

# Protocol Design for Integrity Protection

Stuart G. Stubblebine

USC Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, CA 90292  
stubblebine@isi.edu

Virgil D. Gligor

Electrical Engineering Department  
University of Maryland  
College Park, Maryland 20742  
gligor@eng.umd.edu

## Abstract

*We present a design method for message integrity protection. We illustrate the use of the method by designing large classes of message types whose integrity is provably preserved and by applying the method to the symmetric key option of the Privacy-enhanced Electronic Mail protocol to help discover and eliminate an integrity vulnerability.*

## 1. INTRODUCTION

The integrity of messages in cryptographic protocols has been one of the long-standing assumptions made by most protocol analysis and design methods. Recently, Stubblebine and Gligor [SG92, 93a] showed that a significant number of cryptographic protocols used in practice contain subtle flaws that can expose communications to integrity attacks by active intruders. To date, formal characterizations of message integrity for cryptographic protocols have not been available and, consequently, protocol design for integrity protection has been performed mostly on an ad-hoc basis.

We present a design method for integrity protection in cryptographic protocols that is based on a formal model [SG93b]. The method is independent of the specific encryption system (e.g., symmetric or asymmetric cryptosystem) and checksum/digest functions used. It expresses desirable requirements for message integrity protection in terms of abstract encryption and checksum/digest properties, and relates these properties to the message type, representation, and lifetime of the protocol run and keys used. The use of the method is illustrated by the design of a large class of message types whose integrity is provably preserved in face of active intruder attacks. In particular, the method is used to help discover and eliminate a vulnerability in the symmetric-key option of the Privacy-enhanced Electronic Mail (PEM) protocol for the Internet.

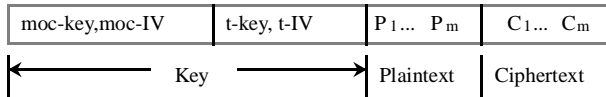
The paper is organized as follows. In section 2, we introduce the terminology used in the description of the design method, and present a message integrity condition. In Section 3, we present the design method, and in Section 4 we include an example of a significant class of message types that preserve message integrity in face of active intruder attacks. In Section 5, we demonstrate the application of the method to the correction of a PEM option vulnerability (i.e., to determine what properties of encryption and checksum/digest can be used to satisfy the integrity protection objectives).

## 2. DEFINITIONS AND MESSAGE INTEGRITY CONDITION

The analysis of message integrity in cryptographic protocols requires the specification of the behavior of both the communicating principals and the attacker and the message structure and representation [SG92, 93b]. These principals, and the attacker, communicate by constructing and sending, and by receiving and recognizing, messages (i.e., protocol data units). For each message type, a principal uses a Key to encrypt or decrypt selected components of the message and, possibly, to compute the redundancy (e.g., checksum, digest) function added to the message. The redundancy functions are generally intended to capture the membership (M), order (O), and cardinality (C) of the blocks within a message. For this reason, these functions are generically called the *MOC* functions [SG92]. If the result of the MOC computation over the (possibly decrypted) blocks of a message equals that MOC value included in that message by the sender, the recipient believes that the received message is genuine provided that only an entity initially possessing a secret Key could have created a message representation of that type.

*Message Structure:* The structure of a message consists of a Key, the plaintext and ciphertext blocks, which are created by using the encryption/decryption functions  $ENC()/DEC()$ . As shown in Figure 1, the Key attribute of a message structure may include multiple component attributes, such as a MOC component key, called the *moc-key*; an encryption/decryption component called the transformation key or, simply, *t-key*; and, depending upon the type of function, initialization vectors for the MOC computation and for the encryption itself, called the *moc-IV* and *t-IV*, respectively. Some components of the Key may be secret; e.g., the *t-* and *moc-keys*. Other components of the Key can be public as is the case with asymmetric cryptographic functions implementing encryption or digital signatures. (In the rest of this paper, whenever it is unambiguous which key and IV is used, or whenever the two keys and IVs are identical, the term key and IV are used without any qualification.)

Message Structure



Message Representation	Attribute Class	Receiver Decrypted-Attributes
P <sub>1</sub> ...P <sub>s</sub> (s ≤ m)	domain	P <sub>1</sub> ...P <sub>s</sub>
C <sub>i</sub> ...C <sub>j</sub>	domain	DEC(t-key; C <sub>i</sub> ...C <sub>j</sub> )
C <sub>m</sub>	range	DEC(t-key; C <sub>m</sub> )
(1 ≤ i ≤ j ≤ m)		

Key Attribute (Key)

t-key	N/A
moc-key	domain

Recognition Condition:

$$MOC(moc\text{-}key; P_1...P_s, DEC(t\text{-}key; C_i...C_j)) = DEC(t\text{-}key; C_m)$$

**Figure 1. Message Structure, Representation, Decrypted-Attributes, and Recognition Condition**

*Message Representation:* The representation of a message consists of a specified subset of the attributes of a message structure; i.e., those attributes that are sent from the sender to the receiver. Not all attributes of the message structure are included in the representation. For example, the secret components of the Key are never included as cleartext in the representation of any message. The representation of a message is defined by a message type. Multiple types of private (confidentiality and integrity protected) and safe (integrity protected) messages can be included in a protocol.

The message sender views the message structure as shown in Figure 1. The message receiver views the message structure as the union of the message representation, Key attribute, and decrypted attributes. The receiver determines values for decrypted attributes from the physical blocks of a message representation and from the Key (see Figure 1). For example, block DEC(t-key; C<sub>m</sub>) represents a decrypted attribute that corresponds to the moc value.

The MOC() function determines the MOC value whose size is a constant, k. The MOC() function divides the message representation into the domain representation and range representation components. The *range representation* is the set of blocks of the message representation whose decrypted attributes correspond to the moc value. The *domain representation* is the set of blocks of the message representation whose decrypted attributes are in the domain of the MOC() function. (An attribute is in the domain of the MOC() function if computing the MOC value to recognize a message requires the attribute to be specified.) Values derived from the (possibly decrypted) blocks of the domain representation are used to decode and check the MOC value. We say that a range (domain) representation *matches* a domain (range) representation whenever the MOC value included in the range representation equals the MOC value

computed over the values derived from the domain representation. The domain and range representations are defined with respect to physical blocks of the message representation that are seen “on the wire” after the message representation has been constructed using the ENC()/DEC() and MOC() functions.

*Message Integrity Condition:* At any time, the protocol state contains the messages that have been constructed and sent, messages that have been received and recognized, and message types each principal can recognize. A protocol run is a finite sequence of states. The message integrity condition is defined as:

The probability that a bogus message representation is recognized by a principal in a state of a protocol run is no greater than a threshold specified by the protocol integrity policy.

The threshold specified by the protocol integrity policy can be different for each message type since messages sent by a principal can have different levels of integrity protection (e.g., weak and strong integrity in DCE [OSF91]). Attackers are assumed to have physical access to the communication media. Thus, the message integrity condition can be rewritten to capture the notion that an attacker can create a recognizable message representation. That is (\*):

The probability that an attacker can create a bogus message representation of type T recognizable to a principal in a state of a protocol run, is no greater than a threshold specified by the protocol integrity policy for message type T.

