RANDOMIZED ENCRYPTION WITH INSERTIONS AND DELETIONS BASED ON PROBABILITY DISTRIBUTIONS

B. John Oommen and P.C. van Oorschot

TR-247 JUNE 1994

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

RANDOMIZED ENCRYPTION WITH INSERTIONS AND DELETIONS BASED ON PROBABILITY DISTRIBUTIONS

B. John Oommen[1] and P.C. van Oorschot[2]
1994 JUNE 16

ABSTRACT

We present a new symmetric block encryption technique involving randomized expansions, substitutions, insertions and deletions. The encryption process differs from previous techniques in that the secret key is a set of probability distributions. It is proven that meaningful estimation of these is not possible. The decryption process utilizes a maximum likelihood strategy in the space of potential ciphertext messages to recover a unique plaintext message. As is the case for other well-known probabilistic encryption methods, the new method does not provide unconditional security in the information-theoretic sense, but like these, is in some sense reminiscent of Shannon perfect secrecy. Modification of the proposed method to make it suitable for practical use is considered.

Keywords: Randomized encryption, symmetric block ciphers, probabilistic encryption, probability distribution functions.

---

[1]School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. The work of this author was partially supported by the Natural Sciences and Engineering Research Council of Canada.
[2]Bell-Northern Research, P.O. Box 3511, Station C, Ottawa, Canada K1Y 4H7. Also with: School of Computer Science, Carleton University, Ottawa , Canada K1S 5B6.

I. INTRODUCTION

Encryption is used to protect the privacy of information stored or transmitted over insecure media. The intention is that, upon seeing an encrypted message or *ciphertext*, only the authorized receiver should be able to recover the corresponding *plaintext*. To be of use in practice, encryption schemes should be resistant to known-plaintext attacks, i.e. an unauthorized user aware of the encryption strategy (but not the secret decryption parameters or *keys*), and given access to even a large amount of plaintext and the corresponding ciphertext, should be incapable of breaking the scheme. Ideally, the system should also withstand chosen-plaintext attacks, i.e. attacks in which the unauthorized user has access to plaintext-ciphertext pairs corresponding to plaintexts of his choosing. Countless encryption techniques have been devised, particularly in recent years. We refer the reader to [Diff79] for an excellent introduction, [Meye82] and [Davi89] for discussion of some of these techniques, and [Schn94a] for a comprehensive review of more recent research in the area.

In this paper we present a new randomized encryption technique[3] which transforms plaintext using random expansions, substitutions, insertions and deletions. The scheme is a symmetric block-cipher scheme, wherein the encryption mapping is a function of both a secret key shared with the intended recipient, and the output from a random number generator (RNG). The method arose from a theoretical perspective apparently not previously explored in relation to encryption; the resulting approach appears distinct from all others in the literature. Indeed, the theoretical foundation is an intricate blend of stochastic processes [Papo65], adaptive learning [Duda73] and discriminant functions ([Duda73], [Fuku72]), but is presented here independent of these concepts.[4] The decryption process implicitly uses a discriminant function in the space consisting of the set of potential ciphertext messages.

The new scheme uses random edit operations in the encryption phase and employs maximum likelihood computations to achieve unique decryption. In this sense, it is distinct from number-theoretic public-key *probabilistic encryption* schemes such as that of Blum and Goldwasser [Blum84]. It is, however, abstractly related to the McEliece encryption scheme [McEl78] (although the latter is a public-key scheme); both rely on a partitioning of the ciphertext space into disjoint subsets and utilize both randomization and a maximum likelihood decoding strategy. The new scheme also fits nicely into the general model of randomized encryption techniques [Rive82]. We pursue these relations in greater detail in Section II.

The new scheme does not offer unconditional, i.e. information-theoretic security, like the well-known *one-time pad*; indeed, as soon as one fixes the size of the shared secret key, an exhaustive

---

[3]The term *BJO Encryption*© (Copyright 1993 by Dr. B. J. Oommen and Carleton University, Ottawa, Canada) will be used to refer to the new method.

[4]Discriminant functions are typically used in pattern classification ([Duda73], [Fuku72]), and are derived from the *a posteriori* distributions of the pattern classes. Philosophically speaking, these discriminant functions specify the Voronoi partitions in the underlying feature space [Prep85].

attack is possible by an adversary with unlimited computing power. It does however avoid the principal disadvantage of such schemes, namely the requirement of a key as long as the plaintext. Indeed, the goal is to avoid this requirement and yet approach the notion of Shannon's perfect secrecy in a practical manner, i.e. such that the probability is negligible that *any* feasible computation on the ciphertext provides *any* information regarding the corresponding plaintext. This was similarly the motivation for probabilistic encryption [Gold84], and is in contrast to standard public-key cryptography, in which case an attacker can trivially gain partial information about the plaintext p corresponding to a given ciphertext c by guessing a ciphertext p_1, and using the public encryption function E and public key k to compute $c_1=E_k(p_1)$ and check if $c=c_1$.

The following are characteristics of the new scheme. The probability of a particular plaintext block being encrypted to the same ciphertext block twice can be made arbitrarily small. The length of the encryption of a particular block is a random variable. In the most abstract setting, the keys used are not integers or elements of a countable field, but rather functions, namely probability distributions and their corresponding parameters. Thus distinct elements of the family of encryption functions offered by the scheme are determined by varying the probability distribution functions and/or their parameters. The encryption completely destroys the statistical information of the plaintext (Corollary to Theorem II).

The encryption mapping is probabilistically consistent[5] and functionally complete[6]. The former is a necessary condition for a maximum likelihood recovery strategy to be possible, and devising such a mapping is difficult when an indefinite number of insertions is permitted; this is perhaps the reason such a scheme has not previously been proposed. Allowing insertions in the encryption mapping yields the powerful characteristic (Theorem VI) that the probability distributions (i.e. keys) used by two authorized users cannot be meaningfully estimated by an adversary because the distributions are unidentifiable[7]. Viewed from the perspective of a systems analyst, this feature is a direct consequence of the unobservability[8] [Kail80] of the underlying state space; consequently an adversary will never be able to consistently or meaningfully estimate the parameters of the distributions themselves. This implies than opponent will be unable to deduce the encryption key. Functional completeness increases the number of possible ways (internal to the encryption mapping) that a particular ciphertext can result even from a particular plaintext and would seem to only increase the complexity of cryptanalysis.

---

[5] A probabilistic framework is considered consistent if it obeys the basic axioms of probability. In particular, the probability mass associated with each event should be non-negative, and the sum of the probabilities of all the mutually exclusive events should be exactly unity. As opposed to this, the estimate of a quantity is consistent if its value converges to the true unknown value as the number of observations is increased indefinitely.

[6] A more detailed explanation of functional completeness is given in the Section III.3.

[7] A concise definition of unidentifiability is given in Section VII.

[8] The concept of *observability* utilized here is formally distinct from the one traditionally used in the literature [Kail80], but is in the same spirit of the traditional black-box definition.

The organization of the remainder of this paper is as follows. Section II reviews related work on randomized encryption, including a general model. Section III outlines the notation used for the new scheme. Section IV introduces the probability distributions used as cryptographic keys. The two central functions that make up the encryption/decryption process, the *Transform* and the *Recover* processes, are described in Sections V and VI, respectively. After proving the properties of these functions, the actual encryption and decryption algorithms are specified in Section VII. A comparison of the characteristics of the new scheme relative to other schemes in Section VIII is followed by concluding remarks in Section IX. The proofs of the technical results are collected in Appendix A. Appendix B considers a streamlined efficient technique for computing the probabilities required to implement the technique.

II. Background on Randomized Encryption

Many techniques have been proposed which make use of randomization during encryption, some of which have received widespread attention ([McEl78], [Gold82], [Gold84], [ElGa85]). One of the most well-known examples is the use of a random "salt" appended to passwords before they are encrypted in the Unix operating system password file. An overview and unified framework for such techniques, including comments on chosen-plaintext and chosen-ciphertext attacks, is given by Rivest and Sherman [Rive82]. In this section we summarize the main ideas in order to allow the proposed scheme, which was devised independent of these, to be put into proper context with the literature.

The fundamental idea behind randomized encryption techniques is to use randomization to increase the cryptographic security of an encryption process, through one or more of the following methods:

i) Increasing the apparent size of the plaintext space (or, from an information-theoretic point of view, increasing the entropy of the plaintext space [Shan49])

ii) Precluding, or decreasing the effectiveness of chosen-plaintext attacks (by virtue of a one-to-many mapping of plaintext to ciphertext blocks); and

iii) Precluding, or decreasing the effectiveness of statistical attacks (by levelling the *a priori* probability distribution of inputs).

A deterministic encryption technique consists of an algorithm parameterized by a secret key, which maps each message from a fixed space of plaintext messages into a unique string from the fixed ciphertext space. For example, in the case of DES ([FIPS77]), both the plaintext and ciphertext spaces consist of all binary strings of length 64. In contrast, a randomized encryption technique consists of an algorithm parameterized by a secret key *and* output from a random number generator. For a fixed key, to each message from the fixed plaintext, there corresponds a (typically very large) set of potential ciphertexts. Each time a fixed plaintext p is encrypted, the output from the random source is used to non-deterministically select one of these ciphertexts. For the encryption process to

be reversible for a fixed key, i.e. to allow unique decryption, the ciphertext sets corresponding to different plaintexts must be disjoint.

More formally, a randomized encryption technique is a ordered quadruple (M, K, C, ϕ) where M is the plaintext message space, K is the key space, C is the ciphertext space, and ϕ is a subset of M×K×C such that the following two conditions hold:

i) For each pair (m, k) where $m \in M$ and $k \in K$, there is at least one ciphertext $c \in C$ such that $(m, k, c) \in \phi$; and

ii) For each pair (k, c) where $k \in K$ and $c \in C$, there is at most one plaintext $m \in M$ such that $(m, k, c) \in \phi$.

Regarding Condition i), in the case that there is more than one such ciphertext $c \in C$, the random source is used to non-deterministically choose one of the eligible ciphertexts. Alternatively, incorporating a randomness space R into the formal description, one can describe a randomized encryption technique by means of a deterministic function f where $f : M \times K \times R \rightarrow C$, and $f(m,k,r) = c$; here the parameter r selects a unique ciphertext $c \in C$ from the set of eligible ciphertexts defined under i) above for the specified (m,k) pair.

