A LAYERED APPROACH TO THE DESIGN OF PRIVATE KEY CRYPTOSYSTEMS

T. E. Moore and S. E. Tavares
Department of Electrical Engineering
Queen's University
Kingston, Ontario, Canada. K7L 3N6

ABSTRACT

This paper presents a layered approach to the design of private key cryptographic algorithms based on a few strategically chosen layers. Each layer is a conceptually simple invertible transformation that may be weak in isolation, but makes a necessary contribution to the security of the algorithm. This is in contrast to algorithms such as DES which utilize many layers and depend on S-boxes that have no simple mathematical interpretation. A property called transparency is introduced to deal with the interaction of layers and how they must be selected to eliminate system weaknesses.

Utilizing this layered approach, a private key cryptographic algorithm consisting of three layers is constructed to demonstrate the design criteria. The algorithm has an adequate key space and valid keys can be easily generated. The design is based on a symmetrical layered configuration, which allows encryption and decryption to be performed using the same algorithm. The algorithm is suitable for VLSI implementation. Some statistical tests are applied to the algorithm in order that its cryptographic performance can be evaluated. The test results and attempts at cryptanalysis suggest that the three-layered algorithm is secure.

1. HISTORY OF LAYERING

The concept of layering cryptographic transformations to produce stronger ones was first suggested by Shannon [14] using substitution and permutation operations as layers. This idea was introduced in 1949 as product ciphers, which made it possible to generate strong cryptosystems by concatenating weak transformations. The 'Lucifer' cipher, developed at IBM by Feistel [6] embodies this approach by alternately applying substitutions and permutations.

A well-known example of an existing private key cryptographic algorithm is the Data Encryption Standard (DES) [3]. The DES algorithm consists of many layers exemplifying the strength of a layering technique. Although DES has been adopted as an encryption standard, it

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 227-245, 1986.

has been subjected to a great deal of criticism and suspicion [4, 7]. Some features of DES, such as the design of the S-boxes for example, are not well understood and instead of trusting in a system which is difficult to analyze, a user may choose a simpler system that can be understood.

Layered encryption has also been explored in the broadcast environment by Spencer and Tavares [15]. Only a few layers were employed in this particular application, each of an arithmetic nature. This is in contrast to layers such as those used in the DES algorithm.

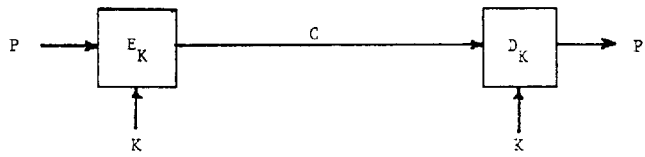
2. OVERVIEW OF LAYERING

In order that the concepts of layered encryption systems can be examined, the basic characteristics of conventional systems are stated. The components necessary in all cryptographic systems are a plaintext space P, ciphertext space C, key space K, a set of enciphering transformations E, and a corresponding set of deciphering transformations D.

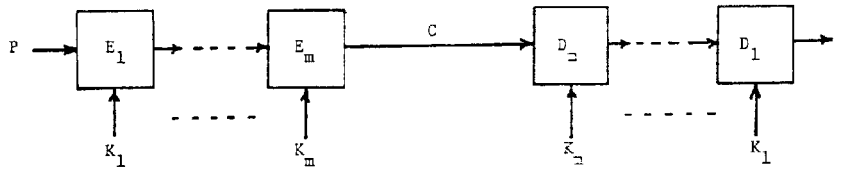
Unlike conventional systems, a layered cryptosystem has several concatenated enciphering transformations for encryption and the same number of deciphering transformations concatenated together for decryption. An m-layered cryptosystem is composed of a plaintext space P, ciphertext space C, a set of m key spaces $K_1, \ldots, K_m$, m sets of enciphering transformations $E_1, \ldots, E_m$, and m corresponding sets of deciphering transformations $D_1, \ldots, D_m$. Schematic diagrams of these two types of cryptosystems are given for comparison in Figure 1.

There are three basic assumptions important to the functionality of layered cryptosystems. The first is that the set of individual layer keys $k_1, \ldots, k_m$ used for encryption are kept secret from unauthorized users. Secondly, each layer is a simple invertible transformation which may be weak cryptographically in isolation, but makes a necessary contribution to the security of the entire system. Lastly, the interlayer results of the enciphering and deciphering transformations are not accessible to unauthorized users. All discussions dealing with layered encryption in this paper apply only to private key cryptosystems.

