Dense Probabilistic Encryption

Josh Benaloh
Clarkson University

Abstract

This paper describes a method of *dense* probabilistic encryption. Previous probabilistic encryption methods require large numbers of random bits and produce large amounts of ciphertext for the encryption of each bit of plaintext. This paper develops a method of probabilistic encryption in which the ratio of ciphertext text size to plaintext size and the proportion of random bits to plaintext can both be made arbitrarily close to one. The methods described here have applications which are not in any apparent way possible with previous methods. These applications include simple and efficient protocols for non-interactive verifiable secret sharing and a method for conducting practical and verifiable secret-ballot elections.

1 Introduction

In 1984, Goldwasser and Micali ([GoMi84]) introduced the notion of *probabilistic encryption*. A probabilistic encryption method allows one to encrypt a fixed value in many different ways. Thus, even when given the encryption of a value and details of the encryption mechanism (including any encryption key), it is not necessarily possible for an adversary to determine whether or not a given ciphertext represents the encryption of a particular value.

Goldwasser and Micali develop a bit encryption function based on the number theoretic problem of quadratic residuosity. Their method has many useful properties, but there is one major drawback: for a given security parameter N, the probabilistic encryption of each bit is N bits long, requires N random bits, and uses several operations on N bit integers.

This work describes a dense method of probabilistic encryption which, unlike the method of Goldwasser and Micali, is capable of encrypting more than one bit at a time. For any given k and security parameter N, this new method allows the encryption of k bits of information into an $N + k$ bit ciphertext using $N + k$ random bits and operations on $N + k$ bit integers. Thus, for any desired security parameter N, the ratio of plaintext size to ciphertext size (as well as to random bits required or to the size of the integers computed upon) can be made arbitrarily close to one.

There are also some applications where one bit at a time probabilistic encryption is unsuitable regardless of efficiency. This paper describes two such applications – non-interactive verifiable secret sharing and a method for obtaining verifiable secret-ballot elections – in which the dense probabilistic encryption method described here can be used while there is no apparent way of developing similar solutions with bitwise probabilistic encryption.

2 The Encryption Method

This section will show how to generate "one-to-many" functions (or *randomized* functions) E_r for any odd integer r with the following basic properties.

- Given a message $M \in \mathbb{Z}_r = \{0, 1, 2, \ldots, r-1\}$, it is computationally easy to for any participant to form *an* encryption $z \in E_r(M)$.
- The decryption of any $z \in E_r(M)$ is unique – that is, if $M_1, M_2 \in \mathbb{Z}_r$ with $M_1 \neq M_2$, then $E_r(M_1) \cap E_r(M_2) = \emptyset$; and this unique decryption can be computed by the creator of E_r.
- Under a suitable cryptographic assumption it is "infeasible" to compute M (or even gain so much as an inverse polynomial advantage at computing any predicate on M) given simply the details of the randomized function E_r and an encryption $z \in E_r(M)$.

In addition to these properties, several other useful properties will be achieved.

- Given a message $M \in \mathbb{Z}_r$ any participant can generate a $z \in E_r(M)$ together with a certificate u which can be used to prove to any other participant(s) that $z \in E_r(M)$.
- Given an encryption $z \in E_r(M)$, it is possible for the creator of E_r to produce a certificate u which is uniformly selected from the set of all possible user certificates and can, likewise, be used to prove that $z \in E_r(M)$.
- There are easily (and universally) computable functions "$\otimes$" and "$\oslash$" which have the property that whenever $z_1 \in E_r(M_1)$ and $z_2 \in E_r(M_2)$ it is the case that $(z_1 \otimes z_2) \in E_r((M_1 + M_2) \bmod r)$ and $(z_1 \oslash z_2) \in E_r((M_1 - M_2) \bmod r)$.

2.1 Encryption

A randomized encryption function E_r satisfying the above properties can be developed as follows.

1. Select two "large" primes p and q such that r divides $(p-1)$, r and $(p-1)/r$ are relatively prime, and r and $(q-1)$ are relatively prime. (Such primes are easy to generate by searching among appropriate arithmetic sequences.) Form $n = pq$.
2. Select $y \in \mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$ such that $y^{(p-1)(q-1)/r} \bmod n \neq 1$.
3. Reveal n and y. The randomized encryption function E_r is defined by the set $E_r(M) = \{y^M u^r \bmod n : u \in \mathbb{Z}_n^*\}$.

It is now a trivial matter for a user given a message $M \in \mathbb{Z}_r$ and randomized function E_r to (by randomly and uniformly selecting $u \in \mathbb{Z}_n^*$) generate a uniform element $z = (y^M u^r \bmod n) \in E_r(M)$. Furthermore, this u serves as a certificate to prove that $z \in E_r(M)$.

To see that decryptions are unique, observe that $y^{M_1} u_1^r \bmod n = y^{M_2} u_2^r \bmod n$ implies that $y^{M_1 - M_2} \bmod n = (u_2 u_1^{-1})^r \bmod n$. By the construction of y, this, in turn, implies that $M_1 \bmod r = M_2 \bmod r$. It then follows immediately that the sets $E_r(0), E_r(1), E_r(2), \ldots, E_r(r-1)$ form a partition of $\mathbb{Z}_n^*$, and this gives the unique decryption property. Also, the two functions given by $z_1 \otimes z_2 = (z_1 \cdot z_2) \bmod n$ and $z_1 \otimes z_2 = (z_1 \cdot z_2^{-1}) \bmod n$ can easily be seen to satisfy the homomorphic properties described above.

In contrast, the original Goldwasser-Micali probabilistic encryption is essentially the special case of this method in which $r = 2$. However, since it is impossible for $r = 2$ to be relatively prime to $q - 1$ when q is a large prime, the Goldwasser-Micali system must restrict consideration to those elements of $\mathbb{Z}_n^*$ with a Jacobi symbol of $+1$. There is no need for such a restriction in the dense system described here.

2.2 Decryption

Security of decryption is based on the cryptographic assumption that deciding *higher residuosity* is computationally difficult: *given z, r, and n of unknown factorization, there is no known polynomial time algorithm to determine whether or not there exists an x such that $z = x^r \bmod n$.*

In the case where $n = pq$ is of the form described above and the prime factors p and q are known, the process of deciding higher residuosity is efficiently computable by the following simple rule:

$$z \in E_r(0) \qquad \text{if and only if} \qquad z^{(p-1)(q-1)/r} \bmod n = 1.$$

Thus, if r is small, one can simply decrypt a message z by determining (exhaustively) the smallest non-negative integer m such that $(y^{-m}z \bmod n) \in E_r(0)$.

This process can be further accelerated by pre-processing. For each $M \in \mathbb{Z}_r$, a canonical value T_M can be computed as

$$T_M = y^{M(p-1)(q-1)/r} \bmod n.$$

It can be shown that for every $z \in E_r(M)$, it is true that $z^{(p-1)(q-1)/r} \bmod n = T_M$. Thus, the r (distinct) values $T_0, T_1, \ldots, T_{r-1}$ can be pre-computed, and any encrypted value z can be decrypted by a table look-up on the value $z^{(p-1)(q-1)/r} \bmod n$.

If r is of moderate size, a combination of the two previous methods can be used to bring the storage, pre-computation time, and decryption time all to $\mathcal{O}(\sqrt{r})$. The idea (sometimes known as "big-step little-step" is to pre-compute T_M for $M \approx k\sqrt{r}$ as k ranges from 1 to $\sqrt{r}$. These values serve as milestones which are only about $\sqrt{r}$ steps apart. Given a z of unknown decryption, one can find the smallest non-negative integer m for which the T_M corresponding to $y^{-m}z$ has been pre-computed. This m is bounded above by $\sqrt{r}$ and can be regarded as an offset from the pre-computed decryption value T_M. The decryption of such a z is therefore the value $M + m$.