Because the encryption function is a one-to-many mapping, and due to the requirement of unique decoding, it follows that the ciphertext space must be larger than the plaintext space in randomized encryption techniques, and data expansion is unavoidable. This is the primary disadvantage of such systems. The *rate* of a randomized encryption technique is the ratio of the size of a plaintext message to size of the corresponding ciphertext. In the most general setting, the rate may vary for different keys and/or plaintexts, in which case one may consider the *average rate*. Data expansion is typically non-existent (the rate is unity) in practical deterministic encryption schemes.

One randomized encryption technique that has been proposed [Rive82] is a symmetric scheme based on block error-correcting codes, motivated by the McEliece public-key system [McEl78]. Consider a k×n generator matrix G (with $n > k$) for a linear error-correcting code C. In standard operation, a k-bit message m is encoded to the n-bit codeword $\mathbf{x} = \mathbf{m}G$ and transmitted to a receiver. The n-k bits of redundancy (parity-check bits) embedded in $\mathbf{x}$ allow correction of up to t bit-errors (for some t depending on C) by standard techniques (e.g. see [MacW77]). To modify this scheme for encryption, the plaintext $\mathbf{m}$ is encrypted by intentionally perturbing $\mathbf{x}$ by a random n-bit string $\mathbf{e}$ of Hamming weight at most t (i.e. containing at most t ones, in randomly-selected components) : $\mathbf{c} = \mathbf{m}G + \mathbf{e}$. The sender now keeps the matrix G secret from all but the intended receiver (whereas for error-correction, G is generally known); this allows the receiver to recover the plaintext by standard error-correction decoding, but the NP-completeness of the general decoding problem for linear error-correcting codes appears to make this task infeasible for adversaries without knowledge of G.

The randomized encryption scheme of the preceding paragraph is based on the theory of linear error-correcting codes, which generally employ a maximum-likelihood decoding strategy, i.e. a

strategy which decodes a received vector (binary n-tuple) y to the codeword x which minimizes the Hamming distance $d_H(x, y)$ -- the number of bit positions differing in x and y. This is the codeword x which maximizes the probability p(y|x). This strategy may be conceptualized as follows: partition the space V of all binary n-tuples into (disjoint) subsets defined by the *spheres of radius t* about each codeword x, plus a single (possibly empty) set consisting of all vectors in V not in any of these. The sphere about a particular codeword x consists of all vectors in V of Hamming distance t or less from x. The decoding strategy is then equivalent to locating the sphere in which the received vector y falls, and decoding y to the codeword at the centre of that sphere. As will be discussed later, the new scheme proposed also makes use of a maximum-likelihood decoding strategy; the partitioning is a partitioning of the ciphertext space V, and is a function of various secret probability distributions (which serve as the secret encryption key). This can be viewed as a generalization of the McEliece scheme; in the latter, the partitioning is determined by the secret generator matrix G (or equivalently, the particular error-correcting code C used).

III. NOTATION AND TERMINOLOGY

Let A be a finite alphabet, and A^* be the set of strings over A. $\lambda \notin A$ is the null symbol, which for reasons explained presently will be called the *output* null symbol. A string $X \in A^*$ of the form $X=x_1x_2...x_N$ is said to be of length $|X| = N$. Its prefix of length i will be written as X_i, where $i < N$. Upper case symbols represent strings, and lower case symbols, elements of the alphabet under consideration. The symbol $\cup$ represents the set union operator.

H, a finite subset of A^*, is called the *dictionary*. The elements of H represent blocks or sub-blocks of plaintext to be encrypted.

As an example, the alphabet could be the set of hexadecimal symbols $\{0,1,..,F\}$ and the dictionary the set of bytes $\{00, 01, ,0F,...,F0, F1, ...,FF\}$.

III.1 The Input and Output Null Symbols : ξ and λ

Let Y' be any string in $(A \cup \{\lambda\})^*$, the set of strings over $(A \cup \{\lambda\})$. Y' is called an *output* edit sequence. The operation of transforming a symbol $a \in A$ to λ will represent the deletion of 'a'.

To differentiate between the operations of deletion and insertion, we introduce the symbol ξ, called the *input* null symbol. Let U' be any string in $(A \cup \{\xi\})^*$, the set of strings over $(A \cup \{\xi\})$. U' is called an *input* edit sequence. ξ is distinct from λ, but is used in an analogous way to formalize symbol insertion. Transforming the symbol ξ to $b \in A$ will represent the insertion of the symbol b.

III.2 The Input and Output Compression Operators : C_I and C_O

The Output Compression Operator, C_O is a mathematical function which maps from $(A \cup \{\lambda\})^*$ to A^*. $C_O(Y')$ is Y' with all the occurrences of the symbol λ removed. Note that C_O preserves the order of the non-λ symbols in Y'. Thus, for example, if $Y'=f\lambda o\lambda r$, $C_O(Y')=for$.

Analogously, the Input Compression Operator, C_I is a mathematical function which maps from $(A \cup \{\xi\})^*$ to A^*. $C_I(U')$ is U' with all the occurrences of the symbol ξ removed. Again, note that C_I preserves the order of the non-ξ symbols in U'.

III.3 The Set of all Possible Edit Operations : $\Gamma(U,Y)$

For every pair (U,Y), $U,Y \in A^*$, the finite set $\Gamma(U,Y)$ is defined by means of the compression operators C_I and C_O , as a subset of $(A \cup \{\xi\})^* \times (A \cup \{\lambda\})^*$:

$$\Gamma(U,Y) = \{(U', Y') \mid (U', Y') \in (A \cup \{\xi\})^* \times (A \cup \{\lambda\})^*, \text{ and each } (U',Y') \text{ obeys (i) - (iii)}\} \quad (1)$$

(i) $C_I(U') = U$; $C_O(Y') = Y$

(ii) $|U'| = |Y'|$

(iii) For all $1 \leq i \leq |U'|$, it is not the case that both $u_i' = \xi$ and $y_i' = \lambda$.

By definition, if $(U', Y') \in \Gamma(U,Y)$, then, $\text{Max}[\,|U|, |Y|\,] \leq |U'| = |Y'| \leq |U| + |Y|$.

The meaning of the pair $(U', Y') \in \Gamma(U,Y)$ is that it corresponds to one way of encrypting (or mutating) U into Y, using the edit operations of substitution, deletion and insertion. The edit operations themselves are specified for $1 \leq i \leq |Y'|$, as (u_i', y_i'), which represents the transformation of u_i', to y_i'. The cases below consider the three edit operations individually.

(i) If $u_i' \in A$ and $y_i' \in A$, it represents the *substitution* of y_i' for u_i'.

(ii) If $u_i' \in A$ and $y_i' = \lambda$, it represents the *deletion* of u_i'. Between (i) and (ii) all the symbols in U are accounted for.

(iii) If $u_i' = \xi$ and $y_i' \in A$, it represents the *insertion* of y_i'. Between (i) and (iii) all the symbols in Y are accounted for.

$\Gamma(U,Y)$ is an exhaustive enumeration of the set of all the operations and *sequence of operations* by which U can be mutated to Y. Observe that we do not permit the deletion a symbol that has once been inserted or substituted. Lemma 0 below specifies the cardinality of $\Gamma(U,Y)$.

In terms of nomenclature, we shall specify that a mutation process transforming a string X to Y is *functionally complete* if it incorporates every way of transforming X to Y considering both the underlying traces[9] and the sequence in which the operations are done.

Lemma O.

The cardinality of $\Gamma(U,Y)$ is given by :

$$\left|\Gamma(U,Y)\right| = \sum_{k=\text{Max}(0,|Y|-|U|)}^{|Y|} \frac{(|U|+k)!}{(k!\,(|Y|-k)!\,(|U|-|Y|+k)\,!)} \quad (2)$$

Proof of Lemma O: See Appendix A. ♦♦♦

[9] A *trace* is a systematic representation for the edit process of transforming one string to another. We refer the reader to [Sank83] for a more formal definition of a trace, and its application in string and tree editing.

Remarks

1. Note that the size of $\Gamma(U,Y)$ increases with the lengths of U and Y. Thus, for example, while $\Gamma(3,3)$ has 63 elements, $\Gamma(4,4)$, $\Gamma(5,5)$ and $\Gamma(6,6)$ have 321, 1683 and 8989 elements respectively. The encryption technique transforms U to Y by randomly selecting one of the elements of $\Gamma(U,Y)$.

2. $\Gamma(U,Y)$ represents the set of all ways by which U can be transformed to Y, and it contains many duplicate entries in terms of the edit operations themselves. Thus, for U = "f" and Y = "go" :

$\Gamma(U,Y) = \{(f\xi, go), (\xi f, go), (f\xi\xi, \lambda go), (\xi f\xi, g\lambda o), (\xi\xi f, go\lambda)\}$.

In particular, the pair $(\xi f, go)$ represents the edit operations of inserting the 'g' and replacing the 'f' by an 'o'. Notice that the last three pairs *all* represent the deletion of 'f' and the insertion of both 'g' and 'o'. The difference between the three is the *sequence* in which these operations are accomplished (cf. definition of functional completeness above).

III.4 The Set Of Valid Expansions : Φ , and the Expansion Operator : T.

Φ is a predefined set of expansions of the symbols of A. We assume that each symbol in A can be transformed into one of J strings, where J is a predefined constant for an instantiation of the encryption. For a symbol $a \in A$, we refer to these strings as Φ_a where,

$$\Phi_a = \{\phi_{a1}, \phi_{a2} \cdots, \phi_{aJ}\}$$

Φ in turn is the union of these sets, Φ_a.

The Expansion Operator $T_j(X)$ expands each x_i of X to the j^{th} element of *its* corresponding expansion set Φ_{x_i}. The intention in introducing expansions is to render cryptanalysis more difficult by making expansions indistinguishable from multiple insertions.

Example 0.

To clarify the use of the operation $T_j(X)$, suppose that A is the set of hexadecimal symbols {0,1,..,F} and that Φ_2 and Φ_7 are {01, 13, 45, AB} and {34, 49, 67, 33} respectively. Note that J, the cardinality of these sets is 4. Then, if X = "272", the expansion of X using j=2 is $T_2(X)$="134913".