Since every block of the message representation is part of either the range or domain representation, to satisfy the message integrity condition it is sufficient to limit the probability that the attacker can (1) choose a domain representation to match a range representation, or (2) choose a range representation to match a domain representation. That is, for every state of a protocol run, given a message type, T, defined using a Key, and policy threshold specified for T,

**if** for every range representation,  $Prob[attacker\ chooses\ a\ domain\ representation\ such\ that\ the\ domain\ representation\ matches\ the\ range\ representation\ of\ type\ T] \leq Threshold,$

**and** for every domain representation,  $Prob[attacker\ chooses\ a\ range\ representation\ such\ that\ the\ domain\ representation\ matches\ the\ range\ representation\ of\ type\ T] \leq Threshold$

**then**  $Prob[attacker\ can\ create\ a\ bogus\ message\ representation\ of\ a\ type,\ T,\ recognizable\ to\ a\ principal] \leq Threshold.$

This above statement of the message integrity condition is used throughout the balance of this paper. Since we refer repeatedly to the clauses of this condition, we denote these

(\*) Communication errors are modeled by attacker actions. Thus, messages sent by the attacker can be considered to be error free.

clauses by S1 and S2, where the condition is “if S1 and S2 then the Message Integrity Condition is satisfied.” The sufficiency of the above condition for message integrity is demonstrated in [SG93b].

To satisfy the message integrity condition in practice, properties of the ENC()/DEC() and MOC() functions and time constraints on the protocol run and keys a) must limit the probability that an attacker discovers the keys during the run (i.e., moc-key or t-key), and b) must limit the probability that the attacker can choose, without discovering the keys, a domain representation to match a range representation or choose a range representation to match a domain representation. (We assume that attackers may only discover keys by analyzing message representations, that principals can be trusted both to maintain the privacy of the key, and that the key is used only to send message representations of designated types.)

An attacker is prevented from discovering the keys if the designer chooses and applies MOC() and ENC()/DEC() functions in a manner that makes it difficult to analytically derive these keys. Furthermore, the designer can choose the MOC() and ENC()/DEC() functions, and apply them to the construction of a message representation, in a manner that makes it difficult for the attacker to analytically discover matching range or domain representations without knowing the key. Note that the MOC() function need not always be computationally complex itself, since the combined use of the MOC() and ENC()/DEC() functions may exhibit the desired complexity properties.

Since analytic attacks can reduce the attacker’s uncertainty in discovering the value of a key, the designer can limit the attacker’s chances of verifying a key by limiting the lifetime of the protocol run. The bound on the protocol run by the key lifetime is denoted by  $d_k$ . Though an analytic attack can reduce the uncertainty of satisfying the recognition condition (viz., Figure 1), an attack can be further constrained (1) by limiting the time period,  $d_d$ , during which the attacker can obtain a solution enabling him to choose a domain representation that matches a range representation, or (2) by limiting the time period,  $d_r$ , during which the attacker can obtain a solution enabling him to choose a range representation that matches a domain representation.

*Scope of the Method:* The design method concerns the integrity protection of protocols as opposed to the design of individual message structures in isolation. This is important since messages of one type, although fully protected, can affect the integrity of other types [SG93a]. The designer can use the method to determine the properties and parameters of the ENC()/DEC() and MOC() functions that satisfy the integrity policy. For example, the method can be used to determine sufficient properties of the MOC() function and the relationship between function parameters (e.g., block size, MOC range size) and constraints on the lifetime of a protocol run and secret key.

Note that use of the method does not necessarily result in a unique message design and protocol. In fact, the method demonstrates that different message integrity designs and

protocols are possible having the same level of integrity protection. The method can be applied to traditional messages types which are built using a single MOC() function and (optionally) encrypted. It does not specify either ENC()/DEC() or MOC() functions; instead, it enables the selection of function properties for particular message types.

We make no attempt to analyze whether specific protocol functions satisfy the MOC() and ENC()/DEC() properties selected by using the method. This analysis is dependent on the algebraic properties of the specific protocol functions. Although important, this analysis is beyond the scope of this method. The method does not attempt to provide protection against message-stream modification. Services that provide such protection can be built into protocols once integrity of individual messages is demonstrably preserved. The method does not attempt to capture information gained by actions of the principals other than sending and receiving messages (e.g., key leakage by un-trusted principals through covert channels).

### 3. DESIGN METHOD

The design method is based on a state-transition model [SG93b]. The general objective of the design method is to ensure that the selected properties of the ENC()/DEC() and MOC() functions, message structures, message representations and protocol assumptions are sufficient to achieve the protocol integrity objective; i.e., to satisfy the conditions defined by clauses S1 and S2 for each message type. The design method consists of the following four steps:

(1) the designer specifies the integrity policy; i.e., for each message type supported by the protocol, a message integrity threshold, *Threshold*, is specified. The *Threshold* is an independent design parameter that states the maximum allowable probability that a bogus message can be accepted as legitimate.

(2) the designer specifies the message structure and representation for each type of message; in particular, the domain and range representations are defined, and the attribute constraints are identified. These constraints state assumptions about whether the value of a secret-domain attribute can be discovered, or limit the number of known values for attributes that may be chosen by a potential attacker. The protocol throughput constraint, or the maximum rate at which messages of each type are created and sent, is also specified.

(3) the designer specifies the properties of the ENC()/DEC() and MOC() functions for each message type. These properties are selected such that the conditions S1 and S2 are satisfied whenever the protocol throughput and run lifetime limit the probability, to the chosen *Threshold*, that (i) the secret components of the Key are discovered, and (ii) that the message representations generated by all the protocol message types enable the creation of bogus messages. The set of properties of the ENC()/DEC() and MOC() functions depend on the constraints placed on the message representation by the protocol (as specified in step (2)).

(4) the designer determines the lifetime constraints on the protocol run as required by each message type, and then specifies the maximum lifetime for the protocol run. The goal of this specification is to ensure that constraints of step (2) are satisfied. In other words, the lifetime is specified such that an attacker can neither guess the Key nor construct bogus messages that can satisfy the message integrity condition.

The protocol design relies on the following set of common protocol assumptions:

**PA1.** Principals are trusted to maintain the privacy of the secret components of the Key (i.e., the key, which could be the t-key, or moc-key).

**PA2.** A key used in the protocol run is generated at random from the set of keys that satisfy the computational needs of the algorithm. (The space of *key* is represented as |key|.)

**PA3.** The space and the computational power of the attacker are known. (Maximum pre-computation is also known.)

### (1) Specify the Policy Threshold

The Threshold (i.e., maximum allowable probability that a bogus message is accepted as legitimate), specified by the integrity policy, is based on the degree of protection desired for the data attributes for the message type. The Threshold may not be smaller than  $1/|key|$ .

### (2) Specify the Message Structure, Representation, and Protocol Constraints

*Message Structure and Representation:* The protocol definition includes a specification of the message formats, including all the attributes of the message structure and representation. The message structure and representation for each protocol message type are “idealizations” of the actual message formats. These idealizations are required for the uniform representation of all attributes of the message types used in a protocol.

*Protocol constraints:* For each message type, the protocol places constraints on the value of a message attribute (e.g., the attribute is encrypted), or limits the number of known values for attributes that can be chosen by a potential attacker. These constraints differ with the type of message because different types of ENC()/DEC() and MOC() functions are used for different types of messages and because these functions are used in different ways for different types of messages [SG92, SG93a]. Identifying the attribute constraints is necessary in message design because it provides the *prima facie* evidence that the attacker is prevented from successfully creating a recognizable bogus message. Identifying the attribute constraints placed by a protocol requires that the attributes of *all* message types be examined. This is necessary because determining whether an attribute should be constrained can depend on all uses of secret components of the Key.

The secret and decrypted attributes may be either constrained or unconstrained. A *secret attribute is constrained* whenever the probability that the attacker may discover the attribute value is less than the given Threshold (\*). Otherwise, the secret component is unconstrained. Examples of constrained secret-attributes include the secret components of the Key;

i.e., the secret transformation key (t-key) the and secret-domain attribute (moc-key). We call the moc-key attribute a *secret-domain attribute* because it is within the domain of the MOC() function. To ensure that a secret-attribute remains constrained, the attacker must be prevented from verifying the value of a chosen attribute. If an attacker is not (adequately) prevented from verifying a chosen attribute that is meant to remain secret, then that attribute is unconstrained.