Finally, even if r is large, decryption is efficient provided that r contains no large prime factors. An extreme case in which decryption is very efficient is when r is of the form $r = 3^k$ for some positive integer k. In this case, the decryption of a value z can be computed in ternary notation. First, the low order "trit" t_k of the decryption of z is the unique value $t_k \in \{0,1,2\}$ such that $(y^{-t_k}z)^{(p-1)(q-1)/3} \bmod n = 1$. Once t_k has been computed, the next-to-last trit $t_{k-1} \in \{0,1,2\}$ is computed as the unique value such that $(y^{-t_k-3t_{k-1}}z)^{(p-1)(q-1)/3^2} \bmod n = 1$. Next, $t_{k-2} \in \{0,1,2\}$ is computed as the unique value such that $(y^{-t_k-3t_{k-1}-3^2t_{k-2}}z)^{(p-1)(q-1)/3^3} \bmod n = 1$. This process is continued until the ternary representation $\langle t_1, t_2, \ldots, t_k\rangle$ of the decryption of z is computed.

3 Some Applications

Besides the advantages of greater density in probabilistic encryption, there are some tasks which can be performed with the methods described here which cannot, in any apparent way, be done by any other means whatsoever.

3.1 Verifiable Secret Sharing

The notion of secret sharing was introduced by Shamir in [Sham79]. Shamir defines *threshold schemes* to be methods of dividing a secret value s into *shares* such that

(1) Any subset of shares which exceeds a predetermined size is sufficient to reconstruct the secret.

(2) Any smaller subset of shares gives no information (in an information theoretic sense) about the secret.

Shamir described a method of secret sharing based on polynomial interpolation and evaluation.

In 1985, Chor, Goldwasser, Micali, and Awerbach ([CGMA85]) described the problem of *verifiable* secret sharing. The problem here is to develop a protocol for secret sharing such that when it is complete each shareholder is confident that its share is meaningful. (Note that a dishonest secret sharer could give some shareholders worthless information rather than actual shares.)

Chor, Goldwasser, Micali, and Awerbach give a protocol which achieves verifiable secret sharing. However, their method is exponential in the number of shareholders.

The application of the encryption method described in this paper to the problem of verifiable secret sharing was first given in [Bena86] in which interactive proof techniques are also required. Feldman ([Feld87]), Ben-Or, Goldwasser, and Wigderson ([BGW88]), Rabin ([Rabi88]), and Rabin and Ben-Or ([RaBO89]) later expanded upon this approach.

The basic technique used in all of these methods is to perform computations on shares of secrets without first combining the shares to form the secrets. This is easily possible if the shares are encrypted using the dense probabilistic encryption method given. Computations on shares can be performed using the homomorphism properties described in the previous section.

One of the simplest of these verifiable secret sharing methods is formed by combining the encryption method described herein with Shamir's polynomial based secret sharing ([Sham79]) and the ideas of Feldman's non-interactive verifiable secret sharing ([Feld87]).

To distribute shares of a secret value $s \in \mathbb{Z}_r$ (r must be prime) to m participants such that any k of the m can determine the secret value, a randomized encryption function E_r is formed by the "dealer" of the secret. The dealer then randomly selects values $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_r$ and forms the polynomial

$$P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_2x^2 + a_1x + a_0$$

where the constant coefficient a_0 is given by $a_0 = s$. The dealer then forms m shares $s_i = P(i) \bmod n$ for $1 \leq i \leq m$ and privately distributes each share s_i to the i^{th} shareholder. Next, the dealer computes encryptions $z_j \in E_r(a_j)$ of the coefficients a_j for $0 \leq j < k$ and publicly reveals these encryptions. It is now possible for any and all participants to (by using the homomorphism properties) compute

$$w_i = \left(\prod_{j=0}^{k-1} z_j^{i^j} \right) \bmod n.$$

Since $z_j \in E_r(a_j)$, the additive homomorphism property of the cryptosystem implies that $(z_j)^\alpha \in E_r(a_j\alpha)$ for any scalar α. In particular, $z_j^{i^j} \in E_r(a_j i^j)$. Since the share $s_i = P(i) \bmod n = \left(\prod_{j=0}^{k-1} a_j i^j\right) \bmod n$, the additive homomorphism property also implies that each $w_i \in E_r(s_i)$. Thus, each w_i is a publicly computable encryption of the share s_i. The dealer can then privately distribute to the i^{th} shareholder a certificate u_i to prove that $w_i \in E_r(s_i)$. It is thereby impossible for the dealer to convince a shareholder that its share is legitimate when it is not. The secret value s can now be reconstructed at any subsequent time by any k of the shareholders. They need only pool their shares to interpolate the polynomial $P(x)$. The constant coefficient $P(0)$ is, by definition, the secret. This method has the added advantage that since shareholders have certificates of their shares, they cannot convincingly lie to each other about the values of their shares. Thus, dishonest shareholders cannot disrupt the reconstruction of the secret.

One final observation is that the dealer need not even be the creator of the randomized encryption function E_r. Careful examination of the process shows that the share certificates u_i, can be computed directly and simply from the random values the dealer chose to encrypt the coefficients. Hence, the dealer need not even be able to decrypt E_r to engage in this protocol.

It should be noted here that it is simply not possible to use ordinary Goldwasser-Micali probabilistic bit encryption for this application. With Shamir's method of secret sharing, the space from which secret and share values are chosen must be of a size which is prime and *greater than the number of shareholders.* Since probabilistic encryption methods will be used to encrypt shares, the Goldwasser-Micali bit encryption method is inadequate. (Note that segmenting share values bit by bit does not work since the Shamir scheme can only be applied when the number of shareholders is smaller than the secret/share space — which in this case is 2.)

3.2 Verifiable Secret-Ballot Elections

The problem of verifiable secret-ballot elections is defined in [Bena87], and a solution is presented there which depends strongly on the dense probabilistic encryption described here. The general problem is quite complex and its solution requires many techniques which are not addressed here. Instead of trying to present here a complete solution to the verifiable secret-ballot election problem, an overview will be given which demonstrates the use of dense probabilistic encryption.

The oversimplified scheme presented here has many omissions and should not be read as a claim of a secure method of holding verifiable secret-ballot elections. The reader is referred to [Bena87] for a complete treatment.

The scheme described in this section is centralized. A central *government* is assumed to exist. The government prepares a dense probabilistic encryption function E_r as described herein with r greater than the number of eligible voters and publicly reveals E_r.

Each voter selects either a random $v_i \in E_r(0)$ to denote a "no vote" or a random $v_i \in E_r(1)$ to denotes a "yes vote". Each voter then publicly releases its v_i. By the homomorphism properties, $W = (\prod v_i) \bmod n$ is an encryption of the *sum* of the unencrypted values. In this case, therefore, W is an encryption of the total number of yes votes cast by voters. Thus, the central government need only decrypt this one value W to determine the tally of the election; and by providing a certificate u of this decryption, the government can prove to all observers that the claimed tally is accurate.

There are, of course, many problems with this scheme. But this is the fundamental idea used in [CoFi85], [BeYu86], [Cohe86], [Bena87], and [BeTu94] to enable a variety of practical verifiable election schemes.

4 Conclusions

This paper has described a method of dense probabilistic encryption which has many similarities to, but many advantages over, the original method of probabilistic encryption introduced by Goldwasser and Micali. These advantages also apply relative to all previous methods of probabilistic encryption.

Many variations of this method are possible depending on the users' willingness to depend upon stronger assumptions in exchange for more efficient decryption. Applications of this method have also been given in which traditional probabilistic encryption methods are not just less efficient, but are instead unusable.

Acknowledgements

The author would like to express many thanks to Don Coppersmith, Mike Fischer, Neil Immerman, Joe Kilian, Dwight Tuinstra, and Moti Yung who offered helpful comments and suggestions regarding this work.

References

[Bena86] **Benaloh, J.** "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret." *Crypto '86*, Santa Barbara, CA (Aug. 1986).

