Towards Making Luby-Rackoff Ciphers Optimal and Practical

Sarvar Patel[1], Zulfikar Ramzan[2]*, and Ganapathy S. Sundaram[1]

[1] Bell Labs, Lucent Technologies,
67 Whippany Road,
Whippany, NJ 07981,
{sarvar, ganeshs}@bell-labs.com

[2] 'Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge MA 02139,
zulfikar@theory.lcs.mit.edu

Abstract. We provide new constructions for Luby-Rackoff block ciphers which are efficient in terms of computations and key material used. Next, we show that we can make some security guarantees for Luby-Rackoff block ciphers under much weaker and more practical assumptions about the underlying function; namely, that the underlying function is a secure Message Authentication Code. Finally, we provide a SHA-1 based example block cipher called Sha-zam.

1 Introduction

The design of block ciphers whose security provably relies on a hard underlying primitive has been a popular area of contemporary cryptographic research. The path breaking paper of Luby and Rackoff [7] described the construction of pseudorandom permutation generators from pseudorandom function generators, which enabled the formalism of the notion of a block cipher. This theoretical breakthrough has stimulated a lot of research and ciphers based on this principle are often termed *Luby-Rackoff Ciphers*. Recall that block ciphers are private-key cryptosystems with the property that the length of the plaintext and ciphertext blocks are the same. Pseudorandom permutations can then be interpreted as block ciphers that are secure against adaptive chosen plaintext and ciphertext attacks. These permutations are closely related to the concept of pseudorandom functions which was defined by Goldreich, Goldwasser and Micali (GGM) [5]. These are functions which are "indistinguishable" from random functions in polynomial time. The GGM construction relied on the notion of pseudorandom bit generators, i.e., bit generators whose output cannot be distinguished from a sequence of random bits by any efficient method.

* Work done while this author was at Lucent Technologies

L. Knudsen (Ed.): FSE'99, LNCS 1636, pp. 171–185, 1999.

The core of a Luby-Rackoff cipher is a Feistel network. The use of Feistel networks for cipher design is not new, in fact it is one of the central design principles of DES. The original construction of Luby and Rackoff consists of four Feistel permutations, each of which requires the evaluation of a pseudorandom function. The proofs of security that were provided were subsequently simplified by Maurer [9] where he provided a rather generalized treatment based on information theoretic (as opposed to complexity theoretic) ideas. In what follows we review some of the more popular results in this field.

We start with the problem of producing a $2n$ bit pseudorandom permutation from n bit pseudorandom functions. Luby and Rackoff used the method of Feistel Networks. Clearly, from an information theoretic point of view, one would need at least two rounds of n bit pseudorandom functions (whose entropy is n bits) since the entropy of the permutations required is $2n$ bits. Now we check if two rounds are enough. Here Luby and Rackoff showed that if only two rounds were used and if an attacker chose two different inputs with the same "right half of n bits" then he can easily distinguish the outputs from a truly random permutation. Hence they suggested the use of at least three rounds, which is secure against chosen plaintext attacks. As it turned out, three rounds was not resistant to adaptive chosen ciphertext attacks, and in fact they showed that four rounds was sufficient to guarantee resistance to adaptive chosen plaintext and ciphertext attacks. The proof involved choosing four different pseudorandom functions in the four rounds.

Following this work, and the paper of Maurer [9], Lucks further generalized the proofs to include *unbalanced Feistel networks*. The main contribution of his work is the notion of a *difference concentrator* which is a non-cryptographic primitive that replaces the pseudorandom function in the first round but yet offers the same security. Parallel to this, a lot of research has concentrated on obtaining variants of Luby-Rackoff constructions where the number of different pseudorandom functions used in the four rounds is minimized. For example, see [12], [16]. This minimization is motivated by the fact that pseudorandom functions are computationally intensive to create and hence any reduction in the number of different functions used directly leads to a more efficient construction. Following these works, Naor and Reingold [10], established a very efficient generalization, where they formalized Lucks' treatment by using strongly universal hash functions. In [10], they achieve an improvement in the computational complexity by using only two applications of pseudorandom functions on n bits to compute the value of a $2n$ bit pseudorandom permutation. The central idea is to sandwich the two rounds of Feistel networks involving the pseudorandom functions between two pairwise independent $2n$ bit permutations. In other words, let f be a pseudorandom function on n bits and let h_1, h_2 be two pairwise independent $2n$ bit permutations (for example $h_i(x) = a_i x + b_i \bmod p$, where a_i, b_i are uniformly distributed). Then Naor and Reingold proved that h_1 followed by two Feistel rounds using f followed by h_2 is a $2n$ bit pseudorandom permutation. They generalized this construction by relaxing the pairwise independence condition on the exterior permutation rounds but changed the interior rounds

to include two different pseudorandom functions. This piece of work represents the state of the art in efficiency related to Luby-Rackoff ciphers.

Another line of research has focused on how to enhance the security of Luby-Rackoff ciphers. Patarin [12] has shown that a Luby-Rackoff permutation can be distinguished from a random permutation using $O(2^{\frac{n}{2}})$ queries. In a related work, [1] showed how to obtain pseudorandom functions on $2n$ bits from pseudorandom functions on n bits using Benes networks. More recently, Patarin [13] has shown that six rounds of the Luby-Rackoff construction (instead of four) results in a pseudorandom permutation which cannot be distinguished from a random permutation with advantage better than $O(\frac{m^3}{2^{2n}})$, where m is the number of queries.

