A Construction of a Cipher From a Single Pseudorandom Permutation

Shimon Even[1] and Yishay Mansour[2]

Abstract

We suggest a scheme for a block cipher which uses only one randomly chosen permutation, F. The key, consisting of two blocks, K_1 and K_2 is used in the following way: The message block is XORed with K_1 before applying F, and the outcome is XORed with K_2, to produce the cryptogram block. We show that the resulting cipher is secure (when the permutation is random or pseudorandom). This removes the need to store, or generate a multitude of permutations.

[1]Comp. Sci. Dept., Technion, Israel Institute of Technology, Haifa, Israel 32000. Supported by the Fund for the Promotion of Research at the Technion, and by Bellcore, Morristown, NJ. E-address: even@cs.technion.ac.il

[2]Computer Science Dept., Tel-Aviv University. Part of the work was done while the author was in IBM T.J. Watson Research Center.

1 Introduction

Shannon defined a random cipher as a collection of randomly chosen permutations, one for each value of the key. Following Shannon, [1], a *cipher*, $\mathbf{C}$, consists of:

1. A finite set of *messages* (and cryptograms), $\mathbf{M}$.
2. A finite set of *keys*, $\mathbf{K}$.
3. Each key $\kappa \in \mathbf{K}$ is assigned a permutation Π_κ: $\mathbf{M} \mapsto \mathbf{M}$.

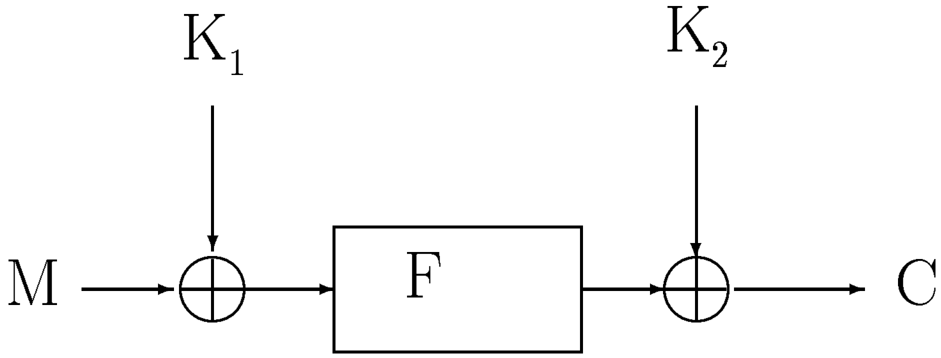
If each of the permutations Π_κ is chosen randomly, with uniform probability, from the set of all $|\mathbf{M}|!$ permutations, then $\mathbf{C}$ is called a *random cipher*. If $\mathbf{M} = \{0,1\}^n$ then $\mathbf{C}$ is called a *block-cipher*. One might view DES [3] as an attempt to realize an approximation of a random block-cipher.

We propose a block-cipher scheme where only one permutation is randomly chosen and used. This permutation is publicly accessible (as a black-box), and anyone that tries to attack the cipher has access to it. The security is proved under the assumption that the permutation is a random permutation (or at least a pseudo random one) and the only way to access it is through a black box.

The key K, which consists of two blocks, K_1 and K_2, is used in the following way: The message block is XORed with K_1, then we apply F, and the outcome is XORed with K_2, to produce the cryptogram block. To decode the cryptogram we perform the reverse order of steps, using the inverse permutation F^{-1}.

As one can immediately see, the key modifies the permutation in a very simple and fast to implement way. Note that the encryption of the messages for different keys are not independent. Thus, in the conceptual level, our system is not a random cipher, and is not even an attempt to approximate one. Yet we show that from the point of view of an efficient adversary the situation is similar to the one when a random block-cipher is used; i.e. the probability of the adversary to crack the system is negligible.

