# Provably Secure and Efficient Block Ciphers

Pat Morin
School of Computer Science
Carleton University
morin@scs.carleton.ca

**Abstract**

The security and efficiency of two recently proposed block ciphers, BEAR and LION, both based on a hash function and a stream cipher, is discussed. Meet-in-the-middle attacks are presented which can be used to dramatically reduce the complexity of a brute-force key search on both these ciphers. A new block cipher is described which is not susceptible to meet-in-the-middle attacks, is provably secure against any chosen plaintext or ciphertext attack, and is more efficient than BEAR or LION.

## 1 Introduction

A number of examples exist which show how cryptographic primitives can be composed to yield other cryptographic primitives. Two examples of this are the well known output-feedback mode of DES which converts a block cipher into a stream cipher, and feedforward mode which converts a block cipher into a hash function. Often, these compositions are provably secure in that an efficient attack on the composite function would lead to an efficient attack on the underlying primitive.

These types of construction are of practical interest, as there often exist efficient cryptographic primitives which, although not provably secure, are widely believed to be secure because of empirical evidence. Composing such primitives to yield higher level functions can lead to new cryptographic functions that are practical to implement and have the advantage that their security is based on the security of trusted primitives.

In this paper we examine the security of two recently proposed constructions, BEAR and LION, which allow block ciphers to be created from a stream cipher and a hash function. The resulting block ciphers are provably secure in that a key-recovery attack that can be mounted with a single plaintext ciphertext pair can be used to break both the hash function and the stream cipher. We show that although these ciphers are provably secure, they are susceptible to meet-in-the-middle attacks which greatly reduce the complexity of a brute force key search.

We also propose a new construction. This is a block cipher which is more efficient in terms of computation than either of the above schemes, is not susceptible to meet-in-the-middle attacks, and is provably secure against any combination of chosen plaintext or ciphertext attacks. This cipher is also of practical interest, as it can be implemented very efficiently in software.

The remainder of the paper is divided as follows: Section 2 gives a brief review of BEAR and LION. Section 3 discusses the security of BEAR and LION and presents attacks on them. Section 4 presents AARDVARK a new block cipher that is similar in construction to BEAR and LION. Section 5 discusses the performance of actual implementations of BEAR, LION, and AARDVARK. Section 6 summarizes this work and presents an open problem in this area.

## 2 Bear and Lion

In this section we present BEAR and LION, two very recent block ciphers due to Anderson and Biham [AnBi96]. Both these ciphers are constructed from a stream cipher, $S$, and a hash function, $H$, using a construction similar to those of Luby and Rackoff [LuRa88]. The requirements on $S$ and $H$ are:

1. $H$ is one-way, given only $H(X)$ it is hard to find $X$;
2. $H$ is strongly collision free, it is hard to find distinct $X$ and $Y$ such that $H(X) = H(Y)$;
3. $S$ resists key recovery attacks, it is hard to find the seed $X$ given $Y = S(X)$;
4. $S$ resists expansion attacks, it is hard to expand any partial stream of $Y = S(M)$, i.e., given some subset of $Y$ it is hard to determine anything more about $Y$.

The block size, $m$, of these ciphers is variable, but is on the order of 1Kbyte-1Mbyte. If we define $k$ as the block size of the hash function used then both these ciphers are unbalanced Feistel networks in which $|L| = k$ and $|R| = m - k$.

Bear performs encryption and decryption using two applications of a hash function and one application of a stream cipher. A BEAR key consists of two sub-keys, $K_1$ and $K_2$, both of size $|K| > k$. Bear encryption and decryption is done by:

| Encryption | Decryption |
|---|---|
| $L = L \oplus H_{K_1}(R)$ | $L = L \oplus H_{K_2}(R)$ |
| $R = R \oplus S(L)$ | $R = R \oplus S(L)$ |
| $L = L \oplus H_{K_2}(R)$ | $L = L \oplus H_{K_1}(R)$ |

Lion is similar in construction to BEAR except that encryption and decryption involve one application of the hash function and two applications of the stream cipher. Again, the key consists of two sub-keys, $K_1$ and $K_2$, both of size $k$. Lion encryption and decryption are done by:

| Encryption | Decryption |
|---|---|
| $R = R \oplus S(L \oplus K_1)$ | $R = R \oplus S(L \oplus K_2)$ |
| $L = L \oplus H(R)$ | $L = L \oplus H(R)$ |
| $R = R \oplus S(L \oplus K_2)$ | $R = R \oplus S(L \oplus K_1)$ |

BEAR and LION are potentially very efficient ciphers given the speeds at which modern stream ciphers and hash functions operate. Anderson and Biham report encryption rates of 13Mbits/sec for BEAR and 16Mbits/sec for LION[1].

