IP Security Policy Working Group
Internet Draft
<draft-ietf-zao-ipsec-policy-semantics-00.txt>
Expires September 2000

J. Zao
BBN Technologies

March 10, 2000

Semantic Model for IPsec Policy Interaction

Status of this Memo

1. Abstract

This Internet Draft discusses three possible forms of interaction among IPsec Policies referred to as Policy Resolution, Policy Correlation and Policy Reconciliation. It also proposes two operators, Policy Composition (*) and Policy Difference (-), to deduce the results of policy interactions. The definitions of these operators establish an algebraic semantics of IPsec Policies based on partially ordered set theory. This formal semantics can be used to ensure consistency of IPsec Policy processing.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

Due to limited availability of special symbols in ASCII character set, conventional mathematical symbols are overloaded in this document to represent logical, set and partial ordering relations. In most cases, the meaning of the symbols are clarified by their context. Notes are included in the cases that confusion may arise. Following are the designated uses of mathematical symbols:

Symbols	Logic Op.	Set Op.	Partial Order Op.
*	AND	Intersection	Composition
+	OR	Union	
-		Difference	Difference
~	Negation	Complement	
[i]		Element Indexing	
^			Least Upper Bound (LUB)
(.,.)			Coordinate Order
(.#.)			Lexicographic Order
(.:.)			Dependency Order

3. Introduction

The use of IPsec Authentication Header (AH) Protocol [3] and Encapsulating Security Payload (ESP) Protocol [4] is specified by packet processing rules commonly known as IPsec Policies. These policies prescribe cryptographic processing of selected IP datagrams in addition to passing and discarding these datagrams at end hosts and/or intermediate gateways. The algorithms and the keys used to perform the cryptographic operations are specified by security associations(SAs), which are negotiated between peer communication nodes using the Internet Key Exchange (IKE) protocol [5]. The IKE protocol permits either of the communication nodes (referred to as the initiator) to propose a suite of SAs and requires the other node (referred to as the responder) to choose a SA from the proposal suite. In most cases, however, the initiator should not construct its SA proposal suite based only on the knowledge of its own IPsec Policies. This is because the responder may have policies that imposed different security actions onto the datagrams. If the initiator proposes SAs that enforce only its own policies without honoring the responder's policies then the responder would have no choice but to reject all the proposed SAs. As a result, the attempt to establish a secure communication between the two nodes would fail. In order for the initiator to honor the IPsec Policies of both nodes, it will need a mechanism to fetch those policies and a method to deduce the resultant policy by combining the relevant policies. To ensure consistency and interoperability, the deduction should be governed by a semantic specification of IPsec Policy Interaction. This document proposes an algebraic semantics that can fulfil that role.

The main body of this document consists of five sections. Following this introduction, Sect. 4 discusses three different forms of IPsec Policy Interaction. Sect. 5 specifies the algebraic semantics we propose for IPsec Policy interaction. The specification includes a formal interpretation of IPsec Policies and the two basic operators, Policy Composition (*) and Policy Difference (-). The use of these operators in deducing the results of policy interactions is explained in Sect. 6. Remarks on security consideration are given

in Sect. 7. Disclaimer, full copyright statement and author's address are appended to the end of this document.

4. Forms of IPsec Policy Interaction

We identified three different forms of IPsec Policy Interaction. In addition to the one described in the last section, which is referred to as Policy Resolution, there are also Policy Correlation / Decorrelation and Policy Reconciliation. These interactions may be best explained using the following example. The topology of the example network and the policies enforced by the network nodes are shown in the following diagrams. Note that the keyword "ANY" implies that all legal values of the field are accepted by the policy.