As in many other applications, if instead of choosing a permutation at random we choose it pseudorandomly, every efficient attack on the system



still has a negligible probability to succeed. It is easy to see that if one has a family of pseudorandom permutations then it can be used to build a secure cipher; simply choose for each key one of the permutations of the family. The novelty of our scheme is that we use only one pseudorandom permutation.

2 The Scheme

Let $\{0,1\}^n$ denote the set of binary words of length n, let F be a common and publicly known permutation on $\{0,1\}^n$ and F^{-1} be its inverse. It is assumed that, for any given $x \in \{0,1\}^n$, it is easy to get $F(x)$ or $F^{-1}(x)$, either by a direct computation or by using an easily and commonly accessible black-box (oracle).

A *key* K consists of two subkeys, K_1 and K_2, each chosen at random from $\{0,1\}^n$. Initially, it is assumed that the key is known to the legitimate parties only; all other parties have no knowledge about it. Also, it is assumed that the key remains fixed and is used, by the legitimate parties, to encrypt messages and decrypt cryptograms, repeatedly, for a relatively long time.

The *encryption* (or cryptogram) $E_K(M)$ of a *message* $M \in \{0,1\}^n$ by the key $K =< K_1, K_2 >$, is performed by

$$E_K(M) = F(M \oplus K_1) \oplus K_2,$$

where $\oplus$ denotes the bit-by-bit exclusive-or operation.

The *decryption* of a *cryptogram* $C \in \{0,1\}^n$, is performed by

$$D_K(C) = F^{-1}(C \oplus K_2) \oplus K_1.$$

It is easy to verify, that for every $M \in \{0,1\}^n$ and $K \in \{0,1\}^{2n}$, the following identity holds $D_K(E_K(M)) = M$, i.e. D_K decrypts messages that where encrypted using E_K.

Before we continue showing the security of our scheme it would be worthwhile to point out why simple variants on the same idea do not work. For example, if $E(M) = F(K \oplus M)$, since the adversary has access to the permutation F, $F^{-1}(E(M)) = K \oplus M$, which is clearly insecure. Another example is $E(M) = K \oplus F(M)$, then the adversary can recover the key by having access to a message M and its encryption $E(M)$, since it may access the permutation with M, receive $F(M)$, and compute $K = E(M) \oplus F(M)$.

3 Definitions of Security and their relations

The two most important applications of conventional crypto-systems are concealment of messages from eavesdroppers and authentication of the identity of correspondents. We assume that F is a random (i.e. randomly chosen), or pseudorandom permutation, and investigate the security of the system from these two points of view.

Our aim is to model an adversary who, for a while, can get the system to encrypt messages and decrypt cryptograms for him, as well as use the permutation F, as a black-box, in both directions (i.e. F and F^{-1}), but has no direct access to the key K. For this reason the access of the adversary to this operations is modeled by oracles, which only answer queries of a particular nature.

Of the different notions of security we discuss two, analogous to the above applications. We show that for our system, these two notions (defined as problems) are hard to solve. We believe that this demonstrates the security of our system.

The *cracking problem*, CP, is an attempt (by an adversary) to decode a given encryption $C_0 = E_K(M_0)$, without any a priori knowledge of the key K. The algorithm, employed by the adversary, has access to the following four oracles:

1. *F-oracle:* Presented with $x \in \{0,1\}^n$, the oracle supplies $F(x)$.

2. F^{-1}*-oracle:* Presented with $x \in \{0,1\}^n$, the oracle supplies $F^{-1}(x)$.

3. E*-oracle:* Presented with $M \in \{0,1\}^n$, the oracle supplies $E_K(M)$.

4. *(C_0-Restricted) D-oracle:* Presented with $C \in \{0,1\}^n$ (such that $C \neq C_0$) the oracle supplies $D_K(C)$.

The algorithm is successful if it outputs $M_0 = D_K(C_0)$. The *success probability* of the algorithm is the probability that on a randomly chosen encryption $C_0 = E_K(M_0)$ it outputs M_0, where all C_0 are equally likely.