A *decrypted attribute is constrained* if the number of known attribute values the attacker can choose is limited. Otherwise, the decrypted attribute is unconstrained. Examples of constrained decrypted attributes include any part of the ciphertext of the domain or range representation when the attacker is prevented from discovering the maximum number of known plaintext-ciphertext pairs. It is important to note that decrypted attributes may be constrained even when the attributes were not encrypted. For example, the computation of a moc value using a keyed MOC() function may be included in the representation without an additional layer of encryption. This moc value is constrained if the number of X,MOC(moc-key,X) pairs that can be obtained by the attacker is limited. Examples of unconstrained decrypted attributes also include higher layer attributes, such as user-data, which are included in the message representation in an unencrypted form.

### (3) Specify the ENC()/DEC() and MOC() Function Properties

ENC()/DEC() and MOC() functions are used (1) to prevent the attacker from discovering a secret-attribute, and to limit the number of known values the attacker may choose for an attribute, and (2) to ensure that, given the designer’s choice of attribute constraints, conditions of clauses S1 and S2 above are satisfied. We do not assume the actual ENC()/DEC() and MOC() functions are secret.

**ENC()/DEC() Functions:** Given that a protocol uses an ENC()/DEC() function satisfying definition EA1 below, properties EA2 and EA3, also defined below, ensure that any protocol can produce constrained, and trivially unconstrained attributes.

**EA1.** The encryption function of block ciphers break a text M into successive blocks  $P_1, \dots, P_n$ , and encipher each block with the same key; i.e.,  $ENC(key, M) = ENC(key, P_1) ENC(key, P_2) \dots ENC(key, P_n) = C_1, \dots, C_n$ . The decryption function, DEC, is the left inverse of ENC; i.e.,  $DEC(key, ENC(key, M)) = M$ . The block size of the cipher is  $b$  bits. (Pairs  $P_i, C_i$  are called the plaintext-ciphertext pairs. For example, in CBC encryption, the plaintext-ciphertext pairs are  $\langle ENC(key, P_{i-1}) \oplus P_i, C_i \rangle$  where  $\oplus$  stands for the “exclusive or (XOR)” operator, and  $i > 1$ .)

**EA2.** Given a random t-key, it is computationally infeasible to determine any bit of  $P_i$  from  $C_i$ , any bit of  $C_i$  from  $P_i$ , and any bit of the key from  $P_i$  and  $C_i$ .

---

(\*) Note PA2 maximizes the attacker’s uncertainty of the attribute’s value.

Constraints on Domain Attributes	Constraints on Decrypted-Range Attributes	
	Unconstrained	Constrained
Both Unconstrained	N/A <sup>1</sup> (vulnerable)	SA1 <sup>2</sup>
Secret-Domain Constrained	MA1 <sup>3</sup> (or SA1)	MA1 or SA1 <sup>4</sup>
Decrypted-Domain Constrained	N/A <sup>5</sup>	MA3 and EA <sup>6</sup> (or SA1)

**Table 1.**

**Property Assumptions Protecting Against Choosing a Domain Representation to Match a Range Representation**

**EA3.** The use of ENC()/DEC() restricts the generation of chosen plaintext-ciphertext pairs by an entity not possessing the t-key(\*).

In what follows, EA is used to represent EA1, EA2, and EA3.

**MOC() Functions:** The properties of MOC() functions defined below are defined by assigning a maximum probability that, given the constraints placed on the decrypted-domain and secret-domain attributes, the conditions of clauses S1 and S2 are satisfied; i.e., an attacker may not (1) choose a value of a decrypted-domain attribute that matches a decrypted-range attribute, or (2) choose a value of a decrypted-range attribute matching a decrypted-domain attribute, with a chance of success higher than that determined by the Threshold.

Since message attributes of different message types can be subject to multiple constraints, the design method can be used to specify multiple MOC() properties, each tailored to a different message type. Our approach to tailoring MOC() function properties to message structures is important since it facilitates the optimal choice of MOC() function for the design application.

Enumerating the many possible MOC() properties is beyond the scope of this paper. However, it is useful to reduce the number of MOC() properties by using the “high-watermark” property selection. That is, if a decrypted- or secret-domain attribute contains both constrained and unconstrained attributes, the selection of the property requiring that the attribute be constrained is sufficient to satisfy the message integrity condition.

Selecting properties of ENC()/DEC and MOC() functions consists of selecting a set of properties satisfying, at a minimum, the same attribute constraints as those specified for the decrypted message attributes. These properties are selected such that, given the designer-specified attribute constraints from step (2), the message types and representations can be defined to satisfy the message

(\*) For example, the ENC() function may XOR a random value (e.g., a confounder, or a random IV) and the first block of plaintext before generating the ciphertext. With CBC encryption, this makes both the first and subsequent plaintext blocks random. This property is important when a server uses the same key on behalf of multiple clients [SG92].

Constraints on Domain Attributes	Constraints on Decrypted-Range Attributes	
	Unconstrained	Constrained
Both Unconstrained	N/A <sup>1</sup> (vulnerable)	EA <sup>2</sup>
Secret-Domain Constrained	MA2 <sup>3</sup> (or none   SA1)	MA2 and/or EA   MA1 <sup>4</sup> or EA   SA1
Decrypted-Domain Constrained	N/A <sup>5</sup>	EA   MA3 and EA <sup>6</sup> or EA   SA1

“|” reads Given

**Table 2.**

**Property Assumptions Protecting Against Choosing a Range Representation to Match a Domain Representation.**

integrity condition. In Tables 1 and 2, we summarize a few of the possible MOC() and ENC()/DEC() properties for typical message structures and representations, which can be selected using designer-specified attribute constraints.

(1) *Properties limiting the probability an attacker can choose a domain representation to match a range representation.*

The properties of the MOC() function, defined below, can be used to limit the probability an attacker can choose a domain representation to match a range representation. In the properties that follow, a block is defined to be “unknown” to the attacker whenever the values of that block are uniformly distributed over all possible values of any block in the range of the MOC() function. Also, the size of each block in the range of the MOC() function, k, is assumed to be sufficiently large so that each probability below approximates  $1/2^k$ .

**MOC Function Properties:**

**MA1.** Given X of length k, and (P1...Ps-1 Pt+1..Pu), that contains m>0 or more unknown blocks, Prob[choosing Ps...Pt such that MOC(P1...Ps-1 Ps...Pt Pt+1...Pu) = X]  $\approx 1/2^k \leq$  Threshold.

In practice, the value m of MA1 would typically be a constant corresponding to the length of a moc-key.

**MA3.** Given X of length k, and a set of p or fewer random but known blocks of length b, Prob[choosing P1...Ps-1 Ps...Pt Pt+1...Pu such that each block of Ps...Pt is a member of the set, and MOC(P1...Ps-1 Ps...Pt Pt+1...Pu) = X]  $\approx 1/2^k \leq$  Threshold.

In practice, the p or fewer random blocks can correspond to the plaintext of known plaintext-ciphertext pairs. Note that when the number of known pairs exceeds p, the approximation by  $1/2^k$  of the MOC()property probabilities becomes invalid.

**SA1.** Given X of length k, Prob[choosing P1...Pu such that MOC(P1...Pu) = X]  $\approx 1/2^k \leq$  Threshold.

In practice, a strong one-way function may satisfy property SA1.

We assume properties SA1 and MA3 have the desirable characteristic that the application of the MOC() function to the set of all possible values in its domain produces all possible MOC range values, with the probability of generating a given MOC range value being uniformly distributed for all possible domain values.

(2) *Properties limiting the probability an attacker can choose a range representation to match a domain representation.*

The MOC() function property MA2, defined below, can be used to limit the probability an attacker can choose a range representation to match a domain representation.