In this paper we show some new constructions of more practical Luby-Rackoff block ciphers from the standpoint of efficiency of computations. In addition, we provide security guarantees for Luby-Rackoff ciphers under weaker and more practical assumptions about the underlying primitive. We start with the Naor-Reingold construction but in a more general context of Abelian groups (as opposed to n bit vectors) and introduce new improvements in efficiency. With the same pseudorandom function in rounds 2 and 3, our universal hash functions in the 1st and 4th rounds operate only on half the data as opposed to the entire data thereby improving on the Naor-Reingold construction. Note that the round functions involve the group operation for encryption, and the difference operation for decryption as opposed to the usual XOR operation for both encryption and decryption.

We employ a novel construct called a *Bi-symmetric $\epsilon - \Delta$-Universal Hash Function*. We also give an example of such a Bi-symmetric $\epsilon - \Delta$-universal hash function that can be implemented much more efficiently than standard universal hash function constructions. Another interesting variation in the proof of security is that we show that even if the underlying function is only a secure Message Authentication Code (as opposed to the much stronger pseudorandom function) no adversary can easily invert Luby-Rackoff block ciphers. Finally, we provide a SHA-1 based example block cipher.

This paper is organized as follows: Section 2 includes the preliminaries, followed by the main results in section 3. Here we define the notion of a *Bi-symmetric $\epsilon - \Delta$* universal hash function, and provide an example. This concept is central to our proof of the main theorem. The security of Luby-Rackoff ciphers using a secure MAC is discussed in section 4. A SHA-1 based example cipher is included in section 5. In section 6 we discuss issues related to the optimality of Luby-Rackoff ciphers.

2 Preliminaries

Let H be a family of functions going from a domain D to a range G, where G is an Abelian Group with '+' and '-' as the addition and subtraction operators respectively. Let ϵ be a constant such that $1/|G| \leq \epsilon \leq 1$. The probabilities denoted below are all taken over the choice of $h \in H$.

Definition 1. *H is a $\Delta-universal-family$ of hash functions if for all $x, y \in D$ with $x \neq y$, and all $a \in G$, $Pr[h(x) - h(y) = a] \leq 1/|G|$. H is called $\epsilon - almost - \Delta - universal$ if $Pr[h(x) - h(y) = a] \leq \epsilon$. An example is the linear hash $h(x) = ax \bmod p$ with $a \neq 0$.*

Definition 2. *H is a strongly universal family of hash functions if for all $x, y \in D$ with $x \neq y$ and all $a, b \in G$, $Pr[h(x) = a, h(y) = b] \leq 1/|G|^2$. H is $\epsilon-almost-strongly-universal$ family of hash functions if $Pr[h(x) = a, h(y) = b] \leq \epsilon/|G|$. An example is the linear congruential hash $h(x) = ax + b \bmod p$ with $a \neq 0$.*

Definition 3 (Basic Feistel Permutation). *Let f be a mapping from G to G. We denote by $\overline{f}$ the permutation on $G \times G$ defined as $\overline{f}(x) = (x_R, (x_L + f(x_R))$ where $x = (x_L, x_R)$, and $x_L, x_R \in G$.*

Definition 4 (Feistel Network). *Let $f_1, \ldots, f_s$ be mappings from G to G, then we denote by $\Psi(f_1, \ldots, f_s)$ the permutation on $G \times G$ defined as*

$$\Psi(f_1, \ldots, f_s) = \overline{f_s} \circ \ldots \circ \overline{f_1} \tag{1}$$

Theorem 1 (Luby-Rackoff). *The permutation on $G \times G$ defined by*

$$\Psi(f_1, f_2, f_3, f_4) \tag{2}$$

where the f_i are keyed pseudorandom functions is a secure block cipher in the sense that it cannot be distinguished from a random permutation by a polynomially bounded adversary who mounts an adaptive chosen plaintext/ciphertext attack.

3 Improving Luby-Rackoff Ciphers

In this section we provide a construction and proof of a more optimized version of a Luby-Rackoff cipher. Our construction is more practical than the one given by Naor and Reingold in [10] – which is currently the state of the art in Luby-Rackoff style block ciphers. Here is the main theorem proven in [10]:

Theorem 2 (Naor-Reingold). *Let f_1, f_2 be keyed pseudorandom functions. Let h_1, h_2 be strongly universal hash functions which as permutations on $G \times G$. Then, the permutation on $G \times G$ defined by*

$$h_2 \circ \Psi(f_1, f_2) \circ h_1 \tag{3}$$

is a secure block cipher in the sense that it cannot be distinguished from a random permutation by a polynomially bounded adversary who mounts an adaptive chosen plaintext/ciphertext attack.

The Naor-Reingold construction was a major breakthrough in the design of Luby-Rackoff ciphers since they were able to completely remove two calls of the

expensive pseudorandom functions, and in some sense replace them with much more efficient non-cryptographic strongly universal hash functions.