We assume that an instantiation of the encryption scheme fixes Φ.

IV. The Encryption Key ($\mathfrak{K}$)

The first component of the key, the *Expansion Distribution*, F, is defined so as to randomly choose the expansion operation. It is a probability distribution defined over the set of integers {1,..J}. F(j) is the probability that the encryption process, while operating on the current plaintext block, utilizes the j^{th} expansion ϕ_{aj} for all $a \in A$, thus expanding a to ϕ_{aj}. Note that F(j) has to satisfy :

$$\sum_{j=1}^{J} F(j) = 1. \qquad (3)$$

The random variable *J* is the index of the expansion used for each symbol.

The second component of the key is a probability distribution G over the set of positive integers. The random variable it describes is Z specifying the number of insertions that are performed in mutating the current plaintext block. G is called the *Quantified Insertion Distribution.* The quantity G(z) is the probability that $Z = z$. Note that G(z) satisfies :

$$\sum_{z \geq 0} G(z) = 1. \tag{4}$$

Typical parametric distributions for G are the Poisson and Geometric distributions ([Devr86], [Papo65]). However, in practical instantiations the specification[10] of the key requires the declaration of G for a finite set of integers bounded by the maximum number of insertions permitted.

The third component of the key is a distribution Q defined over the alphabet, A. Q is called the *Qualified Insertion Distribution* specifying the probability that $a \in A$ will be the inserted symbol conditioned on the fact that an insertion is to be performed. Note that Q satisfies the constraint :

$$\sum_{a \in A} Q(a) = 1. \tag{5}$$

The final component of the key is a probability distribution S over $A \times (A \cup \{\lambda\})$ called the *Substitution and Deletion Distribution.* The quantity S(b|a) is the conditional probability that a given symbol $a \in A$ will be randomly transformed into $b \in (A \cup \{\lambda\})$. Hence, S(c|a) is the conditional probability of $a \in A$ being substituted for by $c \in A$, and $S(\lambda|a)$ is the conditional probability of $a \in A$ being deleted. Observe that S has to satisfy the following constraint for all $a \in A$:

$$\sum_{b \in (A \cup \{\lambda\})} S(b|a) = 1. \tag{6}$$

The key $\mathbb{k}$ consists of the distributions F, G, Q and S, and is assumed secret.

We clarify the above with an example, with parameters chosen for illustrative purposes only.

---

[10] Viewed from an abstract perspective, in general, Z could be a random variable whose value depends on X, the block or sub-block of plaintext being encrypted. However, for the sake of simplicity, for the rest of this paper we shall assume that Z is independent of X.

Example I.

Let the alphabet A be the set of two-bit strings $\{00, 01, 10, 11\}$. Also, let J, the number of possible expansions for a symbol be 3, and the maximum number of insertions permitted be 4. Then a possible key would be :

j	1	2	3
F(j)	0.1	0.6	0.3

The distribution F ; j is the index of the expansion used

z	1	2	3	4	5
G(z)	0.05	0.2	0.5	0.15	0.1

The distribution G ; z is the number of insertions.

a	00	01	10	11
Q(a)	0.1	0.2	0.3	0.4

The distribution Q ; the random variable is the inserted symbol $a \in A$.

a \ b	00	01	10	11	λ
00	0.45	0.15	0.2	0.1	0.1
01	0.23	0.12	0.15	0.1	0.4
10	0.3	0.15	0.5	0.02	0.03
11	0.18	0.22	0.32	0.08	0.2

The distribution S ; the random variable is the substituted or deleted symbol $b \in A \cup \{\lambda\}$ for each $a \in A$.

The distribution F is used to select the expansion done for a particular plaintext block (see Example 0 above). Also, in the encryption of any plaintext block, G would specify the number of insertions, this being any number ranging from zero to four. In this case, the probability of 2 insertions taking place is 0.5. The distribution Q specifies the insertion of a symbol given that an insertion is done. Here, the symbol 10 is inserted with probability 0.3. Finally, S specifies the substitution and deletion operations. In this case, the symbol 01 is transformed into 00 and 10 with probability 0.23 and 0.15 respectively, and it is deleted with probability 0.4. ♦♦♦

V. THE *TRANSFORM* MODULE ($\mathfrak{T}$)

The encryption and decryption stages of the new scheme depend on two central processes *Transform* and *Recover*, denoted $\mathfrak{T}$ and $\mathfrak{R}$ respectively. In this section we describe the former, which transforms a string $X \in H$ into a random element in A^* as a function of the key $\mathfrak{K}$. We subsequently prove its properties.

Transform(X)
Input : The plaintext block or sub-block to be transformed, X.
Assumption : The process assumes the knowledge of the key $\mathfrak{K} = \{F, G, Q, S\}$.
Output : Y, a transformed version of X .
Method:

1. Using the distribution F, randomly choose the variable *J* (i.e., an integer $j \in \{1,..,J\}$). X is mutated by expanding X using the Expansion Operator $T_j(X)$, replacing each x_i by the j^{th} element of *its* corresponding expansion set Φ_{x_i}. Let $U \in A^*$ be the resultant string. Let N = |U|.
2. Using G randomly choose Z = z, the number of symbols to be inserted.
3. Determine the position of the insertions among the individual symbols of U by randomly generating an input edit sequence $U' \in (A \cup \{\xi\})^*$, where each of the (N+z)!/(N!z!) possible strings are equally likely. The positions of the symbol ξ in the string U' represents the positions where symbols will be inserted in U.
4. Randomly substitute or delete the non-ξ symbols in U' using the distribution S.
5. Insert symbols by randomly transforming the occurrences of ξ independently into the individual symbols of the alphabet using Q.

END *Transform*

Transform is pictorially shown in Figure I in Appendix C. Example II is intended to clarify it.

Example II.

Let A, Φ_2 and Φ_7 be defined as in Example 0 (Section III.4) with J=4.

Let X = "27". Then if *J* is randomly assigned the value j=2, X is expanded by using the second expansions in the above sets yielding the resultant string U to be U = "1349"

The random number of insertions to be performed, dictated by G, is now ascertained. Let this random number be 2. The positions of these two insertions is now randomly chosen. Suppose the resultant string is U'= "1ξ3ξ49". The non-ξ symbols of U' are now randomly substituted for or deleted using the distribution S. Suppose that '1' gets transformed to 'A', '3' gets transformed to 'B', '4' is deleted, and '9' gets transformed to '5'. The new string that is to be operated on is thus U'= "AξBξ5". The ξ's in U' are now transformed into the symbols of the A using the distribution Q. Suppose the first ξ gets changed into a '3' and the second gets transformed into a '2'. The resultant string Y, which is the final transformed version of X is thus Y = "A3B25". ♦♦♦

We now prove the properties of $\mathfrak{T}$.

Let |Y| have the value M. Then, using the above notation we have the following results :

Theorem I:

Let Pr[Y|U] be defined as follows :

$$\Pr[Y|U] = \sum_{z=\text{Max}(0,M-N)}^{M} \frac{G(z)\cdot(N!\,z!)}{((N+z)!)} \sum_{U'} \sum_{Y'} \prod_{i=1}^{N+z} p(y_i'|u_i') . \qquad (7)$$

where,

(a) y_i' and u_i' are the individual symbols of Y' and U' respectively,

(b) $p(y_i'|u_i')$ is defined as $Q(y_i')$ if u_i' is ξ, and

(c) $p(y_i'|u_i')$ is defined as $S(y_i'|u_i')$ if u_i' is not ξ. (8)

Then Pr[Y|U] is both functionally complete and probabilistically consistent, as defined above.

Proof of Theorem I: See Appendix A. ♦♦♦

Theorem II:

The quantity Pr [Y|X] can be obtained from Pr [Y|U] by :

$$\Pr[Y|X] = \sum_{j=1}^{J} \Pr[Y|U = T_j(X)]\cdot F(j)) \qquad (9)$$

Proof of Theorem II: See Appendix A. ♦♦♦

Corollary to Theorem II:

The unigram, bigram, and indeed, all the relevant n-gram statistics of the input are completely destroyed by the *Transform* mapping.

Proof: By virtue of Theorems I and II the *total* probability (over all X) of obtaining a transformed string Y is completely dependent on the distributions F, Q and S and the set of permutations, Φ. Thus the symbols occurring at the output of $\mathcal{T}$ are controlled by the above distributions and expansions which operate on the individual characters of the alphabet, and thus the unigram and bigram (and indeed, all the relevant n-gram) statistics of the input are completely destroyed. ♦♦♦

VI. THE *RECOVER* MODULE ($\mathcal{R}$)

The second process that is central to the encryption is *Recover*, denoted $\mathcal{R}$, which processes a transformed string Y and computes the most likely string in the dictionary that it could have originated from. To achieve this, the encryption utilizes a maximum likelihood decision rule which involves computing for every $X \in H$ the probability of having Y as the transformed string given that X was the original word. One who is conversant with the theory of discriminant functions will realize that this essentially implies that we are utilizing the functional form of Pr[Y|X] as a discriminant function to recover X from Y. Indeed, in this case, when the words in H are all equally likely, the discriminant function will also be the minimum probability of error Bayesian decision rule, for it will decide on X based on the corresponding Voronoi partitions of A^*.

We shall now develop the maximum likelihood recovery process for a received string Y. To do this we shall present a fast and efficient way of computing $Pr[Y|X]$ rather than compute it using the brute-force method of exhaustively evaluating the individual components of (7).