# 3 Security of BEAR and LION

BEAR and LION are provably secure in the sense that an oracle that can recover the key of BEAR or LION given one plaintext/ciphertext pair can be used to "break" the underlying keyed hash function and stream cipher. In the case of the hash function this means that the oracle can be used to undermine the one-way and collision free properties. In the case of the stream cipher, the oracle can be used to undermine the "resists key recovery" property.

While provable security is a desirable property, it does not tell the whole story. In the case where an adversary is able to break either the hash function or the stream cipher, the adversary can obtain partial information about the plaintext given just the cipher text. These properties are discussed by the authors. The authors also discuss attacks such as differential and linear cryptanalysis which require many plaintext/ciphertext pairs. They argue that a successful attack on BEAR or LION using these techniques would yield a successful attack on either the hash function, the stream cipher or both. The authors also suggest that these types of attacks can be avoided by using a different key to encrypt each block of data. This is a reasonable approach given BEAR's large block size

An interesting property of BEAR is that given only half the key bits, namely the bits of $K_2$, an attacker can determine *most of* the plaintext. Given a ciphertext $L'$ and $R'$, and the sub-key $K_2$ an attacker can determine the plaintext $R$ by doing a partial decryption:

$$\begin{aligned} &L^* = L' \oplus H_{K_2}(R') \\ &R = R' \oplus S(L^*) \end{aligned}$$

Since $|R|$ is typically much larger than $|L|$ this means that an attacker can determine most of the plaintext without any further cryptanalysis. This is clearly not a desirable property.

A brute force key search on BEAR would be of complexity $2^{2|K|}$, but this can be reduced considerably through the use of a meet-in-the-middle attack [MeHe81]. Given a plaintext/ciphertext pair, $P = (L, R)$, $C = (L', R')$, the attacker computes and stores $H_{K_1}(R) \oplus L$ for all $2^{|K|}$ possible values of $K_1$. Using these stored values the attacker then start computing $H_{K_2}(R') \oplus L'$ for all $2^{|K|}$ possible values of $K_2$ until she finds $K_1$ and $K_2$ such that $H_{K_1}(R) \oplus L = H_{K_2}(R') \oplus L'$. She can then test if this is the correct key-pair by verifying whether $S(H_{K_1}(R) \oplus L) = R \oplus R'$ or not. The correct key-pair will be found using at most $2^{|K|+1}$ encryption operations. Of course, such an attack is largely theoretical, as the value of $|K|$ is likely to be 128 or more.

The same type of partial decryption attack that was demonstrated against BEAR can be used against LION to recover $L$ if $K_2$ is known. However, this attack is less effective

[1] A software implementation on a 133MHz DEC Alpha using SHA as the hash function and SEAL as the stream cipher

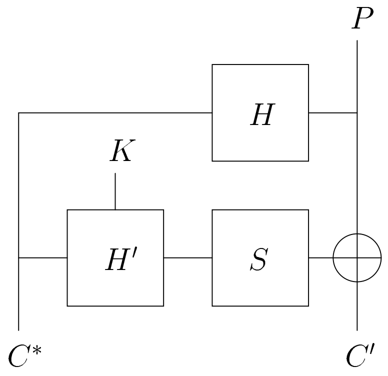


Figure 1: The AARDVARK cipher.

against LION as it only allows the attacker to recover $L$, which would typically not be more than 256 bits in length.

A similar meet-in-the-middle attack can be mounted against LION as was demonstrated against BEAR. In this case the attack is mounted on the stream cipher S and the attacker attempts to find $K_1$ and $K_2$ such that $S(K_1 \oplus L) \oplus R = S(K_2 \oplus L') \oplus R'$ and verifies candidate key pairs by checking whether $H_{K_1}(S(K_1 \oplus L) \oplus R) = L \oplus L'$.

The existence of meet-in-the-middle attacks against these ciphers leads one to believe that they are not as secure as a cipher with a 256+ bit key size could be. In the following section we discuss a new block cipher which is similar in construction to BEAR and LION, uses only a single key of length approximately $k$, and is not susceptible to meet-in-the-middle attacks.