It is important here to clarify that any layer by itself is not secure, given access to its input and output values. Nothing is gained by layering if interlayer results are an allowable resource to a cryptanalyst. It is a reasonable assumption to consider the inter-



(a) Cryptographic System



(b) M-Layered Cryptographic System

FIGURE 1: Comparison of Conventional and Layered Cryptosystems.

layer results as unattainable resources. Unlike plaintext and ciphertext, interlayer values are only transient results which are never stored or accessed at any time by legitimate users of the system. The only manner in which they may be obtained is if an intruder can tap and monitor the hardware between the layers. Physical security can always be employed if monitoring is a possible threat. The remainder of this discussion assumes that interlayer results are never accessible and are adequately protected.

3. THE LAYERED APPROACH

3.1 Layer Selection Criteria

By adopting a few strategically chosen layers, a layered approach can be utilized to design private key cryptosystems. Before a mathematical transformation may be classified as a layer, it must conform to the following specifications:

a) a layer must be well defined in a mathematical sense while remaining simple in concept

b) it must have an adequate key space with easily generated keys and inverse keys

c) be efficient in terms of time and space

d) be easy to program for software simulation and implementation and

e) be suitable for VLSI design and implementation.

It is plausible that if each individual layer in a layered cryptosystem meets these requirements, them the synthesized layered algorithm will conform to them as well.

3.2 Layer Interaction and Transparency

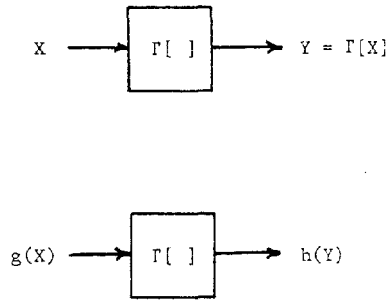
With the layer selection criteria established, it becomes necessary to develop additional guidelines for concatenation of layers. An important consideration in concatenating layers to synthesize a complete algorithm is the problem of layer interaction. There is an obvious disadvantage to concatenate two layers which can each be compromised on an individual basis by the same attack.

A concept is introduced here which helps to deal with layer interaction and is defined as layer transparency. To define transparency, consider the transformation r [ ] of Figure 2 which maps X into Y, where X and Y are n-bit vectors. Let $g(X)$ be the result of a simple operation $g(\cdot)$ on the input X. If $g(X)$ is mapped to $h(Y)$ by r [ ], where $h(\cdot)$ is also a simple operation, then it is said that r [ ] is transparent to $g(\cdot)$, and that $g(\cdot)$ is a transparency of r [ ]. In this discussion, is should be noted that $g(\cdot)$ and $h(\cdot)$ may be the same operation. As an example, $g(X)$ could be a cyclic shift of X by one bit and $h(Y)$ a cyclic shift of Y by t bits, where $1 \leq t \leq n-1$. If $t = 1$, then the two operations $g(\cdot)$ and $h(\cdot)$ would be identical.

As a general rule, two adjacent layers in a layered cryptographic algorithm should not have common transparencies. In addition, it is desirable that all layers in a cryptosystem do not share many of the same transparencies.

3.3 Buffers

The problem of selecting various useful transformations that strictly follow the two transparency rules may not be simple. What is required are simple operations to isolate the main layer transformations. As an example, two nearly compatible transformations may be suitable as adjacent layers except for a single common transparency. If a simple operation can be found that does not preserve this common transparency, then it can be inserted between the two layers. The resultant



Γ[ ] - invertible transformation
X - n-bit input vector
Y - n-bit output vector
g(•), h(•) - simple operations

FIGURE 2: Illustration of Transparency.

new transformation of a simple layer sandwiched between two main layers is no longer hampered by the transparency. The simple operations in question are defined as 'buffers', and for simplicity they can be considered as another layer in the layered cryptosystem. However, buffers differ from the main layers in that they do not possess a key space.

There are two types of buffers defined by their position relative to the main transformations. The first type are positioned before the first and after the last layers. This buffer type is defined as an 'outer buffer'. In a cryptosystem of only a few layers, it is critical that a cryptanalyst not be allowed to probe the outer layers using strategically selected inputs. Knowledge of the transparencies of the first layer for example, can be utilized in such a manner as to derive the result of this transformation without actual knowledge of its key. Hence, for the given strategic input, the first layer is effectively by-passed leaving a weakened algorithm to compromise.

