Multistage Algorithm for Limited One-Way Functions

ABSTRACT

This work describes a multi-stage extension to a previously published algorithm for implementation of a limited one-way function. This algorithm is intended to implement a delaying function with statistically controllable parameters. This cryptographic technique is applicable to Key Escrow Systems for the purpose of implementing a delay function to limit the rate of withdrawal of key material from such an escrow system.

Key Words: Key Escrow, Limited One-Way Function

INTRODUCTION

It is a central theme of modern cryptography to provide security by imposing an intractable amount of computational work on a would be eavesdropper. Cryptographic applications typically rely on strongly one-way functions in order to provide large computational barriers to unlocking the information which they are intended to protect. Only with the aid of special information built into the structure of the encryption algorithm, can this data be recovered. In general, this special information is referred to as cryptographic "keys." Thus, anyone who has the keys quite easily recovers the data, whereas anyone who does not posses the key information faces an intractable task of inverting the function. It is necessary, however, to design systems where individuals, other than the primary key holders, under certain well-defined circumstances, have the ability to gain access to that information.

In particular, this notion gives rise to the idea of Key Escowing. In the most general sense, a key escrow is simply a depository for keying information. If key information is lost or needs to be recovered for some other reason, then it may be withdrawn from the key escrow. Because of the very high value of the escrowed key data, it is necessary to use extreme care in protecting this data. The objective of a key escrow system is to catalogue keying information for later withdrawal while precluding withdrawal for non-legitimate reasons. Therefore, the whole range of security and cryptographic tools are applicable to the development of key escrow systems.

This work proposes an extension to a previously published technique that may be used to provide inherent, algorithmically imposed limitations to deter abuse of a system such as may be used for the purpose of Key Escrow. This extension provides computational advantage over the originally published technique and could serve as the basis for a practical implementation.

BACKGROUND

In a previous work [1], the concept of limited one-way functions was introduced. The term was applied to characterize functions which are provably asymmetric in terms of the work required to compute the value of the function versus the amount of work required to compute the inverse of the function. A limited one-way function is, however, reversible in a measurable (tractable) amount of work. In particular, it is desired to consider functions where the work required can be determined within some known parameters. A system was described in that previous work that permitted the work to be controlled within well defined statistical limitations which can be established as parameters of the system.

The implementation of Key Escrow systems is considered important by the government as a means whereby the general populace may take advantage of the ready availability of strong cryptographic technology while, perhaps, maintaining the current capability of law enforcement to perform legal wiretaps for legitimate criminal investigation purposes. Yet, the actual number of legal wiretaps is quite low, of the order of a thousand a year [2].

Key Escrow systems can be characterized, in part, by the asymmetry of the deposit and withdrawal requirements. Normally, many keys are deposited, however keys are required for recovery only very seldom. Thus, the deposit rate greatly exceeds the required withdrawal rate. Because this depository keeps a copy of all of the keying material, then its economic value is inherently equal to the value of all of the information protected by all of the keys deposited. A key escrow system thus represents a highly valued target for exploitation. Barlow [3] raised the issue of whether key depositories themselves would become the target of criminal or terrorist organizations. The value that it represents, indeed, may serve as a temptation for abuse, even by otherwise authorized individuals who have normal legitimate reasons for access. It is the potential for abuse of a centralized Key Escrow system by those who have access that stands as the greatest deterrent to their implementation. This is especially true in international applications since many countries do not have the tradition of the protection of individual privacy and human rights that we enjoy in the U.S. Even in this country, due care must be taken to protect the rights of individuals. As suggested in the original work, it may be considered advantageous to constrain the possible rate of withdrawals to match legitimate requirements. This would hinder the possibility for widespread or rampant abuse. Thus, the concept of limited one-way functions was introduced in the original work to address the rate of withdrawal problem in an algorithmic manner.

The original algorithm, described in the authors' previous work [1], involved an originator, Alice, creating a set of N trapdoor functions each paired with a corresponding token. These were then to be transmitted to Bob, who in turn, would select one of these pairs at random, add randomization information to the corresponding token, encrypt using the randomly selected trapdoor function, and then return the result to Alice. Alice then

uses the retained trapdoor information to discover which choice Bob made. Hence, we have Alice forming a set of encryption key/token pairs such as:

$$P = \{ (T_1, E_1), (T_2, E_2), (T_3, E_3), (T_4, E_4), \ldots , (T_N, E_N) \}, \quad (1)$$

from which Bob chooses at random the *k*th pair (T_k, E_k). Bob takes the token, T_k, and concatenates randomization information *R*. He then uses the encryption key, E_k, to encrypt the combination. Therefore, Bob forms a cryptogram *C* such that:

$$C = E_k (T_k \,\&\&\, R), \quad (2)$$

where the operator *&&* denotes the concatenation operation. T_k is assumed to be an *n*-bit quantity, where $n = log_2(N)$. *R* is assumed to take on ℝ discrete values and is represented by an *r*-bit number