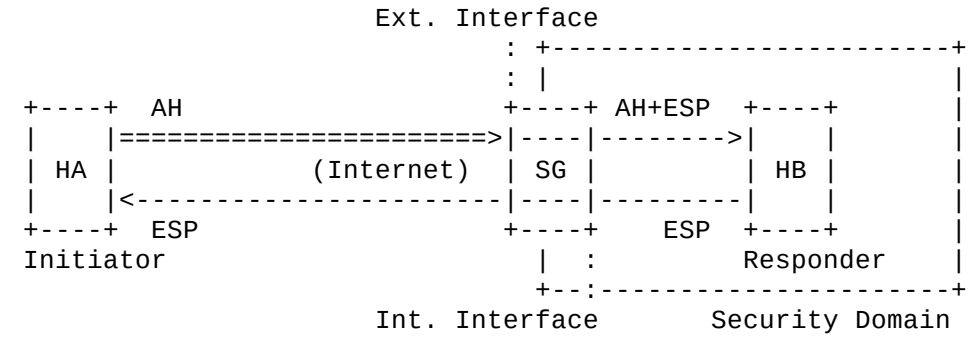

Figure 1. Example network for illustrating IPsec Policy Interaction

	src	dst	prot	sport	dport	dir	action
HA Policies (original)							
Pha1	HA	ANY	TCP	23	ANY	OUT	(PASS ESP TR (DES 56))
Pha2	ANY	HA	TCP	ANY	23	IN	(PASS ESP TR (DES 56))
Pha3	HA	ANY	TCP	ANY	ANY	OUT	(PASS)
Pha4	ANY	HA	TCP	ANY	ANY	IN	(PASS)
SG Policies (original)							
PsgE1	ANY	SD	ANY	ANY	ANY	IN	(PASS AH TN (HMD5 128))
HB Policy (original)							
Phb1	ANY	HB	ANY	ANY	ANY	IN	(PASS AH TR (HMD5 128))

Figure 2. Example policies for illustrating IPsec Policy Interaction

The example network consists of two hosts, HA and HB, and a security gateway SG. The security gateway was configured to protect a

security domain SD {1} that contains the host HB. In the example, we try to establish a TELNET connection between HA and HB by satisfying the IPsec Policies associated with each of the network nodes, HA, HB and SG.

At the host HA, the first two policies (Pha1, Pha2) require the TELNET connection to be protected in IPsec ESP transport mode by encrypting the IP datagrams with single DES algorithm. The other two policies (Pha3, Pha4) permit all other TCP connections to bypass IPsec processing. The absence of other policies implies that no other communication involving HA is allowed; in other words, a default policy of discarding all other traffic is placed after Pha4 without explicit mentioning. Similar default policies are also placed at the other nodes. The policy at the other host HB requires that all incoming datagrams to be authenticated in IPsec AH transport mode using keyed HMAC-MD5 algorithm.

The policies at the security gateway SG are the ones that deserve our attention. Note that the gateway has two relevant interfaces: the internal interface connected to the security domain SD and the external interface connected to the global Internet. Each of these two interface may enforce its own IPsec Policies. In our example, the external interface of SG enforces a policy PsgE1 that requires all traffic coming into the security domain to authenticate their origins to SG in IPsec AH tunnel mode using keyed HMAC-MD5 algorithm. In addition, SG must also have policies that allow IP datagrams transformed by the application of IPsec at HA and HB to pass through. Otherwise, those datagrams will be discarded and the communication between the two hosts will be disrupted by the gateway. As we shall see, these policies will be produced by the process of Policy Reconciliation.

4.1 Policy Resolution

As explained in the introduction, the initiator of IKE negotiation must combine the IPsec Policies of two (or more) network nodes that are relevant to a specific communication in order to determine the security associations necessary to protect the communication. The process is known as Policy Resolution. The result of the process can be formulated as a new policy with its condition clause expressing the common policy conditions of the resolving policies and its action clause capturing the SA proposals that satisfy the resolving policies.

In our example, Pha1 of HA must be resolved with Phb1 of HB to produce a pair of new policies {2}: Phab1 of HA and Phba1 of HB [Figure 3]. The resultant policies requires all TELNET traffic from

{1} A Security Domain is a connected cluster of network entities, that are protected by a common set of security policies enforced by enforcement agents at the domain boundary.
{2} The resultant policies of policy interaction are marked by asterisks (*).

HA to HB to be protected by IPsec AH and ESP for both data confidentiality and origin authentication. Note that the policies must be composed correctly: the outbound policy of source node or interface must be combined with the inbound policy of destination node or interface.

	src	dst	prot	sport	dport	dir	action
HA Policies							
* Phab1	HA	HB	TCP	23	ANY	OUT	(PASS AH+ESP TR (DES 56)(HMD5 128))
Pha1	HA	ANY	TCP	23	ANY	OUT	(PASS ESP TR (DES 56))
Pha2	ANY	HA	TCP	ANY	23	IN	(PASS ESP TR (DES 56))
Pha3	HA	ANY	TCP	ANY	ANY	OUT	(PASS)
Pha4	ANY	HA	TCP	ANY	ANY	IN	(PASS)
SG Policies							
PsgE1	ANY	SD	ANY	ANY	ANY	IN	(PASS AH TN (HMD5 128))
HB Policies							
* Phba1	HA	HB	TCP	23	ANY	IN	(PASS AH+ESP TR (DES 56)(HMD5 128))
Phb1	ANY	HB	ANY	ANY	ANY	IN	(PASS AH TR (HMD5 128))

Figure 3. Example results of Policy Resolution

4.2 Policy Correlation and Decorrelation

Policy Correlation can be considered as a method for discovering mutual dependencies among the IPsec Policies enforced by the same network node. Two IPsec Policies are said to be correlated if they are enforced at the same node/interface and matched with the same IP datagram; they are said to be decorrelated otherwise.