**MA2.** Given  $(P1...Pu)$ , that contains  $m > 0$  or more unknown blocks,  $\text{Prob}[\text{choosing } X \text{ such that } \text{MOC}(P1...Pu) = X] \approx 1/2^k \leq \text{Threshold}$ .

The specification of the MOC() and ENC()/DEC() properties in Table 1 and 2 illustrates the “high-water-mark” property selection; i.e., functions that satisfy required properties for weaker protocol constraints on secret- and decrypted-attributes can also protect message types that impose stronger constraints. For example, property SA1 of entry 2 limits the probability an attacker can choose a domain representation to match a range representation when both the secret- and decrypted- domain attributes are unconstrained. Since property SA1 places no constraints on the MOC() function domain,  $P1...Pu$ , property SA1 also applies when a constrained moc-key attribute is within the MOC() function domain (as could be the case in entry 4 of Table 1). Similarly SA1 also applies when encryption constrains the attacker from choosing known values within the MOC() function domain as in entry 6. The “high water mark” property also applies moving from the left side to the right side. For example, property MA1 of entry 3, is also applicable to entry 4, since a constrained domain attribute is no less constrained when the decrypted-range attribute is also constrained.

To understand the role of all properties in limiting the attacker’s choice of domain and range representations, consider the possibilities that arise from all the combinations of message attribute constraints illustrated in Tables 1 and 2:

Entry 1: When the decrypted-range and decrypted-domain is unconstrained, no function property can restrict the attacker from choosing a range (domain) representation for any domain (range) representation.

Entry 2: SA1 constrains an attacker from finding a domain representation to match a range value, and property EA constrains the attacker from creating a ciphertext representation for a target MOC value. We assume properties SA1 and MA3 (from Table 1) have the desirable characteristic that the application of the MOC() function to the set of all possible values in its domain produces all possible MOC range values, with the probability of generating a given MOC range value being uniformly distributed for all possible domain values. This property is important because, whenever the MOC value is encrypted (assuming EA), it maximizes the attackers uncertainty of choosing the correct range representation.

Entry 3: Property MA1 acts like SA1 with the exception that some of the domain must be constrained. Property MA2 implies that the attacker cannot use analytic methods to determine the MOC value when a secret-domain attribute is constrained. The notation “(or SA1)” in entry 3 of Tables 1 means that SA1 applied in entry 3 since SA1 is a stronger, more general function. That is, a function satisfying property SA1 will also satisfy MA1. The entry “(or none given SA1)” in Table 2 means that no additional property is needed. (This is due to the, previously stated assumption associated with SA1.)

Entry 4: Similar descriptions as entries 2 and 3 apply due to “high-water-mark” property selection. “MA2 and/or EA | MA1” means that given MA1 in Table 1, properties MA2 and/or EA may be applicable. MA2 may be applicable due to the constraint of a keyed MOC() function. EA is applicable when the decrypted-range attribute is constrained by encryption.

Entry 5: This entry could be considered vulnerable since the attacker is not constrained in choosing a range representation for any domain representation consisting of known decrypted-domain attribute values. However, if the decrypted-domain attribute is constrained (perhaps by encryption or by non-cryptographic constraints of protocol semantics that help preserve attribute membership [SG92]) such that the attacker can not create a domain representation that is legitimate, then the addition of properties such as MA3 and EA in entry 5 of Table 1 could prevent the attacker from verifying a domain representation that matches a range representation. These properties could be sufficient to protect message integrity without requiring additional assumptions in Table 2.

Entry 6: MA3 and EA in Table 1 can be applicable when the MOC() function is keyed and the domain is encrypted. EA in Table 2 might apply given “MA3 and EA” in Table 1 since the range may also be encrypted. Although, EA in Table 2 is not necessary if protocol semantics help preserve attribute membership (as discussed above). This is expected since high water mark properties from left (entry 5) to right (entry 6) apply. SA1, and “EA given SA1” also apply due to high-water-mark properties.

#### (4) Lifetime Assumptions

By choosing the lifetime to be no greater than bounds  $dk$ ,  $dd$ , and  $dr$ , we ensure that the specified message types and functions satisfy conditions S1 and S2 (shown formally in [SG93b]).

The message integrity design must constrain the time period during which the attacker may analytically verify the recognition of bogus messages of that type. The time constraints ensure that (1) the property assumptions are satisfied (viz.,  $dk$ ), and (2) the property assumptions limit the probability, to less than the integrity threshold, that an attacker can choose a bogus domain representation to match a range representation (viz.,  $da$ ) and/or choose a range representation to match a domain representation, (viz.,  $dr$ ). Thus by choosing  $dk$ ,  $dr$  and  $dd$ , the conditions of clauses S1

and S2 can be satisfied. We choose the time constraints for a message type T as follows:

*Step i.* Derive constraints on the lifetime,  $d_k$ , such that  $\text{Prob}[\text{assumptions about message attributes of message type T are not satisfied}] \leq \text{Threshold}$ .

Time constraint  $d_k$  limits the probability that assumptions about the use of MOC(), and ENC()/DEC() become invalid. For example, if an unknown value (e.g., a secret key, block decrypted from ciphertext) is assumed to be in the domain of MOC(), or ENC()/DEC(), then  $d_k$  is chosen to limit the probability that the attacker may verify that unknown value. Time constraint  $d_k$  may also limit the lifetime of a protocol run to limit the number of plaintext-ciphertext pairs that may be known (e.g., as required in MOC() property MA3).

The requirements of the next two steps of the design method are satisfied by using the property assumptions of MOC() and ENC()/DEC() to limit the probability that the attacker may successfully create a bogus message, perhaps via repeated trials, by choosing domain or range representations.

*Step ii.* Using assumptions about MOC() and ENC()/DEC(), derive a time constraint,  $d_d$ , such that the condition of clause S1 is satisfied.

*Step iii.* Using assumptions about MOC() and ENC()/DEC(), derive a time constraint,  $d_r$ , such that condition S2 is satisfied.

The conditions of clauses S1 and S2 are satisfied by choosing the lifetime of the protocol run and key to be the minimum of  $d_d$ ,  $d_r$ , and  $d_k$ .

#### 4. EXAMPLES OF MESSAGE TYPES SATISFYING THE MESSAGE INTEGRITY CONDITION

Two message types, together with the stated protocol assumptions and MOC properties, are shown to satisfy the conditions of clauses S1 and S2 of the message integrity condition. Other examples of structures whose attributes are subject to other constraints can be found using our design method [SG92,SG93a]. In the message types that follows,  $R$  is assumed to be the maximum rate the protocol generates known plaintext-ciphertext pairs. Also, we further specify assumption PA3 (i.e., the computational power of the attacker), we specify the rates at which the attacker may compute the ENC and MOC() functions. For simplicity, we will assume these rates take into account parallel computation and are adjusted to accommodate the various cryptanalytical methods of attack.

$V_{\text{enc/dec}}$  is the rate of guessing *key* and verifying the condition,  $C = \text{ENC}(\text{key}, P)$ , such that  $P, C$  is a known plaintext-ciphertext pair. We can assume this rate is quicker than the rate at which known plaintext-ciphertext pairs are generated by the protocol.

$V_{\text{moc}}$  is the rate of computing a MOC value, and equivalently verifying an unknown value in the domain of MOC().

(A separate  $V_{\text{dec}}$  or  $V_{\text{moc}}$  is used for varying rates of encryption and decryption or signing and verifying as is the case when asymmetric cryptosystems are used.)

##### Safe Message Type

(1) Integrity Policy.

$\text{Prob}[\text{Bogus message } P_1 \dots P_{i-1} \text{ } C_i \dots C_j \text{ is recognizable}] \leq \text{Threshold}$ .

(2) Message Structure and Representation.

Message:  $t\text{-key}, P_1 \dots P_{i-1}, P_i \dots P_j, C_i \dots C_j, 1 \leq i \leq j$ .