Consider the problem of editing U to Y, where $|U|=N$ and $|Y|=M$. Suppose we edit a prefix of U into a prefix of Y, using exactly i insertions, e deletions and s substitutions. Since the number of edit operations are specified, this corresponds to editing $U_{e+s} = u_1 \ldots u_{e+s}$, the prefix of U of length e+s, into $Y_{i+s}=y_1 \ldots y_{i+s}$, the prefix of Y of length i+s. Let $Pr[Y_{i+s}|U_{e+s} ; Z=i]$ be the probability of obtaining Y_{i+s} given that U_{e+s} was the original string, and that exactly i insertions took place. Then, by definition,

$$Pr[Y_{i+s}|U_{e+s} ; Z=i] = 1 \qquad \text{if } i=e=s=0 \tag{10}$$

To obtain an explicit expression for the above quantity for values of i, e and s which are nonzero, we have to consider all the possible ways by which Y_{i+s} could have been obtained from U_{e+s} using exactly i insertions. Let r=e+s and q=i+s. Let $\Gamma_{i,e,s}(U,Y)$ be the subset of the pairs in $\Gamma(U_r,Y_q)$ in which every pair corresponds to i insertions, e deletions and s substitutions. Since we shall repeatedly be using the strings U and Y, $\Gamma_{i,e,s}(U,Y)$ will be referred to as $\Gamma_{i,e,s}$. Using (7) and (8),

$$Pr[Y_{i+s}|U_{e+s} ; Z=i] = \frac{(s+e)!\ i!}{(s+e+i)!} \sum_{(U_r',Y_q')} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}'), \qquad \text{if i, e or s} > 0 \tag{11}$$

where, (U_r',Y_q') is the arbitrary element of the set $\Gamma_{i,e,s}$, with u_{rj}' and y_{qj}' as the jth symbols of U_r' and Y_q' respectively.

Let W(. , . , .) be the array whose general element W(i,e,s) is the sum of the product of the probabilities associated with the general element of $\Gamma_{i,e,s}$ defined as below.

$$W(i,e,s) = 0 \qquad \text{if i,e or s} < 0; \text{ and}$$

$$W(i,e,s) = \frac{(s+e+i)!}{i!\ (s+e)!}\ Pr[Y_{i+s}|U_{e+s} ; Z=i] \qquad \text{otherwise.} \tag{12}$$

Using the expression for $Pr[Y_{i+s}|U_{e+s} ; Z=i]$ we obtain the explicit form of W(i,e,s) for all nonnegative values of i, e and s as in (13).

$$W(i,e,s) = 1, \qquad \text{if } i=e=s=0$$

$$= \sum_{(U_r',Y_q')} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}'), \qquad \text{if i, e or s} > 0 \tag{13}$$

To obtain bounds on the magnitudes of the variables i, e and s, we observe that they are constrained by the lengths of the strings X and Y. Thus, if r=e+s, q=i+s and R=Min [M, N], these variables will have to obey the following obvious constraints.

$$\text{Max}[0,M-N] \le i \le q \le M \ ; \quad 0 \le e \le r \le N \ ; \quad 0 \le s \le \text{Min}[M,N].$$

Values of triples (i,e,s) which satisfy these constraints are termed as *feasible*. Let,

$$H_i = \{ j \mid \text{Max}[0,M-N] \le j \le M \},$$

$$H_e = \{ j \mid 0 \le j \le N \}, \text{ and}$$

$$H_s = \{ j \mid 0 \le j \le \text{Min } [M, N] \} \qquad (14)$$

H_i, H_e and H_s are called the set of *permissible values* of i, e and s. Observe that a triple (i,e,s) is feasible if in addition to $i \in H_i$, $e \in H_e$, and $s \in H_s$, the following is satisfied:

$$i + s \le M, \quad \text{and} \quad e + s \le N. \qquad (15)$$

The following result specifies the allowable forms of the feasible triples encountered on transforming U_r, the prefix of U of length r, to Y_q, the prefix of Y of length q.

Theorem III:

To edit U_r, the prefix of U of length r, to Y_q, the prefix of Y of length q, the set of feasible triples is given by $\{(i, r-q+i, q-i) \mid \text{Max } [0, q-r] \le i \le q\}$.

Proof of Theorem III: See Appendix A. ♦♦♦

We shall now derive a relation showing how W(.,.,.) can be recursively defined.

Theorem IV:

Let W(i,e,s) be the quantity defined as in (13) for any two strings U and Y. Then, for all nonnegative i,e and s,

$$W(i,e,s) = W(i-1,e,s)\cdot p(y_{i+s}|\xi) + W(i,e-1,s)\cdot p(\lambda|u_{e+s}) + W(i,e,s-1)\cdot p(y_{i+s}|u_{e+s}) \qquad (16)$$

where p(b|a) is defined as in (8).

Proof of Theorem IV: See Appendix A. ♦♦♦

The computation of the probability Pr[Y|U] from the array W(i,e,s) merely involves weighting the various terms by factors dependent only on the number of insertions, as given by Theorem V.

Theorem V:

If $h(i) = G(i)\cdot\dfrac{N!\ i!}{(N+i)!}$, the quantity Pr[Y|U] can be evaluated from the array W(i,e,s) as :

$$Pr[Y|U] = \sum_{i=\text{Max}(0,M-N)}^{M} h(i)\cdot W(i, N-M+i, M-i). \qquad (17)$$

Proof of Theorem V: See Appendix A. ♦♦♦

To evaluate Pr[Y|U] as defined in Theorem V we make use of the fact that although the latter index itself does not seem to have any recursive properties, the index W(., ., .), which is closely related to it has the interesting properties proved in Theorem IV. We now present the procedure *EvaluateProbabilities* which evaluates the array W(., ., .) for all permissible values of i, e and s

subject to the constraints given in (14). Using the array W(i,e,s) it then computes Pr[Y|U] by adding up the weighted contributions of the pertinent elements in W(., ., .) as specified by Theorem V.

The evaluation of the array W(., ., .) has to be done in a systematic manner, so that any quantity W(i,e,s) is evaluated before its value is required in any further evaluation. This is easily done by considering a three-dimensional coordinate system whose axes are i, e and s respectively. Initially, the contribution associated with the origin, W(0,0,0) is assigned the value unity, and the contributions associated with the vertices on the axes are evaluated. Thus, W(i,0,0), W(0,e,0) and W(0,0,s) are evaluated for all permissible values of i, e and s. Subsequently, the i-e, e-s and i-s planes are traversed, and the contributions associated with the vertices on these planes are evaluated using the previously evaluated values. Finally, the contributions corresponding to strictly positive values of the variables are evaluated. To avoid unnecessary evaluations, at each stage, the variables must be tested for permissibility using the constraints of (14). Finally, Pr[Y|U] is evaluated by adding the weighted contributions of W(.,.,.) associated with the points on the line given by the parametric equation:

$$i = i \;\; ; \;\; e=N-M+i \;\; ; \;\; s=M-i.$$

The procedure is formally given below.

Procedure *EvaluateProbabilities*
Input: The strings $U=u_1u_2...u_N$, $Y=y_1y_2...y_M$, and the key, $\mathfrak{K}$ = {F, G, Q, S}. Let R=Min [M, N].
Output: The array W(i,e,s) for all permissible values of i, e and s and the probability Pr[Y|U].
Method :

W(0,0,0) :=1
Pr[Y|U] := 0
For i := 1 to M **Do** W(i,0,0) := W(i-1,0,0)·Q(y_i) /*Traverse i-axis */
For e := 1 to N **Do** W(0,e,0) := W(0,e-1,0)·S(λ|u_e) /*Traverse e-axis */
For s := 1 to R **Do** W(0,0,s) := W(0,0,s-1)·S(y_s|u_s) /*Traverse s-axis */
For i := 1 to M **Do**
  For e := 1 to N **Do** /*Traverse i-e plane */
    W(i,e,0) := W(i-1,e,0)·Q(y_i) + W(i,e-1,0)·S(λ|u_e)
For i := 1 to M **Do**
  For s := 1 to M-i **Do** /*Traverse i-s plane */
    W(i,0,s) := W(i-1,0,s)·Q(y_{i+s}) + W(i,0,s-1)·S(y_{i+s}|u_s)
For e := 1 to N **Do**
  For s := 1 to N-e **Do** /*Traverse e-s plane */
    W(0,e,s) := W(0,e-1,s)·S(λ|u_{e+s}) + W(0,e,s-1)·S(y_s|u_{e+s})
For i := 1 to M **Do**
  For e := 1 to N **Do**
    For s := 1 to Min[(M-i) , (N-e)] **Do** /*Traverse trellis */
      W(i,e,s) := W(i-1,e,s)·Q(y_{i+s})+W(i,e-1,s)·S(λ|u_{e+s})+W(i,e,s-1)·S(y_{i+s}|u_{e+s})
For i := Min[0 , M-N] to M **Do**

$$Pr[Y|U] := Pr[Y|U] + G(i)\frac{N!\ i!}{(N+i)!}\cdot W(i,N-M+i,M-i)$$

END Procedure *EvaluateProbabilities*

The above implementation evaluates Pr[Y|U] by appropriately utilizing the properties derived in Theorems III-V. A more efficient technique for computing this can be devised. We pursue this important issue in Appendix B where a superior procedure *EfficientlyEvaluateProbabilities* to evaluate Pr[Y|U] has been developed.

We are now in a position to define the process *Recover*, which utilizes the above procedure (or its more efficient version) to compute Pr[Y|U] and attempts to "learn" the identity of X from Y by using a maximum likelihood decision rule. Note that, as mentioned in the preamble to this section, this is tantamount to "carving" A^* into subspaces where each subspace corresponds to a distinct element of H. The process evaluates the probability Pr[Y|X] for every word in H and returns the value of X as the one which maximizes the probability Pr[Y|X]. Note that from a Bayes decision theoretic point of view this is equivalent to using the functions Pr[Y|X] as discriminant functions when the elements in H have an equally likely *a priori* distribution.

Recover (Y)
Input : A string Y.
Assumption : ***Recover*** assumes the knowledge of the key $\mathbb{K}$ = {F, G, Q, S} and the dictionary, H.
Output : $X^+ \in H$, the maximum likelihood estimate of the element from which Y originated.
Method:
For every string $X \in H$ **Do**

Evaluate Pr[Y|X] using Theorem II as:[11] $\Pr[Y|X] = \sum_{j=1}^{J} \Pr[Y|U = T_j(X)] \cdot F(j))$

X^+ := string maximizing Pr[Y|X].
END ***Recover***

VII. ENCRYPTION AND DECRYPTION

Having described the *Transform* and *Recover* processes, we are now in a position to formalize our proposed encryption and decryption algorithms. The method is straightforward: the plaintext is subdivided into blocks where each individual block is an element of the dictionary H. The plaintext block to be encrypted, say X^P, is first mutated using *Transform*. The mutated string, Y, is then subsequently analyzed to determine if, on recovery, it would yield back X^P. If it indeed would be recovered as X^P, it is transmitted as the ciphertext for X^P. If, however, Y cannot be recovered as X^P, the string Y is discarded and *Transform* is invoked again until a string is obtained which can be recovered as X^P.

