Abstract—This paper will layout the design of Bruce Schneier's Blowfish encryption algorithm along with a performance analysis and possible attacks. This examination will lead to a conclusion about the effectiveness of Blowfish in the current era.

I. Introduction

In the early 1990's it was clear that Data Encryption Standard (DES) and even 3DES had begun to show its age. With the advancement of computer hardware it was possible to see the day in which DES would no longer be safe for sensitive data. In 1993 Bruce Schneier released his design of a successor to DES called Blowfish. Blowfish is a general-purpose, symmetric block cipher, as is DES.

A commission started in 1997 by the National Institute of Standard and Technology (NIST) to find a replacement for the DES algorithm that would be called Advanced Encryption Standard (AES). In 1997, Bruce Schneier submitted Twofish, the successor to his Blowfish design. Ultimately his design was not chosen to be the standard and Blowfish (along with Twofish) have fallen out of the spot light.

Although Blowfish is not as well known as what became the AES, Blowfish has been unique and an efficient algorithm that has become popular in the open source community. Bruce Schneier has stated that, “Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone.” Bruce Schneier's website currently states that Blowfish is being used in over 150 products and welcomes adoption in future products.

II. Design Overview

Since Blowfish was designed to be a replacement for DES they share some core similarities. Blowfish and DES are both symmetric block ciphers with one key input per full completion. Blowfish can be integrated as a direct substitution into any system currently running DES with the benefit of executing faster than DES.

A. Data Blocks

Blowfish is a block cipher which means that it encrypts small data pieces at a time of the entire file before repeating the algorithm again
on the next data element if data is still waiting to be processed (Figure 1). This particular algorithm is implemented by encrypting one 64 bit block segments at a time, which is half the size of the block segments in the AES algorithm.

Figure 1

B. Key Expansion

Blowfish can accommodate a variable key length from 32 bits up to 448 bits. This key is not directly used for encryption but used in the creation of many subkeys designed to be patternless.

Most symmetric block ciphers use substitution boxes (S-Boxes) to scatter the data. One unique feature of the Blowfish algorithm is that it dynamically creates the S-Boxes based on the input key commonly known as “nothing up my sleeve number.”

C. Structure

Blowfish is a Feistel cipher (named after physicist Horst Feistel) which means in the design of Blowfish there exists a symmetric structure. This structure is iterated 16 times before reaching completion. A Feistel structure has many advantages, especially in hardware, as to decrypt the cipher text all that is needed is a reversal of the key schedule.

III. Detailed Design

The Blowfish algorithm can be broken down into two sections: data encryption and key schedule/expansion.

A. Data Encryption

Encryption begins with a 64 bit block element of plain text that will be morphed into a 64 bit cipher text. The 64 bit segment is immediately split into two equally sized segments that will be used as the base of the Blowfish algorithm. The exclusive-or-operation (XOR) is performed between the first 32 bit block segment (L) and the first P array (Figure 2). The resulting 32 bit data is passed to the F function (details about the F function can be found in Section C) which permutes the data and yields a 32 bit block segment. This permuted block segment is XOR'ed with the second 32 bit segment (R) created by the 64 bit plain text split. After the XOR operation is complete the 32 bit segments L and R are swapped for future iterations of the Blowfish algorithm.
B. Key Schedule

Before traversal of the algorithm can begin, the P array and S-boxes must be defined. The P array is a reference to 18 independent sub arrays each of 32 bit length. Each P array and S-Box is initially defined using digits of π, which currently is assumed to not have a detectable pattern. The first P array (P1) is defined as the first 32 bits of π, the second P array (P2) is defined as the second 32 bits of π until all P arrays have been defined.

Blowfish allows an encryption key that ranges from 32 bits up to 448 bits. This algorithm is designed to accurately accept a variable key size by XOR the first 32 bits of the key with the first P array, the second 32 bits of the key, if present, with the second P array and continues until the end of the key schedule. If the end of the key is reached and P arrays are still waiting to be created the key rolls back to the first 32 bits and the execution continues. The resulting subkeys are still not considered to be truly cryptic although they come from random numbers.