This basic algorithm can be viewed as an extension to an algorithm originally proposed by Merkle [4]. The computational advantage thus achieved over an eavesdropper in this basic algorithm is dependent on the amount of randomization embedded in the problem. To discover Bob's choice the eavesdropper, Eve has the choice of breaking the *N* trapdoor problems that Alice originally created, or forming *N**ℝ cryptograms of the form that Bob returned. As long as the work required to break the underlying cryptosystems greatly exceeds creating these *N**ℝ cryptograms, the eavesdropper is forced to solve the problem by random search, assuming that there is no structure in the results space which can be exploited. The required work is determined by solving a large number of small problems. Whereas the computational complexity of difficult problems typically do not have well defined bounds, especially lower bounds, it is possible to get tighter results on very simple operations. By forcing the calculation of a large number of simple problems, all of which whose results appear to be randomly related, we may make use of the Law of Large Numbers [5] to statistically control the work required to perform the average withdrawal.

It is this concept of forcing the eavesdropper in through a controlled front-door that serves as the basis for providing a withdrawal capacity on a Key Escrow system while requiring a measurable amount of work to do so. Because the algorithm is built directly into the storage facility, the rate of withdrawals is then limited by its capacity to perform the withdrawal algorithm. This approach is elegant, in that it solves the rate of withdrawal problem in an algorithmic manner.

MULTISTAGE EXTENSION

It is possible to increase the apparent uncertainty in the problem without growing the natural size of the computational engine by use of a technique analogous to cipher chaining. As previously described in the basic algorithm, the initiator, Alice, forms a set

consisting of N pairs of tokens and encryption keys. Also, as before, the recipient, Bob, selects one of these pairs at random and then calculates a cryptogram of the form:

$$C_1 = E_{P_1} (T_{P_1} \&\& R_1 \&\& S), \quad (3)$$

where T_{P_1} is the selected token; E_{P_1} is the corresponding encryption key; $P_1 \in \{1, 2, ..., N\}$ is the index of the choice Bob made from the set P, R_1 is randomization information; and S is information added for signature purposes to permit valid decodings to be distinguished from invalid decodings.

To achieve his computational advantage over the eavesdropper, Bob relies on the uncertainty of his choice of puzzles, as well as randomization information that is added to the problem. Bob can increase this advantage by recursively making additional choices from the originally transmitted puzzle set. It is possible to achieve significant improvement by taking the message from this second choice and concatenating the results from the encryption of the first choice, encrypting this combination with the key from the second choice. Thus Bob chooses, again at random, a new pair (T_{P_2}, E_{P_2}) from the set P. Again, Bob concatenates signature and additional randomization information. This result is subsequently encrypted using the second encryption key. Thus we have:

$$C_2 = E_{P_2} (T_{P_2} \&\& R_2 \&\& S). \quad (4)$$

He then proceeds to take the result from his first selection, the cryptogram C_1, applies the newly selected encryption function, and concatenates this with the second cryptogram. This result is then encrypted using the second encryption key. Thus, we have for the output, O_2, of this stage:

$$O_2 = C_2 \&\& E_{P_2}(C_1) = E_{P_2} (T_{P_2} \&\& R_2 \&\& S) \&\& E_{P_2} (E_{P_1} (T_{P_2} \&\& R_2 \&\& S)) = C_2 \&\& C_{2a}, \quad (5)$$

where C_{2a} is used to denote the term $E_{P_2} (E_{P_1} (T_{P_2} \&\& R_2 \&\& S))$.

It therefore requires two encryption operations to encrypt the information at stage two. The resulting number of bits of output information grows by size of the cryptogram C_{2a}. For two stages only, Bob's response to Alice is to transmit O_2. Thus, the work required to discover both of Bob's choices by random search seems to grow from being $Avg(N*\mathbb{R})$ to $Avg(N^2*\mathbb{R}^2)$, where Avg denotes the average computational complexity. This system is illustrated in the block diagram shown in Figure 1.