It is realized that a constant operation is not suitable for an outer buffer. Since we assume that every feature of the cryptographic algorithm will be public knowledge, except for the key of course, a cryptanalyst can derive the result of any constant operation and have direct access to the outer layers as before. It is thus necessary that outer buffers be computed from key-dependent operations so that

the result of a given buffer operation cannot be determined without knowledge of the keys. For a given key set, this may be accomplished by computing the buffers from a single one-way function of the layer keys. Hence, actual inputs to the first main layer cannot be derived, preventing effective chosen-plaintext attacks.

The second type of buffers are positioned between two main layers. These buffers are defined as 'interlayer buffers' and their purpose is to prevent the preservation of transparencies that exist in common with two adjacent main transformations.

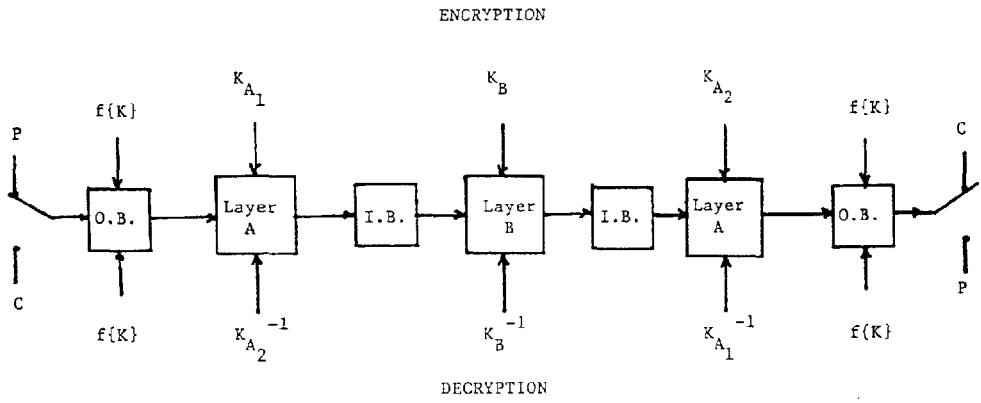
In contrast to an outer buffer, the input to any interlayer buffer is never directly accessible, making it unnecessary for interlayer buffers to be key-dependent operations. Further, it is preferable if the interlayer buffers are key-independent operations as they would not require any pre-computation for a given key set.

3.4 Additional Considerations

In a system where all main layers and buffers are linear, the system transformation may be represented equivalently by a simplified linear operation. An attack based on the principle of superposition can be utilized to compromise a linear cryptosystem. It is thus necessary to ensure that the overall system transformation for the layered algorithm is nonlinear. This can be accomplished by selecting one of the main layers as a nonlinear transformation.

A second consideration when dealing with layer concatenation is symmetry. Carefully selecting the layers in a symmetrical configuration will allow the encryption and decryption functions to be performed using the same algorithm. A schematic diagram of a symmetrical layered configuration is given in Figure 3. For this 3-layered example illustrated in the figure, the essential nonlinear transformation can be either Layer A or Layer B.

In order to facilitate the symmetry in Figure 3, several conditions must be satisfied. First the two outside layers must be selected as identical transformations. In practice, different keys would be used for these two layers to keep the system key space as large as possible. The next requirement for total symmetry is that the two outer buffers must be identical operations. The interlayer buffers must also meet this requirement. The relative positions of these buffers are clearly illustrated in Figure 3. The last requirement is that the outer buffers must be their own inverse operations. The interlayer buffers must also fulfill this requirement.



P - plaintext
C - ciphertext
$K = \{K_{A_1}, K_B, K_{A_2}\}$
= encryption key set
$K^{-1} = \{K_{A_2}^{-1}, K_B^{-1}, K_{A_1}^{-1}\}$
= decryption key set

O.B. - Outer Buffer
I.B. - Interlayer Buffer
f{K} - one-way function

FIGURE 3: Symmetrical Layered Configuration.

With these conditions included in a symmetrical layered configuration, total algorithm symmetry is obtained. As shown in Figure 3, if the encryption key set is $\{K_{A_1}, K_B, K_{A_2}\}$, then the corresponding decryption key set is $\{K_{A_2}^{-1}, K_B^{-1}, K_{A_1}^{-1}\}$. In this notation, $K_{A_1}^{-1}$ represents the mathematical inverse (decryption key) of K_{A_1} for the transformation of Layer A. A benefit resulting from designing a symmetrical algorithm is the reduction in the amount of chip area needed to incorporate both encryption and decryption in a single chip VLSI implementation.