Naor and Reingold mentioned two possible optimizations to their original construction and proved them to be secure block ciphers. The first is to use the same PRF in rounds two and three, thus saving key material: $h_2 \circ \Psi(f, f) \circ h_1$. The other possible optimization is to use the construction $\Psi(h_1, f_1, f_2, h_2)$ where the h_i are $\epsilon - \Delta$-universal hash functions which now operate on only *half* the data, as opposed to the entire $2n$ bit data. This construction saves running time and key material. Unfortunately, trying to realize both optimizations simultaneously $(\Psi(h_1, f, f, h_2))$, leads to an attack. [1] In particular, suppose that the $\epsilon - \Delta$-universal hash function we use is the linear hash $(h_a(x) = ax)$ over the field $GF(2^n)$. First we encrypt $(0, 0)$ which results in the left half of the ciphertext being $V = f(f(h_1(0))) + h_1(0)$. Then we decrypt $(0, 0)$ also resulting in the right half of the plaintext being $R = f(f(h_2(0))) + h_2(0)$. When $h_a(x) = ax$, then $h_1(0) = 0 = h_2(0)$. Thus V=R clearly allowing us to distinguish the cipher from a random permutation. Groups where addition and subtraction are different may not be susceptible to this attack.

This raises the question of whether or not one can use the same PRF's in rounds two and three and have an *efficient* non-cryptographic function operating on only half the bits in rounds 1 and 4. In this paper, we give a construction which answers this question in the affirmative.

We employ a novel construct called a *Bi-symmetric $\epsilon - \Delta$-Universal Hash Function* instead of the standard universal hash functions given in [10]. We also give an example of such a Bi-symmetric-Δ-universal hash function that can be implemented much more efficiently on many platforms than standard universal hash function constructions. Using these *Bi-symmetric $\epsilon - \Delta$-Universal Hash Functions* in rounds one and four, and the same PRF in rounds 2 and 3, we can get a secure and efficient Luby-Rackoff style cipher.

These hash functions, as defined below, possess two properties: the first one is the usual $\epsilon - \Delta$-universal property. The second property is different, and enables us to prevent the aforementioned attack.

Definition 5. *Let G be an Abelian Group and let $'-'$ and $'+'$ denote the subtraction and addition operations with respect to this group. Then H is a family of Bi-symmetric $\epsilon - \Delta$-universal Hash Functions if for all x, y with $x \neq y$, and all δ, $\Pr_{h \in H}[h(x) - h(y) = \delta] \leq \epsilon$ and for all x', y' and for all δ $\Pr_{h_1, h_2 \in H}[h_1(x') + h_2(y') = \delta] \leq \epsilon$.*

Typically we want ϵ to be extremely small – around $O(1/2^n)$ for inputs and outputs of size n bits.

Theorem 3. *Let h_1, h_2 be Bi-symmetric $\epsilon - \Delta$-universal hash functions, and let f be a random function. The block cipher defined by a four round Feistel network $\Psi(h_1, f, f, h_2)$ cannot be distinguished from a random permutation with probability better than $O(m^2 \cdot \epsilon)$ where m is the number of queries made by the*

[1] Daniel Bleichenbacher, personal communication

adversary (who may have unlimited power), and all the operations are performed in an Abelian group G.

Maurer [9] presented a very simple proof of security of the original Luby-Rackoff construction. Naor-Reingold [10] gave a more formal structure for proving adaptive security of Luby-Rackoff ciphers. Fortunately, the conditions that need to be satisfied for the security of the block cipher as presented by a Maurer type treatment are the same as the conditions resulting from the more formal treatment of Naor-Reingold. Hence the proof we present follows the simpler presentation of [9], but only proves security in the non-adaptive case. We can, however, easily prove adaptive security by following the treatment in [10].

Proof. Suppose we make m plaintext/ciphertext queries to a black box which can encrypt/decrypt. Moreover, we make these queries in a non-adaptive fashion; e.g. we make them all at once. Based on the responses of the black box, we must determine whether the black box contains a truly random permutation or a permutation of the form $\Psi(h_1, f, f, h_2)$.

Now, suppose we have a $2n$ bit plaintext message M_i where we denote the leftmost n bits as L_i and the rightmost n bits as R_i. The following equations describe the process by which this plaintext is encrypted by the Feistel network we consider:

$$S_i = L_i + h_1(R_i) \tag{4}$$
$$T_i = R_i + f(S_i) \tag{5}$$
$$V_i = S_i + f(T_i) \tag{6}$$
$$W_i = T_i + h_2(V_i) \tag{7}$$

where $+$ is the addition operation in the group G. Here (V, W) is the encrypted output of the plaintext (L, R). Similarly we can describe the decryption process where the "inputs" are (V, W) and the "outputs" are (L, R). Suppose that following transcript describes the responses of the encryption/decryption black box:

$$\langle (L_1, R_1), (V_1, W_1) \rangle, \ldots, \langle (L_m, R_m), (V_m, W_m) \rangle \tag{8}$$