Layer Functions:

$\text{MOC}(P_1 \dots P_{i-1}) = P_i \dots P_j$ ;

$\text{ENC}(t\text{-key}, P_i \dots P_j) = C_i \dots C_j$

Representation:  $P_1 \dots P_{i-1} \text{ } C_i \dots C_j$

Assumptions:

Protocol: PA1, PA2, PA3 for  $t\text{-key}$

Functions: EA for ENC()/DEC()

MessageAttribute Receiver

RepresentationClass Decrypted Attributes

$P_1 \dots P_{i-1} \text{domain} \quad P_1 \dots P_{i-1}$

$C_i \dots C_j \text{range} \quad \text{DEC}(t\text{-key}; C_i \dots C_j)$

*Constraints on Receiver Attributes*

Decrypted Attribute:  $P_1 \dots P_{i-1}$

Class: Decrypted-Domain

Constraint: Unconstrained

$P_1 \dots P_{i-1}$  is unconstrained since its derivation is independent of any constrained attribute.

Decrypted Attribute:  $\text{DEC}(t\text{-key}; C_i \dots C_j)$

Class: Decrypted-Range

Constraint: Constrained

Since  $P_1 \dots P_{i-1}$  is unconstrained and the MOC() function is public,  $\text{ENC}(t\text{-key}, \text{MOC}(P_1 \dots P_{i-1}))$  generates known plaintext-ciphertext pairs. Thus,  $\text{DEC}(t\text{-key}; C_i \dots C_j)$  must be constrained to satisfy EA3.

Secret-Domain Attribute: None

Summary of Constraints:

Secret-Domain Attribute: None

Decrypted-Domain: Unconstrained

Decrypted-Range Attribute: Constrained

(3) Properties of ENC()/DEC() and MOC()

Against choosing a domain representation: SA1

Against choosing a range representation: EA

(Note any of the combinations in entry 4 or 6 of the Tables may be chosen.)

(4) Message Lifetime

*Step i.* Derive time constraint,  $d_k$ , such that the  $\text{Prob}[\text{constraints within the property assumptions are satisfied}] \leq \text{Threshold}$ .

Property: SA1

Property Constraint: None

Lifetime Constraint: None

SA1 specifies no constraints within the property assumption, thus no additional lifetime constraint is needed.

Property: EA

Property Constraint:  $t\text{-key}$  is Constrained.

Lifetime constraint:  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$ .

By PA1, principals are trusted to maintain privacy of the  $t\text{-key}$ . Time,  $d_k$ , is chosen such that the  $\text{Prob}[\text{the attacker may}$

verify the  $t$ -key]  $\leq$  Threshold. Therefore,  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$ .

*Step ii.* Derive time,  $d_a$ , such that the probability that the attacker chooses a domain representation that matches a range representation is less than the Threshold (i.e., S1).

**Proof:** Suppose all plaintext blocks of  $P_i \dots P_j$  are known to the attacker. Property SA1 limits the probability to  $1/2^k$ , that an attacker can choose an unconstrained domain representation to match a range representation. Since a maximum of  $(R \cdot d_a)$  plaintext ciphertext pairs are known such that  $(R \cdot d_a) < 2^b$ , the maximum number of MOC values  $P_i \dots P_j$ , known to the attacker is  $(R \cdot d_a)^c$ , such that  $R$  is the maximum rate that known plaintext-ciphertext pairs are generated by the protocol, and  $c = 1$ , if  $k < b$ , and  $c = \lfloor k/b \rfloor$ , if  $k \geq b$ . The maximum number of  $[P_1 \dots P_{i-1}, \text{MOC}(P_1 \dots P_{i-1})]$  pairs the attacker knows is  $(n + (V_{\text{moc}} \cdot d_a))$  such that  $n$  pairs are computed before the run and  $(V_{\text{moc}} \cdot d_a)$  message texts may be computed during the run. Therefore, the probability that none of the MOC value of  $[P_1 \dots P_{i-1}, \text{MOC}(P_1 \dots P_{i-1})]$  equals any of the range representation whose MOC value is known is  $(1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_a))}$   $(R \cdot d_a)^c$ . The probability that at least one MOC is equal to any plaintext combination, is  $1 - (1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_a))}$   $(R \cdot d_a)^c$ . However, if at least one plaintext block  $P$  of  $P_i \dots P_j$  unknown to the attacker, then the probability that the attacker chooses the correct ciphertext block from unknown plaintext-ciphertext pairs is less than one when  $(R \cdot d_a)^c < 2^b$ . The probability the attacker chooses a domain representation when the range representation is known is higher, therefore,  $\text{Prob}[\text{at least one bogus message is recognizable}] = 1 - (1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_a))}$   $(R \cdot d_a)^c \leq$  Threshold. $q$

*Step iii.* Derive time,  $d_r$ , such that the probability the attacker chooses a range representation to match a domain representation is less than the Threshold (i.e., P2).

**Proof:** It follows from property assumption SA1 that the MOC() computed over the domain is assumed to be uniformly distributed. Using this property, the proof for deriving  $d_r$ , and choosing a range representation to match a domain representation is analogous to the proof in Step ii. Therefore, choose  $d_r$  such that the following conditions hold for the  $\text{Prob}[\text{at least one bogus message is recognizable}]$ :  $1 - (1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_r))}$   $(R \cdot d_r)^c \leq$  Threshold. $q$

By choosing the lifetime to be no greater than  $d_k$ ,  $d_a$ , and  $d_r$ , we satisfy conditions **S1** and **S2** respectively, therefore  $\text{Prob}[\text{Bogus message } P_1 \dots P_{i-1} \text{ } C_i \dots C_j \text{ is recognizable}] \leq$  Threshold. $q$

We now apply the method to another message type.

## PRIVATE Message Type

### (1) Integrity Policy.

$\text{Prob}[\text{Bogus message } C_1 \dots C_n \text{ is recognizable}] \leq$  Threshold.

---

(\*) If we had not assumed SA1, then it would be advantageous for the attacker to choose range representation that had a higher probability of occurring.

### (2) Message Structure and Representation.

Message:  $t$ -key, moc-key,  $P_1 \dots P_{i-1}$ ,  $P_i \dots P_j$ ,  $P_{j+1} \dots P_n$ ,  $C_1 \dots C_n$ ,  $1 \leq i \leq j \leq n$ .

Layer Functions:

$\text{MOC}(\text{moc-key}, P_1 \dots P_{i-1}, P_{j+1} \dots P_n) = P_i \dots P_j$ ;  $\text{ENC}(t\text{-key}, P_1 \dots P_n) = C_1 \dots C_n$

Representation:  $C_1 \dots C_n$

Assumptions:

Protocol: PA1 - PA3,  $\text{key} = \{t\text{-key}, \text{moc-key}\}$ ,

Functions: EA for ENC()/DEC(), MA1, MA2

MessageAttribute Receiver

RepresentationClass Decrypted-Attributes

$C_1 \dots C_{i-1}$ domain DEC( $t$ -key;  $C_1 \dots C_{i-1}$ )

$C_i \dots C_j$ range DEC( $t$ -key;  $C_i \dots C_j$ )

$C_{j+1} \dots C_n$ domain DEC( $t$ -key;  $C_{j+1} \dots C_n$ )

### Constraints on Receiver Attributes

Decrypted Attribute: DEC( $t$ -key;  $C_1 \dots C_{i-1}$ ), and DEC( $t$ -key;  $C_{j+1} \dots C_n$ )

Class: Decrypted-Domain

Constraint: Constrained

Both DEC( $t$ -key;  $C_1 \dots C_{i-1}$ ), and DEC( $t$ -key;  $C_{j+1} \dots C_n$ ) generate known pairs. Also, by EA, no chosen plaintext pairs exist.