3.5 Summary of Approach

The important concepts pertinent to the layered design approach of cryptographic algorithms were presented. Selection criteria for transformations were established and the concept of transparency was introduced to resolve the problem of layered interaction. System

transparencies can be eliminated by carefully selecting transformations with specified properties, and by utilizing specially designed buffers. The presence of at least one nonlinear operation is essential to the security of the algorithm. The essential nonlinearity can be accommodated by selecting a nonlinear transformation as one of the layers. A symmetrical layered configuration has several advantages, but is not necessary for constructing a secure system. By selecting certain transformations and concatenating them using the established criteria in this section, it may be possible to synthesize a secure cryptosystem.

4. DESIGN OF A LAYERED CRYPTOGRAPHIC ALGORITHM

Before presenting the following discussion, it is important to clarify that the algorithm given here is not intended to represent an unbreakable cryptosystem. It is simply given here in order to illustrate the structured approach to designing cryptographic algorithms given in the previous section.

Utilizing the layered approach given in Section 3, a private key cryptographic algorithm has been designed using the exact configuration given in Figure 3. The Layer A transformations have been selected as linear transformations and Layer B as the essential nonlinear transformations.

4.1 Nonlinear Layer

There are a number of nonlinear transformations that have been used in cryptographic applications. For reasons of dependability and reputation as a strong algorithm, the RSA algorithm [13] was examined for possible foundations for a nonlinear transformation. On that basis, modular exponentiation was chosen to represent the nonlinear layer. A modulus of 2^n-1, for n-bit block encryption, was chosen as an appropriate modulus for this transformation. In this discussion, n is a power of 2, for reasons which will become evident. There are two reasons for choosing this particular modulus. First, the integer 2^n-1 is a product of r distinct prime numbers (at least as far as n = 64). This is an extension of the two prime case used with the RSA modulus. The second reason is an implementation feature of 2^n-1 in that actual division is not required to perform modulo reduction by 2^n-1. This will become clear in Section 5 where implementation considerations are discussed.

To summarize, the following nonlinear transformation is used as Layer

B in the symmetrical layered configuration of Figure 3.

$$Y = 2^n-1 , \quad \text{if } X = 2^n-1$$
$$= X^{K_B} \bmod 2^n-1 , \quad \text{otherwise.}$$

where

X is the n-bit input
Y is the n-bit output
K_B is the key for Layer B

and

$$n = 2^m$$

Ordinarily 0 and 2^n-1 are equivalent modulo 2^n-1, and hence both of these inputs would produce an all zero binary output. The conditional equality in the above definition is necessary to resolve this situation.

In order that we may decrypt correctly, an inverse K_B^{-1} must exist such that

$$X^{K_B * K_B^{-1}} \bmod 2^n-1 = X.$$

This relation can be satisfied for any modulus that is a product of r distinct primes, if the following relationship is true [2]

$$K_B * K_B^{-1} = 1 \bmod \phi(2^n-1)$$

where $\phi(\cdot)$ is the Euler totient function. The above relation reduces to the property that K_B must be chosen relatively prime to $\phi(2^n-1)$.

Blakley and Borosh [2] recognized that transformations of this type always have a certain number of inputs that are mapped to themselves, defined as unconcealed inputs. For this particular transformation, this phenomenon may be represented mathematically as

$$X^{K_B} \bmod 2^n-1 = X.$$

To minimize the number of unconcealed inputs, it is required that K_B be chosen under the following additional constraint:

$$GCD[K_B-1, LCM(p_1-1,\ldots, p_r-1)] = 2$$

where GCD is the Greatest Common Divisor
LCM is the Least Common Multiple
and $(p_1,\ldots, p_r)$ are the r unique prime factors of 2^n-1.

For exponentiation mod2^n-1, the actual minimum number of unconcealed inputs is 3^r+1. This minimum number can only be achieved if K_B satisfies the above relation. For a block length of n = 64, there are a minimum of 2188 unconcealed inputs since 2^{64}-1 is a product of 7 distinct prime numbers. The number of key bits generated by exponentiation modulo 2^n-1, with n = 64, is estimated to be 29.