Viewed from the perspective of adaptive learning and discriminant functions, we see that for a string X^P we effectively invoke *Transform* until we obtain a transformed element which falls within the region in A^* allocated to X^P as per the set of discriminant functions dictated by Pr [Y|X].

---

[11] Here $\Pr[Y|U=T_j(X)]$ is evaluated using *EvaluateProbabilities* described above **OR** *EfficientlyEvaluateProbabilities* developed in Appendix B.

The procedures *EncryptBlock* and *DecryptBlock* follow.

EncryptBlock
Input : A plaintext block X^P to be encrypted.
Assumption : The procedure assumes the knowledge of the key $\mathbb{K} = \{F, G, Q, S\}$.
Output : Y, a randomized encryption of X^P.
Method:
    Repeat
        $Y := Transform(X^P)$
        $X^+ := Recover(Y)$
    Until $(X^+ = X^P)$
END ***EncryptBlock***

EncryptBlock ensures that a ciphertext will be properly decrypted by a single invocation of *Recover*. This is formalized in the procedure DecryptBlock below.

DecryptBlock
Input : A ciphertext block Y.
Assumption : The procedure assumes the knowledge of the key $\mathbb{K}$.
Output : X^P, the plaintext decryption of Y.
Method:
    $X^P := Recover(Y)$
END ***DecryptBlock***

We now prove the primary cryptanalytic property of the encryption which is the fact that the parameters of the distributions F, G, Q, and S can never be consistently or meaningfully[12] estimated. Note that unlike a probabilistic framework which is considered consistent if it obeys the basic axioms of probability, an estimate of a quantity is consistent if its value converges to the true unknown value as the number of observations is increased indefinitely. Closely related to this is the concept of identifiability defined below.

Definition : A distribution or set of distributions is said to be identifiable if a set of parameters $\{\underline{\Theta}\}$ exist, which uniquely characterize it. More formally [Ever81], a set $\mathbb{F}$ of distributions[13] describing a random vector $\underline{x}$ is termed *identifiable* if and only if for any two parameters sets $\{\underline{\Theta}\}$ and $\{\underline{\Theta}^*\}$, the equality of the total distributions defining the random vector $\underline{x}$

$$\mathbb{F}(\underline{x}\,;\,\{\underline{\Theta}\}) = \mathbb{F}(\underline{x}\,;\,\{\underline{\Theta}^*\})$$

implies that $\underline{\Theta}$ equals $\underline{\Theta}^*$.

---

[12] By saying the distributions cannot be *meaningfully* estimated we informally mean that any guess of the distributions is no better than a guess made with or without observation of any ciphertext or plaintext-ciphertext pairs.

[13] The concept highlighted here is that the cryptanalyst will not be able to meaningfully estimate the parameters for $\mathbb{F}$ or any equivalent definition of $\mathbb{F}$.

Theorem VI:

The distributions which serve as the secret key of the scheme, namely F, G, Q, and S, are unidentifiable, and furthermore the parameters of the distributions can never be consistently or meaningfully estimated even if a cryptanalyst has access to infinite samples of chosen plaintext and the corresponding ciphertext.

Proof of Theorem VI: See Appendix A. ♦♦♦

VIII. ANALYSIS OF THE PROPOSED METHOD

For the purpose of analysis, we follow that standard (prudent) assumption that any potential adversary is aware of the encryption technique being used, but not the key. In the proposed system, "technique" is taken to include the type of probability distributions used, but not the particular parameters. As always, the attacker's task will typically be more difficult if, in practice, the technique itself is not known.

The *Transform* module resembles the one-time pad in the following sense : any input may be mapped to any output. Indeed, whereas the one-time pad allows any ciphertext of length n to be the encrypted image of any plaintext of length n, the proposed technique allows any output of length m be the image of any input of length n, for arbitrary n and m. Observe though that if a plaintext block could be encrypted under a fixed key into any arbitrary ciphertext block, then a unique decryption would be impossible. Unique decryption is rendered possible in the proposed system by the module *Transform* working in conjunction with the module *Recover*. Consequently, the encryption of a plaintext block is any recoverable random element in the *subset* of all ciphertext blocks simultaneously tested by both the processes *Transform* and *Recover*. However, averaged over *all* possible keys, the encryption of a particular plaintext block could be any arbitrary ciphertext block.

Regarding cryptanalysis, the above property that any plaintext can get mapped to any ciphertext, seems to rule out chosen ciphertext attacks (see [Rive82, p.159]). In addition, the proposed scheme appears to have most or all of the other advantages of the better of the randomized encryption schemes considered by Rivest and Sherman - including apparent resistance to chosen-plaintext attacks, resistance to spoofing attack based on known-plaintext [Rive82, p.152], intrinsic invulnerability to "bit-twiddling" attacks, and control of amount of data expansion (cf. [Rive82, p.150]).

Before analysis of a particular practical implementation of this scheme is undertaken and a comparison of the implementation with other currently used schemes can be carried out, the following issues must be resolved in addition to the selection of an appropriate alphabet, dictionary, and plaintext block size.

i) *Key size.* In the proposed method, the size of the (shared) secret key is determined by the strategy utilized to specify the distributions. The key size will be impractically large if the

distributions used are explicitly specified, for example using arrays of probabilities. A true random key (for example, of 112-bits) might be used to specify the entire set of distributions by appropriately defining the mapping from the true random key to the actual distributions and parameters.

ii) *Data expansion.* This is an unavoidable characteristic inherent in all randomized encryption techniques. Larger degrees of data expansion typically allow the potential for increased security; this follows from entropy considerations in the plaintext space. The degree of data expansion tolerable will typically control the range of G, the Quantified Insertion Distribution. Additionally, depending on the amount of redundancy in the plaintext, one might consider data compression prior to encryption; this would be at the expense of computing time. The effect that bounding the data expansion would have on the security of the scheme has yet to be fully considered.

iii) *Variable length ciphertext blocks.* To enable proper decoding, information regarding ciphertext block lengths must be available to the authorized receiver. In any instantiation, inclusion of such information may conceivably decrease the cryptographic advantage gained by allowing variable-length ciphertext blocks. Methods to avoid sending cleartext blocklengths (e.g. by encryption) might be considered; methods might also be considered to avoid bandwidth requirements for length indicators themselves.

iv) *Variable throughput.* Internally, the proposed method generates candidate ciphertexts until a ("successful") candidate is found which can be uniquely decoded. Each unsuccessful candidate forces an additional iteration, increasing encryption time. The ratio of successful candidates to unsuccessful depends on the distributions utilized by the specific instantiation. What is desired is to find probability distributions and their corresponding parameterizations which yield "good" Voronoi partitions of the ciphertext space, while keeping the entropy of the key space sufficiently large to make the cryptanalyst's task infeasible. We have yet to study this question in detail, or to examine the expected number of iterations for standard probability distributions.

v) *Speed.* In the absence of decisions regarding specific design and implementation issues, few comments can be made regarding the encryption speed (throughput) of any practical instantiation of the proposed method. To be of practical use, an instantiation should have throughput comparable to that of other schemes currently proposed (e.g. see [Schn94b]). The speed will be influenced by various factors including the distributions used in the instantiation and the cardinality of the plaintext dictionary H.

vi) *Effect of pseudo-randomness.* Ideally, randomized encryption techniques should use true random-bit sources. In a practical instantiation, if the scheme uses sources which are not

truly random, the effect of using pseudo-random number generators, even those which are believed to be cryptographically secure, must be carefully considered.

IX. CONCLUDING REMARKS

Rivest and Sherman [Rive82, p.160] suggested it would be interesting to find operations other than exclusive-or and concatenation for combining messages with random sequences. The proposed method is an example of such a system, which is distinct from both these and the number-theoretic methods based on probabilistic encryption, the latter of which rely on number-theoretic assumptions for their security such as the efficient probabilistic scheme of Blum and Goldwasser [Blum84].

The proposed method provides an interesting new randomized encryption technique. It is of theoretical interest as it offers a novel way, using probability distributions as keys, to generate partitions of a ciphertext space in a one-to-many randomized encryption mapping, such that ciphertexts are uniquely decodable. The security appears comparable to other symmetric randomized encryption techniques. With suitable modifications, the proposed scheme may be adaptable to be sufficiently fast for practical use. However, such modification may make it difficult to analyze the security of the modified technique. Whether the technique can be adapted and implemented in practice to yield both acceptable throughput and security is the subject of further study.

APPENDIX A: PROOFS OF LEMMAS AND THEOREMS

This appendix contains proofs of the technical claims made in the main body of the paper.

Lemma O.

The cardinality of $\Gamma(U,Y)$ is given by :

$$|\Gamma(U,Y)| = \sum_{k=Max(0,|Y|-|U|)}^{|Y|} \frac{(|U|+k)!}{(k!\ (|Y|-k)!\ (|U|-|Y|+k)\ !)}$$

Proof of Lemma O:

First of all note that $|\Gamma(U,Y)|$ depends only on $|U|$ and $|Y|$, and not on the actual strings U and Y themselves. Further, observe that the transformation of a symbol $a \in A$ to itself is also considered as an operation in the arbitrary pair $(U', Y') \in \Gamma(U,Y)$. With this in mind, it is easy to see that if k insertions are permitted, the number of possible strings U' is #(Possible U'), where,

$$\#(\text{Possible U'}) = \binom{|U|+k}{k}.$$

For each element U' which represents k insertions, any corresponding element Y' must contain exactly (|U|-|Y|+k) deletions, and these must be chosen from among the symbols of U. This can be done in #(Possible Y') ways, where,

$$\#(\text{Possible Y'}) = \binom{|U|}{|U|-|Y|+k}.$$

The product of these two quantities yields the result for every value of k. The lemma follows by summing the above product for all permissible values of k. ♦♦♦

Theorem I:

Let Pr[Y|U] be defined as follows :

$$Pr[Y|U] = \sum_{z=Max(0,M-N)}^{M} \frac{G(z)\cdot (N!\ z!)}{((N+z)!)} \sum_{U'} \sum_{Y'} \prod_{i=1}^{N+z} p(y_i'|u_i').$$

where,

(a) y_i' and u_i' are the individual symbols of Y' and U' respectively,

(b) $p(y_i'|u_i')$ is defined as $Q(y_i')$ if u_i' is ξ, and ,

(c) $p(y_i'|u_i')$ is defined as $S(y_i'|u_i')$ if u_i' is not ξ.