Policy Correlation often arises because current IPsec specification [6] mandates that the IPsec Policies maintained in the Security Policy Database (SPD) of a Policy Enforcement Point (PEP) should be searched in linear order when they are matched with a passing IP datagram. This requirement causes the matching condition of an IPsec Policy to depend on its position in the linearly ordered list -- one must search linearly through all the policies that are correlated with a specific policy in order to determine the condition under which the policy is to be enforced. The correlation relation thus complicates both processes of Policy Resolution and dynamic policy update.

We devise the operation of Policy Decorrelation for the purpose of removing possible correlation between IPsec Policies. The operation

transforms two correlated policies into at most three uncorrelated policies, not two of which can be matched with the same IP datagram.

In our example, the four policies of HA are obviously correlated with one another: both Pha1 and Pha3 match with outbound TELNET traffic while Pha2 and Pha4 match with inbound TELNET. The way to decorrelate these policies is to transform Pha2 and Pha4 into corresponding policies which exclude {3} the port number 23 of TELNET from the selectors of source and destination port numbers. Figure 4 lists the set of decorrelated policies of HA.

		src	dst	prot	sport	dport	dir	action
HA Policies								
	Phab1	HA	HB	TCP	23	ANY	OUT	(PASS AH+ESP TR (DES 56)(HMD5 128))
*	Pha1'	HA	~HB	TCP	23	ANY	OUT	(PASS ESP TR (DES 56))
	Pha2	ANY	HA	TCP	ANY	23	IN	(PASS ESP TR (DES 56))
*	Pha3'	HA	ANY	TCP	~23	ANY	OUT	(PASS)
*	Pha4'	ANY	HA	TCP	ANY	~23	IN	(PASS)
SG Policies								
	PsgE1	ANY	SD	ANY	ANY	ANY	IN	(PASS AH TN (HMD5 128))
HB Policies								
	Phba1	HA	HB	TCP	23	ANY	IN	(PASS AH+ESP TR (DES 56)(HMD5 128))
*	Phb1'	~HA	HB	ANY	~23	ANY	IN	(PASS AH TR (HMD5 128))
*	Phb1"	~HA	HB	ANY	ANY	ANY	IN	(PASS AH TR (HMD5 128))

Figure 4. Example results of Policy Decorrelation

4.3 Policy Reconciliation

Policy Reconciliation can be considered to be a method to ensure the consistency between the IPsec Policies enforced by the "upstream" nodes with those enforced by the "downstream" nodes along a communication path. Its main purpose is to allow the IP datagrams transformed by IPsec processing to travel from source to destination without being discarded by the intermediate gateways. Policy Reconciliation is conducted by matching the IP header fields that are modified by the upstream policies with the corresponding selector values of the downstream policies. The operation may be

{3} In the proposed algebraic semantic model of IPsec Policies, exclusion of specific values from a selector may be done using the difference operator.

conducted using static analysis techniques such as abstraction interpretation [7] for flow analysis of computer programs [8].

In our example, SG policies PsgE2 and PsgI1 exist solely for the purpose of reconciling with Pha1 and Pha2 of HA. Similarly, PsgI2 is there to reconcile with Phb1 of HB.

		src	dst	prot	sport	dport	dir	action
HA Policies								
	Phab1	HA	HB	TCP	23	ANY	OUT	(PASS AH+ESP TR (DES 56)(HMD5 128))
	Pha1'	HA	~HB	TCP	23	ANY	OUT	(PASS ESP TR (DES 56))
	Pha2	ANY	HA	TCP	ANY	23	IN	(PASS ESP TR (DES 56))
	Pha3'	HA	ANY	TCP	~23	ANY	OUT	(PASS)
	Pha4'	ANY	HA	TCP	ANY	~23	IN	(PASS)
SG Policies								
	PsgE1	ANY	SD	ANY	ANY	ANY	IN	(PASS AH TN (HMD5 128))
*	PsgE2	HB	HA	ESP	OPQ	OPQ	OUT	(PASS)
*	PsgI1	HB	HA	ESP	OPQ	OPQ	IN	(PASS)
*	PsgI2	ANY	HB	AH	ANY	ANY	OUT	(PASS)
HB Policies								
	Phba1	HA	HB	TCP	23	ANY	IN	(PASS AH+ESP TR (DES 56)(HMD5 128))
	Phb1'	~HA	HB	ANY	~23	ANY	IN	(PASS AH TR (HMD5 128))
	Phb1_	~HA	HB	ANY	ANY	ANY	IN	(PASS AH TR (HMD5 128))

Figure 5. Example results of Policy Reconciliation

<< Further discussions on Policy Reconciliation are not included in the current version of this document. They will be included once relevant results of future work become available. >>