We can assume without loss of generality that these queries are all distinct because making the same query twice doesn't help you. That is for all $1 \leq i < j \leq m$ $(L_i, R_i) \neq (L_j, R_j)$ and $(V_i, W_i) \neq (V_j, W_j)$. Now, let $\mathcal{E}_S$ denote the event that the elements in the set $\mathcal{S} = \{S_1, \ldots, S_m\}$ are all different, and let $\mathcal{E}_T$ denote the event that the elements in the set $\mathcal{T} = \{T_1, \ldots, T_m\}$ are all different (where S_i and T_i are defined as above). Finally, let $\mathcal{E}_{ST}$ denote the event that the sets $\mathcal{S}$ and $\mathcal{T}$ are disjoint. Now, we consider what each event entails. First let's examine the case that the query made was an encryption query; i.e. the input was (L_i, R_i) and the output was (V_i, W_i) . If $\mathcal{E}_T$ occurs, then all the V_i's will be random because $V_i = S_i + f(T_i)$ and since the T_i's are distinct and f is a random function, it follows that $S_i + f(T_i)$ is random. Similarly, if both $\mathcal{E}_S$ and $\mathcal{E}_{ST}$ occur, then W_i will be random. This happens because $W_i = T_i + h_2(V_i) = R_i + f(S_i) + h_2(V_i)$

and $f(S_i)$ is random because f is a random function, and the S_i's are distinct and different from all the T_j's.

Now we look at the case for decryption queries. The inputs in this case are (V_i, W_i) and the outputs are (L_i, R_i). Following the same lines of reasoning as above, we want the "outputs" (L_i, R_i) to be random. This happens if the inputs to the random functions in rounds 2 and 3 are distinct. Therefore, the conditions we need are specified by the events $\mathcal{E}_S$, $\mathcal{E}_T$ and $\mathcal{E}_{ST}$.

Now, if $\mathcal{E}_T$, $\mathcal{E}_S$, and $\mathcal{E}_{ST}$ all occur, then the adversary cannot distinguish our Feistel permutation from a random permutation, except with negligible probability less than $m^2/2^{2n}$ [10]. All that remains is to derive a bound on the probability that at least one of these events does not occur. Let $\mathcal{E}_S^C$, $\mathcal{E}_T^C$, $\mathcal{E}_{ST}^C$ denote the complements of these events. We proceed to derive our bounds:

$$
\begin{aligned}
Pr[\mathcal{E}_S^C \text{ or } \mathcal{E}_T^C] &\leq \Sigma_{1\leq i<j\leq m} \Pr[T_i = T_j] + \Sigma_{1\leq i<j\leq m} \Pr[S_i = S_j] && (9)\\
&\leq \Sigma_{1\leq i<j\leq m} \Pr[W_i - h_2(V_i) = W_j - h_2(V_j)] && (10)\\
&+ \Sigma_{1\leq i<j\leq m} \Pr[L_i + h_1(R_i) = L_j + h_1(R_j)] && (11)\\
&\leq \Sigma_{1\leq i<j\leq m} \Pr[h_2(V_i) - h_2(V_j) = W_i - W_j] && (12)\\
&+ \Sigma_{1\leq i<j\leq m} \Pr[h_1(R_i) - h_1(R_j) = L_j - L_i] && (13)\\
&\leq m(m-1)/2 \cdot (\epsilon + \epsilon) && (14)
\end{aligned}
$$

where the last inequality follows from the previous one by the following argument: if $V_i = V_j$ (similarly $R_i = R_j$) then it follows that $W_i \neq W_j$ (similarly $L_j \neq L_i$) by distinctness of queries; hence the event occurs with probability 0 in this case. If $V_i \neq V_j$ (similarly $R_i \neq R_j$), then the bound follows by the Bi-symmetric $\epsilon - \Delta$-universality property of the h_i.

Moreover, the above analysis holds regardless of whether the S_i or T_i were generated by the process of an encryption query or a decryption query. All that remains is to bound the probability that $\mathcal{E}_{ST}$ does not occur:

$$
\begin{aligned}
\Pr[\mathcal{E}_{ST}^C] &\leq \Sigma_{1\leq i\leq j\leq m} \Pr[S_i = T_j] && (15)\\
&\leq \Sigma_{1\leq i\leq j\leq m} \Pr[L_i + h_1(R_i) = W_j - h_2(V_j)] && (16)\\
&\leq \Sigma_{1\leq i\leq j\leq m} \Pr[h_1(R_i) + h_2(V_j) = W_j - L_i] && (17)\\
&\leq m^2 \cdot \epsilon && (18)
\end{aligned}
$$

where the last inequality follows from the previous one by the Bi-symmetric $\epsilon - \Delta$-universality property of H. Finally, we can apply a union bound to get:

$$\Pr[\mathcal{E}_S^C \text{ or } \mathcal{E}_T^C \text{ or } \mathcal{E}_{ST}^C] \leq 2m^2 \cdot \epsilon \quad (19)$$

This concludes the proof. □

3.1 An Example Bi-symmetric $\epsilon - \Delta$-universal hash function

Definition 6. *Let p be a prime. Define the square hash family of functions from Z_p to Z_p as:*

$$h_x(m) \equiv (m + x)^2 \bmod p \quad (20)$$

Theorem 4. *The square hash family of functions is $\Delta - universal$.*

Proof: For all $m \neq n \in Z_p$, and $\delta \in Z_p$: $\Pr_x[h_x(m) - h_x(n) = \delta] = \Pr_x[(m+x)^2 - (n+x)^2 = \delta] = \Pr_x[m^2 - n^2 + 2(m-n)x = \delta] = 1/p$, where the last inequality follows since for any given $m \neq n \in Z_p$ and $\delta \in Z_p$ there is a unique x which satisfies the equation $m^2 - n^2 + 2(m-n)x = \delta$. □