# 4 Aardvark

This section presents a new block cipher, which we call AARDVARK[2]. Aardvark is based on a stream cipher, $S$, a hash function, $H$, and a keyed hash function, $H'$. Like BEAR and LION, the block size of AARDVARK is variable with values of 1Kbyte-1Mbyte recommended. Unlike BEAR and LION, AARDVARK is not a Feistel network and has the property that the ciphertext is slightly larger than the corresponding plaintext. As we will see, this property is not undesirable as it allows users to verify the integrity of a message during decryption.

An AARDVARK network is shown pictorially in figure 1. An AARDVARK key, $K$, is a bit string suitably long for keying $H'$. It is recommended that $|K|$ be greater than or equal to the number of bits output by $H'$. Encryption in AARDVARK consists of one application each of the hash function and the stream cipher and an application of the keyed hash function on a short ($< 512$ bit) string. This produces two values, $C^*$ and $C'$, both of which make up the ciphertext $C = (C^*, C')$. Encryption is done by:

$$C^* = H(P)$$
$$C' = P \oplus S(H'_K(C^*))$$

[2]All the exotic animal names were already being used.

Decryption is done by:

$$P = C' \oplus S(H'_K(C^*))$$

To verify that a ciphertext was not modified, either maliciously or by accident, the receiver can also verify that $H(P) = C^*$. This is particularly important since an adversary could easily modify the ciphertext in such a way that the plaintext is changed in a predictable manner. This is because flipping a bit in $C'$ has the effect of flipping the corresponding bit in the decrypted plaintext.

To ensure the security of AARDVARK, $S$ and $H$ must have the following properties:

1. $H$ is strongly collision free, it is hard to find distinct $X$ and $Y$ such that $H(X) = H(Y)$;

2. $H'$ resists existential forgery, given an oracle that computes $H'_K$ for an unknown $K$ it is hard to compute $H'_K(X)$ for any $X$ without using the oracle directly;

3. $S$ resists expansion attacks, it is hard to expand any partial stream of $Y = S(M)$;

4. $S$ and $H'$ are independent, there is nothing about $S$ and $H'$ which allows someone to compute $S(H'_K(X))$ easily without knowing $K$.

The strength of AARDVARK lies in the difficulty of computing $H'_K(X)$ without knowing $K$. Since $S$ and $H$ are independent, this makes computing $S(H'_K(X))$ difficult, which in turn makes computing the encryption and decryption functions difficult. We now prove the main theorem about the security of AARDVARK.

**Theorem 1.** *Given two oracles, one which computes $E_K$ and one which computes $D_K$ for an unknown $K$, it is hard to find $(P, C)$ such that $E_K(P) = C$ without using one of the oracles to compute $E_K(P)$ or $D_K(C)$ directly.*

*Proof.* Suppose (bwoc) that we have an efficient attack which allows us to find such a $(P, C)$ without using the oracles directly. Using this attack we find a valid plaintext/ciphertext pair, $(P, C)$, without using the oracles to compute $C = E_K(P)$ or $P = D_K(C)$ directly. During the attack one of two situations occurred:

1. We found two distinct plaintexts $P_1$ and $P_2$ such that $H(P_1) = H(P_2)$.

2. We did not find two distinct plaintexts $P_1$ and $P_2$ such that $H(P_1) = H(P_2)$.

In the first case, we found a collision in $H$ which contradicts the strongly collision free assumption on $H$. In the second case, we were able to find $(P, C^*, C')$ such that $C' = P \oplus S(H'_K(C^*))$ and $C^* = H(P)$. In particular, since we did not find a collision in $H$, we are able to compute $P \oplus C' = S(H'_K(C^*))$, where without using the oracle directly. This contradicts either our assumption that $H'$ resists existential forgery (since we were never given the value of $H_K(C^*)$) or that $S$ and $H'$ are independent.

□

| Block Size | 4096 | 65536 | 1024000 |
|---|---|---|---|
| AARDVARK | 1024000 | 2952072 | 3277849 |
| BEAR | 853333 | 1790601 | 1909735 |
| LION | 602353 | 2184533 | 2568991 |

Table 1: Encryption rates of AARDVARK, BEAR and LION for various block sizes. Block sizes are given in bytes. Encryption rates are given in bytes/second.