5. Algebraic Semantics of IPsec Policies

5.1 Partial Order Relations among IPSec Policies

According to [6], every IPSec Policy must specifies the matching values of seven selectors (source and destination IP addresses, source and destination port numbers, transport layer protocol identifier, user identifier, and security label) and a clause of actions. Each selector may take a specific value or the "wildcard" value ANY. Source and destination IP addresses may also take ranges of address values. Some selectors may be omitted from the explicit expression of a policy; in those cases, they are assumed to have their default values. Together, these selector values specify the

selection condition of the policy, which can be regarded as a logical predicate evaluated to TRUE when the header fields in the passing IP datagrams match with the specified selector values. On the other hand, the action clause specifies whether an IP datagram should be discarded, bypassed without IPSec processing, or passed with IPSec processing when it matches with the policy condition. If IPSec processing is necessary then the action clause also specifies a typed list of security mechanisms to be used in datagram processing:

(esp (DES HMAC_MD5) or (DES HMAC_SHA)) or
((esp DES), (ah (HMAC_MD5) or (HMAC_SHA)))

An IPSec Policy in the above form can be modeled as a mapping from its condition clause C, which is a direct product of seven selector value sets S[i=1..7] {4} to its action clause A, which is a typed list of IPSec-ISAKMP descriptors of security mechanisms:

(1) P = C -> A |= xS[i=1..7] -> A.

Based on the current data model of IPSec Policies [4], we deduce that A is a lattice with the top element T being the NULL action or the least upper bound of conflicting actions and the bottom element B being the UNDEFINED action lower in order than all actions defined. The lattice model of the policy actions enables us to compose two actions by finding their least upper bound using the join operation ^ of a partially ordered set.

5.2 Composition of IPsec Policies

The composition (*) of two IPSec Policies is a commutative and associative operation that produces a new policy, which specifies the resultant actions to be taken when the datagram states match with the conditions of both input policies. The operation has algebraic properties analogous to those of the set operation intersection. In actual computation, the operation is performed by processing the condition clauses and the action clauses separately.

5.2.1 Composition of Policy Conditions

The composition of two policy conditions is simply performed as the intersection of the direct products of their selector value sets:

(2) xS[i,k] * xS[j,k] = x(S[i,k] * S[j,k])

5.2.2 Composition of Policy Actions

Each policy action consists of an ISAKMP action descriptor and an IPSec action descriptor. These action descriptors specify the choices of security mechanisms for supporting Security Association

{4} the selector value sets S[i=1..7] form a direct product space, xS[i].

management and data protection respectively. Each action descriptor can be decomposed into a list of action proposals. Each action proposal contains a list of protection suite, and each protection suite consists of lists of protocol proposals, each of which specify a mechanism to protect AH, ESP or IP compression. The following pseudo-ASN.1 code shows a simplified structure {5} of an IPSec action descriptor:

```
IpsecDescriptor ::=   SEQUENCE OF IpsecProposal
IpsecPropsal ::=      SEQUENCE OF IpsecSuite
IpsecSuite ::=        CHOICE{
                         SEQUENCE OF EspProposal [option],
                         SEQUENCE OF AhProposal [option],
                         SEQUENCE OF IpcompProposal [option]
                         }
```

Furthermore, a protocol proposal such as EspProposal is usually a list of atomic tokens: (ipsecCipherAlg, keyLen[option], keyRound[option], integrity[option], group[option], expiry[option]). All tokens are ASN.1 encoded and thus explicitly typed. The SEQUENCE OF list elements can be interpreted as a disjunctive Boolean expression while IpsecSuite and the protocol proposals (EspProposal, AhProposal, IpcompProposal) should be treated as tuples with specially ordered components.

The composition of two action clauses A[i], A[j] is performed as the join operation ^ defined for the partially ordered set of IPSec policy actions:

[3] $A[i] * A[j] = A[i] \wedge A[j]$

Rules must be established for performing the join operation on different forms of policy actions according to their partial order relations. The following sections provide the rules for both the partial ordering of the actions and their joint operations. Since an ordered set can be constructed by combining simpler ordered sets, we provided two sets of rules: the first set specifies the ordering and the operation for three list structures that appear in the policy actions, and the second set applies the first set of rules onto different lexical tokens.

5.2.2.1 Composition Rules for Action Structures

The nested list of policy actions mentioned above can be parsed into four basic structures.

(A) Disjunctive Boolean Expressions

As mentioned, the list of action proposals and protocol proposals can be interpreted as disjunctive Boolean expressions with OR

{5} Some tokens, e.g. the Boolean flag pfs for Perfect Forward Secrecy, are omitted for simplicity.

connectors between proposals. Composition of these expressions follows Boolean algebraic laws (esp. the absorption and the distribution laws) with ^ equivalent to the AND (*) operation:

(4a) x ^ (x + y) = x :absorption

(4b) (x[1] + .. + x[m]) ^ (y[1] + .. + y[n]) = : distribution
(x[1] ^ y[1]) + (x[1] ^ y[2]) + .. + (x[m] ^ y[n]}

Note that one or more (x[i] ^ y[j]) terms may resolve to T and be dropped from the expression as we asserted x + T = x.