[Bena87] **Benaloh, J.** "Verifiable Secret-Ballot Elections." Ph.D. Thesis presented at *Yale University*, New Haven, CT (Dec. 1987). (Available as *TR-561, Yale University, Department of Computer Science*, New Haven, CT (Sep. 1987).)

[BeTu94] **Benaloh, J.** and **Tuinstra, D.** "Receipt-Free Secret-Ballot Elections." To appear in *Proc.* 26th *ACM Symp. on Theory of Computing*, to be held in Montreal, PQ (May 1994).

[BeYu86] **Benaloh, J.** and **Yung, M.** "Distributing the Power of a Government to Enhance the Privacy of Voters." *Proc.* 5th *ACM Symp. on Principles of Distributed Computing*, Calgary, AB (Aug. 1986), 52–62.

[BGW88] **Ben-Or, M.**, **Goldwasser, S.**, and **Wigderson, A.** "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation." *Proc.* 20th *ACM Symp. on Theory of Computing*, Chicago, IL (May 1987), 1–10.

[CGMA85] **Chor, B.**, **Goldwasser, S.**, **Micali, S.**, and **Awerbuch, B.** "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 383–395.

[CoFi85] **Cohen, J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[Cohe86] **Cohen, J.** "Improving Privacy in Cryptographic Elections." *TR-454, Yale University, Department of Computer Science*, New Haven, CT (Feb. 1986).

[Feld87] **Feldman, P.** "A Practical Scheme for Non-interactive Verifiable Secret Sharing." *Proc.* 28th *IEEE Symp. on Foundations of Computer Science*, Los Angeles, CA (Oct. 1987), 427–437.

[GoMi84] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption." *J. Comp. Sys. Sci. 28*, (1984), 270–299.

[Rabi88] **Rabin, T.** "Robust Sharing of Secrets when the Dealer is Honest or Cheating." *TR 88-1, Hebrew University, Department of Computer Science*, Jerusalem, ISRAEL (July 1988).

[RaBO89] **Rabin, T.** and **Ben-Or, M.** "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority." *Proc.* 21st *ACM Symp. on Theory of Computing*, Seattle, WA (May 1989), 73–85.

[Sham79] **Shamir, A.** "How to Share a Secret." *Comm. ACM 22*, 11 (Nov. 1979), 612–613.

Simple and Effective Key Scheduling for Symmetric Ciphers (Extended Abstract)

Carlisle M. Adams
Bell-Northern Research

1. Introduction

Recently, a design procedure for private-key cryptosystems constructed as substitution-permutation networks (SPNs) was proposed [AT]. This procedure (known as the "CAST" design procedure) incorporates substitution-boxes (s-boxes) with fewer input bits than output bits and leads to a family of cryptosystems which are conceptually simple, are easily implemented, are very efficient in terms of encryption/decryption speed, and appear to have high security.

The focus of this paper is the design of a key scheduling algorithm for DES-like cryptosystems in general and for CAST cryptosystems in particular. Key scheduling is an area which has not seen as much activity as other aspects of cipher design, but which has taken on somewhat more significance in light of Biham's recent work on related-key cryptanalysis [B]. The key scheduling algorithm proposed in this paper is immune to related-key cryptanalysis and, furthermore, has a provable absence of weak and semi-weak keys [KRS, C, MSi] – to the authors' knowledge, this is the first time such a proof has been available for any cipher in the open literature.

The paper is organized as follows: Section 2 provides a brief overview of the design procedure given in [AT]; Section 3 gives a general discussion of key scheduling; and Section 4 gives proofs that the CAST key schedule will possess various cryptographic properties. Concluding comments are given in Section 5.

2. Overview of the CAST Design Procedure

The CAST design procedure for substitution-permutation network cryptosystems which incorporate $m \times n$ s-boxes, $m < n$, is as follows. Let $n=r*m$ (where r is an integer greater than 1), let $2n$ be the blocksize of the cryptosystem, and let s be a positive integer $\leq r$. The general framework of the algorithm is similar to the Data Encryption Standard: the plaintext is initially broken into halves of length n; at each round, one half is modified, is added modulo 2 to the other half, and the two halves are interchanged; after R rounds, the two halves are concatenated to form the ciphertext. However, the modification of a message half at each round is implemented quite differently from DES: here $r+s$ $m \times n$ substitution boxes from separate compatibility classes (see [A, AT]) are used, where each s-box is constructed according to the procedure given in [AT].

The (keyed) message half modification at each round is quite straightforward and is accomplished as follows. The subkey for a round is broken into s m-bit pieces, each piece is input to a separate $m \times n$ s-box, and the n-bit outputs are summed modulo 2 to form a mask for the message half. The masked n-bit message half is then broken into r m-bit pieces, each piece is input to a separate $m \times n$ s-box and the n-bit outputs are summed modulo 2 to form the n-bit modified message half.

2.1. AN EXAMPLE SPN CRYPTOSYSTEM

We have constructed an example substitution-permutation network based on the CAST design procedure to illustrate how it may be used. The network uses parameters m=8, n=32, r=4, and s=2; that is, it has a blocksize and keysize of 64 bits and uses six 8×32 s-boxes $S_1 ... S_6$ (four for the right half R_i and two for the subkey K_i at each round i). It uses 16 bits of key in each round and uses 8 rounds. At round i, the message half modification block f takes as input the four bytes of R_i and the subkey K_i:

$$f(R'_i) = \Sigma\, S_j(R'_{i,j}) \;\; for\; j=1 ... 4, \;\; where\; R'_i = R_i \oplus S_5(K_{i,1}) \oplus S_6(K_{i,2}).$$

(Note that $X_{i,j}$ denotes the j^{th} byte of X_i). Let $KEY = k_1k_2k_3k_4k_5k_6k_7k_8$, where k_i is the i^{th} byte of the input key. The key schedule used is as follows: $K_1=(k_1,k_2)$, $K_2=(k_3,k_4)$, $K_3=(k_5,k_6)$, $K_4=(k_7,k_8)$, $K_5=(k_4',k_3')$, $K_6=(k_2',k_1')$, $K_7=(k_8',k_7')$, $K_8=(k_6',k_5')$, where KEY is transformed to $KEY' = k_1'k_2'k_3'k_4'k_5'k_6'k_7'k_8'$ between round 4 and round 5. The transformation is defined by: $(k_1'k_2'k_3'k_4') = (k_1k_2k_3k_4) \oplus S5[k_5] \oplus S6[k_7]$; $(k_5'k_6'k_7'k_8') = (k_5k_6k_7k_8) \oplus S5[k_2'] \oplus S6[k_4']$. The bytes of KEY' are then used to construct the remaining four subkeys as shown above. It is not difficult to construct a similar schedule for a 12- or 16-round system.

3. COMMENTS ON THE KEY SCHEDULE

Keying is a crucial aspect of cryptosystem design. A key schedule should provide some guarantee of key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion in order to avoid certain key clustering attacks. Our approach to key scheduling is described above. Note that the s s-boxes used for keying (the "key schedule s-boxes") should be from separate compatibility classes and furthermore should be in separate classes from the other r s-boxes used in message half modification (so that there is no guaranteed cancellation of s-box outputs). Note as well that the $s*m$ key bits selected in round i should be different from the $s*m$ key bits selected in round $i+1$ (this is due to the work of Grossman and Tuckerman [GT], who showed that DES-like cryptosystems without a rotating key can be broken). Note finally that if any key bit is used in round R (the last round) for the first time then the network fails the key/ciphertext completeness test, since complementing that bit can only affect a single message half. All key bits must therefore be used by round R-1; in fact, we recommend that they be used by round $R/2$ and reused, after the key transformation given above, in the lower half of the network (this ensures good key avalanche for both encryption and decryption, as well as other properties discussed in Section 5).

