



# 1

## Reasoning about Message Integrity

Rajashekar Kailar<sup>1</sup>  
Virgil D. Gligor<sup>1</sup>  
Stuart G. Stubblebine<sup>2</sup>

### ABSTRACT

We propose an approach for reasoning about message integrity protection in cryptographic protocols. The set of axioms presented herein relate design parameters and assumptions of message integrity protection mechanisms to generic message integrity threats. Comparison of threat properties derived using these axioms with the policy goals for integrity protection aids in assessing the strength (or lack thereof) of message integrity protection mechanisms. We provide examples to illustrate the use of our approach in examining the weaknesses of message integrity protection mechanisms, and also in suggesting modifications in their design parameters.

**Categories and Subject Descriptors:** C.2.4 [Computer-Communication Networks]: General - *Security and Protection*, Distributed Systems; D.4.6 [Operating Systems]: Security and Protection - *cryptographic controls*; E.3 [Data]: Data Encryption.

**Key Words and Phrases:** Message integrity, integrity threshold, effective threshold, block membership, order, cardinality, plaintext, ciphertext.

### 1.1 Introduction

Cryptographic protocols, and in particular, authentication protocols, rely on message integrity protection. However, mechanisms for protecting message integrity have been shown to be error-prone ([3], [4], [14], [18], [20]). Past studies of message integrity have focused primarily on finding attack scenarios in message integrity protection mechanisms, suggesting solutions to eliminate vulnerabilities, and proposing new algorithms for message integrity protection [2], [3], [4], [5], [14], [15], [18], [21]. Recently, an operational model for message integrity, and a general method for designing message integrity protection mechanisms was also proposed [21]. However, to date, analysis of message integrity protection mechanisms has been done in an ad-hoc manner. A method which considers the types of threats that the environment is exposed to and analyzes protection mechanisms to determine whether they achieve their goals<sup>3</sup> has not been proposed. An analysis method that relates design attributes and assumptions of message integrity protection mechanisms to message integrity goals is useful in (1) analyzing extant protection mechanisms for their threat resistance properties, (2) designing new protection mechanisms, and (3) to help gain insight into properties of message integrity protection mechanisms.

In this paper, we propose a set of axioms which model the threats to message integrity in a given environment. These axioms relate the design parameters and assumptions of message integrity protection mechanisms to generic message integrity threats. Comparison of threat properties derived

---

<sup>1</sup>Electrical Engineering Department, University of Maryland, College Park, MD 20742.

<sup>2</sup>USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292.

<sup>3</sup>Message integrity protection mechanism goals are usually stated in terms of a probability threshold; i.e., in the form *Probability with which the protection mechanism is vulnerable to message integrity compromise is less than a specified threshold* [20].

using these axioms with policy goals for integrity protection allows one to assess the strength (or lack thereof) of protection mechanisms. The analysis of protection mechanisms using our axioms has shown that some of the previously flawed mechanisms [18], [19], [20] do not achieve their policy goals. The analysis of protection mechanisms has also suggested modifications in their design parameters to remove existing vulnerabilities. Although the axioms presented herein mainly concern shared key cryptosystems, this reasoning can be extended to public-key cryptosystems (i.e., those that use RSA scheme [17] or variants thereof) as well. We provide some guidelines on how this may be done in Appendix A.

In our model of computation (section 2), we assume that messages are broken up into blocks and that the encryption algorithm is applied to each of these blocks. Hence, the axioms presented herein refer to block enciphering, and cannot be applied to stream encryption. The axioms presented herein model the threats assumed in our model. Hence, these axioms are applicable in analyzing message integrity protection mechanisms only against the types of threats assumed in our model.

The balance of this paper is organized as follows. In the next section, we briefly discuss our model of computation and introduce the notation that we use to reason about message integrity. In section 1.3 we briefly list some of our assumptions about the operating environment. In section 1.5, we discuss the intuition behind the axioms which relate design parameters and assumptions to the goals of protection mechanisms. We use the axioms to analyze some well known (flawed) protection mechanisms and show how the analysis suggests a change in design parameters in order to achieve the specified message integrity goals.

## 1.2 Model of Computation

Our model of computation is derived from the operational model for message integrity of [19]. This model consists of a set of principals (e.g., users, processes or machines), a distinguished principal ( $X$ ) called the *attacker* and an open network consisting of buffers to/from which messages are sent/received by all principals, including  $X$ . Messages sent over the network may or may not be encrypted, depending on whether their application requires confidentiality or not. Encryption algorithms use keys (random numbers) as arguments, and when applied to plaintext, produce ciphertext.

The plaintext message is broken up by the encryption algorithm into a sequence of blocks and the encryption algorithm is applied to each of those blocks. At the receiving end, the ciphertext blocks are decrypted to obtain the original message. If the plaintext message is viewed as an ordered set of plaintext blocks, then the integrity of a received message is said to be preserved if the received (decrypted) message contains all (and only) the plaintext message blocks that are sent, in the proper order (as intended by the sender). In the terminology of [18], these set properties are called *membership* ( $M$ ), *order* ( $O$ ), and *cardinality* ( $C$ ). The  $M$ ,  $O$  and  $C$  properties of plaintext messages are mapped onto checksum functions. We refer to these functions collectively as “MOC” functions, and the checksums that they compute over all message blocks, as MOC-values.

The goal of an attacker is to construct a valid message representation (i.e., a message representation which decrypts properly and passes the message integrity checks at the receiving end). The goal of the system is to preserve message integrity by keeping the probability of a successful message integrity compromise below a specified threshold. The attacker is capable of recording all message exchanges that occur over the network. Further, the attacker can distinguish between ciphertext blocks that are encrypted with two different keys, without necessarily knowing either key. This is a reasonable assumption, since the knowledge of the session identifier of a message is often indicative of the use of a (particular type of) key.

Whenever a key which is used as an argument in a function (e.g., encryption key in encryption) is not assumed to be secret, the attacker is assumed to know the value of the key with certainty. Otherwise (if the key is secret), the attacker can only guess the value of the key randomly from the space of the key. If a random key which is secret is used as an argument in the domain of the encryption function the ciphertext can be predicted with a very small probability by  $X$ , given that  $X$  has chosen the plaintext values.

To construct a valid message representation, the intruder must find (1) the matching encrypted MOC value for the plaintext message or a plaintext message for an encrypted MOC value, and (2) find the ciphertext corresponding to the plaintext that has the proper MOC value. The second objective may not be relevant in cryptographic protocols using public-key cryptosystems (i.e., those using the RSA scheme [17] or variants thereof), since finding the ciphertext for plaintext amounts to finding the encryption key which is publicly known.

In cryptographic protocols using conventional (shared-key) cryptosystems, the attacker can find the ciphertext for a plaintext message in three ways, namely, by (1) finding the encryption key, (2) by matching the plaintext with the plaintext blocks for which the corresponding ciphertext blocks are known (i.e., by collecting known plaintext-ciphertext block pairs), and (3) by using the

knowledge of domain-range pairs of the encryption function (i.e., plaintext-ciphertext pairs) to make a “calculated guess” of the ciphertext representation.

The attacker can find the encrypted MOC value for a plaintext message in two ways: (1) by finding both the encryption key and MOC function key, or (2) by using the knowledge of the plaintext message and MOC value pairs to make a calculated guess of the encrypted MOC value. An attacker can find a plaintext message for a given encrypted MOC value in three ways: (1) by finding the encryption key and the MOC function key, (2) by using the knowledge of the plaintext and MOC value pairs in addition to the *many to one* mapping from the plaintext messages to the MOC values<sup>4</sup>, and (3) by using the knowledge of the plaintext-ciphertext block pairs to construct a plaintext message for an encrypted MOC value.

The attacker action using known plaintext-ciphertext block pairs is made less effective by using random blocks in the domain of the encryption and MOC functions, in addition to the keys. These random blocks are commonly referred to as confounders. In addition, some encryption algorithms use an initial permutation of the plaintext blocks before encrypting them. The permutation is often referred to as an *Initialization Vector*, (IV). We model the use of confounders and Initialization Vectors (IV’s) collectively, and call them *confounding text*. The effect of adding them is to multiply the (range) space of the ciphertext message representation for a given plaintext message (domain) by the space of the confounding block(s). However, the effect of adding confounding components to the domain of the MOC function on the resulting MOC space is not as straight forward. This is because the size of the MOC block is fixed, and in the worst case, for any message, the MOC value can be randomly guessed with probability of success at least equal to the reciprocal of the MOC block space. We discuss this case in greater detail in our axioms.

### 1.2.1 NOTATION AND DEFINITIONS

*Run* : An epoch during which a set of principals (e.g., users, machines or programs) share a communication channel (e.g., a set of cryptographic keys) to exchange information. In particular, the principals which participate in this type of communication, and the messages that they send/receive thereby, are said to “belong” to the *run*.

*X* : A principal which does not belong to the session (i.e., an illegitimate member of the session, or an attacker).

$P_i$ : A plaintext block (with index  $i$  denoting its position in the message), the size of which is a constant defined in the cryptographic protocol.

$P'_i$ : A *modified* plaintext block.  $P'_i$  is the result of applying some function on  $P_i$ ; e.g., in Cipher-Block-Chaining, the modified plaintext block is obtained by Exclusive-OR-ing the plaintext block with the previous ciphertext block.

$\{P'_i\}_{tkey}$ : Ciphertext block obtained by encrypting a *modified* plaintext block  $P'_i$  using the encryption key ( $tkey$ ). In the case of unencrypted messages (i.e.,  $tkey$  is null)  $\{P'_i\}_{tkey} = P'_i$ .

