organizational Datagram Flow: Extended Description*, **University of Southern California TR-88-50. Also, DEC TR 88/50.** 1988.

[10] R. M. Needham and M. D. Schroeder, *Using encryption for authentication in large networks of computers*, **CACM**, vol.21, No.12, pp.993-998, December, 1978.

[11] J. Taber, *An Authentication Method Using DES*, **Unpublished paper**, Available from the author, IBM Corporation, Dallas TX.

[12] *Advanced Micro Devices MOS Mıcroprocessors and Perıpherals Data Book.*, Advanced Micro Devices, Inc., Sunnyvale, CA. 1987

[13] W. Diffie, *The First Ten Years of Public-Key Cryptography*, **Proceedings of the IEEE**, May 1988, pp 560-577.

[14] S. Anderson and D. Estrin, *Representıng and Analyzing Usage Policıes for Interconnected Autonomous Networks*, **University of Southern California TR-88-55**, 1988.