The critical difference between the key schedule proposed in this paper and other schedules described in the open literature is the dependence upon substitution boxes for the creation of the subkeys. Other key schedules (the one in DES, for example) typically use a complex bit-selection algorithm to select the bits of the subkey for round *i* from the input key. As is clear from the work by Biham [B], any weaknesses in this bit selection algorithm can lead to simple cryptanalysis of the cipher, regardless of the number of rounds. The schedule proposed here instead uses a very simple bit-selection algorithm and a set of key schedule s-boxes to create the subkey for each round. These s-boxes must therefore be carefully designed to ensure cryptographically good key schedules (see [A], [AT] for descriptions of s-box design in CAST ciphers).

4. CRYPTOGRAPHIC PROPERTIES OF THE KEY SCHEDULE

In a block cipher, an *inverse key* H for a given encryption key K is defined to be a key such that $ENC_H(p) = ENC_{K^{-1}}(p) = DEC_K(p)$ for any plaintext vector p. Furthermore, a ***fixed point of a key*** K is a plaintext vector x such that $ENC_K(x) = x$.

From work done on cycling properties and key scheduling in DES [KRS, C, MSi], the following definitions have been introduced. A key is *weak* if it is its own inverse (such keys generate a palindromic set of subkeys and have 2^{32} fixed points in DES). A key is *semi-weak* if it is not weak but its inverse is easily found – there are two subclasses: a key is *semi-weak, anti-palindromic* if its complement is its inverse (such keys generate an anti-palindromic set of subkeys and have 2^{32} fixed points in DES); a key is *semi-weak, non-anti-palindromic* if its inverse is also semi-weak, non-anti-palindromic (such keys generate a set of subkeys with the property that $K_i \oplus K_{R+1-i} = V$, where R is the number of rounds and $V = 000...0111...1$ or $111...1000...0$ in DES). DES has 4 weak keys, 4 semi-weak anti-palindromic keys, and 8 semi-weak non-anti-palindromic keys.

Theorem: Ciphers using the key schedule proposed above can be shown to have ***no weak keys.***

Proof: There are two steps to this proof. In the first (general) step, we prove that for the transformation given in the key schedule of Section 2, weak keys (i.e., palindromic sets of subkeys) can arise if and only if $S5[k_5] \oplus S6[k_7] = P(S5[k_2'] \oplus S6[k_4'])$, where $P(\cdot)$ is the byte permutation $(b_1b_2b_3b_4) \rightarrow (b_2b_1b_4b_3)$. The second step, which is specific to each implementation of the CAST design, is to examine the specific s-boxes chosen in the implementation to verify that the equality does not hold (note that s-boxes satisfying this condition do exist). Details of this proof will be given in the full paper. ♦

In fact, a much stronger result can be stated, as follows.

Theorem: Ciphers using the key schedule proposed in this paper can be shown to have ***no inverse key*** $H \in \{0,1\}^{64}$ for any key $K \in \{0,1\}^{64}$ (therefore, such ciphers have ***no semi-weak keys***).
Proof: There are two steps to this proof. Let $S5[k_2'] \oplus S6[k_4']$ be equal to the 4-byte vector $(a_1a_2a_3a_4)$ and let $S5[k_5] \oplus S6[k_7]$ be equal to the 4-byte vector $(b_1b_2b_3b_4)$. In the first (general) step, we prove that for the transformation given in the key schedule of Section 2, inverse keys exist if and only if $a_1=a_2$, $a_3=a_4$, $b_1=b_2$, *and* $b_3=b_4$ all simultaneously hold. The second step, which is specific to each implementation of the CAST design, is to examine the specific s-boxes chosen in the implementation to verify that the equalities do not hold simultaneously (note that s-boxes satisfying this condition do exist). Details of this proof will be given in the full paper. ♦

Open Question: Ciphers using the CAST key schedule have ***no fixed points for any key.***
Discussion: From the above two theorems, these ciphers avoid both palindromic and anti-palindromic sets of subkeys. They therefore are guaranteed to avoid the fixed points associated with weak and semi-weak keys. It appears to be unknown in the open literature whether the existance of such keys is a necessary condition for fixed points in DES-like cryptosystems. Even if it is not a necessary condition, however, it is not clear that any other type of subkey set would produce the large number (2^{32}) of fixed points that result from palindromic and anti-palindromic subkey sets.

The above properties of the key schedule are due to the fact that s-boxes are employed in the schedule itself (i.e., in the *generation* of the subkeys), rather than simply in the *use* of the subkeys. To the authors' knowledge, this is a novel proposal in key scheduling which appears to have some appealing benefits.

Theorem: CAST-designed ciphers are ***immune to related-key cryptanalysis.***
Proof: There are no related keys in the key schedule described in this paper (i.e., the derivation algorithm of the subkeys from the previous subkeys is not the same in all rounds because of the transformation step), and so ciphers using this key schedule are not vulnerable to the "chosen-key-chosen-plaintext", "chosen-key-known-plaintext", or "chosen-plaintext-unknown-related-keys" attacks [B]. Furthermore, the CAST procedure has no known complementation properties (unlike DES, for example) and so appears not to be vulnerable to reduced key searches based on this type of weakness. ♦

5. Conclusions

The CAST design procedure has a theoretical framework (because of the use of bent vectors) and is both feasible (since bent vectors of the necessary size can be generated easily; see [N, P, AT2], for example) and simple (since all phases of the design are defensible and readily understandable). Furthermore, it produces ciphers which possess a number of important cryptographic properties, have good real-time performance, and can be implemented easily in both software and hardware.

This paper has focused on the design of a simple and effective key scheduling algorithm for CAST and other symmetric block ciphers. The proposed schedule appears to meet the needs of cryptographic security (absence of rotating subkeys, related subkeys, weak keys and semi-weak keys) while maintaining conceptual simplicity and ease of implementation. It may therefore serve as a good starting point for further research in this area.

References

[A] C. M. Adams, *A Formal and Practical Design Procedure for Substitution-Permutation Network Cryptosystems*, Ph.D. Thesis, Department of Electrical Engineering, Queen's University, 1990.

[AT] C. M. Adams and S. E. Tavares, *Designing S-Boxes for Ciphers Resistant to Differential Cryptanalysis*, Proceedings of the 3rd Symposium on the State and Progress of Research in Cryptography, Rome, Italy, Feb., 1993, pp.181-190.

[AT2] C. M. Adams and S. E. Tavares, *Generating and Counting Binary Bent Sequences*, IEEE Transactions on Information Theory, vol. IT-36, 1990, pp.1170-1173.

[B] E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys,* in Advances in Cryptology: Proc. of Eurocrypt '93 (to appear) – see the Pre-Proceedings, pp.W124-W143.

[C] D. Coppersmith, *The Real Reason for Rivest's Phenomnenon,* in Advances in Cryptology: Proc. of CRYPTO '85, Springer-Verlag, New York, 1986, pp.535-536.

[GT] E. Grossman and B. Tuckerman, *Analysis of a Feistel-Like Cipher Weakened by Having No Rotating Key*, Technical Report RC 6375, IBM, 1977.

[KRS] B. S. Kaliski Jr., R. L. Rivest, and A. T. Sherman, *Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES),* Journal of Cryptology, vol.1-1, 1988, pp.3-36.

[MSi] J. H. Moore and G. J. Simmons, *Cycle Structure of the DES with Weak and Semi-Weak Keys,* in Advances in Cryptology: Proc. of CRYPTO '86, Springer-Verlag, New York, 1987, pp.9-32.

[N] K. Nyberg, *Constructions of bent functions and difference sets*, in Advances in Cryptology: Proc. of Eurocrypt '90, Springer-Verlag, 1991, pp.151-160.

[P] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, *Propagation characteristics of boolean functions*, in Advances in Cryptology: Proc. of Eurocrypt '90, Springer-Verlag, Berlin, 1991, pp.161-173.