$MOC(mkey, M)$ : *Membership, order and cardinality* value computed over message  $M$  (consisting of a set of  $b$  plaintext or ciphertext blocks), using the MOC function key ( $mkey$ ). The MOC function key refers to the key used as an argument by the function to compute the checksum.

*X has* ( $n, tkey$ ): Principal  $X$  has  $n$  unique plaintext and ciphertext block pairs encrypted with  $tkey$ . In particular,  $X$  can *have* a plaintext-ciphertext pair either by encrypting known plaintext, or by decrypting a ciphertext, or by observing ciphertext messages whose (plaintext) contents are known.

*X has* ( $m, mkey, tkey$ ): Principal  $X$  has  $m$  unique (plaintext) messages and their corresponding MOC-values computed with  $mkey$ , and (optionally,) encrypted with  $tkey$ .  $X$  can *have* plaintext message and encrypted MOC-value pairs either by sending/receiving known messages and observing the encrypted MOC-values, or by observing ciphertext messages whose plaintext contents are known, and noting their encrypted MOC-values. In cryptographic protocols that do not encrypt the moc-function values, the  $tkey$  is treated as a known constant (say zero), and the  $tkey$  space is unity.

*X finds*  $F$ : Principal  $X$  finds information  $F$ , either by deriving  $F$  from the information it possesses or by randomly guessing  $F$ .

---

<sup>4</sup>Given any message, the probability that its MOC value is equal to some  $C$  is at least equal to the reciprocal of the MOC space.

**$X$  finds  $F$  for  $G$ :** Principal  $P$  finds  $F$  such that  $F$  matches  $G$  in accordance with some matching criteria. In our axioms,  $G$  and  $F$  are used to denote either a plaintext and its corresponding ciphertext, or a text string and its corresponding MOC value (or vice-versa), or a plaintext and its corresponding *confounded* text (formed by an initial permutation or by the addition of a confounding block).

**Secret( $F, Run$ ):** Information  $F$  is secret in the *Run*. In particular, members of the session who legitimately share  $F$  are assumed not to reveal  $F$  to anyone outside of the *Run*.

$|F|$ : The number of all possible values that  $F$  can take; i.e., size of the space of  $F$ .

$T_F$ : The threshold of  $F$ . This is defined as the reciprocal of the space of  $F$ .

$T_{tkey}$ : The reciprocal of the space of the encryption key. In the case of an plaintext message, the key is null, and the key space is unity.

$T_{mkey}$ : The reciprocal of the MOC key space. In the case of un-keyed MOC function, the *mkey* is null, and the *mkey* space is unity.

$T_{block}$ : The reciprocal of the ciphertext block space.

$T_{moc}$ : The reciprocal of the MOC block space.

$T_{conf}$ : The reciprocal of the confounder block space.

$T_{int}$ : The policy threshold for message integrity.

$|(P', \{P'\}_{tkey})|$ : The number of all possible (modified) plaintext and ciphertext block pairs that can be generated using the *tkey*.

$ET_F$  The *effective threshold* of  $F$ . This is defined to be the reciprocal of the *effective search space* of  $F$  if  $F$  is a secret and  $F$  has been used as an argument in some function  $n$  times. In particular, if the threshold of  $F$  is  $T_F$ , then the effective threshold

$$ET_F \equiv \frac{T_F}{1-\sigma(n)},$$

where  $\sigma(n) > 0$  iff  $n > 0$ , and  $\sigma(n+1) \geq \sigma(n)$  if  $n \geq 0$ . Intuitively,  $\sigma(n)$  is a measure of the amount of information that the knowledge of  $n$  domain-range pairs of a function which uses  $F$  as an argument provides in “guessing”  $F$ . When  $n$  equals the size of the space of the domain-range pairs,  $ET_F$  equals 1.

$ET_{tkey}$  The *effective encryption key threshold* if *tkey* is assumed to be a secret and the *tkey* has been used  $n$  times in plaintext block encryption.

$$ET_{tkey} \equiv \frac{T_{tkey}}{1-\delta(n)},$$

where  $\delta(n) > 0$  iff  $n > 0$ , and  $\delta(n+1) \geq \delta(n)$  if  $n \geq 0$ .

Intuitively,  $\delta(n)$  is a measure of the amount of information that the knowledge of  $n$  plaintext-ciphertext block pairs provides in “guessing” *tkey*.

$ET_{mkey}$  The *effective MOC key threshold* if *mkey* is a secret and has been used  $m$  times in message checksum computation.

$$ET_{mkey} \equiv \frac{T_{mkey}}{1-\phi(m)},$$

where  $\phi(m) > 0$  iff  $m > 0$ , and  $\phi(m+1) \geq \phi(m)$  if  $m \geq 0$ .

Intuitively,  $\phi(m)$  is a measure of the amount of information that the knowledge of  $m$  message-MOC-value pairs provides in “guessing” *mkey*.

$ET_{block}$  The *effective ciphertext block threshold* after  $n$  plaintext and ciphertext pairs have been generated.

$$ET_{block} \equiv \frac{T_{block}}{1-\epsilon(n)},$$

where  $\epsilon(n) > 0$  iff  $n > 0$ , and  $\epsilon(n+1) \geq \epsilon(n)$  if  $n \geq 0$ .

Intuitively,  $\epsilon(n)$  is a measure of the amount of information that the knowledge of  $n$  plaintext-ciphertext block pairs provides in “guessing” the ciphertext block for a given plaintext block.

$ET_{moc}$  The *effective* MOC block threshold after  $m$  message and encrypted MOC-value pairs have been generated.

$$ET_{moc} \equiv \frac{T_{moc}}{1-\chi(m)},$$

where  $\chi(m) > 0$  iff  $m > 0$  and  $\chi(m+1) \geq \chi(m)$  if  $m \geq 0$ .

Intuitively,  $\chi(m)$  is a measure of the amount of information that the knowledge of  $m$  plaintext message and MOC-value pairs provides in “guessing” the MOC-value for a given plaintext message.

$ET_{cmoc}$  The *effective* MOC block threshold after  $m$  message and encrypted MOC-value pairs have been generated in message types which use confounders.

$$ET_{cmoc} \equiv \frac{T_{moc}}{1-\chi(m) \times T_{conf}},$$

where  $\chi(m) > 0$  iff  $m > 0$  and  $\chi(m+1) \geq \chi(m)$  if  $m \geq 0$ .

Intuitively,  $\chi(m)$  is a measure of the amount of information that the knowledge of  $m$  plaintext message and MOC-value pairs provides in “guessing” the MOC-value for a given plaintext message.

$ET_{pair}$  The probability of success in finding a given plaintext-ciphertext pair among  $n$  such pairs.

If the number of all possible pairs is  $|(P', \{P'\}_{tkey})|$ , and the number of pairs that have already been generated is  $n$ , then, assuming a uniform distribution of blocks,

$$ET_{pair} \equiv \frac{(n+1)}{|(P', \{P'\}_{tkey})|}.$$

In the analysis presented in this paper, *effective thresholds* are used for establishing that if  $ET_F$  is a function of  $n$  (the number of available domain-range pairs of a function), then  $ET_F > T_F$  for  $n > 0$ . In other words, the effective threshold is larger than the real threshold for non-zero values of  $n$ . Further,  $ET_{pair}$  is useful in deriving a limit on the number of pairs that can be generated. We will discuss this in some detail in section 1.5.

### 1.3 Environment Assumptions

In reasoning about message integrity protection mechanisms, we make several assumptions about the operating environment and about the properties of the underlying system functions. We refer to these assumptions collectively as the *environment assumptions*. The axioms that we present, and hence, the results derived using them are based on these assumptions.

- The encryption/decryption functions of block ciphers break a plaintext message  $M$  into  $b$  blocks  $P_1, P_2, \dots, P_b$  and apply a specific function which transforms the  $b$  plaintext blocks into  $P'_1, P'_2, \dots, P'_b$ , (e.g., in cipher-block chaining,  $P'_i$  is a function of  $P_i$  and  $\{P'_{i-1}\}_{tkey}$ ) and encipher each block with the same key; i.e.,

$$\{M\}_{tkey} = \{P_1, \dots, P_b\}_{tkey} = \{P'_1\}_{tkey}, \{P'_2\}_{tkey}, \dots, \{P'_b\}_{tkey}$$

- The block size of the cipher is a constant defined in the design of the protection mechanism. Pair  $(P', \{P'\}_{tkey})$  denotes the modified plaintext  $P'$  and its corresponding ciphertext block  $\{P'\}_{tkey}$ . While the correlation between a plaintext block  $P$  and its corresponding ciphertext block  $\{P'\}_{tkey}$  cannot necessarily be determined, there is a one-to-one mapping between a modified plaintext block  $P'$  and its corresponding ciphertext block  $\{P'\}_{tkey}$ . In the rest of the paper, we refer to *modified* plaintext and ciphertext block pairs simply as plaintext-ciphertext block pairs.
- In message types which do not use checksums, to construct a ciphertext representation for the plaintext message  $P_1, P_2, \dots, P_b$ ,  $X$  must find the *modified plaintext*<sup>5</sup>  $P'_1, P'_2, \dots, P'_b$ , and
  - encrypt each modified block with  $tkey$  to produce the ciphertext representation, or
  - find the corresponding ciphertext blocks  $\{P'_1\}_{tkey}, \{P'_2\}_{tkey}, \dots, \{P'_b\}_{tkey}$ , from among the known plaintext and ciphertext block pairs.

