Bitslice Implementation of AES

Chester Rebeiro, David Selvakumar, and A.S.L. Devi

Real Time Systems Group
Centre For Development of Advanced Computing
Bangalore, India
{rebeiro, david}@cdac.in

Abstract. Network applications need to be fast and at the same time provide security. In order to minimize the overhead of the security algorithm on the performance of the application, the speeds of encryption and decryption of the algorithm are critical. To obtain maximum performance from the algorithm, efficient techniques for its implementation must be used and the implementation must be tuned for the specific hardware on which it is running.

Bitslice is a non-conventional but efficient way to implement DES in software. It involves breaking down of DES into logical bit operations so that N parallel encryptions are possible on a single N-bit microprocessor. This results in tremendous throughput. AES is a symmetric block cipher introduced by NIST as a replacement for DES. It is rapidly becoming popular due to its good security features, efficiency, performance and simplicity. In this paper we present an implementation of AES using the bitslice technique. We analyze the impact of the architecture of the microprocessor on the performance of bitslice AES. We consider three processors; the Intel Pentium 4, the AMD Athlon 64 and the Intel Core 2. We optimize the implementation to best utilize the superscalar architecture and SIMD instruction set present in the processors.

1 Introduction

Security is the most important feature of a cryptographic algorithm. An important secondary requirement is an efficient implementation in hardware and software. The most efficient implementations are generally done in dedicated hardware engines such as in FPGAs and ASICs. However, there are several applications such as networking software, operating system modules, etc., which require fast encryptions but do not have these hardware engines. These applications make an efficient software implementation of cryptographic algorithms important.

The bitslice implementation of DES [1][5] is the most efficient software implementation of DES. It involves converting the algorithm into a series of logical bit operations using XOR, AND, OR and NOT logical gates. When implemented on a microprocessor with a N-bit register width, each bit in the register acts as a 1-bit processor doing a different encryption, therefore N encryptions are done in parallel. This results in significant improvements in throughput.

D. Pointcheval, Y. Mu, and K. Chen (Eds.): CANS 2006, LNCS 4301, pp. 203–212, 2006.

Another advantage of a bitslice implementation is that it is immune to cache-timing attacks. Traditional methods of implementing block ciphers make use of several tables to improve performance [4]. The memory access patterns of the implementation make it vulnerable to cryptanalysis [12] [13]. A bitslice implementation on the other hand is based only on logical operations, there are no tables involved, therefore it is free from attacks based on cache timing analysis.

AES is a symmetric key algorithm and offers higher security compared to DES. The simplicity of its design results in efficient implementations on software platforms. In this paper we try to improve its performance by adapting the bitslice techniques used in DES to AES. We first review the AES algorithm. We then present our implementation of AES encryption using the bitslice technique. In the next section we discuss architecture features of the microprocessor which impact the performance of the encryption. We discuss the Intel Pentium 4 (with EM64T), AMD Athlon 64 and the Intel Core 2 microprocessors. All these microprocessors support 64-bit integer operations and 128-bit SIMD operations. The SIMD operations are supported by the Streaming SIMD Extensions (SSE) instructions. The fourth section has the mode of operation of the ciphers, the fifth has the related work followed by the conclusion in the final section.

1.1 The AES Algorithm

The AES algorithm operates on a 4×4 matrix of bytes called *state*. The state undergoes a series of transformations during the encryption process [*Algorithm:1*]. Each iteration in the encryption process is called a *round*. The number of rounds (N_r) is determined by the size of the AES key. $N_r = 10$, 12 or 14 for key sizes of 128, 192 or 256 bits respectively. All operations on the state are in the Galois Field $GF(2^8)$. The *SubstituteByte* function substitutes each byte with a value from a lookup table called *Sbox*. The entries in the Sbox are obtained by taking the inverse of each element in $GF(2^8)$ followed by a linear affine transformation. The *ShiftRow* function shifts each byte in the row by an offset. The *MixColumn*

Algorithm 1. AES Encryption

Input: 4×4 Plaintext bytes
Output: 4×4 Cyphertext bytes

```
1  AddInitialKey
2  for round = 1 to N_r do
3      SubstituteByte
4      ShiftRow
5      MixColumn
6      AddRoundKey
7  end
8  SubstituteByte
9  ShiftRow
10 AddRoundKey
```