Decrypted Attribute: DEC( $t$ -key;  $C_i \dots C_j$ )

Class: Decrypted-Range

Constraint: Constrained.

DEC( $t$ -key;  $C_i \dots C_j$ ) is constrained since the other attributes of this message type generate known plaintext-ciphertext pairs.

Attribute: moc-key

Class: Secret-Domain

Constraint: Constrained by assumption PA1 and PA2.

### Summary of Constraints.

Secret-Domain: Constrained

Decrypted-Domain: Constrained

Decrypted-Range: Constrained

### (3) Properties of ENC()/DEC() and MOC()

Against choosing a domain representation: MA1

Against choosing a range representation: MA2 & EA

### (4) Message Lifetime

*Step i.* Derive time constraints,  $d_k$ , such that the  $\text{Prob}[\text{constraints within the property assumption are satisfied}] \leq$  Threshold.

Properties: MA1, MA2

Property Constraint: moc-key is Constrained

Lifetime constraint:  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{moc}}$

By PA1, principals are trusted to maintain privacy of the moc-key. Time,  $d_k$ , is chosen such that the  $\text{Prob}[\text{the attacker may verify the moc-key}] \leq$  Threshold. Therefore,  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{moc}}$ .

Property: EA

Property Constraint: key is Constrained.

Lifetime constraint:  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$

By PA1, principals are trusted to maintain privacy of the  $t$ -key. Time,  $d_k$ , is chosen such that the  $\text{Prob}[\text{the attacker may$

verify the  $t$ -key]  $\leq$  Threshold. Therefore,  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$ .

*Step ii.* Derive time,  $d_a$ , such that the probability that the attacker chooses a domain representation that matches a range representation is less than the Threshold (i.e., **S1**).

**Proof:** No additional time constraint is necessary to prove condition **S1** since MA1 restricts the probability to less than the Threshold that the attacker may choose a domain representation that matches a range representation independent of whether the decrypted MOC value or decrypted domain values are known. That is, assume MA1 and let the unknown blocks be  $P_1 \dots P_{s-1} = \text{moc-key}$ ; also, let  $P_s \dots P_t = P_1 \dots P_{i-1} P_{j+1} \dots P_n$ ;  $P_{t+1} \dots P_u = \text{NULL}$ ;  $X = P_i \dots P_j$ . Then,  $\text{Prob}[\text{choosing } P_1 \dots P_{i-1} \text{ such that } \text{MOC}(\text{moc-key}, P_1 \dots P_{i-1}, P_{j+1} \dots P_n) = P_i \dots P_j] = 1/2^k \leq \text{Threshold}.q$

*Step iii.* Derive time,  $d_r$ , such that the probability the attacker chooses a range representation to match a domain representation is less than the Threshold (i.e., **P2**).

**Proof:** No additional time constraints is necessary to prove condition **P2** since MA2 restricts the probability to less than the threshold that the attacker may choose a domain representation that matches a range representation independent of the partial constraints on the decrypted MOC value. Assume MA2 and let the unknown blocks be  $\text{moc-key}$ ; also, let  $X = P_i \dots P_j$ ;  $P_1 \dots P_u = (\text{moc-key } P_1 \dots P_{i-1} P_{j+1} \dots P_n)$ . Then,  $\text{Prob}[\text{choosing } P_i \dots P_j \text{ such that } \text{MOC}(\text{moc-key}, P_1 \dots P_{i-1}, P_{j+1} \dots P_n) = P_i \dots P_j] = 1/2^k \leq \text{Threshold}$ . Thus,  $\text{Prob}[\text{Bogus message } P_1 \dots P_{i-1} P_i \dots P_j P_{j+1} \dots P_n \text{ is recognizable}] \leq \text{Threshold}.q$

By choosing the lifetime to be no greater than  $d_k$ ,  $d_a$ , and  $d_r$ , we satisfy conditions **S1** and **S2** respectively, therefore  $\text{Prob}[\text{Bogus message } C_1 \dots C_n \text{ is recognizable}] \leq \text{Threshold}.q$

### Example Lifetime Computations

We demonstrate how to determine lifetime assumptions for the general message types SAFE Messages and PRIVATE Messages. The message structure, representation and lifetime constraints for SAFE Messages are summarized as follows:

#### SAFE Message

(4) *Message Lifetime Constraints:*

$$d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$$

$$1 - (1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_a))(R \cdot d_a)^c} \leq \text{Threshold}$$

$$1 - (1 - 1/2^k)^{(n + (V_{\text{moc}} \cdot d_r))(R \cdot d_r)^c} \leq \text{Threshold}$$

Therefore,

If

*Integrity Threshold:* 1/1,000,000

*rate of known pairs:*  $R = 1000$  known pairs/second

and if we assume the following attacker assumptions:

$V_{\text{enc/dec}}$ : 10# guesses/second.

$V_{\text{moc}}$ : 10! guesses/second.

$n$ : 2#@.

and assign the following layer function parameters:

*t-key space* =  $2^{|t\text{-key}|} = 2\% \$$

*space of MOC value* =  $2^k = 2\% @$

then, time constraints  $d_a$ , and  $d_r$  are computed as follows:

*Step i:*  $d_k \leq 208.5$  days

*Step ii & iii:*  $d_r = d_a \leq 394$  days.

However  $d_a, d_r \leq d_k$  must hold. Therefore,  $d_a$  and  $d_r$  are chosen such that:  $d_r = d_a \leq 208.5$  days.

#### PRIVATE Messages

(4) *Message Lifetime Constraints:*

$$d_k \leq (\text{Threshold} \cdot 2^{|moc\text{-key}|}) / V_{\text{moc}}$$

$$d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}}$$

No additional time constraints on  $d_a$  and  $d_r$  are needed.

Therefore,

If

*Integrity Threshold:* 1/10,000

*rate of known pairs:*  $R = 1000$  known pairs/second

and if we assume the following attacker assumptions:

$V_{\text{enc/dec}}$ :  $10^4$  guesses/second

$V_{\text{moc}}$ :  $10^4$  guesses/second

and assign the following layer function parameters:

*t-key space* =  $2^{|t\text{-key}|} = 2\% ^$

*space of MOC value* =  $2^k = 2! @*$

then, time constraints  $d_a$ , and  $d_r$  are computed as follows:

*Step i:*  $d_k \leq (\text{Threshold} \cdot 2^{|t\text{-key}|}) / V_{\text{enc/dec}} = 83.4$  days, and

$$d_k \leq (\text{Threshold} \cdot 2^{|moc\text{-key}|}) / V_{\text{moc}} = 3.93 \times 10^4 \#$$

days.