(B) Coordinatewise Ordered Tuples

A few tuples such as EXPIRY = (EXP_TIME, EXP_DATASIZE) in the protocol proposal contain tokens that are ordered independently. (The expiration condition above is satisfied whenever the time limit or the data size limit is reached). The tuples are known to be in coordinatewise order:

(5) (x[1],..,x[n]) =< (y[1],..,y[n]) iff x[i] =< y[i] for all i

and has the following rule for its join operation:

(6) (x[1],..,x[n]) ^ (y[1],..,y[n]) = ((x[1]^y[1]),..,(x[n]^y[n]))

Note that the operation propagates the top element T since x[i]^y[i] = T iff x[i] or y[i] = T.

(C) Lexicographically Ordered Tuples

The protection suites IPSEC_SUITE and the protocol proposals are tuples of tokens with different types. They should be parsed by pattern matching based on the lexicographic ordering of the components. Generally, a lexicographically ordered tuple (p#q) with p in a higher precedence than q obeys the following ordering relation:

(7) (x[1]#x[2]) =< (y[1]#y[2])
iff (x[1]<y[1]) or (x[1]=y[1] & x[2]=<y[2])

The order relation implies the following rule for the join operation:

(8) (x[1]#x[2]) ^ (y[1]#y[2])
= (y[1]#y[2]) iff (x[1]<y[1]) or (x[1]=y[1]) & x[2]=<y[2])
= T iff x[1] = T or y[1] = T

To ensure that the join operation only yields meaningful results (!= T) when the tuples are consisted of tokens with compatible types, we mandate that the join of tokens with incompatible types yields the top element T. With this rule, pattern matching between two disjunctive lists of tuples can be carried out by applying the

Boolean distribution rule and the lexicographic ordering rule to the expressions.

(D) Dependent Component Tuples

The KEY_LEN token in the protocol proposals exist only if ESP_CIPHER is cast, rc5 or blowfish. This means as a tuple (p:q), (ESP_CIPHER, KEY_LEN) obeys a strict dependence relation: for every permissible value of p, there is a mapping Q that defines the set of permissible values of q. If for some p, Q(p) are empty sets then the corresponding q are absent. Under this strong dependence, two tuples are comparable if and only if their first component are equal:

(p[1]:q[1]) =< (p[2]:q[2]) iff (p[1]=p[2]) & (q[1]=<q[2]) in Q(p)

Consequently, the join operation obeys the following rule:

```
(p[1]:q[1]) ^ (p[2]:q[2])
        = (p : (q[1]^q[2])) iff p[1] = p[2] = p
        =  T                iff p[1] != p[2] or p[1] = T or p[2] = T
```

5.2.2.2 Composition Rules for Action Types

In a composition of two policy actions, the structural rules mentioned in the previous section should be applied to every token in the action clauses. Composition of these tokens are guided by a different set of rules, referred to as the type rules. These type rules are associated with individual tokens defined in the policy data model and will continue to evolve as we change the action semantics of the tokens.

Following remarks are applicable to all the type rules stated in this section:

Typing characteristics of action objects: All the tokens in the action clauses are typed. This implies that separate rules may be specified for each individual type. A simple notation scheme is used to refer to these tokens -- the semantic objects are denoted by ALL_CAPITALS_WITH_UNDERSCORES, their types are SmallCaptials and the symbol :: is used as the connector.

Semantic structures for pattern matching: The action clauses are made up of disjunctive Boolean expressions with security proposals connected by OR (+) operators. Composition of these expressions are done by finding the longest common sub-expression within them based on pattern matching between the proposals. In order to perform the matching correctly, elementary objects such as individual cipher algorithms are specified to be incomparable so that they cannot be coerced into one another.

Semantic meaning of default values: Since the default value of an action type specifies an action that must be supported by all IPSec compliant entities, it should be treated as an implicit proposal

embedded in every action specification. Hence, an OR term containing the default action should be added to the semantic expression of every action type.