---

<sup>5</sup>The modified plaintext blocks are a function of the plaintext blocks, and may not always be determined with certainty from the knowledge of the plaintext blocks.

Possession of plaintext blocks  $P_1, P_2, \dots, P_b$  does not necessarily allow  $X$  to find the *modified plaintext blocks*  $P'_1, P'_2, \dots, P'_b$  with certainty. However, there are examples ([18]) where it is possible to obtain  $P'_i$  from  $P_i$  with certainty (e.g., all cryptographic protocols which use DES-CBC [6] encryption mode).

- In message types which use checksums, the MOC function is computed over the entire message (all blocks), and may be encrypted with the *tkey*. In such message types, the possession of ciphertext blocks  $\{P'_1\}_{tkey}, \dots, \{P'_b\}_{tkey}$  corresponding to plaintext blocks  $P_1, \dots, P_b$  does not allow  $X$  to construct a valid message representation unless he can also find the corresponding *modified* MOC value  $\{MOC(mkey, (P_1, P_2, \dots, P_b))'\}_{tkey}$  for the plaintext message  $P_1, P_2, \dots, P_b$ . The valid message representation is idealized as:

$$\{\{P'_1\}_{tkey}, \{P'_2\}_{tkey}, \dots, \{P'_b\}_{tkey} \{MOC(mkey, (P_1, P_2, \dots, P_b))'\}_{tkey}.$$

Note that the placement of the MOC block in the message representation is a function of the message type, and is not considered in this analysis. The probability with which  $X$  can find a valid message representation  $\{M'\}_{tkey} = \{P_1, P_2, \dots, P_b, MOC(mkey, M')\}_{tkey}$  is less than or equal to the probability of finding the ciphertext blocks  $\{P_1\}_{tkey}, \{P_2\}_{tkey}, \dots, \{P_b\}_{tkey}$  and the moc-value  $\{MOC(mkey, M)\}_{tkey}$ .

- The encryption and checksum (MOC) functions are common knowledge. The only unknown arguments in these functions are the keys that they use.
- In any collection of modified plaintext and ciphertext block pairs, modified plaintext blocks occur with uniform probability. This assumption is used to derive the probability of finding a modified plaintext block and its corresponding ciphertext block in a group of plaintext-ciphertext block pairs. In examples where some blocks are more probable than others, this assumption can be relaxed to give a more detailed analysis.
- At any given instance in the run, the number of all possible valid message representations using a *tkey*, *mkey* pair is much larger than the number of valid message representations that are generated during the lifetime of these keys. In particular, the ratio of the maximum number of messages generated to the total number of possible messages is smaller than  $T_{int}$ .
- Legitimate principals maintain the privacy of the secret components of the keys that they possess.
- The encryption key, moc-key and confounding blocks are chosen at random from their candidate spaces.

## Preliminary Observations

- The number of all possible (modified) plaintext-ciphertext block pairs that can be generated using the *tkey* is not larger than the number of all possible values that a ciphertext block can take. While this limit is quite obvious, it is significant because the inequality also supports properties of cryptographic protocols in which not all (modified) plaintext and ciphertext block pairs can exist, since some plaintext message blocks are not considered valid.
- The number of all possible message representations and their corresponding MOC-values is much larger than the number of all possible values that a moc-block can take. This is because the message may consist of several blocks (which can be permuted) and each block can have  $|block|$  values. For instance, if we consider a message containing 5 or more blocks, where each block is 64 bits in length, the space  $|M, \{MOC(mkey, M)\}_{tkey}|$  would be much larger than  $2^{256}$ , which is the space of the largest MOC block used in practice.
- In this analysis, we do not consider the vulnerabilities due to replay of old messages. This type of threat does not compromise message integrity of individual messages. However, our approach can be used to analyze message integrity attacks that are due to the use of old message contents (or blocks) to construct new messages.

## 1.4 Axioms

We present a set of axioms which relate message integrity protection mechanism design parameters and assumptions to the message integrity goals. The axioms follow from our model of computation

and environment assumptions. Hence, the axioms derive message integrity properties in the context of the set of threats that are assumed within our model of computation.

### 1.4.1 SECRECY

This axiom states that if  $F$  is some information that is assumed to be secret throughout the session run, then a principal  $X$  can find  $F$  with probability less than or equal to  $ET_F$ .

$$\mathbf{A}_1: \quad \begin{array}{l} \mathbf{Secret}(F, Run) \Rightarrow Prob[X \text{ finds } F] \leq ET_F \\ \neg \mathbf{Secret}(F, Run) \Rightarrow Prob[X \text{ finds } F] = 1 \end{array}$$

where  $ET_F$  is the *effective threshold* of  $F$  (section 2.1).

That is, if  $F$  is assumed to be a secret, then the choice of  $F$  by some principal  $X$  is uniformly random in the “effective” space of  $F$ . If  $F$  is not a secret, then it can be found with certainty by the intruder  $X$ . We use this axiom in the analysis of protection mechanisms to define the probability with which keys can be predicted. If a key is assumed to be secret, we define the probability with which  $X$  finds the key to be at most equal to the effective threshold of the key. In the application of this axiom, unkeyed functions (e.g., unkeyed MOC) are considered to be equivalent to keyed functions which use known constants as their keys. This is because in both these types of functions, the value of the range can be predicted with certainty for a given domain, if the function does not add any unknown component to its domain. In such mechanisms, the key can be found with certainty.

### 1.4.2 FINDING A CIPHERTEXT REPRESENTATION FOR A GIVEN PLAINTEXT

In this axiom, we compare the three search processes using which  $X$  may be able to find a matching ciphertext representation  $\{M\}_{tkey}$  for a given message  $M = P_1, \dots, P_b$ .

$$\mathbf{A}_2: \quad \frac{X \text{ has } (n, tkey); Prob[X \text{ finds } tkey] = S}{Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M] \leq \max(S, (ET_{pair})^b, (ET_{block})^b)}$$

where  $n \leq |(P', \{P'\}_{tkey})|$ , and  $b$  is the number of blocks in message  $M$ . If  $tkey$  is a secret,  $S$  is  $ET_{tkey}$ . Otherwise,  $S$  is 1. By definition,  $ET_{pair}$  is the probability of finding a given plaintext-ciphertext pair among  $n$  pairs. The intruder would require  $b$  such pairs, each of which would have to be searched independently. Hence, we obtain the second term  $(ET_{pair})^b$ . By definition,  $ET_{block}$  is the effective threshold of a ciphertext block. The intruder would need  $b$  such blocks to compose a message. Hence, the term  $(ET_{block})^b$ .

The probability with which  $X$  finds  $\{M\}_{tkey}$  for a given  $M$  is at most as large as the maximum of these three probabilities (i.e., the one which is most likely to be successful). In this axiom, we do not take into consideration the probability with which  $X$  can also find a matching checksum for the plaintext (We describe this in another axiom, namely,  $A_3$ ). In the analysis of message integrity protection mechanisms, we use this axiom in conjunction with the property that the ciphertext block space is no smaller than the plaintext-ciphertext block pair space. That is,  $ET_{pair}$  is greater than  $(n+1) \times T_{block}$ . In this axiom, we do not take into account any *confounding text* that the layer may add to the domain of the encryption function. This omission is intentional, since we treat this feature separately in axiom  $A_4$ .

An interesting result of axiom  $A_2$  is that if  $(ET_{pair})^b > T_{int}$  or if  $(ET_{block})^b > T_{int}$  then  $Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M]$  is not limited by  $T_{int}$ . From the definitions of these effective thresholds, it follows that if  $n > |(P', \{P'\}_{tkey})| \times (T_{int})^{\frac{1}{b}}$  or if  $\epsilon(n) > 1 - \frac{T_{block}}{(T_{int})^{\frac{1}{b}}}$ , then  $Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M]$  is not limited by  $T_{int}$ . This is interesting because we can define a lifetime restriction on the  $tkey$  based on this condition. From our observation that the ciphertext block space is larger than the plaintext and ciphertext block pair space, this result also implies that if  $n > \frac{(T_{int})^{\frac{1}{b}}}{(T_{block})^{\frac{1}{b}}}$ , then  $Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M]$  is not limited by  $T_{int}$ .

### 1.4.3 CHECKING THE VALIDITY OF MESSAGES

In this axiom, we describe the process by which  $X$  may find either the MOC-value for a given plaintext message, or a plaintext message that corresponds to a given MOC-value.

$$A_3: \frac{X \text{ has } (m, mkey, tkey); X \text{ has } (n, tkey); \\ Prob[X \text{ finds } tkey] = S; Prob[X \text{ finds } mkey] = T}{\frac{Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M] \leq \max(S \times T, ET_{moc})}{Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}] \leq \max(S \times T, ET_{moc}, (ET_{block})^b)}}$$

That is, if principal  $X$  has  $m$  pairs of plaintext messages  $M$  and their corresponding MOC values computed with key  $mkey$  and encrypted with  $tkey$ , and  $X$  has  $n$  plaintext-ciphertext block pairs, then the probability with which  $X$  finds  $tkey$  is  $S$ , and the probability with which  $X$  finds  $mkey$  is  $T$ , then  $X$  can find  $\{MOC(mkey, M)\}_{tkey}$  for  $M$  with probability less than or equal to the maximum of  $S \times T$ , and  $ET_{moc}$ .  $X$  can find  $M$  for  $\{MOC(mkey, M)\}_{tkey}$  with a probability which is not greater than the maximum of  $S \times T$ ,  $ET_{moc}$ , and  $(ET_{block})^b$ .