Theorem 5. *The square hash is a Bi-symmetric $\epsilon - \Delta$-universal family of hash functions.*

Proof: We have already proved the first property and need to prove the property that for all m, n and for all δ $\Pr_{x,y \in H}[h_x(m) + h_y(n) = \delta] \leq \epsilon$. $Pr_{x,y}[(m+x)^2 + (n+y)^2 = \delta \bmod p] = Pr_{x,y}[x^2 + 2xm + m^2 + n^2 + y^2 + 2ny - \delta = 0 \bmod p]$ There are at most 2 solutions for x for a given key y, so altogether there are at most 2^{n+1} solutions out of 2^{2n} possible keys x, y. So $\Pr_{x \in H}[h_x(m) + h_x(n) = \delta] = \frac{2}{2^n} = \frac{1}{2^{n-1}}$. If we count the number of solutions by further reducing this modulo 2^n, then ϵ increases by a small factor. □

We remark that the function $h_x(m) \equiv (m+x)^2 \bmod p \bmod 2^n$ is also a Bi-symmetric $\epsilon - \Delta$-universal family of hash functions for a slightly larger value of ϵ. This function is more useful for implementation purposes since we often work over addition modulo 2^n (e.g. bit strings of length n). The Square Hash family requires much less computation time on many platforms than the traditional linear hash. The speedup occurs because squaring an n-bit number requires roughly half the number of basic word multiplications than actually multiplying two n-bit numbers. Thus square hash requires fewer operations and instructions to implement. More details can be found in [14].

4 Proving Security Under MAC Assumption

We give an alternate proof of security of our construction. This proof utilizes a weaker, but perhaps more practical, assumption, and makes a weaker claim on security of our block cipher. In particular, we show that if the underlying function f in our Feistel Network works as a secure *Message Authentication Code (MAC)*, then it is infeasible for an adversary to come up with a random plaintext/ciphertext pair after mounting an adaptive chosen plaintext/ciphertext attack. Some earlier work on the relationship between unpredictability (MACs) and indistinguishability was studied in [11]. We now define the relevant notions and then prove our claim.

The goal of message authentication codes is for one party to efficiently transmit a message to another party in a way that enables the receiving party to determine whether or not the message he receives has been tampered with. The setting involves two parties, Alice and Bob, who have agreed on a pre-specified secret key x. There are two algorithms used: a signing algorithm S_x and a verification algorithm V_x. If Alice wants to send a message M to Bob then she first computes a message authentication code, or MAC, $\mu = S_x(M)$. She sends (M, μ) to Bob, and upon receiving the pair, Bob computes $V_x(M, \mu)$ which returns 1 if

the MAC is valid, or returns 0 otherwise. Without knowledge of the secret key x, it should be next to impossible for an adversary to construct a message and corresponding MAC that the verification algorithm will be accept as valid.

The formal security requirement for a Message Authentication Code was defined by Bellare, Canetti and Krawczyk [3]. In particular, we say that an adversary forges a MAC if, when given oracle access to (S_x, V_x), where x is kept secret, the adversary can come up with a valid pair (M^*, μ^*) such that $V_x(M^*, \mu^*) = 1$ but the message M^* was never made an input to the oracle for S_x. Here is a formal definition of an alternate, but equivalent formulation:

Definition 7. *We say that a function f_k is an (ϵ, q)-secure Message Authentication Code (MAC), if the probability that an adversary can successfully perform the following experiment is bounded by ϵ:*

1. *(Adversary is given black box access to f_k) The adversary makes q possibly adaptively chosen queries to a black box for f_k – that is the adversary comes up with a message m_1 gets to see $f_k(m_1)$, and from this information comes up with a query m_2, gets to see $f_k(m_2)$, and so on until making a query m_q, and getting to see $f(m_q)$.*
2. *The adversary comes up with a pair $(m, f_k(m))$ where m is different from any message queried in the first part. That is, $m \neq m_i$ for $1 \leq i \leq q$.*

Message Authentication has been a well studied problem, and there are a number of schemes which are widely believed to be secure. We show that we can use any of these schemes as the underlying function f in our block cipher and still get a fairly secure encryption scheme.

We now explain what it means for a cryptosystem to be *randomly-secure* against adaptive chosen message/ciphertext attacks. The definition has a similar flavor to the above definition for MACs.

Definition 8. *We say that a secret key encryption algorithm E_k is (ϵ, q)-randomly-secure against adaptive chosen plaintext/ciphertext attacks if no adversary can successfully perform the following experiment with probability better than ϵ:*

1. *(The adversary is given black box access to E_k) The adversary makes q possibly adaptively chosen plaintext/ciphertext queries to a black box for E_k; for example, the adversary comes up with a message m_1 gets to see $E_k(m_1)$, and from this information comes up with a query m_2, gets to see $E_k(m_2)$, and so on until making a query m_q and getting to see $E_k(m_q)$.*
2. *For a given randomly chosen message (or ciphertext) the adversary comes up with the corresponding ciphertext (or plaintext). This message (ciphertext) is different from any message (ciphertext) queried in the first part.*

We remark that there are alternate, and more stringent, definitions of security for symmetric key encryption schemes.