4.2 Linear Layers

By concatenating a simple linear transformation processing specified properties with a nonlinear layer, it is plausible that a stronger transformation will result from the concatenation. A particular family of linear transformations used in cryptographic applications is modular multiplication. These transformations have been studied previously by Leung and Tavares [12] for modulus values of 2^n and 2^n-1. Multiplication modulo 2^n-1 is a fundamental transformation in a cryptographic algorithm proposed by Akl and Meijer [1].

Multiplication modulo 2^n was chosen over 2^n-1 for two reasons. First, this modulus is different from the modulus of the nonlinear exponentiation transformation. If each layer was some operation modulo 2^n-1, and ignoring the effect of any interlayer buffers, then it is possible to simplify the overall mathematical representation of the 3-layered concatenation by applying the principles of modular arithmetic [5].

The second reason is that multiplication modulo 2^n is not transparent to complements, whereas multiplication mod2^n-1 is transparent to this complement operation. To further clarify this operation, a bit-wise complement of any input produces a bit-wise complement of its corresponding output. It can be shown that both multiplication and exponentiation mod2^n-1 are transparent to complements. Thus selecting multiplication mod2^n-1 would tend to violate the layer interaction criteria established in Section 3.2. Proof of the complement transparency for exponentiation mod2^n-1 is given in Appendix A.

In summary, the linear transformations indicated by Layer A in Figure 3 may be represented analytically as:

$$Y = X * K_A \bmod 2^n$$

where X is the n-bit input
Y is the n-bit output
and K_A is the key for Layer A.

In order to estimate the size of the key space for this transformation, the number of keys K_A that allow X to be recovered from Y must

be known. From elementary number theory, X has a unique inverse mod 2^n if the integers K_A and 2^n are relatively prime. The number of integers relatively prime to 2^n is $\phi(2^n) = 2^{n-1}$. Hence for n = 64, there are 2^{63} keys K_A that will allow successful decryption. Since the GCD (K_A, 2^n) must equal 1, the K_A must be an odd integer and thus valid 64-bit keys may be selected by simply setting the least significant bit of K_A to binary one.

4.3 Common Transparencies and Buffer Selection

The design of the buffers depends on the common transparencies that exist between the main transformations of the algorithm. The following is a summary of the known transparencies and weaknesses that are common to exponentiation mod 2^n-1 and multiplication mod 2^n.

i) The all-binary zero input maps to itself in both transformations.

ii) Both transformations are transparent to shifting; although multiplication is transparent to logical shifts, and exponentiation is transparent to a variation of cyclic shifts.

iii) Multiplication is preserved under modular exponentiation and modular multiplication.

The transparencies and weaknesses above can be easily verified for each transformation.

Recall that the purpose of outer buffers is to inhibit an intruder from launching chosen-plaintext attacks. Outer buffers must also be key-dependent operation. For simplicity and ease of implementation, the exclusive - OR addition of a key-dependent n-bit sequence V is suitable for the outer buffers. To determine a particular value of V for a given key set, it is necessary that V be derived from a one-way function of the three keys. Therefore, the sequence V cannot be computed unless the three layer keys are known. Exclusive - OR addition is also its own inverse operation and thus satisfies the conditions needed to maintain the symmetrical layered configuration depicted in Figure 3. By coincidence, this buffer operation also eliminates the zero input mapping to itself.

The second and third common transparencies listed at the beginning of this section are left to be resolved by the interlayer buffers. It should be pointed out that the third transparency is true only when the product of the inputs in question is less than 2^n. If the product is greater than this value, then multiplication is not preserved through the concatenation of the three layers. This result stems from the fact that two different modulus values are used in the transforma-

tions.

An n-bit permutation ρ is a suitable interlayer buffer under the following constraints:

i) ρ does not preserve shifts
ii) ρ does not preserve multiplication
and iii) ρ is its own inverse operation

The first two constraints rectify the second and third common transparencies and the last constraint is necessary to satisfy the symmetrical layered configuration conditions.

4.4 Summary of 3-Layered Algorithm

A block diagram summary of the 3-layered cryptographic algorithm is shown in Figure 4. The layers indicated by Layer A in Figure 3 are multiplication modulo 2^n transformations, and Layer B is the nonlinear exponentiation modulo 2^n-1 transformation. To easily distinguish between the two multiplication layers, a notation change from letters to numbers is done. The layers are labelled as 1, 2 and 3 going from left to right in Figure 4. The outer and interlayer buffers shown in the figure are as defined in Section 4.3.