The cracking problem is sometimes called *chosen plaintext/ciphertext attack* or *two-sided attack*.

In the *existential forgery problem*, EFP, the adversary has access to four oracles: F-oracle, F^{-1}-oracle, E-oracle and (unrestricted) D-oracle. The latter is defined as follows: Presented with any $C \in \{0,1\}^n$, the oracle supplies $D_K(C)$. The task is to find a *new* pair $< M, C >$, $C = E_K(M)$; i.e. a pair which does not consist of a query and an answer, as previously supplied by either the E-oracle or the D-oracle. Actually, EFP is the authentication problem. A forgery here mean, intuitively, that the adversary can send messages as though it was one of the parties in the protocol.

Let $f(n)$ be a function defined from the positive integers to the interval $[0,1]$. We say that $f(n)$ is *polynomially negligible* if for every polynomial $p(n)$ there is an n_0, such that if $n > n_0$ then $f(n) < 1/p(n)$.

Assume that the adversary employs a randomized algorithm, whose time is polynomially bounded (in n), to solve one of these problems. We say that the problem is *hard* if, for every such algorithm, the success probability is polynomially negligible; the probability is taken over all choices made in the design of the system (i.e. the choice of F), the keys chosen by the user and coin-flips performed by the adversary algorithm.

Instead of proving that CP and EFP are hard separately, we first reduce the EFP to CP; i.e. we show that the existence of a (successful) CP attack implies the existence of an EFP attack. The following theorem, which is a folklore theorem and is given here for the sake of completeness, shows the reduction from EFP to CP holds for *any* block cipher scheme.

Theorem 3.1 *Let $t : \mathcal{N} \rightarrow \mathcal{N}$ and $\epsilon : \mathcal{N} \rightarrow [/, \infty]$. If there exists a CP attack which runs in time $t(n)$ and its success probability is $\epsilon(n)$, then there is an EFP attack which runs in time $t(n)$ and its success probability is $\epsilon(n)/t(n)$.*

Proof: We show how to construct an attacking algorithm A, for the EFP, making use of a cracking procedure P. Fix $n \in \mathcal{N}$ for which P performs the CP (Cracking Problem) attack in time $t(n)$ and its success probability is $\epsilon(n)$.

We use the fact that the range of the encryption is samplable; for our encryption scheme this holds trivially, since the range is $\{0, 1\}^n$.

Also, we may assume, without loss of generality, that P queries the E-oracle on its output; i.e. the cracking procedure P, while attempting to decrypt its input C_0, checks the correctness of its output M, by feeding M to the E-oracle, and comparing $E_K(M)$ with C_0. If P inverts a cryptogram C_0 successfully, then there is a *critical* time $i \leq t(n)$, such that at time i, P queries the E-oracle about M_0, and at no prior time M_0 has been queried.

We construct A as follows.

1. Randomly, choose a cryptogram C_0.
2. Feed it, as an input, to P.
3. Randomly, with uniform distribution, choose $1 \leq \tau \leq t(n)$.
4. Let P run exactly $\tau - 1$ steps. If at the τ'th step, P queries the E-oracle about a value M', output $< M', C_0 >$ (without querying the E-oracle).

The probability that A generates a legitimate pair, i.e. that $M' = D_K(C_0)$, is at least $\epsilon(n)/t(n)$. This follows from the fact that with probability $\epsilon(n)$, P inverts C_0 successfully, and with probability $1/t(n)$, P has been stopped at the critical time. $\square$

From the previous theorem we immediately deduce the following corollary.

Corollary 3.2 *If for every polynomial-time EFP attack the success probability is polynomially negligible, then for every polynomial-time CP attack the success probability is polynomially negligible.*

In general the converse of Theorem 3.1 does not hold for any block cipher, since, for example, there might be a message whose encryption is always known, regardless of the key. (For example, in RSA when $M = 1$.)