Theorem 6. *Let f be a $(\epsilon_1, 2q)$-secure MAC, and let h_1, h_2 be ϵ_2-Δ-universal hash functions. Then, $\Psi(h_1, f, f, h_2)$ is $(\epsilon_1 + q^2\epsilon_2^2, q)$-randomly-secure under adaptive chosen plaintext/ciphertext attacks.*

Proof. The overall idea is to show that given any adversary A' who can break the encryption scheme (given black box access to the encryption and decryption mechanisms), we can construct an adversary A who can break the underlying MAC f (given black box access to the MAC function f). A will work as follows. First, A picks two hash functions h_1 and h_2 at random from a family of ϵ_2-Δ-universal hash functions. Then A proceeds simulating A'. At some point A' is going to make a query which could be in either of two forms: "Please give me the encryption of a message m" or "Please give me the decryption of a ciphertext c." In either case, A must answer this query for A' in a legitimate fashion. A can do this easily by making two calls to the black box for f and simulating the encryption or decryption algorithms. For example, if the ith query is an encryption query on a message $M_i = (L_i, R_i)$, then A computes values S_i, T_i, V_i, W_i according to equations 4, 5, 6, and 7.

We see that A calls the black box for f whenever it computes T_i and V_i. Decryption queries are handled in a similar fashion. Now, after A' finishes making q queries, (which results in A having made $2q$ queries) it will output a ciphertext (V, W) and the corresponding plaintext (L, R) – if the plaintext is different from plaintexts given during all the encryption queries made by A', and the ciphertext differs from the ones given during the decryption queries made by A', then A' has successfully broken the encryption scheme. Now, we translate this break of the encryption into something that breaks the underlying MAC f with high probability. So, we now may have derived two potential (Message, Tag) pairs. A computes:

$$S = L + h_1(R) \tag{21}$$
$$T = W - h_2(V) \tag{22}$$
$$Tag(S) = T - R \tag{23}$$
$$Tag(T) = V - S \tag{24}$$

If you work out the equations, it's easy to see that $Tag(S) = f(S)$ and $Tag(T) = f(T)$. Therefore, $(S, Tag(S)$ and $(T, Tag(T))$ are valid Message/Tag pairs. It appears as if we are done, but there is still one technicality remaining. Recall that when we defined the notion of a secure MAC, we require that the message output by the adversary must differ from the messages that the adversary gave to the black box during the query phase. We must now bound the probability that $S = S_i$ for some i $(1 \leq i \leq q)$.

$$Pr[\exists i \ 1 \leq i \leq q \text{ such that } S = S_i] \tag{25}$$
$$= Pr[\exists i \text{ such that } L + h_1(R) = L_i + h_1(R_i)] \tag{26}$$
$$\leq \Sigma_{i=1}^{q} Pr[L + h_1(R) = L_i + h_1(R_i)] \tag{27}$$
$$\leq q\epsilon_2 \tag{28}$$

The last equation follows from the previous because h_1 is an ϵ_2-Δ-universal hash function. Similarly, one can show that

$$Pr[\exists i \ 1 \leq i \leq q \text{ such that } T = T_i] \leq q\epsilon_2 \tag{29}$$

Since the hash functions h_1 and h_2 were chosen independently at random, it follows that likelihood that both $S = S_i$ and $T = T_i$ is at most $(q\epsilon_2)^2$.

What remains is to find a lower bound for the success probability of A'. We can do this by first deriving a lower bound for the success probability of A. A is successful whenever both of two conditions happen:

Condition 1 A' outputs a message and a valid ciphertext for that message. (i.e. A' is successful.)

Condition 2 At least one of the S, T that are generated do not coincide with something that was queried before; i.e. you can use the message and ciphertext to generate a valid message/tag pair where the message was not part of a previous query.

An easier way to proceed is to derive an upper bound for the probability that A fails, and subtract that number from 1. Suppose that the probability of condition 1 being met is at least ϵ_3 – then Pr[Condition 1 doesn't happen] $\leq (1 - \epsilon_3)$. Now, we know that condition 2 fails to occur with probability at most $(q\epsilon_2)^2$. Therefore, by a union bound, Pr[A is unsuccessful] $\leq 1 - \epsilon_3 + (q\epsilon_2)^2$. Therefore, Pr[$A$ is successful] $\geq \epsilon_3 - (q\epsilon_2)^2$. Now, by assumption we have that Pr[A is successful] $\leq \epsilon_1$. Therefore, it follows that $\epsilon_3 \leq \epsilon_1 + q^2\epsilon_2^2$ – which is a bound on success probability of A'. We have thus shown that we break the cipher with probability at most $\epsilon_1 + q^2\epsilon_2^2$ after making q queries. And this concludes the proof. □

5 An Example Block Cipher: Sha-zam

In this section we discuss the design of *Sha-zam*, a block cipher based on constructions and theorems proved earlier. We use SHA-1 as the underlying primitive instead of a family of pseudorandom functions. Replacing pseudorandom functions by cryptographic functions (with desired properties) is not new. Biham and Anderson [2], propose the use of SHA-1 in conjunction with stream ciphers to design block ciphers. Also, Lucks used MD5 with an unbalanced Feistel network and Guttman's construction uses SHA-1 but different from the Luby Rackoff construction. In our design we do not use any stream ciphers. Instead we rely entirely on the improved versions of the Luby Rackoff construction and use SHA-1 as our underlying primitive. Recall that in section 4 we showed that Luby-Rackoff ciphers are secure if the underlying primitive is a secure MAC. Here security is with respect to some form of invertibility. From a practical standpoint the use of SHA-1 is justified. For example, the internet RFC HMAC-SHA-1 [4] assumes that forging a tag using SHA-1 as the underlying MAC is hard.