*Step ii & iii:*  $d_r, d_a \leq d_k$

Therefore, choose  $d_a$  and  $d_r$ , such that:

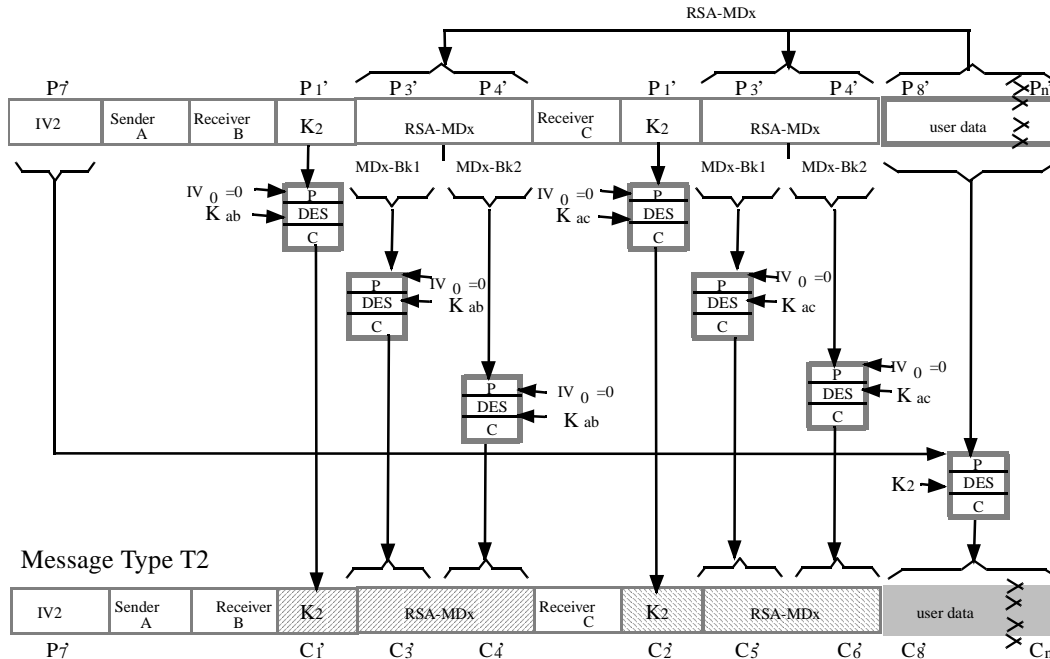
$$d_a \leq 83.4 \text{ days and } d_r \leq 83.4 \text{ days.}$$

By choosing  $d_a$  and  $d_r$  as shown in these examples, the designer obtains message structures that satisfy conditions **S1** and **S2**. Therefore, the sufficiency theorem of Section 3 tells us that the message integrity condition is satisfied. (Of course, a weakness in the ENC()/DEC() or MOC() function could have a significant impact on the adequacy of  $d_r$ ,  $d_a$ , and  $d_k$ .)

## 6. Applying the Method to Correct a Protocol Vulnerability

We demonstrate how to apply the method to determine what properties of the redundancy function can be used to satisfy an integrity protection objective.

In applying the method to Privacy Enhanced Electronic Mail (PEM) [Linn89], not only properties of the redundancy function are determined, but also a potential vulnerability is indicated. This potential vulnerability surfaced when it was realized that the actual mechanism chosen in PEM, DES-



**Figure 2. The Representation of PEM Messages using RSA-MDx\***

MAC [ANSI82], was never intended to satisfy the properties that were determined from the application of the method. Further investigation led to the discovery of the attack scenario confirming the vulnerability. We now present the application of the method, whereas the attack scenario can be found in [SG93a].

PEM provides confidentiality, authentication, and message integrity services for electronic mail transfer. This version is vulnerable since known pairs from a message of type T2, in Figure 2, can be used to create a bogus message of type T1, Figure 3.

A message of type T2 uses the “unkeyed” RSA-MDx redundancy function, DES CBC mode of encryption, and can be sent to any number of addressees. The key used to encrypt the user data,  $K_2$ , and the MDx digest are encrypted keys shared pair-wise between sender and recipients (e.g,  $K_{ab}$ , and  $K_{ac}$  in Figure 2). A message sent from principal  $a$  to principals  $b$  and  $c$  and has the following message structure and representation:

**Message - Type T2**

Message:  $K_{ab}, K_{ac}, K_2, C_1', C_2', P_3', P_4', C_3', C_4', C_5', C_6', P_7', P_8' \dots P_n', C_8' \dots C_n'$

Layer Functions:

- Random()=  $K_2, P_7'$
- ENC( $K_{ab}, 0 \dots 0; K_2$ ) =  $C_1'$
- ENC( $K_{ac}, 0 \dots 0; K_2$ ) =  $C_2'$
- MOC( $P_8' \dots P_n'$ ) =  $P_3', P_4'$
- ENC( $K_{ab}, 0 \dots 0; P_3'$ ) =  $C_3'$
- ENC( $K_{ab}, 0 \dots 0; P_4'$ ) =  $C_4'$
- ENC( $K_{ac}, 0 \dots 0; P_3'$ ) =  $C_5'$
- ENC( $K_{ac}, 0 \dots 0; P_4'$ ) =  $C_6'$
- ENC( $K_2, P_7'; P_8' \dots P_n'$ ) =  $C_8' \dots C_n'$

Representation:  $P_7', C_1', C_3', C_4', C_2', C_5', C_6', C_8' \dots C_n'$

*Assumptions:*

Protocol: PA1, PA2, and PA3 where

Principal =  $P^a$  and  $P^b$  for key =  $\{K_{ab}\}$

Principal =  $P^a$  and  $P^c$  for key =  $\{K_{ac}\}$

Also  $P^a, P^b,$  and  $P^c$  maintain privacy of  $K_2$ .

Functions: EA such that ENC()/DEC() is DES CBC mode.

MOC() is the RSA-MDx function and  $P_3', P_4'$  are random.

Space of MOC():  $2^k = 2!@*$

Key Space (of  $K_{ab}, K_{ac}, K_2$ ):  $2\%^\wedge$

Space of Cipherblock:  $2^b = 2^\wedge\$$

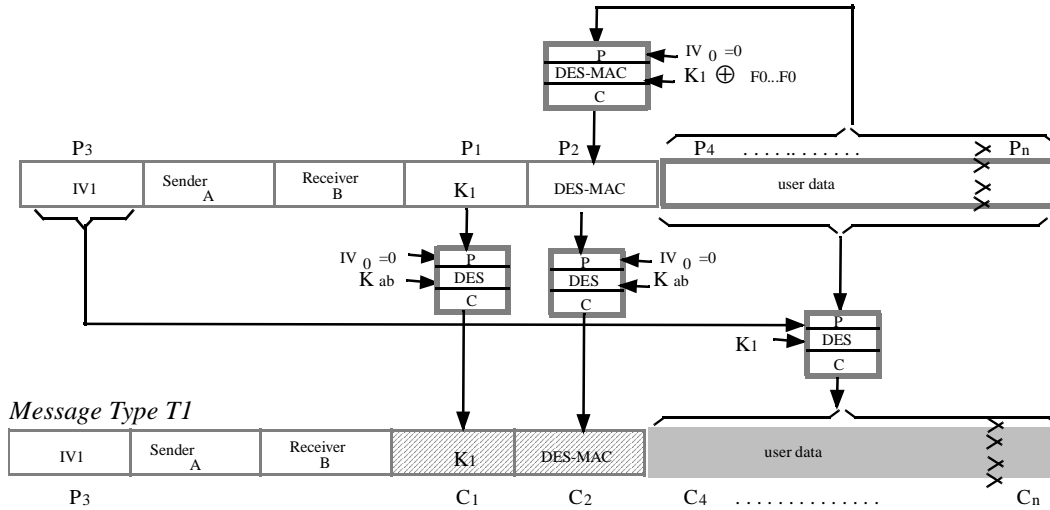
MessageAttribute

Representation Class Decrypted-Attributes

$P_7'$	domain	$P_7'$
$C_1'$	domain	DEC( $K_{ab}, 0 \dots 0; C_1'$ )
$C_2'$	domain	DEC( $K_{ac}, 0 \dots 0; C_2'$ )
$C_3'$	range	DEC( $K_{ab}, 0 \dots 0; C_3'$ )
$C_4'$	range	DEC( $K_{ab}, 0 \dots 0; C_4'$ )
$C_5'$	range	DEC( $K_{ac}, 0 \dots 0; C_5'$ )
$C_6'$	range	DEC( $K_{ac}, 0 \dots 0; C_6'$ )
$C_8' \dots C_n'$	domain	DEC(DEC( $K_{ab}, 0 \dots 0;$
$C_1'), P_7'; C_8' \dots C_n'$ ), or		DEC(DEC( $K_{ac}, 0 \dots 0;$

$C_2'), P_7'; C_8' \dots C_n')$

The method is now applied for message type T1. Note that a complete analysis requires all message types in



**Figure 3. The Representation of PEM Messages using DES-MAC**

the protocol to be specified, not just message types T1 and T2.