The reasoning behind this axiom is similar to that of  $A_2$ .  $X$  may be able to find the MOC-value corresponding to a given text in two ways, namely:

1. By finding both the  $mkey$  (in the case of a keyed MOC function) and the  $tkey$  (in the case of an encrypted MOC-value), The probability of this is  $S \times T$  if  $tkey$  and  $mkey$  are unrelated. In some cryptographic protocols, this may not be the case, as  $mkey$  and  $tkey$  may be identical (e.g., Kerberos V4), or the discovery of  $tkey$  may aid in finding  $mkey$ .
2. By computing a MOC-value for the message  $M$  using the knowledge of the  $m$  message and MOC-value pairs.  $X$  can find the MOC-value for the message by searching the *effective moc-block space*. This search succeeds with probability  $ET_{moc}$ .

$X$  may be able to find the plaintext for a given (possibly encrypted) MOC-value in three ways, namely

1. By finding both  $mkey$  and  $tkey$ , and using the knowledge of the moc-function to construct a message  $M$  which has the required MOC-value. The probability of this is at most  $S \times T$  (discussed above).
2. By making use of the many to one mapping from messages to MOC-values. Since any given message has a particular MOC-value with probability  $T_{moc}$ , the probability with which any chosen message  $M$  will have the MOC-value  $\{MOC(mkey, M)\}_{tkey}$ , given that  $X$  has  $m$  message and MOC-value pairs is at most equal to  $ET_{moc}$ , the *effective moc threshold* (in this axiom, non-existence of a confounder is assumed), using the same arguments as before.
3. Lastly,  $X$  can choose a plaintext message representation  $M$  by randomly choosing  $b$  blocks to match the message by searching the *effective block space*. Using the environment assumption we have that the probability of success in this type of search is limited by the *effective block threshold*.

The probability of finding  $\{MOC(mkey, M)\}_{tkey}$  for a given  $M$  (or vice-versa) is at most as large as the maximum of the probability of success in the corresponding search options. In this axiom, like in  $A_2$ , we assume that the MOC function does not add any *confounding text* to the text  $M$ .

This assumption will be relaxed in  $A_4$  and  $A_5$ .  $ET_{moc} > T_{int}$  or  $(ET_{block})^b > T_{int}$  then  $Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M]$  or  $Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}]$  are not limited by  $T_{int}$ . From the definitions of the effective thresholds, this implies that if  $\chi(m) > (1 - \frac{T_{moc}}{T_{int}})$  or  $\epsilon(n) > 1 - \frac{T_{block}}{(T_{int})^b}$ , then the  $Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M]$  or  $Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}]$  are not limited by  $T_{int}$ .

#### 1.4.4 CONFOUNDING THE DOMAIN OF ENCRYPTION FUNCTION

In practice, encryption and MOC functions can add *confounding text* to their domain in order to counter known plaintext attacks. In the previous axioms ( $A_2$  and  $A_3$ ), we have not taken into consideration the effect of *confounding text* on the probability of the success of attacker actions. In this axiom, we take this into account and relate the probabilities that ensue due to adding a confounding component to the domain of an encryption function. The effect is to multiply the search space as derived in the previous axioms ( $A_2$  and  $A_3$ ) by a constant factor, namely the search space of the *confounding text*.

$$A_4: \frac{Prob[X \text{ finds } \{M'\}_{tkey} \text{ for } M'] \leq Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M] \leq W}{\frac{Prob[X \text{ finds } M' \text{ for } M] \leq T_{conf}}{Prob[X \text{ finds } \{M'\}_{tkey} \text{ for } M] \leq W \times T_{conf}}}}$$

where  $W = \max(S, (ET_{pair})^b, (ET_{block})^b)$ , is derived using axioms A<sub>2</sub>.  $S$  is the probability with which  $tkey$  is compromised.  $ET_{pair}$  and  $ET_{block}$  are defined in section 1.2.1.

That is, if the probability with which  $X$  finds  $\{M'\}_{tkey}$  for  $M'$  is not greater than the probability with which  $X$  finds  $\{M\}_{tkey}$  for  $M$ , which in turn is not greater than  $W$ , and if  $M'$  is obtained from  $M$  by the application of some layer function (i.e., a confounding function) with a range space of  $\frac{1}{T_{conf}}$ , then the probability with which  $X$  finds  $\{M'\}_{tkey}$  for  $M$  is less than or equal to the product of  $W$  and  $T_{conf}$ .

The rationale behind this axiom is the following: We treat  $[X \text{ finds } \{M'\}_{tkey} \text{ for } M]$  as the simultaneous occurrence of two events, namely (1)  $[X \text{ finds } M' \text{ for } M]$ , and (2)  $[X \text{ finds } \{M'\}_{tkey} \text{ for } M']$ . If  $X$  finds both these matching pairs,  $X$  can find  $\{M'\}_{tkey}$  for  $M$ . The probability of this joint occurrence would then be equal to the product of their individual probabilities. The probability with which  $X$  finds  $M'$  for  $M$  is at most equal to  $T_{conf}$ . The joint probability of the events (1) and (2) above is at most  $W \times T_{conf}$ . The statement  $Prob[X \text{ finds } \{M'\}_{tkey} \text{ for } M] \leq Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M]$  holds for any message type in which the space of  $M'$  is not smaller than the space of  $M$ . Since this is always the case, we do not explicitly write this as an assumption in our analysis. The reason for including this in our axiom is to allow the reader to understand the implication.

### 1.4.5 CONFOUNDING THE DOMAIN OF MOC FUNCTION

In this axiom, we describe the effect of adding a confounding component to the domain of the MOC function.

$$\begin{array}{l}
 \text{A}_5: \quad \frac{\begin{array}{l} Prob[X \text{ finds } \{MOC(mkey, M')\}_{tkey} \text{ for } M'] \leq \\ Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M] \leq Z_1; \\ Prob[X \text{ finds } M' \text{ for } \{MOC(mkey, M')\}_{tkey}] \leq \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}] \leq Z_2; \end{array}}{Prob[X \text{ finds } M' \text{ for } M] = Prob[X \text{ finds } M \text{ for } M'] \leq T_{conf}} \\
 \frac{Prob[X \text{ finds } \{MOC(mkey, M')\}_{tkey} \text{ for } M] \leq \max(Z_1 \times T_{conf}, ET_{cmoc})}{Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M')\}_{tkey}] \leq \max(Z_2 \times T_{conf}, ET_{cmoc})}
 \end{array}$$

where  $Z_1 = \max(S \times T, ET_{moc})$ , and

$$Z_2 = \max(S \times T, ET_{moc}, (ET_{block})^b)$$

are derived using axiom A<sub>3</sub>.  $S$  and  $T$  are the probability with which  $X$  can find  $tkey$  and  $mkey$  respectively.  $T_{conf}$ ,  $ET_{moc}$  and  $ET_{cmoc}$  are defined in section 1.2.1.

That is, if

- the probability with which  $X$  finds  $\{MOC(mkey, M')\}_{tkey}$  for  $M'$  is not greater than the probability with which  $X$  finds  $\{MOC(mkey, M)\}_{tkey}$  for  $M$ , which in turn is not greater than  $Z_1$  ( $Z_1$  is derived using axiom A<sub>3</sub>),
- the probability with which  $X$  finds  $M'$  for  $\{MOC(mkey, M')\}_{tkey}$  is not greater than the probability with which  $X$  finds  $M$  for  $\{MOC(mkey, M)\}_{tkey}$ , which in turn is not greater than  $Z_2$  ( $Z_2$  is derived using axiom A<sub>3</sub>), and
- the probability with which  $X$  finds  $M'$  (which is the confounded version of  $M$ ) for  $M$  is at most  $T_{conf}$ ,

then the probability with which  $X$  finds  $\{MOC(mkey, M')\}_{tkey}$  for  $M$  is at most equal to the maximum of two quantities  $Z_1 \times T_{conf}$  and  $ET_{cmoc}$ . The reasoning about the term  $Z_1 \times T_{conf}$  is similar to that of A<sub>4</sub>. The second term, which is the *effective moc threshold* is not a linear function of  $T_{conf}$ . This term represents the “calculated” choice of the MOC-value for a given message from the knowledge of the  $m$  message and MOC-value pairs. The presence of a confounder with a large space (and hence, a small  $T_{conf}$ ) has the effect of bringing the *effective moc threshold* close to  $T_{moc}$ . The statements (1) and (2) above are true in any message type in which the space of  $M'$  is not less than the space of  $M$ . These statements are included in the axioms to enable the reader to understand the implication. The probability of finding  $M$  for a given  $\{MOC(mkey, M')\}_{tkey}$  is limited by the maximum of two quantities  $Z_2 \times T_{conf}$ , and  $ET_{cmoc}$ . The reasoning here is similar.

An important consequence of axioms A<sub>2</sub> through A<sub>5</sub> is that if  $X$  has  $(n, tkey)$  where  $n$  is not zero, and if  $T_{conf}$  is 1, then  $X$  can mount a verifiable plaintext attack, subject to  $X$ 's ability to find a suitable MOC-value. If  $n$  is equal to the plaintext-ciphertext pair space, and if  $T_{conf}$  is one, then  $X$  can mount a chosen plaintext attack, again, subject to the same moc-constraint. If, however,  $X$

has known plaintext-ciphertext pairs ( $n > 0$ ), and  $T_{conf}$  is less than one, then possession of known pairs does not allow  $X$  to mount known or chosen plaintext attacks. Attacks in this case would be probabilistic in nature. The same reasoning holds for text and MOC value pairs (i.e., when  $X$  has  $(m, mkey, tkey)$ ). Attacks using the vulnerability of the moc function would be successful, subject to the ability of  $X$  to compromise the encryption scheme, if one is used.