Key Clustering in Substitution-Permutation Network Cryptosystems

H. M. Heys and S. E. Tavares
Department of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada K7L 3N6
email: tavares@ee.queensu.ca

Abstract — In this paper we examine the key clustering characteristics of a class of block cryptosystems referred to as substitution-permutation networks or SPNs. Specifically, we investigate the relationship between the property of key avalanche and the success of a key clustering attack. Further, we develop an analytical model of the key avalanche property and use this to estimate a lower bound on the complexity of a key clustering attack as a function of the number of rounds of substitutions.

I. Introduction

Substitution-permutation networks (SPNs) evolved from the work of Shannon [1] and Feistel [2] and form the foundation for many modern private key block cryptosystems such as DES [3], FEAL [4], and LOKI [5]. Such cryptosystems belong to the class of product ciphers which obtain their cryptographic strength by iterating a cryptographic operation several times. The basic SPN consists of a number of rounds of nonlinear subsitutions connected by bit position permutations. The substitutions are performed by dividing the block of bits into small sub-blocks and using a mapping stored as a table lookup and referred to as an S-box. It has been shown that this basic SPN structure can be used to construct ciphers which possess good cryptographic properties such as completeness [6] and resistance to differential and linear cryptanalysis [7][8].

We shall consider a general N-bit SPN as consisting of R rounds of $n \times n$ S-boxes. The plaintext and ciphertext are N-bit vectors denoted as $\mathbf{P} = [P_1\ P_2\ ...\ P_N]$ and $\mathbf{C} = [C_1\ C_2\ ...\ C_N]$, respectively. An S-box in the network is defined as an n-bit bijective mapping $S\ :\ \mathbf{X} \rightarrow \mathbf{Y}$ where $\mathbf{X} = [X_1\ X_2\ ...\ X_n]$ and $\mathbf{Y} = [Y_1\ Y_2\ ...\ Y_n]$. We shall assume that an S-box is keyed by XORing n bits of key with the S-box input vector, $\mathbf{X}$, before the substitution operation is performed. Hence, the network is keyed from a τ-bit key $\mathbf{K} = [K_1\ K_2\ ...\ K_\tau]$ by XORing N bits of the key with the network bits before each round of substitutions. The method for determining where each key bit is applied in the network is referred to as the key scheduling algorithm. Decryption is performed by running the data "backwards" through the network (i.e., applying the key scheduling algorithm in reverse and using the inverse S-boxes). A simple example of an SPN cryptosystem with $N = 16$, $n = 4$, and $R = 4$ is illustrated in Figure 1.

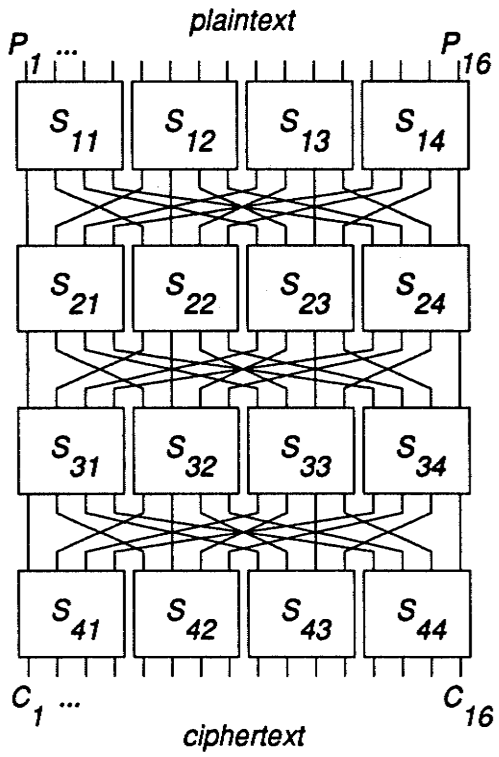

Figure 1. Simple SPN with $N = 16$, $n = 4$, and $R = 4$

II. Key Clustering Attacks

A block cryptosystem is considered weak if keys which are close to each other in Hamming distance result in a number of corresponding ciphertexts which are also close in distance. For example, consider two keys, $\mathbf{K}'$ and $\mathbf{K}''$, for which $wt(\Delta\mathbf{K} = \mathbf{K}' \oplus \mathbf{K}'')$ is small where $wt(\cdot)$ represents the Hamming weight of the specified argument, $\oplus$ is the bit-wise XOR operator, and Δ is used to indicate the XOR difference of the specfied vector. The encryption of l plaintexts, $\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_l$, under the two different keys results in l ciphertext pairs, $(\mathbf{C}'_1, \mathbf{C}''_1), (\mathbf{C}'_2, \mathbf{C}''_2), ..., (\mathbf{C}'_l, \mathbf{C}''_l)$. If there are a number of ciphertexts that are close in distance due to the proximity of the two keys — for example, if $E\{wt(\Delta\mathbf{C} = \mathbf{C}' \oplus \mathbf{C}'')\}$ is small where $E\{\cdot\}$ is the expectation operator — then we refer to the cryptosystem as having key clustering.

Key clustering can be exploited by a cryptanalyst to improve upon an exhaustive key search. Such an attack requires an appropriate number of known plaintexts to be able to determine whether a key is close to the correct key. The cryptanalyst proceeds by randomly selecting and testing keys until a key is found that is in the neighborhood of the correct key. Once such a key is found, the cryptanalyst can test all keys within a suitable distance of this key until the correct key is established. As an example, consider a cryptosystem with a key of 64–bits and which has the property that keys within distance 5 of the actual key (about 2^{23} keys) result in ciphertexts that are (on average) close enough to the actual ciphertexts to be distinguishable using only about 1000 known plaintext-ciphertext pairs. By randomly executing about $2^{64}/2^{23} = 2^{41}$ trials of

1000 encryptions we expect to be able to discover a key in the neighorhood (i.e., within distance 5) of the actual key. Testing all keys in the neighborhood of this experimental key (about 2^{23} encryptions) will reveal the correct key. As a result, the complexity of the key clustering attack is approximately $(1000)2^{41} + 2^{23} \approx 2^{51}$ which is a significant improvement on the complexity of about 2^{64} required for an exhaustive key search.

III. Key Avalanche Property

The relationship between the key avalanche property of a cryptosystem and key clustering attacks was noted in [9]. In this section we develop a model for the key avalanche property of an SPN as a function of the number of rounds of substitutions.

(a) Definitions

Consider the following definition of key avalanche.

Definition 1: A cryptosystem is said to satisfy the *key avalanche criterion* if each ciphertext bit changes with a probability of 1/2 when a single key bit is changed.

This definition is analagous to the definition of the strict avalanche criterion (SAC) for a cryptosystem [10] which refers to the probability of a ciphertext bit change given a one bit plaintext change. The key avalanche criterion, of course, refers to key rather than plaintext changes. As well, we can extend Definition 1 to consider the effect of changes involving more than one key bit.

Definition 2: A cryptosystem is said to satisfy the *extended key avalanche criterion order* κ if each ciphertext bit changes with a probability of 1/2 when a set of κ key bits are changed.

Let the *key avalanche probability*, p_{ka}, represent the probability that a particular ciphertext bit changes given a particular set of κ key bit changes. Ideally, we desire a cryptosystem to exactly satisfy the extended key avalanche criterion and $p_{ka} = 1/2$ for any ciphertext bit and set of κ key bits. In reality, cryptosystems will likely only approximately satisfy the criterion and the key avalanche probability can be represented as

$$p_{ka} = (1/2) - \epsilon. \tag{1}$$

We refer to ϵ as the *key avalanche error* and note that the value of $|\epsilon|$ is typically very small. A key clustering attack may be mounted against a cryptosystem which has poor extended key avalanche characteristics (i.e., a large value for $|\epsilon|$). The complexity of the key clustering attack is essentially a product of the number of key trials required to find a key in the neighborhood of the correct key and the number of known plaintexts required to determine that a test key is in the neighborhood. Cryptosystems with large values of κ and large corresponding values of $|\epsilon|$ require few key tests before finding a key close to the actual key and few known plaintexts to determine that a tested key is in the neighborhood of the actual key and, hence, are susceptible to key clustering attacks.