In addition to SHA-1 we use the Square Hash function (SQH) which we have introduced earlier. This cipher is driven by a *key scheduling generator* whose

security is also related to SHA-1. Hence we get efficient re-use of code. Based on our results in earlier sections, under the assumption SHA-1 is pseudorandom we have that Sha-zam is a block cipher secure against adaptive chosen plaintexts and ciphertext attacks. Under the weaker assumption that SHA-1 is a secure MAC we show that no adversary can invert Sha-zam.

If C is an n-bit string, we denote by $prefix_k(C)$ the $n-k$ bit prefix of C (i.e. the first $n-k$ bits of C). We denote by $SHA(IV, x)$ the 160 bit output produced by SHA-1 on a 512 bit user specified input x and the standard IV. Our block cipher, which we call Sha-zam, takes as input a 320 bit block M and outputs a 320 bit ciphertext C. We denote $M = (L, R)$ where L is the left 160 bits of M and R is the right 160 bits of M. Also, we prefer to keep IV secret. In our construction we use three keys k_1, k_2, k_3.

Encryption with Sha-zam

Input: Plaintext Stored in L, R – each of which is 160 bits
Private Key: $k = (k_1, k_2, k_3)$ where:
k_1, k_3 are 160 bits each, and k_2 is 352 bits.
If IV not secret: then use the standard 160 bit IV.

Output: Ciphertext stored in V,W – each of which is 160 bits

Procedure:
$S = L + SQH_{k1}(R) \bmod 2^{160}$
$T = R + SHA(IV, S, k_2) \bmod 2^{160}$
$V = S + SHA(IV, T, k_2) \bmod 2^{160}$
$W = T + SQH_{k_3}(V) \bmod 2^{160}$

Decryption with Sha-zam

Input: Ciphertext Stored in V,W – each of which is 160 bits
Private Key: $k = (k_1, k_2, k_3)$ where:
k_1, k_3 are 160 bits each, and k_2 is 352 bits.
If IV note secret: then use the standard 160 bit IV.

Output: Plaintext stored in L,R

Procedure:
$T = W - SQH_{k_3}(V) \bmod 2^{160}$
$S = V - SHA(IV, T, k_2) \bmod 2^{160}$
$R = T - SHA(IV, S, k_2) \bmod 2^{160}$
$L = S - SQH_{k_1}(R) \bmod 2^{160}$

Our block cipher can be implemented efficiently. Specifically, it can encrypt messages in roughly the same time as it would take DES to accomplish this same task.

We describe a practical and secure pseudo-random generator, which we use for key scheduling. The security is based on SHA-1. If we run our generator using a randomly selected key as an input seed, we can securely generate the necessary bits needed for the secret key of our block cipher. We make use of a 512 bit prespecified global constant C. We almost never use the entire constant C but often take some specified prefix of it depending on the length of the key we're working with. We now describe our generator. Given a seed s we generate pseudorandom bits as follows:

Description of Generator
Input: seed s

1. Let $s_0 = s$
2. For $i = 1$ to m do
   Let $s_i = SHA(IV, prefix(C), s_{i-1})$
3. Output: $\langle h(s_1), \ldots, h(s_m) \rangle$

In step 3 above, h refers to a hash function chosen from a universal class. For example, the linear congruential hash function is any finite field is a very good candidate. The proof of security of this key scheduling generator will be presented elsewhere, [15].

6 A Discussion on Optimality

Since the invention of Luby-Rackoff ciphers, considerable progress has been made with respect to making the construction more efficient. Specifically, as noted in the introduction, most of the focus has been in "reducing" the number of invocations of a random function and the amount of key material used. Following the work of Lucks [8], Naor-Reingold have produced extremely efficient constructions with the help of hash functions and just two calls to a random function. In the present work, we have described a further generalization by using a different class of hash functions operating on only half the size of the input, in addition to a reduction in the key material. Is the end of progress in sight? We now discuss what it might mean for a Luby-Rackoff cipher to be optimal. We present various parameters of interest, and discuss how our proposal fits within this discussion.

1. Minimal number of rounds: Luby-Rackoff in their original paper showed that two rounds are not enough and 3 rounds are needed for plaintext security. Furthermore in order to resist adaptive attacks four rounds are needed. Our construction also consists of 4 rounds.
2. Maximal Security: Patarin [12] showed that the 4 round Luby-Rackoff construction can be distinguished from a random permutation with probability $O(\frac{m^2}{2^n})$ with m queries. We meet this bound as stated in Theorem 1. We can reduce the distinguishing probability by increasing the number of rounds, but this would violate the previous critirea.