4 The immunity of the system when F is truly random

In this section we assume that the permutation F is a truly random permutation, i.e. it has been chosen randomly, out of the set of all $2^n!$ permutations, where all permutations are equally likely to be picked. In addition, the key K is chosen uniformly from $\{0,1\}^{2n}$ and remains fixed. Under this assumption, we show that solving EFP for our system is *hard*; i.e. that for every polynomial-time EFP attack on our system, the success probability is polynomially negligible. By Corollary 3.2, solving CP is also hard. In fact we prove a stronger result, namely, any algorithm that makes only a polynomial number of queries has an exponentially small probability of success, regardless of its running time.

Recall that the adversary asks queries of two forms:

1. *E/D queries.* For a given message M_i, the E-oracle returns $E_K(M_i)$, or for a given a cryptogram $E_K(M_i)$, the D-oracle returns M_i. We say that $< M_i, E_K(M_i) >$ is an *E-pair.*
2. *F/F^{-1} queries.* For a given value A_j, the F-oracle returns $F(A_j)$, or for a given $F(A_j)$, the F^{-1}-oracle returns A_j. We say that $< A_j, F(A_j) >$ is an *F-pair.*

An algorithm A for EFP asks various queries of the four types and then computes a new E-pair $< M, E_K(M) >$ (i.e. an E-pair which was not generated through the E/D-oracles). We show that for every algorithm A, if it asks only polynomially many queries, then its probability to succeed is exponentially small. The probability is taken over the random choices of the algorithm and the choice of the random permutation F and the key K.

The following notion is central to the analysis of the scheme. Consider two pairs (candidates to be E-pairs), $< M_1, C_1 >$ and $< M_2, C_2 >$. If either $M_1 = M_2$ or $C_1 = C_2$, then we say that the two pairs *overlap*. Two overlapping pairs $< M_1, C_1 >$ and $< M_2, C_2 >$ are *identical* when $M_1 = M_2$ and $C_1 = C_2$. Note that the replies of the (genuine) E-oracle and D-oracle are such that if two E-pairs overlap then they are identical. Thus, without loss of generality, we may assume that all queries are non-overlapping. The definition of 'overlap' and 'identical' for F-pairs is similar.

The basic outline of the proof is the following. We define a set of "good" keys, and show that after each query, with high probability, almost all the keys $K =< K_1, K_2 >$ are "good" and are equally likely to be the encryption key. We use this to argue that the adversary cannot "know" which of the keys is the "true" one. This leads to the observation, that as long as we know that the encryption key is from that "good" set of keys, we can choose each time a random key from this set, and the adversary will not be able to notice this, since the probability distribution that we will generate is identical to the true one. Finally, when the adversary outputs a pair $< M, C >$ his chances of being successful are very small, since there are still too many "good" keys.

First, we investigate the number of possible choices of the key, $K =< K_1, K_2 >$, and the amount of freedom in the choice of the permutation F, such that these choices are consistent with a given set of query-answer pairs.

We start by defining when the first subkey K_1 is bad. This definition depends only on the sets of E-pairs and F-pairs generated by the queries. Consider all E-pairs, $< M_i, C_i >$. Since they are non-overlapping, all M_i's are different, and therefore all the corresponding values $M_i \oplus K_1$ (potential inputs to F) are different. Also, since all F-pairs, $< A_j, B_j >$, are non-overlapping, all A_j's are different. (But this does not exclude the possibility that $M_i \oplus K_1 = A_j$, for some i and j.)