## 1.5 Goals of Analysis

Message integrity is said to be established if the probability with which  $X$  (who is assumed not to have the encryption key) can find a suitable ciphertext representation for a given message, is less than the assumed integrity threshold. This is stated below as goal [G<sub>1</sub>]. In addition, message types which make use of checksums for detection of modification of messages may want to ensure that  $X$  can neither compute a checksum for a given message [G<sub>2</sub>], nor find a message corresponding to a given checksum [G<sub>3</sub>] with a probability that is greater than the threshold.

$$\begin{aligned} Prob[X \text{ finds } \{M'\}_{tkey} \text{ for } M] &\leq T_{int} \dots\dots\dots[G_1] \\ Prob[X \text{ finds } \{MOC(mkey, M')\}_{tkey} \text{ for } M] &\leq T_{int} \dots\dots\dots[G_2] \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M')\}_{tkey}] &\leq T_{int} \dots\dots\dots[G_3] \end{aligned}$$

**Definition:** A cryptographic protocol is said to satisfy message integrity conditions if it satisfies the condition  $[G_1 \vee (G_2 \wedge G_3)]$ .

The condition G<sub>1</sub> implies that  $X$  can not find the ciphertext  $\{M'\}_{tkey}$  corresponding to plaintext  $M$  with a probability greater than  $T_{int}$ . G<sub>2</sub> implies that no attacker  $X$  can find a  $\{MOC(mkey, M')\}_{tkey}$  for some  $M$  with a probability greater than  $T_{int}$ , and G<sub>3</sub> implies that  $X$  cannot find a  $M'$  for  $\{MOC(mkey, M')\}_{tkey}$  with a probability greater than  $T_{int}$ . Here,  $M'$  denotes the message string that is obtained when the layer function for encryption and MOC-value computation adds some unknown component to  $M$ . This *confounding text* is assumed to be chosen at random from a uniform space of size  $T_{conf}^{-1}$ . If there is no such unknown component in  $M'$ , then  $M' = M$ .

Mechanisms which use only the properties of encryption algorithm to enforce message integrity protection without having any redundancy (using checksums) may aim to achieve only goal G<sub>1</sub>. Goal G<sub>1</sub> may be relevant only to cryptographic protocols which use encryption for preserving secrecy. In cryptographic protocols which do not use encryption for secrecy (i.e., use non-confidential messages), G<sub>1</sub> would reduce to  $Prob[X \text{ finds } M' \text{ for } M] \leq T_{int}$ . This is precisely the definition of *confounding text*. Hence, cryptographic protocols which do not use encryption would need to use confounders in the domain to achieve goal G<sub>1</sub>.

### 1.5.1 LIFETIME CONSTRAINTS ON THE ENCRYPTION KEY

The goals of message integrity analysis translate to lifetime constraints on the encryption key, since the ease with which an attacker can find the *tkey*, or construct the ciphertext representation by having known plaintext and ciphertext block pairs increases with the lifetime of the key for a fixed encryption rate and computational resources available to the attacker. In this section, we briefly discuss the interpretation of the goals of integrity analysis in terms of lifetime constraints of the *tkey*.

- In axioms A<sub>2</sub> and A<sub>4</sub>, using  $(ET_{pair})^b < T_{int}$  and our assumption that  $\frac{1}{|(P', \{P'\}_{tkey})|} \leq T_{block}$ , we have that the total number of message blocks that are encrypted using *tkey* during its lifetime must be less than  $\frac{(T_{int})^{\frac{1}{b}}}{(T_{block} \times T_{conf})}$  if the goal G<sub>1</sub> should be achieved. If the maximum number of ciphertext blocks that are encrypted in any given time is  $R$ , then

$$lifetime(tkey) < \frac{1}{R} \times \frac{(T_{int})^{\frac{1}{b}}}{(T_{block} \times T_{conf})}$$

where  $lifetime(tkey)$  should be read as “lifetime of *tkey*.”

- From A<sub>2</sub> and A<sub>4</sub>, using  $(ET_{block})^b < T_{int}$ , we derive a lifetime restriction (i.e., limit on  $n$ ) based on the limiting value of  $\epsilon(n)$ . That is,

$$lifetime(tkey) < \frac{1}{R} \times \epsilon^{-1} \left( 1 - \frac{T_{block}}{(T_{int})^b} \right)$$

A precise calculation of this limit would require the analysis of the encryption function to find the function  $\epsilon(n)$ , and is beyond the scope of our work.

- If the maximum rate at which an attacker can exhaustively try  $tkey$  (possibly off-line) to verify a certain computed ciphertext block with known plaintext and ciphertext block pairs is  $C$ , then

$$lifetime(tkey) < \frac{1}{C} \times \left( \frac{T_{int}}{T_{tkey}} \right)$$

The lifetime of  $tkey$  must not exceed the minimum of the above three quantities in order to ensure that the message integrity goals are established. That is,

$$lifetime(tkey) < \min\left(\frac{1}{R} \times \frac{(T_{int})^{\frac{1}{b}}}{(T_{block} \times T_{conf})}, \frac{1}{R} \times \epsilon^{-1} \left(1 - \frac{T_{block}}{(T_{int})^b}\right), \frac{1}{C} \times \left( \frac{T_{int}}{T_{tkey}} \right)\right)$$

In this analysis, we do not explicitly consider the lifetime limits, but analyze the constraints on  $n$ , the number of available plaintext-ciphertext block pairs.

### 1.5.2 LIFETIME CONSTRAINTS ON MOC-FUNCTION KEY

The goals of message integrity analysis can also be translated to moc-key lifetime constraints, similar to those of encryption keys. In what follows, we interpret the goals in terms of necessary conditions on the lifetime of moc-keys.

- From  $A_3$  and  $A_5$ , using  $ET_{moc} < T_{int}$ , we have that the value of  $m$  must be limited such that

$$\chi(m) \leq 1 - \left( \frac{T_{moc} \times T_{conf}}{T_{int}} \right)$$

This gives us a lifetime restriction on  $mkey$ . If  $K$  is the maximum number of valid message representations that can be constructed per given time,

$$lifetime(mkey) < \frac{1}{K} \times \chi^{-1} \left(1 - \frac{T_{moc} \times T_{conf}}{T_{int}}\right)$$

A precise calculation of this limit would require the analysis of the MOC-function to find the function  $\chi(m)$ , and is beyond the scope of our work.

- If the maximum rate at which an attacker can exhaustively try  $mkey$  (possibly off-line) is  $D$ , then

$$lifetime(mkey) < \frac{1}{D} \times \frac{T_{int}}{T_{tkey}}$$

The lifetime of  $mkey$  must not exceed the minimum of the above two quantities in order to ensure that the message integrity goals are established.

## 1.6 Examples of Message Integrity Analysis

Message integrity protection weaknesses can be subtle, and can escape the scrutiny of designers if there are no guidelines on relating the design parameters to the message integrity goals of the protection mechanisms. In this section, we provide examples to show that using our approach can aid in explaining these weaknesses and also suggest modifications in the design parameters. Our approach to the analysis of message integrity protection properties involves the following steps.

1. Explicitly state the premises and design parameters of the message integrity protection mechanisms using semantics of the model of computation. The results derived are based on these premises. In particular, our approach cannot determine the soundness of these premises.
2. Apply the axioms on the premises and design parameters to derive expressions of the form  $Prob[\text{message integrity compromise}] \leq P$ .
3. Compare the probability limit ( $P$ ) derived using the axioms with the policy threshold for message integrity. Depending on the choice of the integrity threshold, the analysis may (not) indicate a vulnerability. In our analysis, unless otherwise stated, we assume an integrity threshold of  $\frac{1}{2^{64}}$ . In other words, we assume that the level of message integrity protection is adequate if the probability of a successful integrity attack is at most equal to  $\frac{1}{2^{64}}$ .
4. Analyze the implications of the policy goals on the lifetimes of the keys. If these constraints are strong, then the protection mechanism is vulnerable due to the availability of known domain-range pairs of encryption or moc-functions.

5. Suggest modifications in protection mechanism design parameters to remedy existing vulnerabilities.

We use three protocol examples to illustrate the use of the axioms presented in this paper. The examples are chosen to illustrate different types of weaknesses that can be detected using this method. Additional analysis examples are given in [8].