(A) ISAKMP Descriptors

Syntactically, the ISAKMP descriptor is the following ASN.1 object:

```
IsakmpDescriptor ::=
     SEQUENCE {
       exchange ENUMERATED {
         MainMode,
         AggressiveMode } OPTIONAL,
       proposal SEQUENCE OF IsakmpProposal
     }
```

Semantically, an ISAKMP descriptor IKE_DSPR::IsakmpDescriptor is a tuple formed by two completely independent components, X_MODE and IKE_PROP:

(11) IKE_DSPR |= (X_MODE, IKE_PROP)

(A.1) Exchange Modes

ISAKMP may use two exchange modes, main mode and aggressive mode, to negotiate the security associations for protecting ISAKMP communication. Among them, main mode is the default choice, and aggressive mode is the stronger mode.
The explicit mode specification X_MODE in exchange field may be either of the two modes:

(12) X_MODE in { M_MODE, A_MODE }

Taking the default value into consideration, the full mode specification XModeF should be interpreted as a Boolean expression of the following form:

(13) X_MODE_F = DEFUALT + X_MODE

In the above expression, + denotes a set union operation.

(A.2) ISAKMP Proposals

The specification IKE_PROP contained in proposal field should be interpreted as the following Boolean expression of ISAKMP proposals:

(14) IKE_PROP |= + IKE_PTERM[i]

Syntactically, an IsakmpProposal object obey the following ASN.1 expression:

```
IsakmpProposal ::=
     SEQUENCE {
```

```
          cipher IsakmpCipherAlg,
          keylength INTEGER OPTIONAL,
          hash HashAlg,
          group INTEGER OPTIONAL,
          expiry Expiry
        }
```

Semantically, an ISAKMP proposal IKE_PROP::IsakmpProposal is a tuple consists of components in all three orderings: independent, lexicographic dependent and completely dependent order. Following is its semantic expression:

(15) IKE_PTERM |= (((IKE_CIPHER : KEY_LEN) # IKE_HASH # GROUP), EXPIRY)

consisting of the following types of the semantic objects:

```
     IKE_CIPHER :: IsakmpCipherAlg
     IKE_HASH :: HashAlg
     GROUP :: INTEGER    ;; ISAKMP group number for DH computation
     KEY_LEN :: INTEGER  ;; key length for selected crypto
algorithms
     EXPIRY :: Expiry    ;; expiration limits of security
associations
```

(B) IPSec Descriptors

Syntactically, the IPSec descriptor is the following ASN.1 object:

```
IpsecDescriptor ::=
     SEQUENCE {
       pfs BOOLEAN,
       proposal SEQUENCE OF IpsecProposal
     }
```

Similar to the ISAKMP descriptor, an IPSec descriptor IPS_Ptr::IpsecDescriptor is a tuple formed by two completely independent components, PFS and IPS_PROP:

(16) IPS_DPTR |= (PFS, IPS_PROP)

Like X_MODE in the ISAKMP descriptor, the Boolean value of PFS (perfect forward secrecy) should be combined with its default value to produce its full semantic expression:

(17) PFS_F |= DEFAULT + PFS

IpsecProposal and IpsecSuite have the following ASN.1 expressions:

```
IpsecProposal ::=
     SEQUENCE OF {
       protectionSuite IpsecSuite
     }
```

```
IpsecSuite ::=
     CHOICE {
       espProtocol SEQUENCE OF EspProposal,
       ahProtocol SEQUENCE OF AhProposal,
       compProtocol SEQUENCE OF IpcompProposal
     }
```

The syntax of the two object types allow them to produce complex Boolean expressions of proposal tuples. Basically, an IPSec proposal object IPSprop is a disjunctive Boolean expression of IPSec suite object IPS_SUITE while an IPS_SUITE object is a triplet with each component being a disjunctive Boolean expression of ESP_PTERM, AH_PTERM and COMP_PTERM, which specify the proposed mechanisms for ESP, AH and Compression respectively.

(18) IPS_PROP |= + IPS_SUITE[i]

```
IPS_SUITE |=
        ((+ EPS_PTERM[i]) # (+ AH_PTERM[j]) # (+ COMP_PTERM[k]))
```

The components of IPS_SUITE must obey lexicographic order because each triplet must be taken as an integral proposal suite and their comparison must be done by pattern matching.

(B.1) ESP Proposals

Syntactically, an IsakmpProposal object obey the following ASN.1 expression:

```
EspProposal ::=
     SEQUENCE {
       cipher IpsecCipherAlg,
       keylength INTEGER OPTIONAL,
       keyrounds INTEGER OPTIONAL,
       integrity IntegrityAlg OPTIONAL,
       group INTEGER OPTIONAL,
       expiry Expiry OPTIONAL
     }
```

Semantically, an ESP proposal term ESP_PTERM::EspProposal is a tuple of components in all three possible orderings. Following is its semantic expression:

(20) EPS_PTERM |=
((((ESP_CIPHER : (KEY_LEN, KEY_ROUND)) # ESP_HASH # GROUP), EXPIRY)

consisting of the following types of the semantic objects:

```
ESP_CIPHER :: IpsecCipherAlg
ESP_HASH :: IntegrityAlg
GROUP :: INTEGER     ;; ISAKMP group number for DH computation
KEY_LEN :: INTEGER   ;; key length for selected crypto algorithms
```

```
  KEY_ROUND :: INTEGER ;; key rounds for selected crypto algorithms
  EXPIRY :: Expiry     ;; expiration limits of security associations
```

Objects KeyRound is currently unused and undefined.