We say that first subkey K_1 is *bad*, with respect to a set S of E-pairs and a set T of F-pairs if there is an E-pair $< M_i, C_i > \in S$ and an F-pair $< A_j, B_j > \in T$, such that $M_i \oplus K_1 = A_j$; otherwise the first subkey K_1 is *good*. Similarly, we say that a second subkey K_2 is *bad*, with respect to a set S of E-pairs and a set T of F-pairs, if there is an E-pair $< M_i, C_i > \in S$ and an F-pair $< A_j, B_j > \in T$, such that $B_j \oplus K_2 = C_i$; otherwise the second subkey K_2 is good. A key $K =< K_1, K_2 >$ is good if both the first subkey K_1 is good and the second subkey K_2 is good. Note that if all queries have been answered "honestly", with respect to a fixed pair $< K, F >$, where

$K =< K_1, K_2 >$, then K_1 is good iff K_2 is good.

Lemma 4.1 *Let S be a set of non-overlapping E-pairs and T a set of non-overlapping F-pairs. Then, the number of bad first (second) subkeys, with respect to S and T, is at most lm, where $l = |S|$ and $m = |T|$.*

Proof: A first subkey K_1 is bad if there are queries i and j such that, $M_i \oplus K_1 = A_j$, or alternately, $K_1 = A_j \oplus M_i$. Therefore, at most lm subkeys K_1 are bad. A similar argument holds for the second subkey K_2. □

Lemma 4.1 implies that for every set of l non-overlapping E-pairs and m non-overlapping F-pairs, there are at least $2^n - lm$ good first subkeys K_1 and as least $2^n - lm$ good second subkeys K_2, which implies that there are at least $2^{2n} - 2lm2^n$ good keys $K =< K_1, K_2 >$, or, alternatively, the fraction of bad keys is at most $\frac{2lm}{2^n}$.

We define a permutation Π to be a $< K, S, T >$ *extension*, where $K =< K_1, K_2 >$ is a key, S is a set of non-overlapping E-pairs and T is a set of non-overlapping F-pairs, if, for each $< A, B >\in T$ it holds that $\Pi(A) = B$, and for each $< M, C >\in S$ it holds that $\Pi(M \oplus K_1) = C \oplus K_2$. We say that a pair $< K, \Pi >$, where $K =< K_1, K_2 >$, is consistent with a set S of E-pairs if for every $(M, C) \in S$, $C = \Pi(M \oplus K_1) \oplus K_2$. In a similar way, a pair $< K, \Pi >$, where $K =< K_1, K_2 >$, is consistent with a set T of F-pairs if for every $(A, B) \in T$, $\Pi(A) = B$. We say that a pair $< K, \Pi >$, where $K =< K_1, K_2 >$, is *consistent with all query-answer pairs* if it is consistent both with the set of E-pairs and the set of F-pairs.

Lemma 4.2 *Let S be a set of non-overlapping E-pairs and T a set of non-overlapping F-pairs. Let $K =< K_1, K_2 >$ be a good key with respect to S and T. For any permutation Π which is a $< K, S, T >$ extension, the pair $< K, \Pi >$ is consistent with all query-answer pairs.*

Proof: Consider the set V of pairs $< x, y >$, such that $x, y \in \{0, 1\}^n$ and either $< x, y >$ is equal to one of the given F-pair (i.e. $< x, y >\in T$), or $M_i \oplus K_1 = x$ and $y \oplus K_2 = C_i$, for one of the given E-pairs $< M_i, C_i >\in S$. Since K is good with respect to S and T, both K_1 and K_2 are good, hence, every two pairs in the set V share neither their first component, nor their second component. Since Π is a $< K, S, T >$ extension, it is consistent with the set V, i.e. $\Pi(x) = y$, for every $< x, y > \in V$. Therefore, $< K, \Pi >$ is consistent with both the F-pairs and the E-pairs. $\square$

The next lemma states that all the good keys are, in some sense, equally likely.

Lemma 4.3 *Let S be a set of non-overlapping E-pairs and T a set of non-overlapping F-pairs. The probability that the encryption key is κ, given S and T, is the same for any key κ which is good with respect to S and T.*

Formally, given that the probability over the keys is uniform (i.e. $Prob_{K,F}[K = \kappa] = 1/2^{2n}$, for any $\kappa \in \{0,1\}^{2n}$) and the probability over the permutations is uniform (i.e. $Prob_{K,F}[F = \pi] = 1/(!2^n)$, for any permutation π), then

$$Prob_{K,F}[K = \kappa \mid < K, F > \text{ is consistent with } S, T]$$

is the same for any key $\kappa \in \{0,1\}^{2n}$ which is good with respect to S and T.