### 1.6.1 ANALYSIS OF KERBEROS V4 PROTOCOL [2]

In this section, we analyze the Kerberos V4 protocol which uses DES (Data Encryption Standard [6]) for encryption. The Kerberos V4 was previously shown to be vulnerable in [2]. We show that this protocol does not achieve its goals, namely, that of keeping the probability of a successful message integrity compromise below the specified threshold. The Kerberos V4 protocol has the property whereby the same key is used both for the encryption function as well as for the moc function. This key is 7 bytes long, and hence, has a space of size  $2^{56}$ . Hence, the key threshold is equal to  $2^{-56}$ . The ciphertext blocks in this protocol are 8 bytes long. The *moc* value occupies a whole block (i.e., 8 bytes). The protocol does not use confounding text. Hence, the confounder threshold is 1.<sup>6</sup>

#### Protocol Premises and Design Parameters

$$\begin{aligned} &\mathbf{Secret}(tkey, Run) \\ &\mathbf{Secret}(mkey, Run) \\ &mkey = tkey = key \\ &T_{key} = \frac{1}{2^{56}} \\ &T_{block} = T_{moc} = T_{int} = \frac{1}{2^{64}} \\ &T_{conf} = 1 \end{aligned}$$

The first two premises are about the secrecy of the encryption key (*tkey*) and the moc-key (*mkey*) in the session (Run). These premises are based on the fact that this protocol uses keyed encryption and moc-functions, the keys of which are secret within a session (Run). The remaining assumptions are about the design parameters of the protocol. The *mkey* and the *tkey* are the same, and their threshold is  $\frac{1}{2^{56}}$ . The ciphertext block, MOC-value, and the assumed policy integrity threshold are all  $2^{-64}$ . The protocol uses no confounders (i.e., confounder space is 1).

#### Analysis

From the assumptions about *secrecy*, and about *tkey* and *mkey* being identical (both equal to *key*), using axiom A<sub>1</sub>, we have

$$\begin{aligned} Prob[X \text{ finds } tkey] &= ET_{key}. \\ Prob[X \text{ finds } mkey] &= ET_{key}. \end{aligned}$$

Application of axioms A<sub>2</sub> and A<sub>3</sub> yield

$$\begin{aligned} Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M] &\leq \max(ET_{tkey}, (ET_{pair})^b, (ET_{block})^b) = W \\ Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M] &\leq \max(ET_{key}, ET_{moc}) = Z_1 \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}] &\leq \max(ET_{key}, ET_{moc}, (ET_{block})^b) = Z_2 \end{aligned}$$

Note that since *tkey* = *mkey* = *key*, in the application of A<sub>3</sub>, the joint probability of finding both *tkey* and *mkey* is equal to the probability of finding one of them (i.e., *key*).

We now apply axioms A<sub>4</sub> and A<sub>5</sub>, with the assumption that  $Prob[X \text{ finds } M' \text{ for } M] \leq T_{conf} = 1$ . Using this, we obtain,

$$\begin{aligned} Prob[X \text{ finds } \{M'\}_{key} \text{ for } M] &\leq W \dots [R_1] \\ Prob[X \text{ finds } \{MOC(key, M')\}_{key} \text{ for } M] &\leq \max(Z_1, ET_{moc}) \dots [R_2] \\ Prob[X \text{ finds } M' \text{ for } \{MOC(key, M')\}_{key}] &\leq \max(Z_2, ET_{moc}) \dots [R_3] \end{aligned}$$

#### Observations

In result R<sub>1</sub>, the term *W* is at least as large as  $T_{key}$  (because  $W = \max(ET_{tkey}, (ET_{pair})^b, (ET_{block})^b)$ ). From the design parameters,  $T_{key} > T_{int}$ . Hence,  $W > T_{int}$ . Using this in

---

<sup>6</sup>We view this message type as being equivalent to a message type which uses a known constant as a confounder. The use of a known constant as a confounder does not serve any purpose. This view allows us to analyze both these message types with the same notation.

result  $R_1$ , we can see that this protocol does not achieve goal  $G_1$ . Consider goals  $G_2$  and  $G_3$ . In results  $R_1$  through  $R_3$ , terms  $Z_1$  and  $Z_2$  are at least as large as  $T_{key}$ , which in turn is greater than  $T_{int}$ . Hence, the protocol does not achieve  $G_2$  and  $G_3$ .

We now reason about these results. On first glance, it appears that the goal  $G_1$  is not achieved due to the relatively small size of the key space. We will now show that this intuition is not entirely correct. Consider a variant of this protocols' design parameters; i.e., the same protocol with  $T_{key} \leq \frac{1}{288}$ . Hence,  $T_{key} \leq T_{int}$ . However, the term  $W = \max(ET_{key}, (ET_{pair})^b, (ET_{block})^b)$  can be less than or equal to  $T_{int}$  only if

$$(ET_{pair})^b \leq T_{int}$$

From the definition of  $ET_{pair}$ ,

$$\left(\frac{(n+1)}{(|P', \{P'\}_{key}|)}\right)^b \leq T_{int} \dots\dots\dots (1)$$

Using our assumption that the number of plaintext-ciphertext pairs that can be generated is not larger than the number of all possible ciphertext blocks, we can derive the condition  $((n+1) \times T_{block})^b \leq T_{int}$  from (1). This means that

$$n < \frac{(T_{int})^{\frac{1}{b}}}{T_{block}} \leq 1$$

The limit on  $n$  derived above suggests that even if  $T_{key}$  is decreased from  $2^{-56}$  to  $2^{-64}$ , all other design parameters being the same, this protocol can still not achieve the analysis goals for non-zero values of  $n$ . Choosing a large key size for this protocol does not turn out to be useful in achieving the goal  $G_1$ , since the ciphertext block size is still equal to the reciprocal of the assumed integrity threshold, making the search of the ciphertext block space easier than searching the key space when  $n > 0$ . This is the case because this protocol does not use confounding text, and hence, makes known plaintext-ciphertext block pairs available (hence, reducing the *effective* search space for unknown ciphertext blocks).

Again, consider the goals  $G_2$  and  $G_3$ . On first glance, the reason for this protocol not being able to achieve goals  $G_2$  and  $G_3$  seems to be because the *mkey* is the same as the *tkey*. However, this intuition is also not entirely true. Consider the variant of this protocol which uses a *mkey* which is different from *tkey*. The terms  $Z_1$  and  $Z_2$  would then be

$$Z_1 = \max(ET_{tkey} \times ET_{mkey}, ET_{moc}), \text{ and}$$

$$Z_2 = \max(ET_{tkey} \times ET_{mkey}, ET_{moc}, (ET_{block})^b)$$

Since this protection mechanism has  $T_{moc} = T_{int}$ , the *effective moc threshold* (i.e.,  $ET_{moc}$ ) is larger than  $T_{int}$  for non-zero values of  $m$ . Hence, both  $Z_1$  and  $Z_2$  are greater than  $T_{int}$  for non-zero values of  $m$ . Hence, (from results  $R_2$  and  $R_3$ ) this protocol will not achieve the goals ( $G_2 \wedge G_3$ ) if  $m > 0$ , in spite of having unique *mkey* and *tkeys*.

#### *Suggested Solution for Remedying Vulnerability*

The vulnerability of this protection mechanism can be remedied by choosing a larger block size (i.e.,  $T_{block} < T_{int}$ ), in addition to a larger encryption key size (i.e.,  $T_{key} < T_{int}$ ) or adopting one of the following solutions:

- Choosing a larger moc block space. By doing this, the *effective moc threshold* is reduced to less than  $T_{int}$ .
- Using a confounder which has a large space, such that  $\frac{T_{moc}}{1-T_{conf}}$  is at most equal to the assumed message integrity threshold.
- Choosing a moc function and the lifetime constraints on the *mkey* such that the value of  $\epsilon(m)$  is much smaller than 1 for the maximum number that  $m$  can attain during the lifetime of the *mkey*. By this choice, the term  $\frac{T_{moc}}{1-\epsilon(m)}$  in  $Z_1$  and  $Z_2$  can be kept close to  $T_{moc}$ .

The rationale for these suggestions is that by increasing the size of both the encryption key space (such that  $T_{key} > T_{int}$ ) as well as the ciphertext block space (such that the term  $\frac{(T_{int})^{\frac{1}{b}}}{T_{block}}$  is larger than the number of plaintext-ciphertext block pairs that will be generated during the lifetime of the key), the goal  $G_1$  can be achieved. To achieve  $G_1$  or  $G_2 \wedge G_3$ , the design parameters can also be chosen such that the goals  $G_2$  and  $G_3$  hold. This can be done using the three alternative solutions presented above.

This example shows that vulnerabilities of message integrity protection mechanisms can be subtle and that adhoc and intuitive reasoning about them can be erroneous. Relating protection mechanism design parameters to the integrity goals and the environment threats allowed us to reason about the weaknesses of this protocol, and to discover some weaknesses that were not obvious on intuitive reasoning. In the following section, we provide an example in which we consider a protocol which uses two message types, and consider the effects of having a moc-function key that can be common to both types of messages.

### 1.6.2 ANALYSIS OF PRIVACY ENHANCED EMAIL

The Privacy Enhancement for Internet Electronic Mail (PEM) [11,12,13] supports confidentiality, integrity, and authentication for electronic mail transfer in the Internet. An earlier version of PEM [10] was shown to be vulnerable to integrity attacks against messages using the DES-MAC integrity check when these messages are sent to more than one receiver [16]. The PEM services support both single- and multiple-receiver messages.

The moc-key (*mkey*) of the single recipient session is known to members of the multiple receiver session, since the same *mkey* can be used in both types of sessions. In the analysis that follows, we consider the effect of having the *mkey* of the single receiver session known to the members of the multiple receiver session. In this protocol analysis, a principal who belongs to the multiple receiver session (and hence, knows the *mkey* of the multiple receiver session as well as the single receiver session) and uses this knowledge of the *mkey* to construct a single receiver session message between two other parties which belong to the multiple receiver session, is viewed as the attacker. It should be noted that the secrecy assumptions in the analysis are made from this viewpoint, and hence, may not apply in general, to any attacker.