If we let $T(\cdot)$ represent the overall transformation depicted in Figure 4, then the encryption operation may be represented as

$$C = T_K(P)$$

where P is the plaintext
C is the ciphertext
and $K = \{K_1, K_2, K_3\}$ is the encryption key set. Since the algorithm is symmetrical, the decryption operation may also be represented in terms of the same transformation as

$$P = T_{K^{-1}}(C)$$

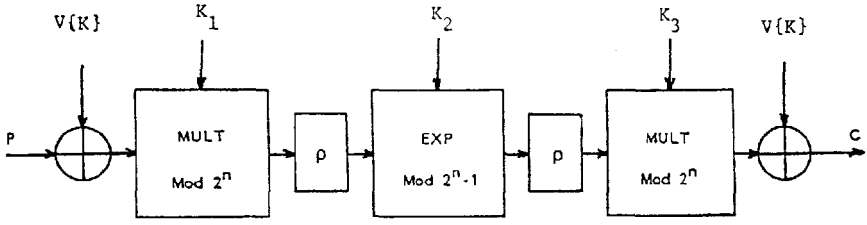
where $K^{-1} = \{K_3^{-1}, K_2^{-1}, K_1^{-1}\}$ is the decryption key set. Thus, the distinguishing feature between encryption and decryption with this algorithm is the key set used in each case. The decryption keys are related to their encryption key counterparts by the following equations:

i) $K_1 * K_1^{-1} \bmod 2^n = 1$

ii) $K_2 * K_2^{-1} \bmod (2^n-1) = 1$

iii) $K_3 * K_3^{-1} \bmod 2^n = 1$

Calculating the integer values of each decryption key can thus be accomplished by using Euclid's algorithm [9].



P: n-bit plaintext

C: n-bit ciphertext

K_1, K_2, K_3: user selected keys for layers 1, 2, 3

K: $\{K_1, K_2, K_3\}$

V: n-bit sequence derived from a one-way function of the keys

ρ: interlayer buffer operation

⊕: bit-wise exclusive-OR operation

FIGURE 4: Block Diagram of the 3-Layered Cryptographic Algorithm.

5. IMPLEMENTATION CONSIDERATIONS

The discussion included in this section is intended to illustrate the relatively simple algorithms that are needed to implement the main transformations in the algorithm of Figure 4. The primary consideration of the design criteria was to facilitate a VLSI (very large scale integration) application. Pseudo-code algorithms suitable for VLSI implementation of the main transformations are contained in Appendix B. Simple shift-registers and adders are the primary components necessary to implement these algorithms.

The first pseudo-code algorithm given in Appendix B is for modular exponentiation. It uses repeated squaring and multiplication to implement exponentiation. The algorithm scans the bits of the binary representation of the exponent, starting with the least significant bit. For each bit of the exponent, a squaring operation is performed if the bit is a binary zero, squaring followed by multiplication if the current exponent bit is a binary one. All squaring and multipli-

cation operations are reduced modulo 2^n-1, and can be implemented using the second algorithm given in Appendix B.

The second and third algorithms of Appendix B are for multiplication modulo 2^n-1 and 2^n respectively. Both utilize "shifting and adding" techniques to implement multiplication. These algorithms are efficient since actual division is not performed when modulo reducing by either 2^n or 2^n-1.

For a modulus of 2^n, all overflow bits resulting from the repeated addition operations are simply truncated in order to modulo reduce. The overflow bits represent integer multiples of 2^n, and hence truncating these bits is equivalent to dividing by 2^n. For 2^n-1, the overflow bits are not truncated, but are cyclicly shifted and added to the least significant bit of the result. This is equivalent to subtracting a value of 2^n-1.

To implement the outer buffer operations in VLSI, n two-input exclusive - OR gates in parallel can be used for each buffer. Since interlayer buffers can be selected as constant permutations, they can be hard-wired in a VLSI implementation.

6. PERFORMANCE

Since the 3-layered algorithm cannot be proven secure, we must rely on certain tests and analyses to provide confidence in the algorithm. A few statistical tests have been applied to the algorithm in order to evaluate its cryptographic performance. The tests listed below were used in the evaluation:

i) Plaintext/ciphertext Complexity Test
ii) Avalanche Complexity Test
iii) Bit Distribution Test
iv) Cycle Test

The above tests were performed on a 32-bit software implementation of the algorithm. Using a VAX 11/750 computing facility, assembly language routines were written to simulate each layer.