Proof: We are interested in computing the probability of a good key κ, given that the sets S and T are consistent. By Bayes formula, this is equivalent to the probability of S and T given that the key is κ, times the a priori probability that the key is κ divided by the probability that a random $< K, F >$ is consistent with S and T. Formally,

$$Prob_{K,F}[K = \kappa \mid < K, F > \text{ is consistent with } S, T] =$$
$$\frac{Prob_{K,F}[K = \kappa] \times Prob_{K,F}[< K, F > \text{ is consistent with } S, T \mid K = \kappa]}{Prob_{K,F}[< K, F > \text{ is consistent with } S, T]}$$

From the hypothesis of the lemma we have that $Prob_{K,F}[K = \kappa] = 1/2^{2n}$, for any $\kappa \in \{0,1\}^{2n}$. Furthermore, the probability of S and T being consistent a priori, i.e., $Prob_{K,F}[< K, F > \text{ is consistent with } S, T]$, is also the same for all keys, since this event does not depend on κ. Therefore, we should focus on the probability of S and T being consistent given that the key is κ, i.e. $Prob_{K,F}[< K, F > \text{ is consistent with } S, T \mid K = \kappa]$. Since we are given the key $\kappa = < K_1, K_2 >$, we can transform the E-pairs to restrictions on the permutation F. Similarly to Lemma 4.2, let,

$$S' = \{< M \oplus K_1, C \oplus K_2 > \mid < M, C > \in S\}.$$

(Note that if the messages where encrypted using $< K, F >$ and $< x, y > \in S'$ then $F(x) = y$.) Clearly, $|S| = |S'|$ since there is a one to one mapping between the pairs in S and S'. Since the key κ is good, it implies that there is no overlap between S' and T, i.e. $S' \cap T = \phi$. Therefore, the probability of S and T being consistent, given that the key is κ is the probability that the permutation F obeys the $|T| + |S|$ constraints in $V = S' \cup T$, which is

$$Prob_F[\forall < x, y > \in V : \quad F(x) = y] = \Pi_{i=0}^{|S|+|T|-1} \frac{1}{2^n - i}.$$

This probability does not depend on the particular choice of κ, therefore it is identical for all good keys, which completes the proof of the lemma. $\square$

Now we are ready to show the main theorem.

Theorem 4.4 *The probability of an algorithm A to solve the EFP problem, when F and K are chosen randomly and uniformly, is bounded by*

$$O(\frac{lm}{2^n}) \tag{1}$$

where l is the number of E/D queries and m is the number of F/F^{-1} queries.

Proof: Given an algorithm A that can solve the EFP problem, we would like to bound the success probability of A. The algorithm A asks queries of various types and get replies from the appropriate oracle using a pair $< K, F >$.

In our analysis we plan to deviate from this way of generating the replies. First let us define how we plan to reply to A's queries. Latter we justify the way that we generate the answers to the queries, and at the end we compute an upper bound on the success probability of the algorithm A.

Initially we choose a pair $L_0 =< K, F >$, where $K =< K_1, K_2 >$, with a uniform distribution. Let $S^{(i)}$ be the set of E-pairs and $T^{(i)}$ be the set of F-pairs up to the ith query of A. After the ith query of A, we choose, at random, a good key $K^{(i)} =< K_1^{(i)}, K_2^{(i)} >$ with respect to $S^{(i)}$ and $T^{(i)}$, and a random permutation $F^{(i)}$ which is a $< K^{(i)}, S^{(i)}, T^{(i)} >$ extension. Let $L_i =< K^{(i)}, F^{(i)} >$.