The PEM protocol uses ciphertext blocks which are 8 bytes long, and hence, have a space of size  $2^{64}$ . The moc block size is equal to the ciphertext block size. The protocol does not use confounding text. Hence, the confounder threshold is 1. The encryption key is 7 bytes long. Hence, its space is  $2^{56}$  in size.

#### Protocol Premises and Design Parameters

$$\begin{aligned} & \mathbf{Secret}(tkey, Run) \\ & \neg \mathbf{Secret}(mkey, Run)^7 \\ T_{block} &= T_{moc} = T_{int} = \frac{1}{2^{64}} \\ T_{tkey} &= \frac{1}{2^{56}} \\ T_{conf} &= 1 \\ & (\text{i.e., } T_{block} = T_{moc} = T_{int} < T_{tkey} < T_{conf} = 1) \end{aligned}$$

The protocol uses encrypted messages, applying a keyed encryption function on plaintext blocks. The encryption key is assumed to be secret in the protocol run. For the reasons discussed at the start of this section, the *mkey* of the single receiver session is not assumed to be secret within the multiple user session. Although the protection mechanism uses a *mkey* which has space =  $2^{56}$ , the assumption about the lack of secrecy of this key makes  $T_{mkey} = 1$ . The remaining premises listed above are about design parameters.

#### Analysis

From the assumptions about secrecy, using  $A_1$ , we have that

$$\begin{aligned} Prob[X \text{ finds } tkey] &= ET_{tkey} \\ Prob[X \text{ finds } mkey] &= 1 \end{aligned}$$

Application of the axioms  $A_2$ ,  $A_3$  and the results derived above yields

$$\begin{aligned} Prob[X \text{ finds } \{M\}_{tkey} \text{ for } M] &\leq \max(ET_{tkey}, (ET_{pair})^b, (ET_{block})^b) = W \\ Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M] &\leq \max(ET_{tkey}, ET_{moc}) = Z_1 \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}] &\leq \max(ET_{tkey}, ET_{moc}, (ET_{block})^b) = Z_2 \end{aligned}$$

We now apply axioms  $A_4$  and  $A_5$ , with the assumption that  $Prob[X \text{ finds } M' \text{ for } M] \leq T_{conf} = 1$ , and obtain,

$$\begin{aligned} Prob[X \text{ finds } \{M'\}_{tkey} \text{ for } M] &\leq W \dots\dots\dots [R_1] \\ Prob[X \text{ finds } \{MOC(mkey, M')\}_{tkey} \text{ for } M] &\leq \max(Z_1, ET_{moc}) \dots\dots\dots [R_2] \\ Prob[X \text{ finds } M' \text{ for } \{MOC(mkey, M')\}_{tkey}] &\leq \max(Z_2, ET_{moc}) \dots\dots\dots [R_3] \end{aligned}$$

#### Observations

The term  $W = \max(ET_{tkey}, (ET_{pair})^b, (ET_{block})^b)$  on the right hand side of result  $[R_1]$  is larger

---

<sup>7</sup>The *mkey* of the single user session is not a secret to the members of a multiple session which contains the members of the single user session. For example, if parties  $A, B$ , and  $C$  belong to a multiple receiver session, then the *mkey* of this session can be used by  $A$  to send (single receiver session) messages to  $B$  or to  $C$ .

than  $T_{int}$ , since  $ET_{tkey} > T_{int}$ . Hence, goal  $G_1$  does not hold. Similarly,  $Z_1 = \max(ET_{tkey} \times T_{mkey}, ET_{moc}) > T_{int}$  and  $Z_2 = \max(ET_{tkey} \times ET_{mkey}, ET_{moc}, (ET_{block})^b) > T_{int}$  because  $ET_{tkey} > T_{int}$  and  $ET_{mkey} = 1$ . Hence,  $G_2 \wedge G_3$  does not hold.

Since neither  $G_1$  nor  $(G_2 \wedge G_3)$  hold, this protocol does not achieve the goals of analysis. Although the analysis seems to indicate that the protocol is unable to achieve goal  $G_1$  due to the relatively small space of the *tkey*, we observe that increasing the size of the encryption key space alone would not suffice for this protocol to achieve goal  $G_1$ . This is because the ciphertext block space in this protocol is equal to the reciprocal of the assumed integrity threshold, and hence, the *effective block space* would be smaller than the reciprocal of the integrity threshold for non-zero values of  $n$ . Hence, goal  $G_1$  can be achieved if the key space as well as the ciphertext block space are increased such that  $ET_{tkey} < T_{int}$ , and the *effective block threshold* is smaller than  $T_{int}$  for the largest value of  $n$  possible during the lifetime of the encryption key. Another alternative to remedy the protocol weakness is to use a confounder which has a sufficiently large space such that the *effective moc threshold* is close enough to  $T_{int}$ .

This protocol falls short of goals  $G_2$  and  $G_3$  because of three reasons: (1) because it uses the same *mkey* for the single as well as the multiple recipient sessions (i.e., the key for a single recipient session is not a secret) (2) the moc-block space is equal to the reciprocal of the assumed integrity threshold, and hence, the *effective moc threshold* is larger than the assumed integrity threshold for non-zero values of  $m$ , the number of available plaintext message and MOC-value pairs. (3) the ciphertext block size is equal to the reciprocal of the message integrity threshold, hence making the *effective block threshold* larger than the integrity threshold, for non-zero values of  $n$ , the number of available plaintext-ciphertext block pairs available. The goals  $G_2$  and  $G_3$  can be achieved if the three conditions above which cause the protocol weakness, are negated.

The solution to this problem suggested in [20] is to use two moc blocks, so that the resulting moc-space is the product of the individual spaces. This solution does not establish goal  $G_1$ , but achieves goals  $G_2$  and  $G_3$ , hence satisfying the condition  $G_1 \vee (G_2 \wedge G_3)$ . Below, we analyze the proposed solution of [20] and show that it satisfies the message integrity goals. In [8], we present the analysis of this proposed solution and show that it does not exhibit the weakness of the original protocol.

### 1.6.3 ANALYSIS OF MESSAGE DIGEST OF KRB\_SAFE IN KERBEROS V5

In this protocol example, we show that choosing the ciphertext block space equal to or smaller than the reciprocal of the integrity threshold may reduce the effectiveness of choosing a large moc-block space to enforce the assumed message integrity threshold. The KRB\_SAFE protocol in Kerberos V5 uses a checksum which is the DES-CBC encryption of the 128-bit RSA-MD4 digest. The ciphertext block size is 8 bytes. The encryption key and the *mkey* are 7 bytes long. The protocol does not use confounders. By using a MOC-value which has a space  $2^{128}$ , the protocol aims to keep the message integrity threshold anywhere between  $\frac{1}{2^{128}}$  and  $\frac{1}{2^{64}}$ . In the analysis that follows, we show that this goal is not actually achieved by the protocol, owing to the fact that it uses a ciphertext block which has a space much smaller than the space of the MOC-value.

#### *Protocol Premises and Design Parameters*

**Secret**(*tkey*, *Run*)

**Secret**(*mkey*, *Run*)

$$T_{moc} = \frac{1}{2^{128}}; T_{block} = \frac{1}{2^{64}}$$

$$T_{tkey} = T_{mkey} = \frac{1}{2^{56}}; T_{conf} = 1$$

$$T_{int} \in \left(\frac{1}{2^{128}}, \frac{1}{2^{64}}\right)$$

$$(i.e., T_{moc} \leq T_{int} < T_{block} < T_{tkey} = T_{mkey} < T_{conf} = 1)$$

This protocol does not use encrypted messages. However, in this example, we are concerned with the DES-CBC encryption of the RSA-MD4 checksum, for which the protocol uses a *tkey* which is assumed to be secret. The RSA-MD4 is a keyed moc-function, with the *mkey* being secret. The digest space is  $2^{128}$ , since it is 16 byte long. However, the digest is encrypted (using cipher-block-chaining) into two blocks of size 8 bytes each.

#### *Analysis*

From the assumption that the *tkey* and *mkey* are secret, using  $A_1$ , we have,

$$Prob[X \text{ finds } tkey] = ET_{tkey}$$

$$Prob[X \text{ finds } mkey] = ET_{mkey}$$

Since this protocol does not use encryption for secrecy, goal  $G_1$  is not relevant. We focus on finding the encrypted MOC-value for a given message (or vice-versa). Application of the axiom  $A_3$  yields

$$\begin{aligned} Prob[X \text{ finds } \{MOC(mkey, M)\}_{tkey} \text{ for } M] &\leq \max(ET_{tkey} \times ET_{mkey}, ET_{moc}) = Z_1 \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M)\}_{tkey}] &\leq \max(ET_{tkey} \times ET_{mkey}, ET_{moc}, (ET_{block})^b) = Z_2 \end{aligned}$$

We now apply axiom  $A_5$ , with the assumption that  $Prob[X \text{ finds } M' \text{ for } M] \leq T_{conf} = 1$ , and obtain

$$\begin{aligned} Prob[X \text{ finds } \{MOC(mkey, M')\}_{tkey} \text{ for } M] &\leq \max(Z_1, ET_{moc}) \dots\dots\dots[R_2] \\ Prob[X \text{ finds } M \text{ for } \{MOC(mkey, M')\}_{tkey} M] &\leq \max(Z_2, ET_{moc}, (ET_{block})^b) \dots[R_3] \end{aligned}$$

*Observations*

For this protocol to achieve the goal  $G_3$ , it is necessary that  $ET_{block} \leq T_{int}$  for a one block message. That is,

$$\left(\frac{T_{block}}{1-\epsilon(n)}\right) \leq T_{int}$$

since the term  $Z_2$  is at least as large as  $\frac{T_{block}}{1-\epsilon(n)}$ . But  $\frac{T_{block}}{1-\epsilon(n)} > T_{block} > T_{int}$  for values of  $n > 0$  in this protocol. The ciphertext space in this protocol is much smaller than the MOC block space. Hence, the protocol can achieve only a weaker threshold, namely  $\frac{1}{2^{64}}$ , although it uses a 128 bit moc block. The protocol fails to achieve the desired result, namely that of keeping the integrity threshold in the range  $(\frac{1}{2^{128}}, \frac{1}{2^{64}})$ .