(b) Network Model Assumptions

An analytical model for the strict avalanche characteristics of SPNs is presented in [11]. We now extend the methodology to develop a model of the key avalanche characteristics for a one bit key change, i.e., $\kappa = 1$. The model approximates each S-box in the network as a stochastic mapping and calculates the key avalanche probability for each round recursively assuming a one bit key change.

We shall assume that the cryptosystem of interest is an N-bit block cryptosystem constructed using $n \times n$ S-boxes such that $N = n^2$. For example, a 64–bit SPN constructed using 8×8 S-boxes is a practical cryptosystem that satisfies these constraints. The network of Figure 1 is also a simple illustration of such a network with $N = 16$ and $n = 4$.

In the model, each S-box behaves as a random variable selected uniformly from the set of possible bijective mappings. Hence, an input change to an S-box results in a number of output changes represented by the random variable $D = wt(\Delta\mathbf{Y})$ where all possible values of $\Delta\mathbf{Y}$ belonging to the set of $2^n - 1$ non-zero changes are equally likely. Therefore, the probability distribution of D is given by

$$P_D(D=0) = \begin{cases} 1 & , wt(\Delta\mathbf{X}) = 0 \\ 0, & , wt(\Delta\mathbf{X}) \geq 1 \end{cases} \tag{2}$$

and

$$P_D(D=d) = \begin{cases} 0 & , wt(\Delta\mathbf{X}) = 0 \\ \frac{\binom{n}{d}}{2^n-1} & , wt(\Delta\mathbf{X}) \geq 1 \end{cases} \tag{3}$$

for $1 \leq d \leq n$. For ease of notation, we will represent $P_D(D = d)$ by $P_D(d)$.

We assume that the network uses a simple, effective permutation defined by bit i of the output of round r being connected to bit j of the input of round $r + 1$ such that

$$j = n \cdot ((i-1) \ \ mod \ \ n) + \lfloor (i-1)/n \rfloor + 1. \tag{4}$$

This permutation belongs to the class of permutations identified Kam and Davida [6] as providing provable completeness in networks for which $N = n^R$. Ayoub [12] further identified this permutation as belonging to a class of permutations cryptographically equivalent to Kam and Davida's structure.[1]

The cryptosystem that we shall consider in the model is keyed using $\tau = N$ key bits which are all applied at each round by XORing with the S-box input bits. We assume that the ordering of the key bits at the input to each round can be represented as a random variable. Hence, the model determines probabilities by averaging over all possible placements of the one bit key change in each round.

[1] The class of permutations identified by Ayoub are particularly useful for networks of an arbitrary number of rounds because they are optimal (in the sense that they provide completeness in the fewest number of contiguous rounds) and are easy to implement since they allow the use of the same permutation for each round.

(c) Computation of Key Avalanche Property

The recursive model is based on finding the probability distribution of the number of S-boxes in a round which have changes at their outputs given the probability distribution for the previous round when a one bit key change is applied. From the probability distribution of the number of S-boxes with output changes, it is possible to derive, under the assumptions of the model, a probability distribution of the number of bit changes at the output of a particular round. From this, the expected number of bit changes and, hence, the key avalanche probability are easily determined.

Let W_r represent the random variable corresponding to the number of bit changes (caused by the complementation of one key bit) after round r, i.e.,

$$W_r = \sum_{s=1}^{n} wt(\Delta \mathbf{Y}_{rs}) \tag{5}$$

where $\Delta \mathbf{Y}_{rs}$ is the output change of an S-box numbered s, $1 \leq s \leq n$, in round r, $1 \leq r \leq R$. Assuming symmetry in the location of a key bit change and the resulting output bit changes, the key avalanche probability after r rounds corresponding to any ciphertext bit and key bit change is given by the expected value of the number of bit changes divided by the block size, i.e., $p_{ka} = E\{W_r/N\}$. Therefore, we are interested in determining the probability distribution $P(W_r)$. For notational convenience, we let $P(W_r = w_r) \equiv P(w_r)$.

Let L_r be a random variable representing the number of S-boxes in round r which have changes at their outputs, i.e., $L_r = \#\{s \mid wt(\Delta \mathbf{Y}_{rs}) \neq 0\}$. The probability distribution of bit changes for the output of round r may be determined from

$$P(w_r) = \sum_{l_r=0}^{n} P(w_r \mid l_r) \cdot P(l_r) \tag{6}$$

where the variable l_r represents a particular value of the random variable L_r. Since the single key bit change must cause a change in the output of one, and only one, of the first round S-boxes, we can write the first round probability distribution of L_r as

$$P(L_1 = l_1) = \begin{cases} 1 & , l_1 = 1 \\ 0 & , l_1 \neq 1. \end{cases} \tag{7}$$

Consequently, the probability of l_{r+1} S-boxes in round $r+1$ with output changes may be determined recursively using

$$P(l_{r+1}) = \sum_{l_r=0}^{n} P(l_{r+1} \mid l_r) \cdot P(l_r). \tag{8}$$

In order to determine $P(W_r)$ and, subsequently, the expected number of bit changes, we must therefore derive expressions for $P(w_r \mid l_r)$ and $P(l_{r+1} \mid l_r)$.

Consider first the determination of $P(w_r \mid l_r)$. Let $\mathbf{d} = [d_1\ d_2\ ...\ d_{l_r}]$ where $d_i \in \{1,...,n\}$ is the number of output changes, $wt(\Delta\mathbf{Y})$, of the i-th S-box of round r that has an output change. Define

$$\Lambda = \left\{ \mathbf{d} \;\middle|\; \sum_{i=1}^{l_r} d_i = w_r \right\} \tag{9}$$

to represent the values of $\mathbf{d}$ for which there are a total of w_r bit changes at the output of round r. The probability of w_r output bit changes given that l_r S-boxes have output bit changes is given by

$$P(w_r \mid l_r) = \sum_{\mathbf{d}\in\Lambda} P(\mathbf{d}) \tag{10}$$

where

$$P(\mathbf{d}) = \prod_{i=1}^{l_r} P_D(d_i). \tag{11}$$

Consider now the determination of $P(l_{r+1} \mid l_r)$. Let t be the number of S-boxes in round $r+1$ that do not have any input changes and let b be the number of S-boxes in round $r+1$ that have input changes of one bit only, i.e., $t = \#\{s \mid wt(\Delta\mathbf{X}_{(r+1)s}) = 0\}$ and $b = \#\{s \mid wt(\Delta\mathbf{X}_{(r+1)s}) = 1\}$. In order to determine $P(l_{r+1} \mid l_r)$, we initially consider the joint probability of b and t given l_r S-boxes with output changes in round r, $P(b,t \mid l_r)$.

Define the probability $P(\delta \mid \mathbf{d})$ to be the probability that at least δ particular S-boxes in round $r+1$ are not affected by input changes given that a specific $\mathbf{d}$ occurs. Further, let $P(\rho \mid t, \mathbf{d})$ represent the probability that at least ρ particular S-boxes have only one input bit changing given that t S-boxes do not have any input changes and a specific $\mathbf{d}$ occurs. Letting $\mathrm{T} = \{1,...,n\}^{l_r}$, the probability of interest is given by

$$\begin{aligned} P(b,t \mid l_r) &= \sum_{\mathbf{d}\in\mathrm{T}} P(b,t,\mathbf{d}) \\ &= \sum_{\mathbf{d}\in\mathrm{T}} P(b \mid t,\mathbf{d}) \cdot P(t \mid \mathbf{d}) \cdot P(\mathbf{d}) \\ &= \sum_{\mathbf{d}\in\mathrm{T}} \sum_{\rho=b}^{n-t} (-1)^{\rho-b} \binom{\rho}{b} \binom{n-t}{\rho} P(\rho \mid t,\mathbf{d}) \\ &\qquad \cdot \sum_{\delta=t}^{n} (-1)^{\delta-t} \binom{\delta}{t} \binom{n}{\delta} P(\delta \mid \mathbf{d}) \cdot P(\mathbf{d}). \end{aligned} \tag{12}$$

Equation (12) is derived by considering the application of the extension of the inclusion-exclusion principle [13, p.271] in order to determine $P(b \mid t, \mathbf{d})$ and $P(t \mid \mathbf{d})$.