At the receive end, Alice recovers Bob's selection by undoing the work that Bob performed. Alice does posses unique information. Alice has the trapdoor key information which allows Alice to quickly reverse the encryption that Bob performed. Thus, Alice has the set, D, of decryption keys corresponding to the transmitted (hence public) keys.

$$D = \{D_1, D_2, D_3, D_4, \ldots, D_N\}. \quad (6)$$

Alice tries keys one at a time, until a match is made on the second message. This enables Alice to recover the cryptogram from the first choice.

$$D_{P_2}(C_2) = D_{P_2}(E_{P_2}(T_{P_2} \,\&\&\, R_2 \,\&\&\, S)) = T_{P_2} \,\&\&\, R_2 \,\&\&\, S. \quad (7)$$

Alice recognizes the successful decode because of the signature information *S*. Consequently, there is some finite measurable probability of a spurious decode. That occurs when a incorrect choice of the decode key accidentally maps to a pattern that matches the signature.

Alice uses the discovered choice of D_{P_2} to unroll the second term, C_{2a}. Thus, Alice gets the intermediate result:

$$D_{P_2}(C_{2a}) = D_{P_2}(E_{P_2}(C_1)) = C_1. \quad (8)$$

Alice then, once again, selects keys one at a time until the first choice is recovered,

$$D_{P_1}(C_1) = D_{P_1}(E_{P_1}(T_{P_1} \,\&\&\, R_1 \,\&\&\, S)) = T_{P_1} \,\&\&\, R_1 \,\&\&\, S. \quad (9)$$

Finally, the result is determined by the successful recognition of the signature *S*.

The work required by Alice to do this decode operation is therefore still *Avg*(*N*). This process can be extended further. If Bob makes *k* choices then the work required by Alice grows to $Avg(k^2*N)$ while the work required of the eavesdropper grows to $Avg\ k(N*\mathbb{R})^k$.

We can express the general case of a *k*-stage version of the algorithm with the recursive relationship:

$$O_k = C_k \,\&\&\, E_{P_k}(O_{k-1}), \quad (10)$$

where it should be recognized that the encryption function must be applied *k-1* times in order to encrypt all of the information associated with the term O_{k-1}. A block diagram of this algorithm is illustrated in Figure 1. As can be seen from this figure, the output space of the final stage of the algorithm grows as 2^{kl}, where *l* is the number of bits in C_i. It is only the final result, O_k, that is passed back to Alice. Therefore, neither Alice nor the eavesdropper see any of the intermittent results.

Once Bob transmits O_k back to Alice, it becomes Alice's task to reverse Bob's selection process. As before, Alice tries keys randomly to unravel the encryption to get T_k. She performs the operation:

$$D_{P_k}(C_k) = D_{P_k}(E_{P_k}(T_{P_k} \,\&\&\, R_k \,\&\&\, S)) = T_{P_k} \,\&\&\, R_k \,\&\&\, S. \quad (11)$$

$$D_{P_k}(O_{k-1}) = D_{P_k}(E_{P_k}(O_{k-1})) = D_{P_k}(E_{P_k}((C_{k-1} \,\&\&\, E_{P_{k-1}}(O_{k-2})). \quad (12)$$