The protocol is rendered weak because it uses DES-CBC to encrypt the 128 bit RSA-MD4 digest. The use of a 64 bit cipher-block makes the search space for the encrypted checksum smaller than  $2^{128}$ . This is because the protocol does not use confounders, and hence, makes available known plaintext-ciphertext block pairs. Having a known plaintext-ciphertext pair which corresponds to one of the encrypted digest blocks allows the attacker to exhaustively search for the other encrypted digest block, from a space of size  $2^{64}$ .

One possible solution to remedy this weakness is to use a confounding block of space  $2^{64}$  which makes known plaintext-ciphertext block pairs available with a probability  $\frac{1}{2^{64}}$ , and hence, allows this protocol to achieve its message integrity goals. Another alternative solution (although probably less practical) is to use 16 byte long ciphertext blocks and 16 byte long encryption keys to encrypt the digest, instead of the 8 byte blocks and keys, respectively.

## 1.7 Conclusions

Our model of computation allows reasoning about integrity design parameters of cryptographic protocols. The reasoning provides a set of axioms that relate protocol design parameters and protocol premises to the goals of message integrity protocols, considering the set of threats that the system is exposed to (as assumed in our model). As shown in these examples, using simple reasoning, we have accounted for the vulnerabilities in protocols which have been previously discovered to be flawed, and have suggested modifications in design parameters to strengthen these protocols.

*Acknowledgements:* The first two authors are grateful to Tom Tamburo and Marty Simmons of IBM Federal Systems Company for their continued support and encouragement.

## 1.8 REFERENCES

- [1] M.Burrows, M.Abadi and R.Needham, "A Logic of Authentication," *ACM Transactions on Computer Systems*, Vol.8, No.1, February 1990.
- [2] S.M.Bellovin and M.Merritt, "Limitations of the Kerberos Authentication System," *Computer Communications Review*, Vol.20, No.5, October 1990.
- [3] G.I.Davida, "Chosen Signature Cryptanalysis of the RSA (MIT) Public-Key Cryptosystem," Tech. Report TR-82-2, Dept. of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, WI, October 1982.
- [4] D.E.Denning, "Digital Signatures with RSA and Other Public-Key Cryptosystems," *Communications of the ACM*, Vol. 27, No. 4, April 1984.

- [5] D.Dolev and A.C.Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, Vol. IT-29, No. 2, March 1983.
- [6] Federal Information Processing Standard, "DES Modes of Operation," National Bureau of Standards, December, 1980.
- [7] L.Gong, R.Needham and R.Yahalom, "Reasoning about Beliefs in Cryptographic Protocols," *Proceedings of the IEEE Symposium on Research and Privacy*, May 1990.
- [8] R.Kailar, V.D.Gligor, S.G.Stubblebine, "Reasoning about Message Integrity," Technical Report (93-065), Electrical Engineering Department, University of Maryland, College Park, MD 20742.
- [9] J.Kohl and B.C.Neuman, "Kerberos Version 5, RFC, Draft #4", Project Athena, MIT, December 20, 1990.
- [10] J.Linn, "Privacy Enhancement for Internet Electronic Mail: Part I - Message Encipherment and Authentication Procedures," Internet Working Group, RFC-989, February, 1987.
- [11] J.Linn, "Privacy Enhancement for Internet Electronic Mail: Part I - Message Encipherment and Authentication Procedures," Internet Working Group, RFC-1113, August 1989.
- [12] J.Linn, "Privacy Enhancement for Internet Electronic Mail: Part II - Certificate-Based Key Management," Internet Working Group, RFC-1114, August 1989.
- [13] J.Linn, "Privacy Enhancement for Internet Electronic Mail: Part III - Algorithms, Modes, and Identifiers," Internet Working Group, RFC-1115, August 1989.
- [14] J.M.Moore, "Protocol Failures in Cryptosystems," *Proceedings of the IEEE*, Vol. 76, No. 5, May 1988.
- [15] R.C.Merkle, "One Way Hash Functions and DES," in *Advances of Cryptology, Proceedings of Crypto '89* Santa Barbara, California, 1989.
- [16] C.Mitchell and M.Walker, "Solutions to the Multidestination Secure Electronic Mail Problem," *Computers and Security*, Vol. 7, No. 5, 1988.
- [17] R.L.Rivest, A.Shamir, and L.Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, February 1978.
- [18] S.G.Stubblebine and V.D.Gligor, "On Message Integrity in Cryptographic Protocols," TR.No:2843, University of Maryland, College Park, Maryland 20742.
- [19] S.G.Stubblebine and V.D.Gligor, "On Message Integrity in Cryptographic Protocols," *Proceedings of the IEEE Symposium on Research in Privacy*, May 1992.
- [20] 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, CA, February 1993.
- [21] S.G.Stubblebine and V.D.Gligor, "Protocol Design for Integrity Protection," *Proceedings of the IEEE Symposium on Research in Privacy*, May 1993.

## Appendix A

# Reasoning about Message Integrity of Public-Key Cryptosystems

In this section, we provide some guidelines on extending our approach to the analysis of public-key cryptosystems. The set of axioms presented here is not necessarily complete. Other axioms may be necessary to model threats to the integrity of digital signatures.

In public-key cryptosystems (i.e., those using the RSA scheme [17] or variants thereof), the goals of message integrity protection are somewhat different from those of shared-key cryptosystems. Finding the ciphertext  $\{M\}_{key}$  for a given plaintext  $M$  is not an issue here, since the *key* is publicly known (by definition). Hence, the goals are simply  $G_2$  and  $G_3$ . Axiom  $A_2$  is relevant only for computing the probability of finding the encrypted signatures. The *membership*, *order*, and *cardinality* properties are mapped onto digital signatures using a secret key (i.e., the secret counterpart of the public-key). Hence, finding the signature for a plaintext message is equivalent to finding the MOC-value for a plaintext message, and vice-versa.

If the plaintext message  $M = P_1, P_2, \dots, P_b$ , where each  $P_i$  is the integer value corresponding to a block, then the digital signature using the RSA scheme is

$$D(M) = P_1^d \pmod{pq} \parallel P_2^d \pmod{pq} \parallel \dots \parallel P_b^d \pmod{pq},$$

where  $d$  is the secret key of the party which signs the message, and  $p$  and  $q$  are large prime numbers. The symbol  $\parallel$  denotes signature block concatenation.

In [4], it was shown that if  $P_i$  can be factorized, and the signature blocks of the factors can be obtained, then the signature block corresponding to the  $i^{th}$  block can be computed. Possession of  $b$  such computed blocks can enable an attacker to compute the signature. In [14], it was shown that possession of signatures to  $b$  arbitrary message blocks (where each plaintext block is some constant times  $P_i$ ) enables an attacker to compute the plaintext message for a chosen signature.

With this introduction, we proceed to present a modified set of axioms for public-key cryptosystems. The axiom  $A_1$  is still the same, and applies primarily to the secret counterpart of the public-key. As observed before, axiom  $A_2$  is not relevant since obtaining ciphertext for plaintext is not a problem in public-key cryptosystems. Axiom  $A_3$  for a  $b$  block message with the signature computed as above becomes:

$$A3' \quad \frac{X \text{ has } (n, d) ; Prob[X \text{ finds } d] \leq T_d}{Prob[X \text{ finds } D(M) \text{ for } M] \leq \max(T_d, \prod_{i=1}^b \frac{t_b}{|P_i, D(P_i)|}, (\frac{T_{block}}{1-\epsilon(n)})^b)}$$

where  $b$  is the number of blocks in the message and in the signature, and  $t_b$  is the number of factors for each block which maximizes the probability of finding the signatures of all the factors, and  $|P_i, D(P_i)|$  is the number of all possible message blocks and their corresponding signatures.

The reasoning here is as follows: the signature can be computed by (1) finding the key, or (2) by computing each of the signature blocks, or (3) by making a calculated guess at the signature representation using the knowledge of message blocks and their corresponding signatures. The first search has a probability  $T$  of success. Each block in the signature can be factored into  $t_b$  numbers, where the factors are chosen such that the probability of finding their corresponding signatures is maximum. Hence, the probability of finding the signature for a block by computing it from known message block- signature pairs is  $\frac{t_b}{|P_i, D(P_i)|}$ . The probability of finding the entire signature in this manner is at most equal to the product of the probability of finding the signature blocks of individual message blocks. Finally, a calculated guess at the signature may be done with a probability of success defined by  $\frac{T_{block}}{1-\epsilon(n)}$ , in a manner similar to guessing a ciphertext block for a plaintext block. The function  $\epsilon(n)$  is monotonically non-decreasing, and is non-zero for values of  $n$  greater than zero.

Axioms  $A_4$  and  $A_5$  are identical to the ones presented here, except that the term *confounding text* now refers to the effect of hashing a plaintext message prior to encrypting it.