The probability $P(\mathbf{d})$ is determined as in equation (11) and from [11], we have

$$P(\delta \mid \mathbf{d}) = \prod_{i=1}^{l_r} \frac{\binom{n-\delta}{d_i}}{\binom{n}{d_i}}. \tag{13}$$

We shall determine the probability $P(\rho \mid t, \mathrm{d})$ in the following manner. Without loss of generality, consider that the last t S-boxes in round $r+1$ are the S-boxes which do not have any input changes. Let N_t represent the number of arrangements for the output bit changes of round r satisfying d such that the last t S-boxes in round $r+1$ have no input changes and the remaining $n-t$ S-boxes each have one or more input bit changes. Hence, N_t can be determined by computing the number of arrangements with exactly zero of the remaining $n-t$ S-boxes having no input changes. Using the inclusion-exclusion principle this is given by

$$N_t = \sum_{\mu=0}^{n-t} (-1)^{\mu} \binom{n-t}{\mu} \prod_{i=1}^{l_r} \binom{n-t-\mu}{d_i}. \tag{14}$$

Further, assume that the first ρ S-boxes in round $r+1$ have only one input change. Define the vector $\mathrm{h} = [h_1\ h_2\ ...\ h_{l_r}]$ where $h_i = \{0, ..., \rho\}$ represents the number of outputs of the i-th S-box in round r with output changes which provide an input change to the first ρ S-boxes. Hence, $\mathrm{h} \in \mathrm{H}$ where

$$\mathrm{H} = \left\{ \mathrm{h} \mid \sum_{i=1}^{l_r} h_i = \rho \right\}. \tag{15}$$

Let N_ρ represent the number of arrangements of the S-boxes in round r which originate input changes to the first ρ S-boxes. Hence

$$N_\rho = \rho! / \left[\prod_{i=1}^{l_r} h_i! \right]. \tag{16}$$

Lastly, define N_γ as the number of valid arrangements of bit changes such that the remaining $n-t-\rho$ S-boxes in round $r+1$ have one or more bit changes at their input. Once again applying the inclusion-exclusion principle, this is given by

$$N_\gamma = \sum_{\mu=0}^{n-t-\rho} (-1)^{\mu} \binom{n-t-\rho}{\mu} \prod_{i=1}^{l_r} \binom{n-t-\rho-\mu}{d_i - h_i}. \tag{17}$$

The probability $P(\rho \mid t, \mathrm{d})$ can now be computed as

$$P(\rho \mid t, \mathrm{d}) = \sum_{\mathrm{h} \in \mathrm{H}} N_\rho N_\gamma / N_t. \tag{18}$$

Using the probability $P(b, t \mid l_r)$ it is possible to compute $P(l_{r+1} \mid l_r)$ by determining the expected result given that a key bit change is randomly XORed to one input bit of round $r+1$. If we ignore the effect of the key bit change, the number of S-boxes with output changes is simply given as $l_{r+1} = n - t$. However, in determining the effect of the key bit change we must consider the following three cases, their probability of occurrence, and their resulting implications:

1. Key change XORed with bit from S-box with no input changes (probability = t/n) $\Rightarrow$ $l_{r+1} = n - t + 1$.
2. Key change XORed with bit from S-box with one input change (probability = b/n) $\Rightarrow$ $l_{r+1} = n - t - 1$ with probability $1/n$, or $l_{r+1} = n - t$ with probability $(n-1)/n$.
3. Key change XORed with bit from S-box with more than one input change (probability = $1 - t/n - b/n$) $\Rightarrow$ $l_{r+1} = n - t$.

Hence, given $P(b, t \mid l_r)$, we can compute $P(l_{r+1} \mid l_r)$ by:

$$\begin{aligned} P(l_{r+1} = n - t + 1 \mid l_r) &= \sum_{b=0}^{n-t} \frac{t}{n} \cdot P(b, t \mid l_r) \\ P(l_{r+1} = n - t - 1 \mid l_r) &= \sum_{b=0}^{n-t} \frac{b}{N} \cdot P(b, t \mid l_r) \\ P(l_{r+1} = n - t \mid l_r) &= \sum_{b=0}^{n-t} \left(1 - \frac{t}{n} - \frac{b}{N}\right) \cdot P(b, t \mid l_r). \end{aligned} \tag{19}$$

Using this analysis, we have estimated the key avalanche probability for a 64–bit SPN and, subsequently, determined the key avalanche error ϵ as a function of the number of rounds. The results are listed in the second column of Table 1.

IV. Methods of Determining Close Keys

Using the key avalanche model presented in the previous section, we can now determine the security of an SPN against exploitation of weak key avalanche for a key clustering attack as a function of the number of rounds of substitutions. In this section we examine two methods for determining that an experimental key is close to the actual key. Specifically, for each method we determine the number of known plaintexts, $\mathcal{N}_P$, required to reveal that two keys are close to each other.

In developing a lower bound on the complexity of the key clustering attack, we only consider the number of known plaintexts required to determine if an experimental key is close to the actual key; we do not consider how many trials are required before we expect to pick a close experimental key. Hence, although $\mathcal{N}_P$ gives an approximate lower bound on the complexity of the key clustering attack, in practice, the complexity of the attack will be much higher than $\mathcal{N}_P$ since it will typically take a large number of trials before a selected key is close to the actual key. In the approach we make the reasonable assumption that the magnitude of the key avalanche error $|\epsilon|$ is maximized for $\kappa = 1$.

(a) Ciphertext Correlation

In a cryptosystem with weak key avalanche, an obvious method for determining whether an experimental key is close to the actual key is to search for correlation in the ciphertext output bits. If there is a high enough degree of correlation it is likely that the experimental key is a small Hamming distance from the actual key.

The problem may be considered to be a hypothesis testing problem with one hypothesis, H_0, being that the test key comes from the neighborhood (i.e., for $\kappa = 1$, within one bit) of the correct key and the other hypothesis, H_1, being that the key is not in the neighborhood of the correct key. Let ϵ_R represent the value of the key avalanche error for $\kappa = 1$ after R rounds of substitutions. Assume that the probability that a ciphertext bit changes under hypothesis H_0 or H_1 is given by $p_0 = 1/2 - \epsilon_R$ or $p_1 = 1/2$, respectively[2]. Let η represent the number of samples of ciphertext bit changes required to test a key and, hence, the number of known plaintexts required to test a key is given by $\mathcal{N}_P = \eta/N$. The number of bit changes in η ciphertext bit change samples follows the binomial distribution for each hypothesis. Therefore the expected number of bit changes and variances are given by

$$\begin{aligned} H_0 : \quad & \mu_0 = \eta/2 - \eta\epsilon_R, \quad && \sigma_0^2 = \eta\left(1/4 - \epsilon_R^2\right) \\ H_1 : \quad & \mu_1 = \eta/2, && \sigma_1^2 = \eta/4. \end{aligned} \tag{20}$$

Since ϵ_R is typically very small, $\epsilon_R^2 \ll 1/4$ and $\sigma_0^2 \approx \sigma_1^2 = \eta/4$. Therefore, let $\sigma^2 = \eta/4$ represent the variance of both hypothesis distributions.

Since η is typically large, the binomial distribution for each hypothesis may be approximated as a Gaussian distribution with the means, μ_0 and μ_1, and variance σ^2. For convenience, we shall assume that the acceptable probability of error in selecting a hypothesis is the same for both H_0 and H_1. Hence, considering the symmetry of the hypotheses, we require η large enough so that

$$\mu_0 + \alpha\sigma \approx \mu_1 - \alpha\sigma \tag{21}$$

with the significance level α selected to provide a suitably small probability of error in the hypothesis test where the probability of error is given by

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-x^2/2} dx. \tag{22}$$

Hence, $\eta\epsilon_R - \sqrt{\eta}\alpha \approx 0$ and, consequently, $\eta \approx (\alpha/\epsilon_R)^2$.