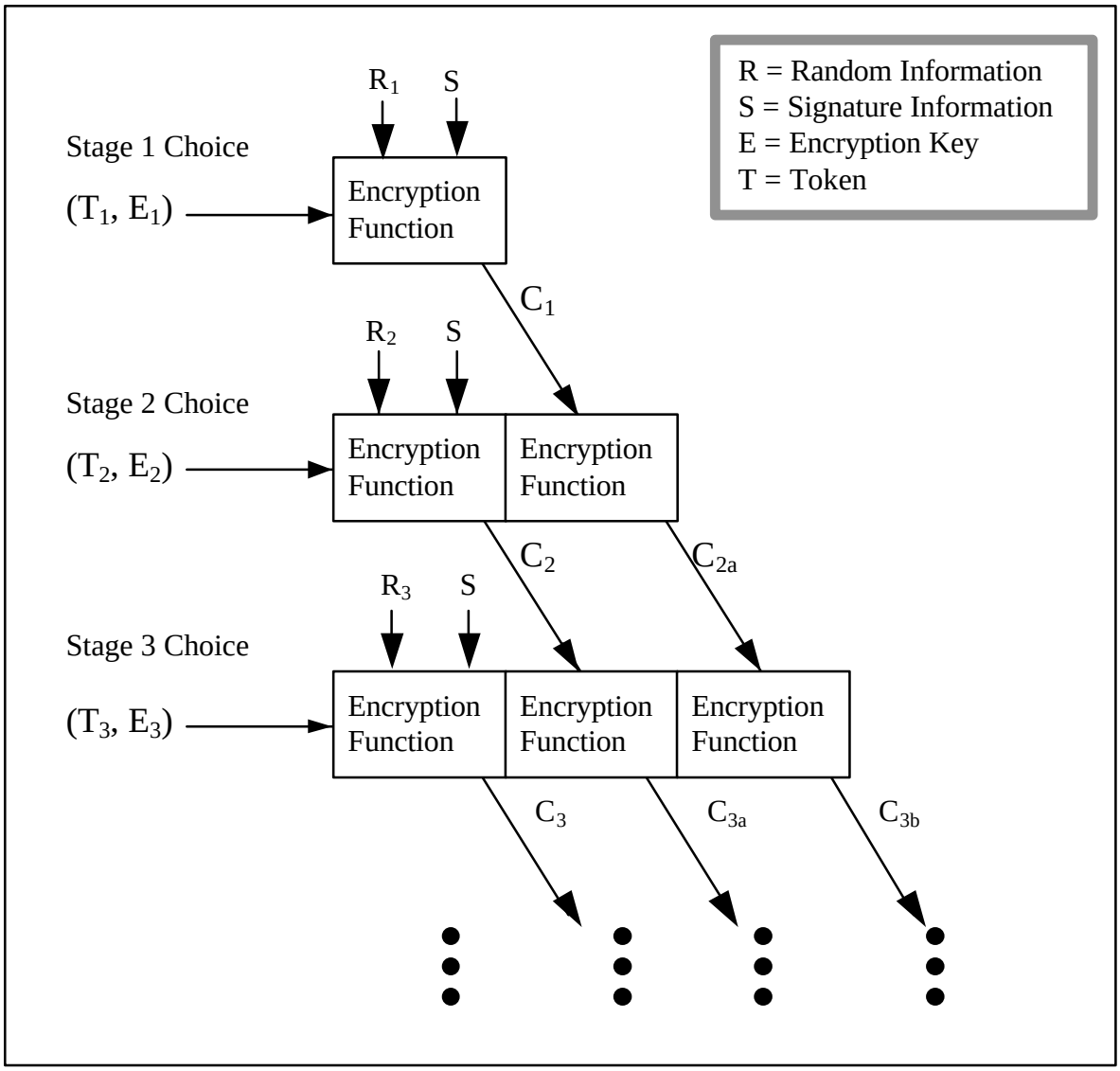

Figure 1 - Block Diagram, Multistage Algorithm

Alice continues this process recursively until all of Bob's choices are discovered.

There is, however, a basic problem with this scheme. The problem is that, to achieve the benefit, one assumes that the eavesdropper, Eve, must randomly search among all the $(N*\mathbb{R})^k$ choices to find a match, performing k encryptions for each choice at each stage. However, the outcome can be attacked in a piecemeal manner. Unfortunately, if the information from each choice or stage is segregated, or if there is discernible structure, then each stage can be attacked separately, thus greatly reducing the number of operations that the eavesdropper must perform. The system illustrated previously can be attacked because the output O_k is readily separated into two parts which can be attacked piecemeal.

Consequently, to overcome this problem, it is necessary to spread (diffuse) the information prior to input at each encryption stage. This is accomplished by mixing the token information for the current stage, the cryptogram information from the previous stage, randomization information, and the signature information together to break down structure before encryption. To do this effectively, it is necessary to use a reversible mixing function so that the structure built into the problem is spread out, yet such that the function can be easily inverted by Alice. The objective of this mixing function is to remove recoverable structure. This precludes the eavesdropper from attacking the problem piecemeal. Eve must now search the entire results space for possible matches to the *k*th stage message, otherwise break the underlying encryption problems.

The process Bob goes through is now modified to be:

$$C_k = E_{P_k} (M(T_{P_k} \,\&\&\, S \,\&\&\, R_k \,\&\&\, C_{k-1})) , \tag{13}$$

and Alice's decryption process becomes:

$$\begin{aligned} M^{-1}(D_{P_k}(C_{P_k})) &= M^{-1}(D_{P_k}(E_{P_k} (M(T_{P_k} \,\&\&\, S \,\&\&\, R_k \,\&\&\, C_{k-1})))) \\ &= T_{P_k} \,\&\&\, S \,\&\&\, R_k \,\&\&\, C_{k-1}. \end{aligned} \tag{14}$$

The added mixing function does not impose significant cost on Alice. Since Alice retains the decryption keys, Alice may do the decryption operation, invert the mixing function, and perform a match on the signature field information. Thus, the additional step of reversing the mixing function is imposed essentially with minimal cost. Consequently, the work that Alice performs at each stage of the decryption process in discovering Bob's set of choices is still $Avg(kN)$ and the overall cost is $Avg(k^2N)$. A block diagram illustrating this process, which includes the mixing function, is illustrated in Figure 2.