This theorem says that AARDVARK is secure against any combination of chosen plaintext/ciphertext attack provided that the above mentioned requirements on $H$ and $S$ are met. Security is defined in a very general sense: it is not computationally feasible for an attacker to find a single valid plaintext/ciphertext pair. This is a powerful result, since hash functions and stream ciphers exist which (for practical purposes) satisfy these requirements.

The attentive reader may have noticed that we have not made use of the "resists expansion attacks" requirement placed on $S$. It is, however, an important requirement. Since the block size of AARDVARK is large, it is possible that an adversary may know some of the plaintext of a message which means that the adversary knows a partial output of $S$. The "resists key expansion" requirement on $S$ prevents such an adversary from learning any more about the plaintext.

AARDVARK is clearly resistant to meet-in-the-middle attacks since the key is not divided in any way, and has a built in method of ensuring ciphertext integrity. Since AARDVARK uses only one application each of the hash function and stream cipher plus an application of the keyed hash function on a short bit string, one would also hope that it would run faster than BEAR or LION.

# 5 Performance of AARDVARK, BEAR, and LION

In order to test the relative performance of AARDVARK, BEAR, and LION we implemented them. SHA [NBS93] was used as the hash function and SEAL [RoCo93] as the stream cipher. In BEAR and AARDVARK, SHA was keyed using $H'_K(M) = \text{SHA}(K\|M\|K)$. The stream cipher was given by $S(M) = \text{SEAL}_M(0)\|\text{SEAL}_M(1)\|\cdots$. The test machine was a Sun Ultra-Sparc with a 140MHz Ultra-1 processor. All source code was written in C and compiled using *gcc* with optimization enabled.

All three ciphers were tested with various block sizes.The results are shown in table 1. As we would expect, AARDVARK outperforms BEAR and LION for all block sizes, with encryption rates varying from 1Mbyte/sec with a 4KByte block size to 3.2MBytes/sec with a 1MByte block size.

A perhaps surprising result is that BEAR outperforms LION for small block sizes, but the opposite is true for larger block sizes. This is because SEAL runs faster than SHA, but has a high key setup time. Because of this, LION which uses two applications of SEAL is slower for small block sizes, but as the block size is increased, the key setup times become less important and LION overtakes BEAR. This suggests that a stream cipher with a low key setup time is called for.

# 6 Conclusions

Meet-in-the-middle attacks on both BEAR and LION have been presented which, although largely theoretical, suggest that these ciphers are not as strong as they could be. A new block cipher has been proposed which is more efficient than BEAR or LION, is provably secure against any combination of chosen plaintext and/or ciphertext attacks, and has a built-in method of verifying ciphertext integrity. This work is of practical interest as SHA/SEAL based AARDVARK makes for a very fast software block cipher.

As noted in section 5 and in [AnBi96], the key setup times for the SEAL stream cipher have a significant negative impact on the performance of block ciphers based on it. Bear, LION, and AARDVARK could be made substantially faster by the development of a stream cipher with lower key setup times. This is the subject of ongoing research.

As with all new cipher proposals, we encourage cryptanalysis of AARDVARK. Such analysis could take the form of attacks against general AARDVARK or against specific instances of AARDVARK (e.g. SHA/SEAL based AARDVARK).

# Acknowledgments

The author would like to thank Mike Just and Carlisle Adams for their comments and suggestions on earlier drafts of this paper.

# References

[AnBi96] R. Anderson, E. Biham, "Two Practical and Provably Secure Block Ciphers: BEAR and LION," in *Proceedings of the Third International Workshop on Fast Software Encryption*, Cambridge, UK, pp.113-120, 1996.

[BeRo93] M. Bellare, P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," in *First ACM Conference on Computer and Communications Security*, ACM, November, 1993.

[LuRa88] M. Luby, C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," in *SIAM Journal on Computing*, v. 17, no. 2, pp.373-386, 1988.

[MeHe81] R. Merkle, M. Hellman, "On the Security of Multiple Encryption," in *Communications of the ACM*, v.24, n.7, pp.465-467, 1981.

[NBS93] "Secure Hash Standard," National Bureau of Standards FIPS Publication 180, 1993.

[RoCo93] P. Rogaway, D. Coppersmith, "A Software-Optimized Encryption Algorithm," in *Proceedings of the 1993 Cambridge Security Workshop*, Cambridge, UK, pp.56-63, 1993. es