C. F Function

The F function is arguably the most complex section of the algorithm and the only section that uses the S-Boxes. The F function accepts a 32 bit stream of data and divides the input into four equal sections. Each 8 bit subdivision is transformed into a 32 bit data stream by means of their corresponding S-Box. The resulting 32 bit data is XOR'ed or added together to provide a final 32 bit value for further permutations of the Blowfish algorithm, see Figure 3 for details (note that all addition is modulo $2^{32}$).
IV. Execution

When Blowfish is invoked the first step operation is to compute the P array and S-Boxes known as the key schedule. Upon completion of the key schedule the first iteration of Blowfish algorithm begins. The initial input uses zeros as the 64 bit block. The resulting cipher text is 64 bits and is used to overwrite the first P array and the second P array (note: the P arrays are 32 bits in size). The cipher text is used as input for the next iteration resulting in a new P3 and P4 subkeys. This cycle will continue until all P arrays have been successfully filled.

The first nine iterations are necessary to compose the P array, the tenth iteration is the start the encryption of the first 64 bits of plain text. Note that the first through the ninth iteration computes the P array once and will be reused until all plain text has been encrypted.

Due to the fact that the Blowfish algorithm must dynamically compute all subkeys and S-boxes before the start of encryption, there is a slight overhead in the computation. This computational overhead of subkeys and S-box creation results in roughly the equivalent of encrypting an additional four kilobytes of data per data file.

V. Sudo Code

Main:
Get input
L = input[0-31]
R = input[32-63]
For i = 0 to i = 15
   L = L XOR π
   L = f-Function(L) (detailed F function below)
   R = L XOR R
   Swap L and R
Swap L and R (intended to nullify the last swap)
L = L XOR P18
R = R XOR P17
output = combined result of L and R
return output

f-Function:
Get input
L = input[0-7]
centerL = input[8-15]
centerR = input[16-23]
R = input[24-31]
L = S-Box(L) (note that the result of the S-Box is a 32 bit data stream)
centerL = S-Box(centerL)
centerR = S-Box(centerR)
R = S-Box(R)
L = L + centerL (note: this is mod 2^{32} addition)
L = L XOR centerR
L = L + R
return L

VI. Performance

Blowfish is still one of the more efficient algorithms due to the fact that it can be implemented using simple operations that a
microprocessor can execute quickly. Quality implementation of Blowfish do not use variable length shifts or conditional jumps that take more processing time for a microprocessor to execute. Abdel-Karim Al Tamini at Washington University in St. Louis published a performance evaluation paper [6] of multiple algorithms speeds. The results from the Washington University study (Figure 4) show that Blowfish is a worthy contender in the cryptographic community outperforming its intended competition.

![Performance with CBC](image)

**Figure 4**

V. Vulnerability

Currently there is no known successful attack against Blowfish other than an exhaustive key search of every possible key (brute force attack). Although Serge Vanudenay's argued in his paper “On Weak Keys of Blowfish” [4] that it remains a possibility that the key size could be reduced. Vanudenay states that by dynamically creating the S-Boxes there exists a potential of accidental replication such that byte \( x \) is equal to byte \( x' \). Tests of this theory proved to be successful through round 8 of Blowfish, unfortunately not practical in real use. In Vanudenay's tests he foreknew the S-Box values before creation, a luxury attackers should not have.

VII. Security

Since the inception of Blowfish there has been vast scrutiny by the cryptographic community with virtually no success. Some have theorized weaknesses but none have effectively reduced the key size in real world simulations.

Conclusion

Cryptography is an evolving area and what is safe today may not be safe tomorrow. The unfortunate reality is that encrypted data can always be converted back to its original form by anyone who is willing to try every possible key until the answer is found, the purpose of encryption is to make them wait so long that it is irrelevant once they get the key. Currently Blowfish is able to provide long term data security without any known backdoor vulnerability or ability to reduce the key size. For the foreseeable future Blowfish is a safe and effective design although future reevaluations will be needed.

References

1. Bruce Schneier, Description of a New Variable-Length Key, 64-bit Block Cipher


6. Abdel-Karim Al Tamimi, Performance Analysis of Data Encryption Algorithms. [link]