Then the above definition is both functionally complete and consistent.

Proof of Theorem I:

The functional completeness is clear since the definition of $Pr[Y|U]$ involves computing the product of the probabilities of the individual elements of *every single pair* in $\Gamma(U,Y)$. Thus, for every element (U', Y') the product of the individual probabilities is its contribution to $Pr[Y|U]$.

The consistency of the definition is, however, a little more difficult to see. It involves proving that the value of the infinite summation :

$$\sum_{Y \in A^*} Pr[Y|U]$$

is exactly unity. The fact that this result is true is definitely not obvious.

To prove the result, we first go through the mechanics of explicitly writing down the expression for the probability $Pr[Y|U]$. This is done by exhaustively summing up all the probability contributions of the various ways by which U could have been transformed to Y. Notice that this information is contained in $\Gamma(U,Y)$. Let τ be the summation of this quantity over all the possible values of Y. We intend to prove that τ is exactly unity.

We shall prove this by successively considering the case when $Z = z$. Consider the set of strings that can be obtained by having z insertions occur in the mutation. Indeed, since inserted symbols cannot be subsequently deleted, whenever z insertions occur, the set of all strings that can be obtained is the union of the sets A^j, where j takes the value from z to N+z. Let ${}_zH_{N+z}$ be this set and let τ_z be :

$$\tau_z = \sum_{{}_zH_{N+z}} \frac{(N!\ z!)}{((N+z)!)} Pr[Y|U\ ;\ Z=z].$$

Indeed, if with no loss of generality we assume that z can be any non-negative integer, we have :

$$\tau = \sum_{Z=0}^{\infty} G(z)\ \tau_z.$$

Notice now that for each $Y \in {}_zH_{N+z}$ the number of insertions must be bounded by $Max(0, |Y|-N)$ and $|Y|$. Thus,

$$\sum_{Y \in {}_zH_{N+z}} Pr[Y|U\ ;\ Z=z] = \sum_{Y \in {}_zH_{N+z}} \sum_{U'} \sum_{Y'} \prod_{i=1}^{N+z} p(y_i'|u_i'),$$

$$= \sum_{Y \in {}_zH_{N+z}} \sum_{(U',Y') \in (\Gamma_{U,Y})} \prod_{i=1}^{N+z} p(y_i'|u_i')). \qquad (A.1)$$

But for each U', the last product is over all the letters of the finite alphabet as in (8). Hence,

(i) $p(y_i'|u_i')$ is $Q(y_i')$ if u_i' is ξ, and, summed over all y_i' this quantity is unity, and,

(ii) $p(y_i'|u_i')$ is interpreted as $S(y_i'|u_i')$ if u_i' is not ξ, and, summed over all u_i' *this* is unity.

Hence, for every U' this sums to unity, and since, for each z, there are $\binom{N+z}{z}$ elements of U', (A.1) has the value $\frac{(N+z)!}{N!\ z!}$. Hence, τ_z has the value unity. Consequently τ itself is unity because G is a valid distribution in itself. Thus the result. ♦♦♦

Theorem II:

The quantity Pr [Y|X] can be obtained from Pr [Y|U] by :

$$\Pr[Y|X] = \sum_{j=1}^{J} \Pr[Y|U = T_j(X)] \cdot F(j))$$

Proof of Theorem II:

U could have been obtained from X by any of the j expansions of each symbol. Note that the probability that the j^{th} expansion is used is F(j). Thus, to evaluate Pr [Y|X], we have to enumerate all the various mutually exclusive ways by which X could have been transformed to U and then sum these quantities using the law of total probability. Hence the result. ♦♦♦

Theorem III:

To edit U_r, the prefix of U of length r, to Y_q, the prefix of Y of length q, the set of feasible triples is given by $\{(i, r-q+i, q-i) \mid \text{Max}[0, q-r] \le i \le q\}$.

Proof of Theorem III:

Consider the constraints imposed on feasible values of i, e and s. Since we are interested in the editing of U_r to Y_q we have to consider only those triples (i, e, s) in which i+s = r and e+s = q. But the number of insertions can take any value from Max[0,q-r] to q. For every value of i in this range, the feasible triple (i,e,s) must have exactly q-i substitutions.

Similarly, since the sum of the number of substitutions and the number of deletions is r, the triple (i,e,s) must have exactly r-q+i deletions. Hence the result. ♦♦♦

Theorem IV:

Let W(i,e,s) be the quantity defined as in (13) for any two strings U and Y. Then, for all nonnegative i,e and s,

$$W(i,e,s) = W(i-1,e,s)\cdot p(y_{i+s}|\xi) + W(i,e-1,s)\cdot p(\lambda|u_{e+s}) + W(i,e,s-1)\cdot p(y_{i+s}|u_{e+s})$$

where p(b|a) is defined as in (8).

Proof of Theorem IV:

The proof of the result is divided into three main divisions, written as Cases(a)-(c) respectively.

Case (a) : Any two of the three variables i, e and s are zero.

Case (b) : Any one of the three variables i, e and s is zero.

Case (c) : None of the variables i, e and s are zero.

The most involved of these cases is Case (c). The first two cases are merely one and two parameter cases respectively of Case (c). To avoid repetition, we shall prove only Case (c). Thus, for the rest of the proof, we encounter only strictly positive values for the variables, i, e and s. Let r=e+s, q=i+s, $U_r=u_1 \ldots u_r$, and $Y_q=y_1 \ldots y_q$. Then, by definition,

$$W(i,e,s) = \sum_{(U_r',Y_q')} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}'), \tag{A.2}$$

where (U_r',Y_q') is the arbitrary element of the set $\Gamma_{i,e,s}$ with u_{rj}' and y_{qj}' as the jth symbols of U_r' and Y_q' respectively. In the above expression and in all the expressions used in this proof, we shall assume that p(b|a) is defined by (8).

Let the lengths of the strings U_r' and Y_q' in the arbitrary element of $\Gamma_{i,e,s}$ be L. Then the last symbols of U_r' and Y_q' are u_{rL}' and y_{qL}' respectively. We partition the set $\Gamma_{i,e,s}$ into three mutually exclusive and exhaustive subsets.

$$\Gamma^1_{i,e,s} = \{ (U_r',Y_q') \mid (U_r',Y_q') \in \Gamma_{i,e,s}, u_{rL}' = u_r, y_{qL}' = y_q \} \tag{A.3}$$

$$\Gamma^2_{i,e,s} = \{ (U_r',Y_q') \mid (U_r',Y_q') \in \Gamma_{i,e,s}, u_{rL}' = u_r, y_{qL}' = \lambda \} \tag{A.4}$$

$$\Gamma^3_{i,e,s} = \{ (U_r',Y_q') \mid (U_r',Y_q') \in \Gamma_{i,e,s}, u_{rL}' = \xi, y_{qL}' = y_q \} \tag{A.5}$$

By their definitions, we see that the above three sets are mutually exclusive. Further, since u_{rL}' and y_{qL}' cannot be ξ and λ respectively simultaneously, every pair in $\Gamma_{i,e,s}$ must be in one of the above sets. Hence these three sets partition $\Gamma_{i,e,s}$. Rewriting (A.2) we obtain,

$$W(i,e,s) = \Big[\sum_{(U_r',Y_q') \in (\Gamma^1_{i,e,s})} S' \Big] + \Big[\sum_{(U_r',Y_q') \in (\Gamma^2_{i,e,s})} S' \Big] + \Big[\sum_{(U_r',Y_q') \in (\Gamma^3_{i,e,s})} S' \Big] \tag{A.6}$$

$$\text{where, } S' = \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}'). \tag{A.7}$$

Consider each term in (A.6) individually. In every pair in $\Gamma^1_{i,e,s}$, $u_{rL}' = u_r$ and $y_{qL}' = y_q$. Hence,

$$\sum_{(U_r',Y_q') \in (\Gamma^1_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}')$$

$$= \left[\sum_{(U_r',Y_q')\in(\Gamma^1_{i,e,s})} \prod_{j=1}^{|U_r'|-1} p(y_{qj}'|u_{rj}') \right] \cdot p(y_q|u_r) \qquad (A.8)$$

For every element in $\Gamma^1_{i,e,s}$ there is a unique element in $\Gamma_{i,e,s-1}$ and vice versa. Hence, the first term in the above expression is exactly W(i,e,s-1). Since r=e+s and q=i+s,

$$\sum_{(U_r',Y_q')\in(\Gamma^1_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}') = W(i,e,s-1)\cdot p(y_{i+s}|u_{e+s}) \qquad (A.9)$$

Consider the second term in (A.6). In every pair in $\Gamma^2_{i,e,s}$, $u_{rL}' = u_r$ and $y_{qL}' = \lambda$. Hence,

$$\sum_{(U_r',Y_q')\in(\Gamma^2_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}')$$

$$= \left[\sum_{(U_r',Y_q')\in(\Gamma^2_{i,e,s})} \prod_{j=1}^{|U_r'|-1} p(y_{qj}'|u_{rj}') \right] \cdot p(\lambda|u_r) \qquad (A.10)$$

For every element in $\Gamma^2_{i,e,s}$ there is a unique element in $\Gamma_{i,e-1,s}$ and vice versa. Hence, the first term in the above expression is exactly W(i,e-1,s). Thus,

$$\sum_{(U_r',Y_q')\in(\Gamma^2_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}') = W(i,e-1,s)\cdot p(\lambda|u_{e+s}) \qquad (A.11)$$

Consider the third term in (A.6). In every pair in $\Gamma^3_{i,e,s}$, $u_{rL}' = \xi$ and $y_{qL}' = y_q$. Hence,

$$\sum_{(U_r',Y_q')\in(\Gamma^3_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}')$$

$$= \left[\sum_{(U_r',Y_q')\in(\Gamma^3_{i,e,s})} \prod_{j=1}^{|U_r'|-1} p(y_{qj}'|u_{rj}') \right] \cdot p(y_q|\xi) \qquad (A.12)$$