3. Minimal Rounds of PRFs: Since the output of the block cipher is $2n$ bits long, it would seem that two n bit PRFs are necessary to insure cryptographic security. We also use only two PRFs in rounds two and three. For rounds one and four we use non-cryptographic called Bi-symmetric $\epsilon - \Delta$-universal functions, which add to the efficiency significantly.
4. Reusing the same PRF: It has been the goal of many papers to reduce the number of different PRFs that are used, ultimately hoping to use just one PRF. We achieve this goal by just using one PRF in rounds two and three.
5. Minimal Data Size Operated on by Non-cryptographic function: Our Bi-symmetric $\epsilon - \Delta$-universal hash function in rounds one and four operate on n bits of the data. If we operated on any smaller part of the data then we would open ourselves to collisions that can be detected with lower number of queries, thus increasing the distinguishing probability and decreasing security.
6. Reusing Hash Functions: It might be tempting to use the same universal hash function in rounds 1 and 4 to save key material. However, using the same hash h in both rounds, in groups where $g = -g$ for all $g \in G$, unfortunately leads to an attack which we now describe.

   Consider the group of n bit vectors with respect to the usual XOR operation. When we encrypt (0,0), the left half of the resulting ciphertext is $V = f(f(h(0))) + h(0)$. Then we decrypt (0,0) also, thus resulting in $R = f(f(h(0))) + h(0)$. Since V and R are equal, we immediately distinguish the block cipher from a random permutation.

   In our construction we may be able to use the same h. This seems plausible due to the fact our network operates with Abelian groups whose operations are not symmetric (i.e. + and - are different), where we may be able to exploit the use of specialized hash functions with the property $Pr_{h \in H}[h(x) + h(y) = \delta] \leq \epsilon$ for all x, y, and δ. When working in the cyclic group of integers modulo 2^n, we note that the Square hash function satisfies this specialized property. In particular, even if the above property does not suffice, other properties of square hash might aid in proving our claim. We leave this for further study.

7 Conclusion

In this paper we have described some novel improvements to Luby-Rackoff ciphers. We introduce the concept of a Bi-symmetric $\epsilon - \Delta$ universal hash function and provide an example of such a class. This concept when applied to the Naor Reingold construction of a Luby-Rackoff cipher improves the efficiency. In addition we show that under the weaker and more practical assumption of secure MAC we show that Luby Rackoff ciphers are hard to invert. We discuss the design of a new cipher based on these improvements - Sha-zam, whose security is related to SHA-1.

Acknowledgments

We would like to thank Daniel Bleichenbacher and Peter Winkler for useful comments on earlier versions of this paper. We would also like to thank Omer Reingold for very helpful comments.

References

[1] W. Aiello, R. Venkatesan, Foiling birthday attacks in length-doubling transformations, *Advances in Cryptology - EUROCRYPT '96*, LNCS 1070, 307–320, 1996.

[2] R. Anderson, E. Biham, Two practical and provably secure block ciphers, BEAR and LION, *Fast Software Encryption*, LNCS 1039, 113–120, 1996.

[3] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, *Advances in Cryptology*, LNCS 1109, 1996.

[4] M. Bellare, R. Canetti, H. Krawczyk. HMAC: Keyed-Hashing for Message Authentication, *Internet RFC*, 2104, February 1997.

[5] O. Goldreich, S. Goldwasser, and A. Micali, How to construct random functions?, *Journal of ACM*, 33: 792–807, 1986.

[6] P. Gutmann, documentation to SFS release 1.20 - SFS7.DOC, URL:http:/www.cs.auckland.ac.nz/pgut01/sfs.html, 1995.

[7] M. Luby, and C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions, *SIAM Journal of Computing*, 17: #2, 373–386, 1988.

[8] S. Lucks, Faster Luby-Rackoff ciphers, *Proc. Fast Software Encryption*, LNCS, 1039, 189–203, 1996.

[9] U. Maurer, A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators, *Advances in Cryptology - EUROCRYPT '92*, LNCS 658, 239–255, 1992.

[10] M. Naor, O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff revisited, *J. of Cryptology*, vol. 12, 29–66, 1999. Preliminary version in: *Proc. 29th Annual ACM STOC*, 189–199, 1997.

[11] M. Naor, O. Reingold, ¿From unpredictability to indistinguishability: A simple construction of pseudo-random functions from MACs, *Advances in Cryptology - CRYPTO '98*, LNCS, 267–282, 1998.

[12] J. Patarin, New results on pseudorandom permutation generators based on the DES scheme, *Advances in Cryptology - CRYPTO '91*, LNCS, 301–312, 1991.

[13] J. Patarin, Improved security bounds for pseudorandom permutations, *4th ACM Conference on Computer and Communications Security*, 142–150, 1997.

[14] S. Patel, Z. Ramzan, Square hash: Fast message authentication via optimized universal hash functions, *preprint.*

[15] S. Patel, Z. Ramzan, G. S. Sundaram, On constructing pseudorandom generators based on cryptographic hash functions, *In preparation.*

[16] J. Pieprzyk, How to construct pseudorandom permutations from single pseudorandom functions, *Advances in Cryptology - EUROCRYPT '90*, LNCS 473, 140–150, 1991.

[17] U. S. Department of Commerce/ N. I. S. T, *Secure Hash Algorithm*, FIPS 180, April 1995.