For an R round SPN, the number of known plaintexts required to test a key is, therefore,

$$\mathcal{N}_P \approx \frac{\alpha^2}{N\epsilon_R^2} \tag{23}$$

where α is selected to provide a suitably small probability of error in the hypothesis test.

[2] Note that $p_1 = 1/2$ implies that $\epsilon = 0$ for $\kappa \neq 1$. In practice, the key avalanche error ϵ for different values of $\kappa \neq 1$ would not necessarily be exactly zero. However, since the assumption results in a lower bound on the security analysis, it is therefore suitable for our purposes.

(b) Meet-in-the-Middle Correlation

Similarly to the ciphertext correlation approach we may consider identifying close keys by using an experimental key to encrypt the known plaintexts for the first $R/2$ rounds and to decrypt the known ciphertexts for the last $R/2$ rounds. This generates two values for a middle block of N bits which can be checked for correlation. Let $\epsilon_{R/2}$ represent the value of ϵ for $R/2$ rounds. If $\epsilon_{R/2}$ is large, then the two sets of middle bits are significantly correlated to the actual bits and, therefore, are highly likely to be the same. We refer to the correlation of the two sets of middle bits as meet-in-the-middle correlation. Note that it is not necessary or possible for the cryptanalyst to know the actual middle bits.

Let $p_{R/2} = 1/2 - \epsilon_{R/2}$ represent the probability that a middle bit is different than the actual middle bit given that the experimental key selected is within distance one of the actual key. Assume that the key avalanche probability is the same backwards and forwards. The probability that two experimental middle bits are the same, p_M, is given by the probability that both bits are correct or that both bits are incorrect. That is,

$$\begin{aligned} p_M &= \left(p_{R/2}\right)^2 + \left(1 - p_{R/2}\right)^2 \\ &= 1/2 + 2\epsilon_{R/2}^2. \end{aligned} \tag{24}$$

As before, we define hypothesis H_0 to be that an experimental key is close to the actual key and hypothesis H_1 to be that it is not. The expected number of ciphertext bit changes and variances for each hypothesis are

$$\begin{aligned} H_0 &: \quad \mu_0 = \eta/2 - 2\eta\epsilon_{R/2}^2, \quad \sigma_0^2 = \eta\left(1/4 - 4\epsilon_{R/2}^4\right) \\ H_1 &: \quad \mu_1 = \eta/2, \qquad\qquad\quad \sigma_1^2 = \eta/4 \end{aligned} \tag{25}$$

Using an analysis similar to the previous case of ciphertext correlation, for an R round SPN, we may determine the number of known plaintexts required to test a key to be

$$\mathcal{N}_P \approx \frac{\alpha^2}{4N\epsilon_{R/2}^4} \tag{26}$$

where α is selected to provide a suitably small probability of error in the hypothesis test.

V. Results

Clearly the advantage of using one form of the attack over the other depends on the relative values of ϵ_R and $\epsilon_{R/2}$. For an SPN with $N = 64$ and a key size of $\tau = 64$, using the values of the key avalanche error determined by the model of Section III and presented in the second column of Table 1, we have calculated the number of known plaintexts required in order to test a key to determine whether it is within distance one of the actual key as a function of the number of substitution rounds. The results for both methods of determining close keys are presented

Rounds R	Key Avalanche Error ϵ	Ciphertext Correlation $\mathcal{N}_P$	Meet-in-Middle Correlation $\mathcal{N}_P$
1	4.37×10^{-1}	5	-
2	2.21×10^{-1}	20	7
3	2.98×10^{-2}	1124	-
4	1.59×10^{-3}	3.98×10^{5}	105
5	6.75×10^{-5}	2.20×10^{8}	-
6	2.55×10^{-6}	1.54×10^{11}	3.16×10^{5}
7	9.09×10^{-8}	1.21×10^{14}	-
8	3.12×10^{-9}	1.03×10^{17}	3.96×10^{10}
9	1.05×10^{-10}	9.16×10^{19}	-
10	3.45×10^{-12}	8.42×10^{22}	1.21×10^{16}

Table 1. Results for SPN with $N = 64$, $n = 8$, and $\alpha = 8$

in columns 3 and 4 of Table 1. The significance level for the hypothesis test was selected to be $\alpha = 8$. Note that it can be shown [14, p.569] that $Q(\alpha) \leq e^{-\alpha^2/2}/2$ and, hence, although increasing the value of α does not significantly change the value of $\mathcal{N}_P$, it does significantly decrease the likelihood of an error in the hypothesis test.

From the table we can determine the number of rounds required by an SPN in order to provide a level of security against key clustering equivalent to exhaustive key search. The results suggest that the meet-in-the-middle approach requires fewer known plaintexts to identify a close key. For both methods, when $R = 10$, the number of plaintexts required to test a key satisfies $\mathcal{N}_P \gtrsim 2^{53}$. Combining this bound on $\mathcal{N}_P$ with the number of trials required to select a key close to the actual key results in a complexity much greater than the 2^{64} key trials required in exhaustive key search. We conclude that a 64–bit 10–round SPN with a 64–bit key and 8×8 S-boxes is expected to be unbreakable using a key clustering attack exploiting a key avalanche weakness. Further, since the complexity of the attack is likely to be far greater than $\mathcal{N}_P$, our analysis suggests that, in practice, an 8–round SPN with $\mathcal{N}_P \gtrsim 2^{35}$ will have adequate resistance to key clustering.

VI. Conclusion

We have presented an analysis of the relationship between the key avalanche property and key clustering. Using a stochastic model of the key avalanche property we are able to determine the minimum number of rounds required for an SPN to ensure that a key clustering attack, exploiting weak key avalanche, will fail.

References

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.

[2] H. Feistel, "Cryptography and computer privacy," *Scientific American*, vol. 228, no. 5, pp. 15–23, 1973.

[3] "National Bureau of Standards - Data Encryption Standard," *Federal Information Processing Standard Publication 46*, 1977.

[4] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm: FEAL," *Advances in Cryptology: Proceedings of EUROCRYPT '87*, pp. 267–278, 1988.

[5] L. Brown, J. Pieprzyk, and J. Seberry, "LOKI - a cryptographic primitive for authentication and secrecy applications," *Advances in Cryptology: Proceedings of AUSCRYPT '90*, pp. 229–236, 1990.

[6] J. B. Kam and G. I. Davida, "A structured design of substitution-permutation encryption networks," *IEEE Transactions on Computers*, vol. 28, no. 10, pp. 747–753, 1979.

[7] L. J. O'Connor, "On the distribution of characteristics in bijective mappings," *Advances in Cryptology: Proceedings of EUROCRYPT '93*, pp. 360–370, 1994.

[8] H. M. Heys and S. E. Tavares, "The design of product ciphers resistant to differential and linear cryptanalysis," *presented at CRYPTO '93*, Santa Barbara, Calif., Aug. 1993.

[9] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, vol. 67, no. 3, pp. 397–427, 1979.

[10]A. F. Webster and S. E. Tavares, "On the design of S-boxes," *Advances in Cryptology: Proceedings of CRYPTO '85*, pp. 523–534, 1986.

[11]H. M. Heys and S. E. Tavares, "Avalanche characteristics of a class of product ciphers," tech. rep., Department of Electrical Engineering, Queen's University, Aug. 30, 1993.

[12]F. Ayoub, "The design of complete encryption networks using cryptographically equivalent permutations," *Computers and Security*, vol. 2, pp. 261–267, 1982.

[13]F. S. Roberts, *Applied Combinatorics*. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

[14]S. S. Haykin, *Digital Communications*. New York: Wiley, 1988.