For every element in $\Gamma^3_{i,e,s}$ there is a unique element in $\Gamma_{i-1,e,s}$ and vice versa. Hence, the first term in the above expression is exactly W(i-1,e,s). As in the above cases,

$$\sum_{(U_r',Y_q')\in(\Gamma^3_{i,e,s})} \prod_{j=1}^{|U_r'|} p(y_{qj}'|u_{rj}') = W(i-1,e,s)\cdot p(y_{i+s}|\xi) \qquad (A.13)$$

Re-substituting (A.9), (A.11) and (A.13) into (A.6) proves the result. ♦♦♦

Theorem V:

If $h(i)= G(i)\cdot\frac{N!\ i!}{(N+i)!}$, the quantity $Pr[Y|U]$ can be evaluated from the array W(i,e,s) as :

$$Pr[Y|U] = \sum_{i=Max(0,M-N)}^{M} h(i).W(i, N-M+i, M-i).$$

Proof of Theorem V:

Consider the constraints imposed by (14) on i, e and s. Since we are interested in the editing of the entire string U to the entire string Y, we have to consider only those elements of W(i,e,s) in which i+s=M and e+s=N. But the number of insertions can take any value from Max[O,M-N] to M. For every value of i in this range, the term in W(i,e,s) that will give a contribution to $Pr[Y|U]$ must have exactly M-i substitutions. Since the sum of the number of substitutions and the number of deletions is N, the term in W(i,e,s) that will give a contribution to $Pr[Y|U]$ must have exactly N-M+i deletions. Hence we are only interested in the terms of the form W(i, N-M+i, M-i) with i varying from Max[0, M-N] to M. The result is proved by noting that the weighting factor h(i) is only dependent on i, the number of insertions. ♦♦♦

Theorem VI:

The distributions of the scheme, F, G, Q, and S, are unidentifiable, and furthermore the parameters of the distributions can never be consistently or meaningfully estimated even if a cryptanalyst has access to infinite samples of chosen plaintext and the corresponding ciphertext.

Proof of Theorem VI:

Suppose that a cryptanalyst has access to infinite samples of chosen plaintext and the corresponding ciphertext. Consider the encryption of a plaintext block $X \in H$. Since the cryptanalyst operating in a chosen plaintext mode has access to encryptions of X, he can determine whether a *particular* string Y is one of the possible encryptions of X. However, in any particular encryption of X into Y, he will not have access to the actual number of substitutions, insertions and deletions that took place in *Transform* so as to have caused the encryption. Neither will he be able to know which

element of $\Gamma(X,Y)$ was responsible for the mutating process. The process *Transform* is thus, by definition, *unobservable* from a black-box perspective[14] [Kail80].

To be more specific, let us suppose that |X| is N and that |Y| is M. First of all, in the process of mutating X, it could have first been transformed into a string U through any of the J expansions supported by T. Subsequent to this, Y could have been obtained by deleting all of U and inserting all of Y. Alternatively, Y could have been obtained by inserting k symbols of Y, where $\text{Max}[0, M-|U|] \leq k \leq M$, and deleting and substituting the remaining symbols of U such that |Y|=M. (Indeed, even for a particular value of k, there are $\frac{(|U|+k)!}{(k!\ (|Y|-k)!\ (|U|-|Y|+k)\ !)}$ possible ways by which U could have been mutated to Y.) Thus by merely processing X and Y the cryptanalyst has no (information-theoretic) possibility of deducing which of these ways was responsible for the mutations. Observe, that neither outcomes of the random variable Z nor of any algebraic function of Z is obtainable by studying the plaintext and the corresponding ciphertext. Thus it is (information theoretically) impossible for him to estimate the parameters of the distribution of Z.[15] The same is true for the distribution S.

To prove the unidentifiability of the distributions, consider the trivial case of a key $\mathbb{k}_1$, in which the symbol 'a' is always deleted, one insertion is always performed and whenever a symbol is inserted the symbol 'b' is inserted with probability 0.5 and the symbol 'c' is inserted with probability 0.5. Then, a straightforward evaluation of total probabilities using Theorem I yields :

$\Pr[Y=\text{"b"} \mid U=\text{"a"}, \mathbb{k} = \mathbb{k}_1] = 0.5$, and,

$\Pr[Y=\text{"c"} \mid U=\text{"a"}, \mathbb{k} = \mathbb{k}_1] = 0.5$.

Consider now the alternate case of a key $\mathbb{k}_2$, in which the symbol 'a' is never deleted but always substituted into 'b' with probability 0.5 and into 'c' with probability 0.5, and additionally, the key never permits an insertion. A similar evaluation of the corresponding probabilities yields :

$\Pr[Y=\text{"b"} \mid U=\text{"a"}, \mathbb{k} = \mathbb{k}_2] = 0.5$, and,

$\Pr[Y=\text{"c"} \mid U=\text{"a"}, \mathbb{k} = \mathbb{k}_2] = 0.5$.

Processing the string U="a" through *Transform* would in both cases yield the same output string, namely, Y='b'. But, even in this trivial case, a cryptanalyst would not be able to distinguish between $\mathbb{k}_1$ and $\mathbb{k}_2$ because the distributions are unidentifiable -- the total probability of obtaining the ouput string is the same in both these cases. This is a consequence of the fact that the operations causing the mutations are unobservable.

It is easy to extend these arguments for more complicated alphabets and distributions. Furthermore in this case, if the set of possible keys is $\{\mathbb{k}_1, \mathbb{k}_2\}$, and if these keys can be chosen with equal *a priori* likelihood, then:

---

[14] As mentioned earlier, the concept of *observability* utilized here is formally distinct from the one traditionally used in the literature [Kail80], but is in the same spirit of the traditional black-box definition.

[15] This is true for each output of the *Transform* module, and thus the same result holds for the output of EncryptBlock, i.e. the overall encipherment process.

$Pr[\mathbb{K} = \mathbb{K}_1 \mid Y="b"] = Pr[\mathbb{K} = \mathbb{K}_2 \mid Y="b"] = Pr[\mathbb{K} = \mathbb{K}_1] = Pr[\mathbb{K} = \mathbb{K}_2] = 0.5$, and,

$Pr[\mathbb{K} = \mathbb{K}_1 \mid Y="c"] = Pr[\mathbb{K} = \mathbb{K}_2 \mid Y="c"] = Pr[\mathbb{K} = \mathbb{K}_1] = Pr[\mathbb{K} = \mathbb{K}_2] = 0.5$.

This implies that the *a posteriori* probability of computing the key given the ciphertext is exactly the *a priori* probability - reminiscent of the concept of Shannon's perfect secrecy.

Augmenting this situation is the fact that the mutation of a symbol $a \in A$ into an element of Φ_a using F could have also occurred due to the effect of a sequence of random insertions and substitutions based on the distribution S. This information about whether expansions or insertions occurred in a particular transformation is also not obtainable by merely studying the plaintext and the corresponding ciphertext further justifying the unidentifiability of the distributions. Thus, even if the cardinality and details of Φ are available, the distribution F itself cannot be estimated because neither j (the index of the expansion used as defined in (3)) nor of any algebraic function of j is obtainable by merely studying the plaintext and the corresponding ciphertext. ♦♦♦

APPENDIX B: EFFICIENT EVALUATION OF PROBABILITIES PR[Y|U]

In the procedure *EvaluateProbabilities*, to compute Pr[Y|U] we made use of the fact that though the latter index itself does not seem to have any recursive properties, the index W (.,.,.), which is closely related to it had the interesting properties stated in Theorem IV. As is quite obvious, the latter process requires cubic time and space (in terms of the lengths of the processed blocks or sub-blocks) respectively. To do this we have taken advantage of the following fact : For a particular value of i, in order to compute W(i,e,s) for all permissible values of e and s, it is sufficient to store only the values of W (i-1,e,s) for all the corresponding permissible values of e and s. This is true from Theorem IV, since the computation of any one quantity requires *at most* three previously computed quantities, namely W(i-1,e,s), W(i,e-1,s) and W(i,e,s-1).

We shall now present a more efficient technique to compute Pr[Y|U]. Consider a three dimensional trellis, in which the coordinates are i, e and s. The strategy which we utilize involves successively evaluating the array W in planes parallel to the plane i = 0. Thus we would need four arrays namely, (a) Wie: the plane in which s = 0, (b) Wis: the plane in which e = 0, (c) Wes0 : the plane parallel to i = 0, maintained for the previous value of i, and, (d) Wes1 : the plane parallel to i = 1 maintained for the current value of i.

The technique for computing the probabilities would then evaluate these arrays in a systematic manner. Initially the quantities associated with the individual axes are evaluated. The i-e and i-s planes are then computed and stored in the arrays Wie and Wis respectively. The trellis is then traced *plane by plane* - always retaining only the current plane parallel to the plane i = 0. Thus, prior to updating Wes0, its pertinent component required to compute Pr[Y|U] is used to update the latter.

Suppose that instead of storing all the four arrays Wie, Wis, Wes0 and Wes1 we maintained only the values corresponding to the e and s axes and the previous plane perpendicular to the i-axis. It can be shown that the entire current plane perpendicular to the i-axis can be computed. The result follows from the following facts which cover the cases of the lines parallel to the e and s axes, and the case of computing the index for an internal point in which i, e, s > 0 :

(i) $W(i,e,0) := W(i-1,e,0)\cdot Q(y_i) + W(i,e-1,0)\cdot S(\lambda|u_e)$

(ii) $W(i,0,s) := W(i-1,0,s)\cdot Q(y_{i+s}) + W(i,0,s-1)\cdot S(y_{i+s}|u_s)$

(iii) $W(i,e,s) := W(i-1,e,s)\cdot Q(y_{i+s})+W(i,e-1,s)\cdot S(\lambda|u_{e+s})+W(i,e,s-1)\cdot S(y_{i+s}|u_{e+s})$

From (i) - (ii) we observe that to compute the axes for a current plane (perpendicular to the i-axis) values that are required are only the values that have been already stored in the same location, and the value that is computed for the previous location on the axis. Similarly, from (iii) we note that for any interior point, (i.e., for the case when i, e, s are positive), the only values that are required are the previous values for the same point in which the value of i is decremented by unity, and the values corresponding to the locations that are already computed on the same value of i, but for the values of e and s which are also decremented by unity. Thus, in the plane-tracing phase, only the rows

corresponding to the current and previous values of the loop variable "i" are required, and using these, both the axes of the current plane can be computed and also the values for an internal node can be computed.