When A performs its ith query, we reply using L_{i-1}. If the ith query of A caused the key $K^{(i-1)}$ to become bad, then we stop and declare the algorithm A successful, otherwise we continue. (Therefore we can assume that a key $K^{(i-1)}$ remains good, then all the previous keys $K^{(j)}$ where also good.)

By the assumption of the theorem, after $l + m$ queries the algorithm A outputs a pair $< M, C >$, and it is successful if $F^{(l+m)}(M \oplus K_1^{(l+m)}) \oplus K_2^{(l+m)} = C$; otherwise it fails.

We first need to justify why the way that we generate the oracle replies is valid. Later, in order to compute the success probability of A, we bound two probabilities: the probability that a query will cause a good key to become bad, and the probability that A outputs a consistent pair $< M, C >$ given that the key is a random good key.

By Lemma 4.3 any good key K has the same probability given the sequence of queries and replies to the oracles so far. This implies that the posterior probabilities, given the conversation, is uniform over all the good keys. In addition, by Lemma 4.2, any good key K with a permutation which is a $< K, S^{(i)}, T^{(i)} >$ extension, are consistent with all previous query-answer pairs. Therefore, when we choose after each query a new random pair L_i, we are consistent with all previous replies, and we generate the same distribution.

To be more precise, consider the distribution of conversations of A. One way to sample this distribution is to chose a random key K (from the prior distribution over keys, which is uniform) a random permutation F, and generate the conversation with A using K and F. An alternative way is, after each query of A to sample a new key $K^{(i)}$, using the posterior distribution over the keys (i.e., the conditional distribution of the keys given the conversation), and a new permutation $F^{(i)}$, which is chosen from the posterior distribution (i.e., the conditional distribution of the permutations given the conversation and $K^{(i)}$). By the definition of the posterior probability, those two methods would generate the same distribution, and thus they are indistinguishable. We chose to consider the second way of generating conversations, however we add another twist. At certain events (namely, if we chose a bad key) we decided that the algorithm A is successful and do not continue. Clearly this can only increase the probability of A to win.

For any fixed i, now we can compute the probability that the key $K^{(i)} =<$

$K_1^{(i)}, K_2^{(i)} >$ becomes bad during the $i+1$th query. We need to consider the four different types of queries which could have been the ith query.

Consider an E-query on the message M_i. If there is a previous E-pair $< M_i, C >$, then, since L_i is consistent with all the E-pairs, the reply will be consistent with $< M_i, C >$, i.e. it will be C. In this case, clearly, the key $K^{(i)}$ remains good.

Assume the E-query on the message M_i is non-overlapping with the previous E-pairs. Such a query adds at most m bad first subkeys K_1 and m bad second subkeys K_2. By Lemma 4.1, the number of good keys is at least $2^{2n} - 2lm2^n$. Thus, no matter how A chooses the query M_i, his chance of making $K^{(i)}$ bad is bounded by

$$\frac{2m2^n}{2^{2n} - 2lm2^n} = \frac{2m}{2^n - 2lm}.$$

The analysis for a query to the D oracle is almost identical, yielding exactly the same bound. In the case of an F query, or an F^{-1} query, a similar analysis yields the bound

$$\frac{2l}{2^n - 2lm}.$$

It follows that the probability A makes $K^{(i)}$ bad by a single E/D query is bounded by $\frac{2m}{2^n-2lm}$, and the probability of this happening during any of the l E/D queries is bounded by $\frac{2lm}{2^n-2lm}$. The same expression bounds the probability of making $K^{(i)}$ bad by any of the m F/F^{-1} queries. Thus, the probability that any $K^{(i)}$ is bad is bounded by

$$\frac{4lm}{2^n - 2lm} = O(\frac{lm}{2^n}).$$

From now on we assume that none of the $K^{(i)}$ becomes bad.