Candidate functions to consider for use as suitable mixing functions include simply rearranging the bits in a predetermined manner, linear transformation over the Galios Fields $GF(2^n)$, or even going so far as applying a symmetric cryptosystem such as DES. One measure of the effectiveness of the selected mixing function can be ascertained by taking into account the number of bits of the output of the mixing function which change, on average, any time a particular input bit changes value. To understand the procedure for obtaining this result, first consider the output pattern resulting from each possible input pattern where a given bit is value logic zero. Then consider the output pattern that results from that same pattern, except where the bit that was previously held to logic zero is now set to logic one. The total number of bits that change over the range of possible input patterns are counted and a percentage derived. The results of applying a linear transform to the various combinations of input are tested for bit changes.

Consider the example of using a linear transform as a mixing function. We consider functions of the form $aX + b \pmod{n}$. We can visualize the effectiveness of this type of

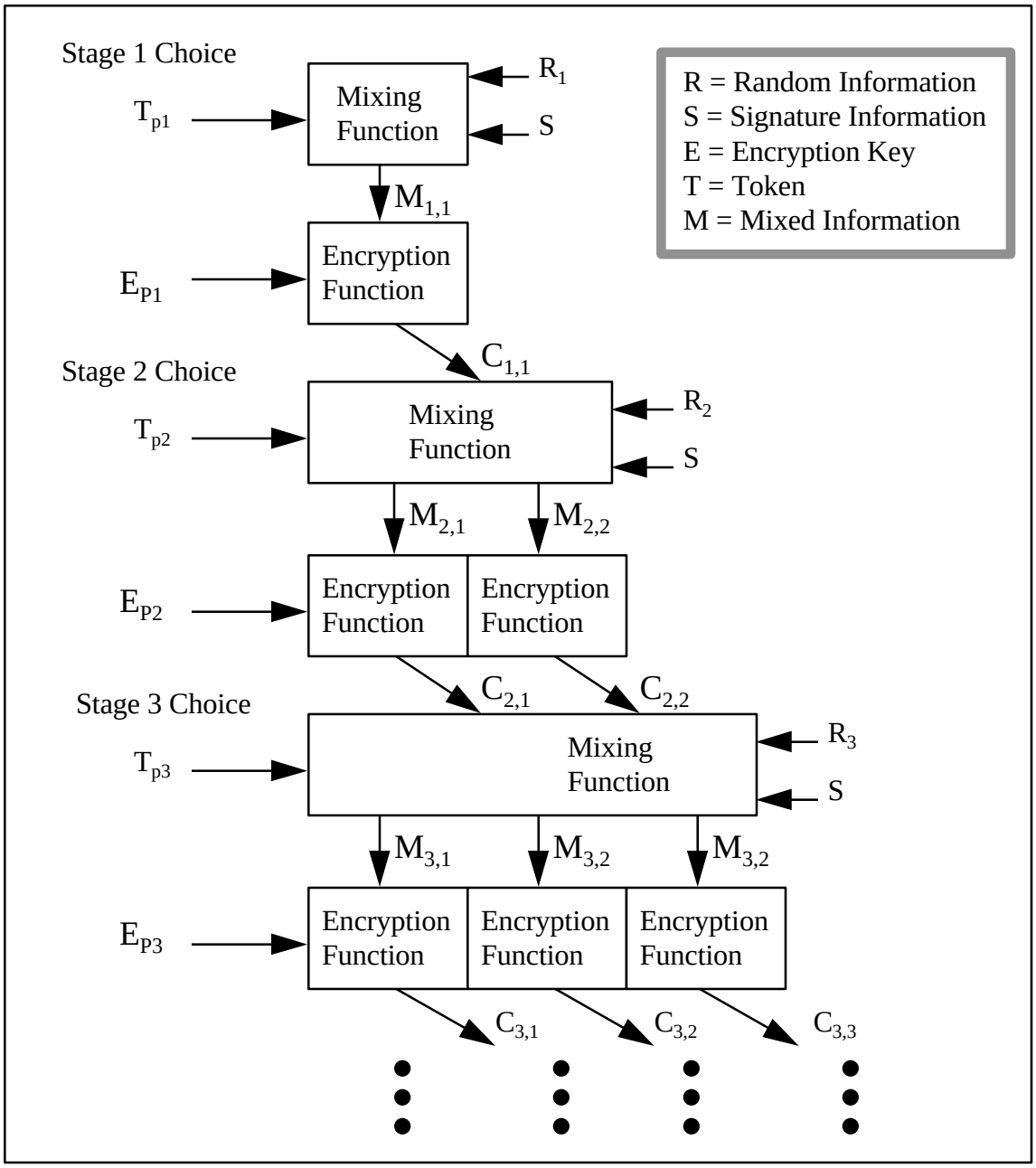

Figure 2 - Block Diagram, Multistage Algorithm with Mixing Function

function by plotting the results for vectors of limited size. An example of the effectiveness of linear transformation for use as a mixing function is illustrated in Figure 3 for the case of an eight bit multiplication and where the parameter, *b*, is set to zero. As can be seen from these examples, the mixing effects are good for the lower bits but ineffective for the upper bits. Another anticipated result that is highlighted by these results is that there are both good and bad choices for parameters as well.