As a consequence of the above result we notice that the computation of any element on the Wes1 plane does not require the storage of the values of the entire array Wes0. Indeed, because the values of the "i-1" plane persist in the Wes array until they are replaced by values of the "i" plane, the entire calculation may be done with a single quadratic array, say, Wes(.,.). The formal procedure to efficiently compute Pr[Y|U] follows.

Procedure EfficientlyEvaluateProbabilities
Input & Output : Same as in **Procedure EvaluateProbabilities**
Method :

Wes(0,0) := 1
For s := 1 to Min[M,N] **Do** /* initialize s-axis */
  Wes(0,s) := Wes(0,s-1)·S($y_s|u_s$)
For e := 1 to N **Do** /* initialize e-axis */
  Wes(e,0) := Wes(e-1,0)·S($\lambda|u_e$)
For e := 1 to N **Do** /* Compute Wes-plane
  For s := 1 to Min[M,N-e] **Do** /* parallel to i=0 */
    Wes(e,s) :=Wes(e-1,s)·S($\lambda|u_{e+s}$) + Wes(e,s-1)·S($y_s|u_{e+s}$)]
If N ≥ M **Then** /* Initialize Pr[Y|U] */
  Pr[Y|U] := G(0)·Wes(N-M, M)
Else
  Pr[Y|U] := 0
For i := 1 to M **Do** /* Trace planes parallel
Begin /* to i=0 */
  Wes(0,0) := Wes(0,0)·Q(y_i)
  For e := 1 to N-M+i **Do** /*Line parallel to s-axis*/
    Wes (e,0) := Wes(e,0)·Q(y_i) + Wes(e-1,0)·S($\lambda|u_e$)
  For s := 1 to Min [N, M-i] **Do** /*Line parallel to e-axis*/
    Wes(0,s) := Wes(0,s)·Q(y_{i+s}) + Wes(0,s-1)·S($y_{i+s}|u_{e+s}$)
  For e := 1 to N-M+i **Do** /*Body of trellis */
    For s := 1 to Min [N-e,M-i] **Do**
      Wes(e,s) := Wes(e,s-1)·S($y_{i+s}|u_{e+s}$) + Wes(e-1,s)·S($\lambda|u_{e+s}$) + Wes(e,s)·Q(y_{i+s})
  If N-M+i ≥ 0 **Then**

$$Pr[Y|U] := Pr[Y|U] + G(i) \cdot \frac{N!\ i!}{(N+i)!} \cdot Wes(N-M+i, M-i)$$

End
END Procedure EfficientlyEvaluateProbabilities

APPENDIX C: SCHEMATIC OF THE *TRANSFORM* PROCESS

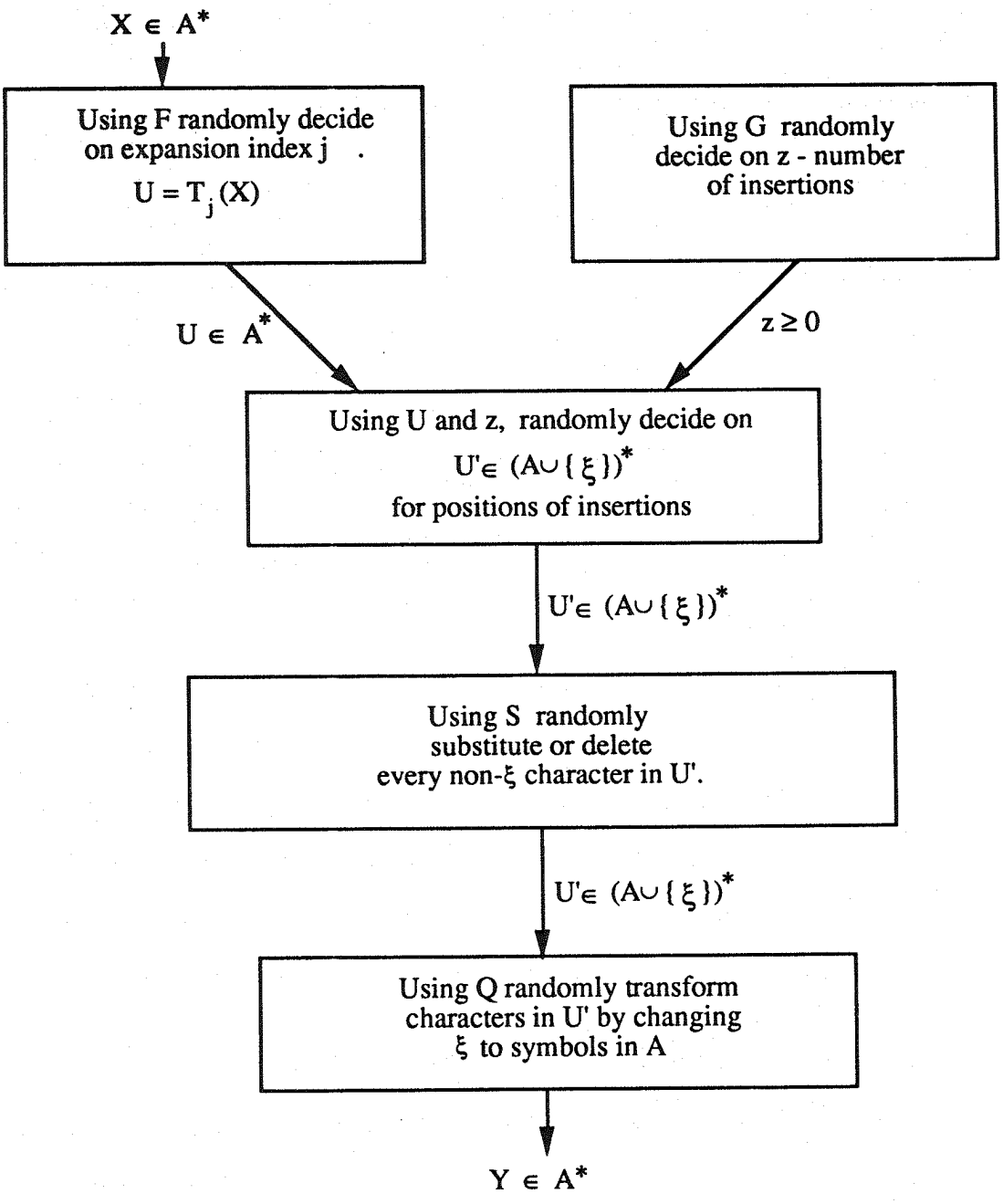

Figure I: The *Transform* process $\mathfrak{T}$, which mutates a string $X \in A^*$ using random expansions, substitutions, insertions and deletions. Distributions F, G, Q and S specify the secret key $\mathfrak{K}$.

REFERENCES

[Blum84] M. Blum, S. Goldwasser. "An efficient probabilistic public-key encryption scheme which hides all partial information". *Advances in Cryptology - Proceedings of Crypto'84*, Springer-Verlag (1985), 289-299.

[Bras88] G. Brassard. *Modern Cryptology*. Springer-Verlag, 1988.

[Davi89] D.W. Davies, W.L. Price. *Security for Computer Networks*, 2nd edition. Wiley & Sons, 1989.

[Devr86] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[Diff79] W. Diffie, M. Hellman. "Privacy and authentication: an introduction to cryptography". *Proc. of the IEEE*, **67** (Mar. 1979), 397-427.

[Duda73] R. O. Duda, P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, 1973.

[ElGa85] T. ElGamal. "A public key cryptosystem and signature scheme based on discrete logarithm". *IEEE Trans. Info. Theory*, **IT-31** (1985), 469-472.

[Ever81] B.S. Everitt, D.J. Hand. *Finite Mixture Distributions*, Chapman and Hall. London, 1981.

[FIPS77] National Bureau of Standards. U.S. Dept. of Commerce, FIPS PUB 46, *Data Encryption Standard*. Washington, D.C. (Jan. 15, 1977).

[Fuku72] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.

[Gold82] S. Goldwasser, S. Micali. "Probabilistic encryption and how to play mental poker keeping secret all partial information". *Proc. 14th ACM STOC* (1982), 365-377.

[Gold84] S. Goldwasser, S. Micali. "Probabilistic encryption". *J. Computer and System Sciences* **28** (1984), 270-299.

[Kail80] T. Kailath. *Linear Systems*. Prentice Hall, Englewood Cliffs, 1980.

[MacW77] F.J. MacWilliams, N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[McEl78] R. McEliece. "A public-key cryptosystem based on algebraic coding theory". *Pasadena JPL report 42-22* (Jan-Feb 1978), 114-116.

[Meye82] C.H. Meyer, S.M. Matyas. *Cryptography: A New Dimension in Computer Data Security*. Wiley & Sons, 1982.

[Papo65] A. Papoulis. *Probability, Random Variables and Stochastic Processes*.McGraw Hill Book Company, 1965.

[Prep85] F.P.Preparata, M.I. Shamos. *Computational Geometry : An Introduction*. Springer-Verlag, 1985.

[Rive82] R. Rivest, A. Sherman. "Randomized encryption techniques". *Advances in Cryptology - Proceedings of Crypto 82*, Plenum Press (1983), 145-163.

[Sank83] D. Sankoff, J. B. Kruskal.*Time wraps, string edits, and macromolecules : Theory and Practice of Sequence Comparsion*. Addison-Wesley, (1983).

[Schn94a] B. Schneier. *Applied Cryptography*. Wiley & Sons, 1994.

[Schn94b] B. Schneier. "The Cambridge algorithms workshop". *Dr. Dobb's Journal*. (April 1994), 18-24.

[Shan49] C.E. Shannon. "Communication theory of secrecy systems". Bell System Technical *Journal*, **28** (Oct. 1949), 659-715.