(B.2) AH Proposals

Syntactically, an IsakmpProposal object obey the following ASN.1 expression:

```
AhProposal ::=
     SEQUENCE {
       integrity IntegrityAlg,
       group INTEGER OPTIONAL,
       expiry Expiry OPTIONAL
     }
```

Semantically, an AH proposal term AH_PTERM::AhProposal is a tuple with the following semantic expression:

```
AH_PTERM |=  ((AH_HASH : GROUP), EXPIRY)
                                 if AH_GROUP depends on AH_HASH
         |=  ((AH_HASH, GROUP), EXPIRY)
                                 if AH_GROUP NOT depends on AH_HASH
```

The expression contains the semantic objects of the following types:

```
  AH_HASH :: IntegrityAlg
  GROUP :: INTEGER  ;; ISAKMP group number for DH computation
  EXPIRY :: Expiry  ;; expiration limits of security associations
```

(B.3) IP Compression Proposals

The IpcompProposal object has the following simple ASN.1 syntax:

```
IpcompProposal ::=
     SEQUENCE {
       compression CompressionAlg,
       expiry Expiry OPTIONAL
     }
```

The semantic expression of an compression proposal term COMP_PTERM::IPcompProposal is as follow:

```
(21)   COMP_PTERM |=  (COMP_ALG, EXPIRY)
```

with the following two types of semantic objects:

```
     COMP_ALG :: CompressionAlg
     EXPIRY :: Expiry  ;; expiration limits of security associations
```

(C) Security Algorithms

The security algorithm types, including IsakmpCipherAlg, HashAlg, IpsecCipherAlg, IntegrityAlg, CompressionAlg, all have the same semantic structure. We discuss the common semantics of these types without going through each type.

In order to enforce pattern matching between the security proposals, individual objects in each algorithm type are made incomparable in their ordering. Also, a bottom object B is added to each set to accommodate the error cases.

Assuming a set L contain N explicit algorithm objects l[i=1..N] and B, then the following rule governs the join operation of this set:

(22) l[i] ^ l[j] = l iff l[i] = l[j] = l
= B otherwise

In other words, any mismatch between the algorithm objects will produce the bottom object B. Note that the ordering of the algorithm objects does not reflect the strength of the algorithms. Therefore, NULL encryption is treated as a distinct object incomparable but not lower in order than other cipher algorithm objects.

(D) Group

The object GROUP::Group in all proposals denotes the choices of pre-selected parameters and algorithms (MODP vs. EC2N) for performing Diffie-Helman computation. The set of semantic objects has the same structure as X_MODE::ExchangeMode and PFS::pfs -- i.e. a Boolean expression with an OR operator connecting its default value and the optional explicit value, both of which belong to a set of incomparable values:

(23a) GROUP in {1,2,3,4}

(23B) GROUP_F = DEFAULT + GROUP

Any semantic reference to GROUP should be regarded as a reference to GROUP_F so that the default value can be taken into consideration. Note that since the default value is always included in the disjunctive Boolean expression, the join operation on GROUP (also, on X_Mode and PFS) will never produce the bottom object B.

(E) Key Lengths

The object KEY_LEN in the proposals of cipher algorithm is a positive integer equal to the number of bits in the cryptographic key used by the algorithm. As such, it belong to a linearly ordered set with the increase in the number in proportion to the increase in strength of cipher computation. The linear ordering of KEY_LEN is used in the join operation of two matched proposals to increase the strength of the computation:

(24) KEY_LEN[i] ^ KEY_LEN[j] = KEN_LEN[j] iff KEY_LEN[i] >= KEY_LEN[j]

Note that the default specification of KEY_LEN is to accept any proposed value; hence the default KEY_LEN value is 0.

(F) Expiry

The object EXPIRY::Expiry in a proposal is a couplet specifying the maximum duration or the maximum processed data size of the corresponding security association:

(25) EXPIRY = (EXP_TIME, EXP_DATASIZE)

Both EXP_TIME (in seconds) and EXP_DATASIZE (in kilobytes) are positive integers with the increase in their values in inverse proportion to the increase in strength of cipher computation. This is because the longer or the more a security association and its key parameters are used, the more likely it is to be compromised by the adversaries. Hence, the linear ordering of both EXP_TIME and EXP_DATASIZE are used in reverse order the join operation of proposals:

(26a) EXP_TIME[i]^EXP_TIME[j] = EXP_TIME[j]
iff EXP_TIME[i] =< EXP_TIME[j]

(26b) EXP_DATASIZE[i] ^ EXP_DATASIZE[j] = EXP_DATASIZE[j]
iff EXP_DATASIZE[i] =< EXP_DATASIZE[j]

Note that the default specification of both EXP_TIME and EXP_DATASIZE is to accept any proposed value. Their reverse ordering hence requires their default values to be the maximum permissible values of the types.