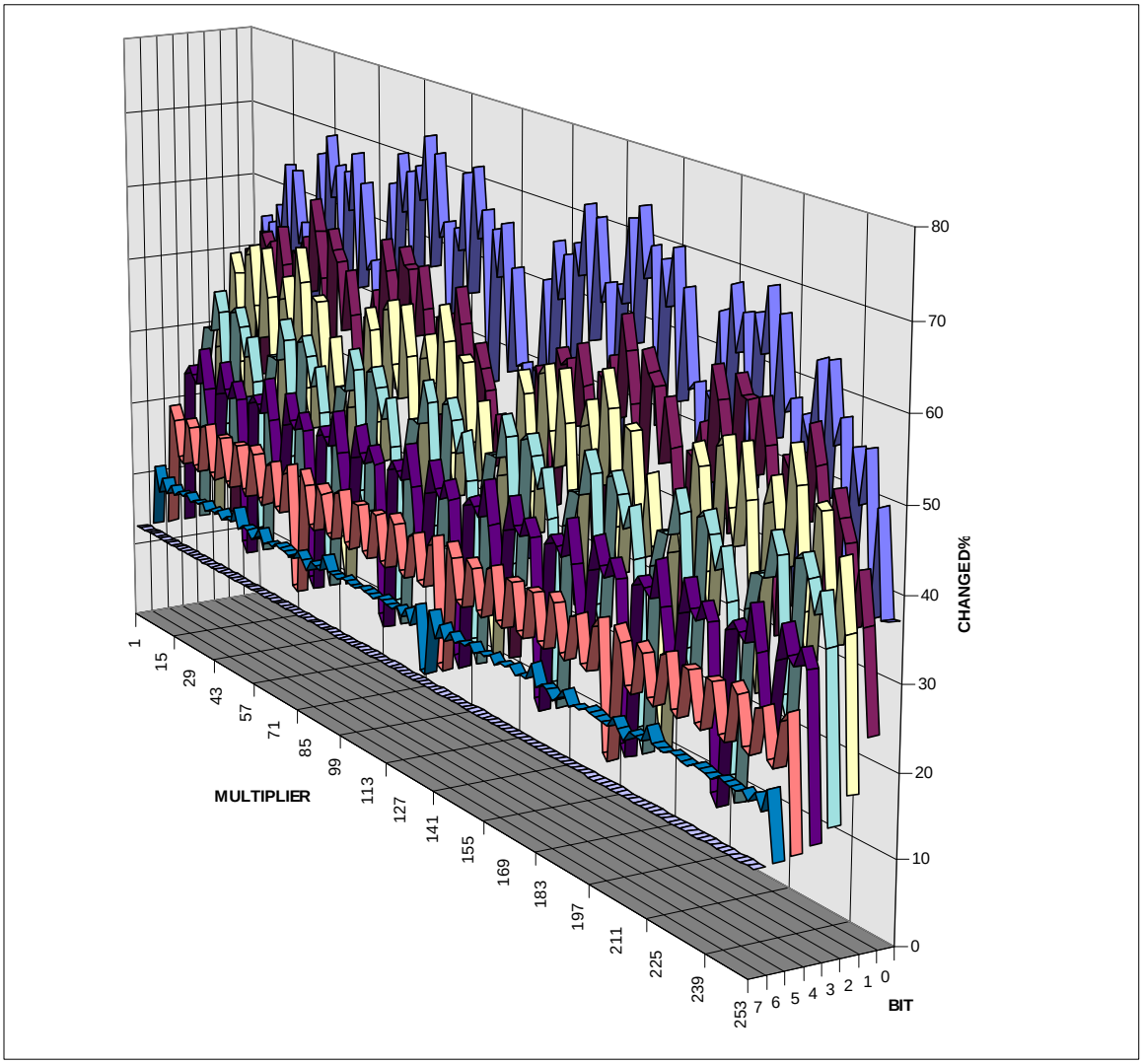

Figure 3 - Percentage Bits Change, For Multiplication by Constant

Therefore, one reasonable compromise may be to perform the transform, invert the bit order, and then to perform a second linear transform. This would roughly even out the amount of mixing that occurs from bit to bit. The mixing results for this combination are illustrated in Figure 4. It can be seen now that there are excellent choices for parameters to achieve the desired mixing. Ideally, a value would be selected that results in an expected value of half of the output bits changing for randomly selected input. Linear transformation is thus one potential reasonable choice to use as the basis for a mixing function for the multistage encryption application. It is easily invertable, and its contribution to the overall computational complexity is easily measured.