The first two tests listed above depend on the concept of complexity. The complexity criterion [12] was used extensively for performing these statistical tests, and a measure of complexity developed by Lempel and Ziv [11] was used to evaluate the randomness properties of the algorithm. In general, the difference between any plaintext and its corresponding ciphertext should have high complexity with a high probability [12]. This complexity is referred to as plaintext/

ciphertext complexity and is measured using the first test. In addition, the difference between two ciphertexts whose corresponding plaintexts differ by a predetermined bit must also have this high complexity. Horst Feistel [6] termed this property the Avananche Effect, and it is measured using the avalanche complexity test.

An additional test is a bit distribution test which simply counts the number of binary ones (or zeros) in the two variations of difference sequences mentioned above. Over a large sample of randomly selected plaintext, the resulting bit distribution should resemble the binomial distribution if the differences are indeed random.

The last test that was applied to the algorithm is a cycle test [10]. The purpose of this particular test is to determine if the set of permutations for the overall algorithm transformation is closed under functional composition. If the transformation is closed, then the set of transformations may generate a small group and hence contain a weakness that is vulnerable to a known-plaintext attack [8]. The cycle test that was implemented examines the orbits of plaintext messages under fixed keys which are produced by the algorithm in output-feedback mode. Although this is not the most efficient closure test [8], it was felt that this particular version of the test was the simplest and best suited for the available resources.

The results of the first three statistical tests listed at the beginning of this section indicate that the 3-layered algorithm performs well cryptographically. It appears that the algorithm in fact possesses good randomness properties. The cycle test results are inconclusive as only a few tests have been completed. Results thus far indicate that the overall transformation of the 3-layered algorithm is not closed under functional composition.

7. CLOSING REMARKS

We have presented a layered approach to designing strong cryptographic algorithms using conceptually simple mathematical transformations. Although the layers themselves are weak in isolation, they make a necessary contribution to the overall strength of the algorithm. This is a simplified approach which can reduce the complexity of designing a cryptographic algorithm.

In addition, a three-layered cryptographic algorithm has been designed using the layering technique. Although the algorithm was presented to illustrate the design criteria, it in fact appears strong and possesses several attractive features. Naturally, it is possible that

cryptanalysis could show that the algorithm is weak, or under certain conditions, may be compromised completely. In either case, the analysis would be interesting due to the simple concepts and mathematical properties inherent in the design. The layered approach would still be considered useful as it reduces the complexity of algorithm design. It also allows a designer to develop layered algorithms reasonably fast, since previously studied transformations can be chosen as layers.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada under Strategic Grant #G1364.

8. REFERENCES

[1] Akl, S.G. and Meijer, H., "Two New Secret Key Encryption Algorithms", presented at Eurocrypt '85, Linz, Austria, Apr. 1985.

[2] Blakley, G.R. and Borosh, I., "Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages", Comp. & Maths with Appls., Vol. 5, pp. 169-178, Pergamon Press Ltd., 1979.

[3] "Data Encryption Standard", FIPS PUB 46, National Bureau of Standards, Washington, D.C., Jan. 1977.

[4] Davies, D.W., "Some Regular Properties of the DES", Advances in Cryptology: Proceedings of Crypto '82, pp. 89-96, Plenum Press, 1983.

[5] Denning, D.E., Cryptography and Data Security, Addison-Wesley, Reading, Mass., 1982.

[6] Feistel, H., "Cryptography and Computer Privacy", Sci. Am., Vol. 228, pp. 15-23, May 1973.

[7] Hellman, M.E., et al., "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard", Information Systems Lab., Dept. of Electrical Eng., Stanford Univ., 1976.

[8] Kabiski, B.S., Rivest, R.L. and Sherman, A.T., "Is the Data Encryption Standard a Group?", presented at Eurocrypt '85, Linz, Austria, Apr. 1985.

[9] Knuth, D., The Art of Computer Programming; Vol. 2, Seminumerical Algorithms, Addison-Wesley, Reading, Mass., 1969.

[10] Konheim, A.G., Cryptography: A Primer, John Wiley and Sons, New York, 1981.

[11] Lempel, A. and Ziv, J., "On the Complexity of Finite Sequences", IEEE Trans. on Info. Theory, Vol. IT-22, pp. 75-81, Jan. 1976.