Consider the situation after $l+m$ queries, when A outputs a pair $< M, C >$. We compute an upper bound on the probability that the pair is "correct", i.e. $F^{(l+m)}(M \oplus K_1^{(l+m)}) \oplus K_2^{(l+m)} = C$. The probability that $< M, C >$ makes $K^{(l+m)}$ bad is, using the previous argument, is bounded by

$$\frac{2m}{2^n - 2lm} = O(\frac{lm}{2^n}).$$

Now we bound the probability that $C = F(M \oplus K_1^{(l+m)}) \oplus K_2^{(l+m)}$ given that $K^{(l+m)} =< K_1^{(l+m)}, K_2^{(l+m)} >$ is a good key with respect to $S^{(l+m)} \cup \{< M, C >\}$ and $T^{(l+m)}$. Since $K^{(l+m)}$ is good, it implies that there is no F/F^{-1}-pair $< A, B >$ with either $A = M \oplus K_1^{(l+m)}$, or $B = C \oplus K_2^{(l+m)}$. Now we consider the probability that $F(M \oplus K_1^{(l+m)}) = C \oplus K_2^{(l+m)}$.

Since we have not committed to the value of F at $M \oplus K_1^{(l+m)}$ either directly (using the l F/F^{-1}-pair) or indirectly (using the m E/D-pair), there are exactly $2^n - (m + l)$ possible values for $F(M \oplus K_1^{(l+m)})$, and hence the probability that it is $C \oplus K_2^{(l+m)}$ is

$$\frac{1}{2^n - m - l} = O(\frac{lm}{2^n}).$$

The theorem follows from summing the probability that $K^{(l+m)}$ is bad and the probability that A's output is correct, given that $K^{(l+m)}$ is good. $\square$

From the above theorem we deduce the following important corollary.

Corollary 4.5 *Consider our system, with a randomly chosen F. For every polynomially bounded algorithm to solve EFP, the probability of success is polynomially negligible.*

In general, Theorem 4.4 implies a lower bound of $2^{n/2}$ on the cracking problem. Thus for this scheme to be secure, n must be chosen so as to make $2^{n/2}$ steps infeasible. We mention that J. Daemen [2] investigated the question whether the bound in Theorem 4.4 is tight. He shows that our scheme, which uses $2n$ key-bits, can be cracked in time $O(2^n)$, using Known Plaintext Attack, and in space and time $O(2^{n/2})$, using Chosen Plaintext Attack.

5 The immunity of the system when F is pseudorandom

Our system remains secure even if F is just known to be chosen pseudorandomly. As before, we assume that the adversary has access to the four oracles, but has no access to the innards of the box which implements F. This issue is meaningless when F is chosen randomly, but when it is chosen pseudorandomly, the difference may be crucial.

Our claims are restricted to the oracle-type attacks. The reason why our result extends to a pseudo-random permutations is that if an efficient algorithm violates the immunity of the system when F is pseudorandom, then this algorithm can be transformed into a distinguisher of pseudorandom F's from random ones (since for the latter the immunity must hold – by Theorem 4.4).

Theorem 5.1 *If F is chosen pseudorandomly and K is chosen uniformly, then for every polynomially bounded algorithm to solve EFP for our system, the probability of success is polynomially negligible.*

Acknowledgments

The authors would like to thank Reuven Bar-Yehuda, Shai Ben-David, Benny Chor, Guy Even, Oded Goldreich, Refael Heiman and Silvio Micali.

References

[1] C.E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Tech. J.*, Vol. 28, 1949, pp. 656-715.

[2] J. Daemen, "Limitations of the Even-Mansour Construction", Proceedings of *AisaCrypt*, 1991.

[3] National Bureau of Standards, "Data Encryption Standard", *Federal Information Processing Standard*, U.S. Department of Commerce, FIPS PUB 46, Washington, DC, 1977.

[4] M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions", *SIAM J. on Computing*, Vol. 17, No. 2, 1988, pp. 373-386.