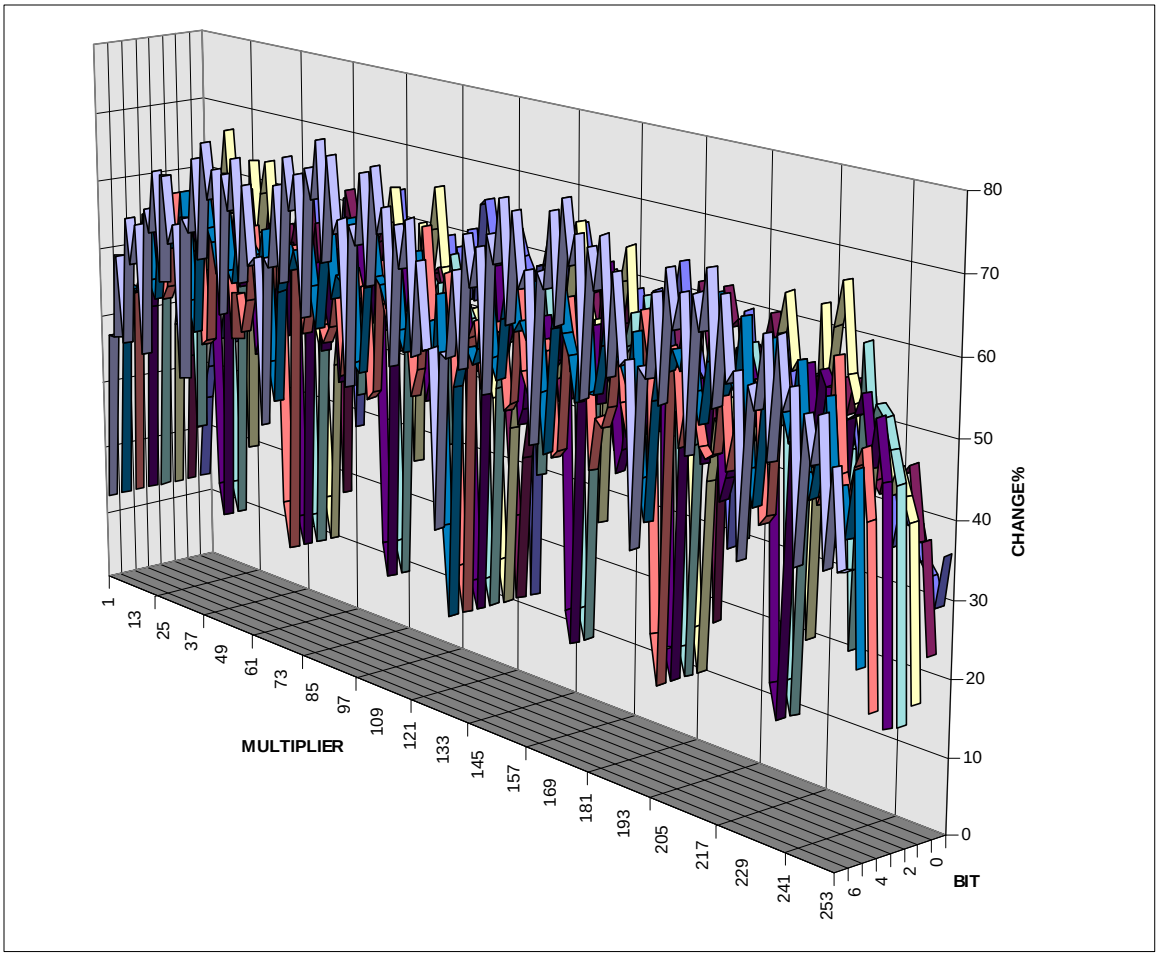

Figure 4 - Percentage Bits Change, Multiply After Bit Reversal

COMPUTATIONAL ADVANTAGE

One measure of the computational complexity of the work required by the various participants is the number of fixed size encryption or decryption operations required. Bob obviously performs k encryption operations at each stage. Alice must perform $Avg(kN)$ decryption operations for each stage, starting with stage k and working backwards. Eve, lacking the secret keys is forced to work in the forward direction, else solve the N trap door problems. Thus Eve must try, on average, all combinations of Bob's possible choices at each stage. The mixing function prevents Eve from segmenting the problem and attacking it by observing partial results. The number of decryption operations necessary to perform the work required of Alice is $Avg(k^2N)$, whereas the number of encryption operations that are required by Eve to discover the choices that Bob made at each stage of the algorithm is given by