[12] Leung, A.K. and Tavares, S.E., "Sequence Complexity as a Test for Cryptographic Systems", Proceedings of Crypto '84, pp. 468-474, Springer-Verlag, 1985.

[13] Rivest, R.L., Shamin, A. and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Comm. ACM, Vol. 21, pp. 120-126, Feb. 1978.

[14] Shannon, C.E., "Communication Theory of Secrecy Systems", Bell Syst. Tech. J., Vol. 28, pp. 656-715, Oct. 1949.

[15] Spencer, M.E. and Tavares, S.E., "Layered Broadcast Cryptographic Systems", Advances in Cryptology: Proceedings of Crypto '83, pp. 157-170, Plenum Press, 1984.

APPENDIX A

Proof of Complement Transparency for Exponentiation Mod 2^n-1

It is required to show that a bit-wise complement of any input produces a bit-wise complement of its corresponding output for exponentiation mod 2^n-1.

Proof: More formally, it is required to show that if

$$X^K \equiv Y \bmod(2^n-1)$$

then

$$\bar{X}^K \equiv \bar{Y} \bmod(2^n-1).$$

where $\bar{X}$ and $\bar{Y}$ are the bit-wise complements of X and Y respectively and K is odd.

We can write

$$X + \bar{X} = 2^n-1$$

or $$X + \bar{X} \equiv 0 \bmod(2^n-1)$$

or $$X \equiv -\bar{X} \bmod(2^n-1)$$

Squaring,

$$X^2 \equiv \bar{X}^2 \bmod(2^n-1)$$

and thus

$$X^U \equiv \bar{X}^U \bmod(2^n-1), \qquad U \text{ even}$$

and

$$X^V \equiv -\bar{X}^V \bmod(2^n-1), \qquad V \text{ odd}$$

Let $$X^K \equiv Y \bmod(2^n-1), \qquad K \text{ odd}$$

then $$\bar{X}^K \equiv -X^K \bmod(2^n-1)$$

$$\equiv -Y \bmod(2^n-1)$$

$$\equiv \bar{Y} \bmod(2^n-1).$$

APPENDIX B

Pseudo-code Algorithms for Layer Transformations

In this appendix are pseudo-code algorithms for the two transformations of the 3-layered algorithm. A total of three algorithms are included as follows:

i) Algorithm #1: Exponentiation mod 2^n-1

ii) Algorithm #2: Multiplication mod 2^n-1

iii) Algorithm #3: Multiplication mod 2^n

Algorithm #1 requires the use of Algorithm #2 in the form of a subroutine in order to perform the full modular exponentiation transformation. Algorithms #2 and #3 are shifting and adding based routines which include the appropriate variations necessary to perform modulo reduction.

The three algorithms are presented here purposely for a VLSI application. Each algorithm can be implemented almost entirely with shift-registers, adders, and a carry-bit function. For these algorithms, the value of 'n' is not considered variable, but is in fact a constant equal to the specified block length of the cryptosystem. Thus n will govern certain design parameters such as the size of internal registers. For notation, all input and output variables (denoted as capital letters) are regarded as n-bit integers, and the i^{th} bit position of X is expressed as X(i) in these algorithms.

ALGORITHM #1

Exponentiation mod 2^n-1

Returns: $Y = X^K \bmod 2^n-1$

```
Input:  X,K
  i = 0
  if (K(i) = 1) then
              Y = X
          else
              Y = 1
  end if
  i = 1
  do while (i < n)
         X = X * X mod 2^n-1
         if (K(i) = 1) then
             Y = Y x Y mod 2^n-1
         end if
         i = i + 1
  end do
Output: Y
```

ALGORITHM #2

Multiplication mod 2^n-1

Returns: P = A * B mod 2^n-1

```
Input: A,B
  P = 0
  i = 0
  do while (i < n)
        if (B(i) = 1) then
            P = P + csl(i,A)
            if (carry = 1) then
                 P = P + 1
            end if
        end if
        i = i + 1
  end do
Result: P
```

csl(i,A) = cyclic shift left of A by i bits
carry = carry bit function

ALGORITHM #3

Multiplication mod 2^n

Returns: Y = X * K mod 2^n

```
Input: X,K
  Y = 0
  i = 0
  do while (i < n)
       if (K(i) = 1) then
           Y = Y + lsl(i,X)
       end if
       i = i + 1
  end do
Output: Y
```

lsl(i,X) = logical shift left of X by i bits