**Message - Type T1**

**(1) Integrity Policy.**

Prob[Bogus message P3 C1 C2 C4...Cn is recognizable] ≤ Threshold.

**(2) Message Structure and Representation.**

Message:  $K_{ab}, K1, C1, P2, C2, P3, P4...Pn, C4...Cn$

Layer Functions:

Random()=  $K1, P3$

$ENC(K_{ab}, 0...0; K1) = C1$

$MOC(K1 \oplus F0...F0, 0...0; P4...Pn) = P2$

$ENC(K_{ab}, 0...0; P2) = C2$

$ENC(K1, P3; P4...Pn) = C4...Cn$

Representation:  $P3 C1 C2 C4...Cn$

(Initial) Assumptions:

Protocol: PA1 - PA2 where

Principal =  $P^a$  and  $P^b$ , key =  $\{K_{ab}\}$ , maintain privacy of  $K1$ ;

Functions:

EA such that  $ENC()/DEC()$  is DES CBC mode.  $MOC()$  is DES-MAC and is computed using  $ENC()$  except only the last ciphertext block is saved.

Space of  $MOC()$ :  $2^k = 2^{\wedge}\$$

Key Space (of  $K_{ab}$ ):  $2\% ^{\wedge}$

Space of Cipherblock:  $2^b = 2^{\wedge}\$$

MessageAttribute Receiver

RepresentationClass Decrypted-Attributes

P3	domain	P3
C1	domain	$DEC(K_{ab}, 0...0; C1)$
C2	range	$DEC(K_{ab}, 0...0; C2)$
C4...Cn	domain	$DEC(DEC(K_{ab}, 0...0; C1), P3; C4...Cn)$

Recognition condition for Message Type 1:

$$MOC(DEC(K_{ab}, 0...0; C1) \oplus F0...F0, 0...0; DEC(DEC(K_{ab}, 0...0; C1), P3; C4...Cn)) = DEC(K_{ab}, 0...0; C2)$$

Constraints on receiver attributes.

Decrypted Attribute: P3

Class: Decrypted-Domain (\*)

Constraint: Unconstrained

P3 is unconstrained since it is independent of other attributes and thus is determined directly from the message representation.

Decrypted-Attribute:  $DEC(K_{ab}, 0...0; C1)$

Class: Decrypted-Domain

Constraint: Unconstrained

By checking other decrypted-attributes we discover from message type T2, that an entity other than  $P^a$  and  $P^b$  (i.e., entity  $P^c$ ) knows  $DEC(K_{ac}, 0...0; C2')$  is equal to  $DEC(K_{ab}, 0...0; C1')$ . Thus, for entity  $P^c$  choosing  $C1 := C1'$ ,  $C1$  is known. Since  $DEC(K_{ab}, 0...0; C1)$  represents a shared secret between  $P^a$  and  $P^b$ , we will assume that the knowledge of any  $DEC(K_{ab}, 0...0; C1')$  leaves this attribute unconstrained.

Decrypted-Attribute:  $DEC(K_{ab}, 0...0; C2)$

Class: Decrypted-Range

Constraint: Constrained

Values of  $DEC(K_{ab}, 0...0; C2)$  may be known for the same reason as above. However, since  $DEC(K_{ab}, 0...0; C2)$  represents a decrypted-range attribute, we assume that the knowledge of some values of  $DEC(K_{ab}, 0...0; C1')$  need not require that this attribute be considered unconstrained. For example, properties of the  $MOC()$  function and the space of the range value can compensate for an attacker's knowledge of some values for  $DEC(K_{ab}, 0...0; C2)$ .

Decrypted-Attribute:  $DEC(DEC(K_{ab}, 0...0; C1), P3; C4...Cn)$

Class: Decrypted-Domain

Constraint: Unconstrained

(\*) P3 is within the domain of  $MOC()$  since the derivation of  $P4...Pn = DECcbc(DECcbc(K_{ab}, 0...0; C1), P3; C4...Cn)$  requires P3 to be specified.

Since  $DEC(K_{ab}, 0...0; C1)$  is unconstrained and  $P3$  is unconstrained,  $DEC(DEC(K_{ab}, 0...0; C1), P3; C4...Cn)$  is unconstrained for known values of  $DEC(K_{ab}, 0...0; C1)$ .

Attribute:  $DEC(K_{ab}, 0...0; C1) \oplus F0...F0$

Class: Secret-Domain

Constraint: Unconstrained

Since  $DEC(K_{ab}, 0...0; C1)$  is unconstrained then  $DEC(K_{ab}, 0...0; C1) \oplus F0...F0$  is unconstrained.  $F0...F0$  is known since  $F0...F0$  is a constant and the protocol is assumed to be known.

#### Summary of Constraints.

Secret-Domain: Unconstrained

Decrypted-Domain: Unconstrained

Decrypted-Range: Constrained

This set of constraints corresponds with entry 2 in Tables 1 and 2.

### **(3) Properties of ENC()/DEC() and MOC()**

Protection against choosing a domain representation: SA1

Protection against choosing a range representation: EA

Integrity protection for message type T1 is achieved by choosing an actual function that satisfies the properties in (3) and by limiting the lifetime of the protocol run. A general form of a message structure satisfying these constraints is the Private Message presented earlier. The lifetime constraints are determined in an analogous fashion.

Although important, it is beyond the scope of this paper to determine as to whether actual chosen functions satisfy the properties derived using the design method. In PEM, the DES-MAC was chosen as the MOC() function for the design of message type T1. However, a brief comparison of the DES-MAC function with property SA1 indicates an incompatibility. The design of DES-MAC was intended to require a secret key (i.e., constraints on the domain like that required in property MA1). Choices of actual redundancy functions and alternate message structures to correct this problem are given in [SG93a].

## **7. CONCLUSIONS**

We presented a design method based on a formal model of integrity protection. The use of the method was illustrated by the design of a large class of message types whose integrity is provably preserved in face of active intruder attacks. The use of the method was also illustrated in the elimination of a recently discovered vulnerability of the symmetric key option in the Privacy-enhanced Electronic Mail (PEM) protocol for the Internet.

## **Acknowledgements**

The work reported herein was supported in part by IBM Corporation under contract number YC313314. We thank Rajashekar Kailar and the anonymous reviewers for their noteworthy comments and we are grateful to Tom Tamburo, Marty Simmons, and Peter Will for their support and encouragement.

## **9. REFERENCES**

- [ANSI82] ANSI X9.9-1982, *American National Standard for Financial Institution Message Authentication*, April 13, 1982.
- [Linn89] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I -- Message Encipherment and Authentication Procedures," Internet Working Group, RFC-1113, August 1989.
- [OSF91] Open Software Foundation, OSF, Distributed Computing Environment (DCE), Remote Procedure Call Mechanisms, Code Snapshot 3, Release 1.0, March 17, 1991.
- [SG92] S.G. Stubblebine and V.D. Gligor, "On Message Integrity in Cryptographic Protocols," Proc. IEEE 1992 Research in Security and Privacy, IEEE Computer Society Press, Oakland, California, May 1992. Also extended version in University of Maryland, Technical Report CS-TR-2843, February 1992.
- [SG93a] S.G. Stubblebine and V.D. Gligor, "Protecting the Integrity of Privacy-Enhanced Electronic Mail with DES-Based Authentication Codes", PSRG Workshop on Network and Distributed System Security, San Diego, California, February 11-12, 1993.
- [SG93b] S.G. Stubblebine and V.D. Gligor, "A Model and Protocol Design Method for Integrity Protection", University of Southern California, Information Sciences Institute, Research Report, RR-93-306, 1993.