5.3 Difference of IPSec Policies

The difference operation (-) is a non-commutative operation that takes two IPSec Policies and produces a new policy which specifies the actions to be taken when the condition of the second policy is excluded from that of the first policy. The operation is analogous to the difference of two sets. Its main use is to perform policy decorrelation: given two policies P[1] and P[2], their composition P[1] * P[2] and their differences P[1] - P[2], P[2] - P[1] represent three uncorrelated parts of the original policies.

Similar to composition, policy difference can also be performed by processing the condition clauses and the action clauses separately. Let P[1] = C[1] -> A[1] and P[2] = C[2] -> a[2] be two policies, and P[2,1] = P[2] - P[1] = C[2,1] -> A[2,1] be one of their differences then the formulas to compute the condition C[2,1] and the action A[2,1] are

(27a) C[2,1] = C[2] = C[1]

(27b) A[2,1] = A[2]

However, we never need to compute the policy differences explicitly. In the next section, we present a method that allows us to express the difference of policies in set operations and perform the policy matching in Boolean logic.

6. Application of IPsec Policy Semantics

As we have discussed in the introduction {sect.3], deducing the necessary security associations to enforce the IPsec Policies of the nodes along the path of a communication can be a complex task. This is because the IPsec Policies only prescribe the security requirements of individual nodes, but the security associations must satisfy matching pairs of policies in corresponding node pairs. The situation is further complicated by the fact that the policies to be enforced at a network node may come from several different sources including the users (of the node) who intend to protect their applications, the administrators (of the enterprise network) that own and manage the node and/or the Internet service providers that connect the enterprise network to the global Internet. Each party may specify IPsec Policies according to his/her own understanding of the security requirements of communications involving that node. These policies often capture only partial or local perspectives of the security requirements and may be inconsistent with one another. Hence, before negotiating the security associations that enforce the policies, we must eliminate any inconsistencies among these policies, discover the matching pairs and deduce the resultant actions -- i.e., the combined requirements of cryptographic operations -- from every matching pair. These operations should be performed according to a formal set of rules carefully articulated by policy developers and adopted by the Internet community in order that predictable and consistent results can be obtained from the policy processing we have discussed. Our response to this need of formal processing rules was to develop an algebraic semantic model for IPsec Policies. The semantic model defines two operators, Policy Composition and Policy Difference, which are essential for conducting Policy Resolution and Decorrelation.

Policy Composition is the operator that captures the essence of Policy Resolution. It takes two IPsec Policies and produces a new one, which specifies the actions to be taken when the selection conditions of the two given policies become TRUE.

Policy Difference is the operator that performs Policy Decorrelation. It takes two IPsec Policies and produces a new one, which specifies the actions to be taken when the selection condition of the first policy is TRUE but that of the second policy is FALSE.

Together, Policy Composition and Difference can produce three new IPsec Policies from a given pair. The first new policy, known as the composite policy, specifies the actions to be taken when the

selection conditions of both given policies are satisfied. This new policy represents the resolution of the two given policies. The other two, known as the difference policies, specify the actions to be taken when the selection condition of one policy is satisfied while the condition of the other policy is not. The three new policies are decorrelated because at most one of them may be selected when matched with IP datagrams.

7. Security Considerations

<< Remarks on security consideration will be added to the next revision of this document. >>

8. References

[1] S. Branded, "The Internet Standards Process - Revision 3", BCP 9, RFC 2026, October 1996.

[2] S. Branded, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] S. Kent and R. Atkinson, "IP Authentication Header" RFC 2402, November 1998.

[4] S. Kent and R. Atkinson. "IP Encapsulating Security Payload (ESP)" RFC 2406, November 1998.

[5] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)" RFC 2409, November 1998.

[6] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol" RFC 2401, November 1998.

[7] P. Cousot and R. Cousot, "Abstract Interpretation: A unified lattice model for static analyses of programs by construction or approximation of fixpoints" in Proceedings of ACM SIGPLAN Conference on Principles of Programming Languages (1977), pp.238-252.

[8] S. S. Muchnick and N. D. Jones, "Program Flow Analysis, Theory and Applications" Prenctice-Hall, 1981.

Disclaimer and Copyright Statement

Author's Address

John K. Zao
BBN Technologies
10 Fawcett Street
Cambridge, MA 02138
USA
Email: jzao@bbn.com
Telephone: +1 (617) 873-2438