$$\sum_{i=1}^{k} i\,(N\mathbb{R})^{i} \;=\; \frac{(N\mathbb{R})\left[k(N\mathbb{R})^{k+1} - (k+1)(N\mathbb{R})^{k} + 1\right]}{(N\mathbb{R}-1)^{2}}. \qquad (15)$$

A summary of the number of operations required at each stage by Alice, Bob, and Eve is detailed in Table 1 below.

STAGE	ALICE	BOB	EVE
1	N	1	$\mathbb{R}N$
2	$2N$	2	$2*(\mathbb{R}N)^2$
3	$3N$	3	$3*(\mathbb{R}N)^3$
k	kN	k	$k*(\mathbb{R}N)^k$
Total	$O(k^2N)$	$O(k^2)$	$O(k*(\mathbb{R}N)^k)$

Table 1 - Work Required at Each Stage

SUMMARY

This new extension to the algorithm proposed in the original work on limited one-way functions [1] provides considerable additional utility over the previous results. This previous algorithm can be viewed as an extension to Merkle's puzzle scheme [4] to situations with *Avg(N)* advantage over a restricted range. The overall security of the herein described multi-stage system is still bounded by the strength of the underlying cryptosystem upon which it is based. The backdoor approach, requiring the breaking of the underlying cryptosystem, represents the upper bound on the work required. The new utility results, however, from growing the computational advantage on the front door path to $Avg \mathbb{R} (\mathbb{R} N)^{k-1}$. Combined with embedding randomization information into the algorithm, this technique provides two parameters which can be used to control the statistical properties of the work required to perform the front-door decryption. It also provides a methodology for rapidly growing the computational advantage, given fixed size encryption blocks. This is the case, since the technique is based on chaining fixed size encryption engines, and thus can potentially economize the assets required for implementation. This improves the potential utility of the algorithm in applications such as Key Escrow systems.

As explained in some detail in the previous work, it is desirable to impose a work function on the withdrawal process for a Key Escrow system so that the rate of withdrawals can be limited to a reasonable rate. An example of this need relates to the possibility of implementing a key escrow system as part of a national system for encryption of communications assets. An algorithm for inclusion of a limited one-way function for inclusion in the implementation of such a key escrow system was proposed. This is to take advantage of the great disparity between deposit rates for the system, which could be one the order of many millions or hundreds of millions per year, and

legitimate withdrawals, which currently would be on the order of a few thousand per year for legal wiretap purposes. An advantage of this proposed technique over other possible methods for delaying withdrawals is that it can be implemented algorithmically and has measurable statistical parameters of solving very large numbers of randomly related simple problems.

This approach is contrasted against the possibility of applying a cryptographic key of limited size. Such a technique would perhaps suggest that solution be achieved by random search techniques, but there are frequently methods for achieving a solution in less time. There are also normally some values in the solution space which are easier to break. Thus there may be a wide range of solution times. Normally such cryptographic systems correspond to a complex problem where the computational complexity does not have a tight lower bound and hence having a considerable difference between measured complexity and a known lower bound. In contrast, the solution represented by this algorithm represents an opportunity to control the work within statistical limitations which are determined by requiring the solution of a large number of simple problems of measurable complexity. This method is elegant, in that it solves the problem algorithmically, does not rely on any specific cryptosystem as a basis, and can potentially have a range of applications.

REFERENCES

[1] W. T. Jennings, J. G. Dunham, "Key Escrowing Systems and Limited One Way Functions," *Conference Proceedings, 19th National Information Systems Security Conference*, Baltimore, Md, Oct. 1996.

[2] Froomkin, Michael A., "The Metaphor is the Key: Cryptography, the Clipper Chip, and the Constitution," *U. Penn Law Rev. 709*, 1995.

[3] J. P. Barlow, "A Plain Text on Crypto Policy," *Communications of the ACM*, Vol. 36, No. 11, Nov. 1993. pp. 21-26.

[4] R. C. Merkle, "Secure Communications Over an Insecure Channel," *IEEE Trans. on Information Theory*, 1976, IT-22, pp. 664-654.

[5] Mood, Graybill, and Boes, *Introduction to the Theory of Statistics*, 3rd Ed. 1974, McGraw Hill.