Arjun Rao
B2 - 41
140905484

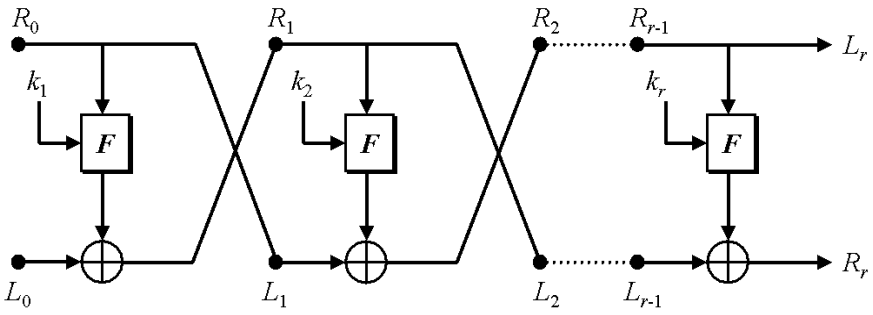
Avikant Saini
B2 - 43
140905508

Parallel Programming Lab
Mini Project Report

Counter Block Code Encryption

Abstract

The need for encryption of data does not require an explicit justification. Block coding involves encryption of blocks of plaintext data into cipher text blocks. We will be using 64-bit blocks in our encryption algorithm. The process uses a 64-bit key known only to the sender and receiver. In counter block coding a consecutively changing counter is used to encrypt the key to form a unique key for each individual block of plaintext. This involves encrypting and decrypting the cipher text by encrypting the key using a secret encryption algorithm along with the counter value. The counter value is changed for consecutive blocks of data using some incremental manipulation technique. Once the unique 64-bit key is formed it is XOR'ed with the 64-bit plaintext block to form the cipher text block. The cipher text blocks are the joined together in order to form the encrypted data file.



During decryption the same technique is used to extract the unique for each 64-bit block of cipher text, which is then inverted to form the decryption key. This decryption key is XOR'ed with the cipher text blocks to retrieve the original plaintext data file.

The encryption algorithm will involve alternating rounds of substitution and transposition ciphers, like in a Fiestel cipher.

Introduction

In cryptography, a mode of operation is an algorithm that uses a block cipher to provide an information service such as confidentiality or authenticity. A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called a block. A mode of operation describes how repeatedly to apply a cipher's single-block operation securely to transform amounts of data larger than a block.

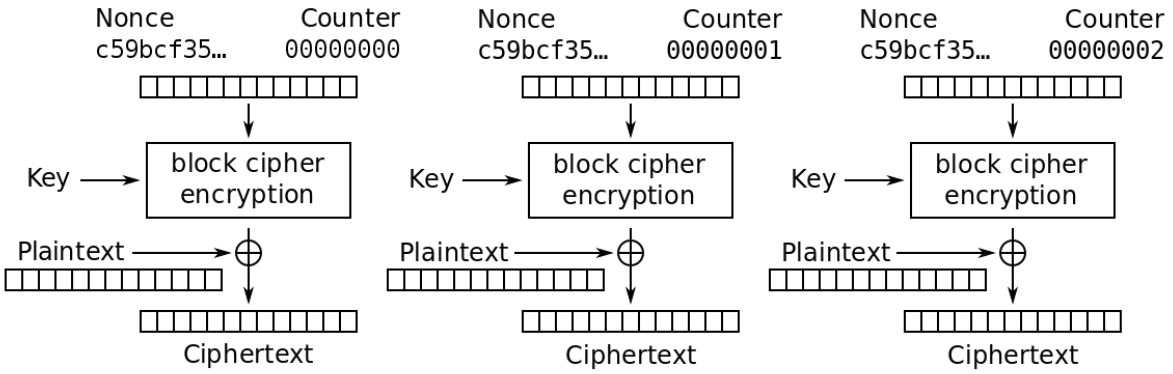
Most modes require a unique binary sequence, often called an initialization vector (IV), for each encryption operation. The IV has to be non-repeating and, for some modes, random as well. The initialization vector is used to ensure distinct ciphertexts are produced even when the same plaintext is encrypted multiple times independently with the same key.

Block ciphers have one or more block size(s), but during transformation the block size is always fixed. Block cipher modes operate on whole blocks and require that the last part of the data be padded to a full block if it is smaller than the current block size. There are, however, modes that do not require padding because they effectively use a block cipher as a stream cipher.

Historically, encryption modes have been studied extensively in regard to their error propagation properties under various scenarios of data modification. Later development regarded integrity protection as an entirely separate cryptographic goal. Some modern modes of operation combine confidentiality and authenticity in an efficient way, and are known as authenticated encryption modes.

Counter Block Mode Encryption

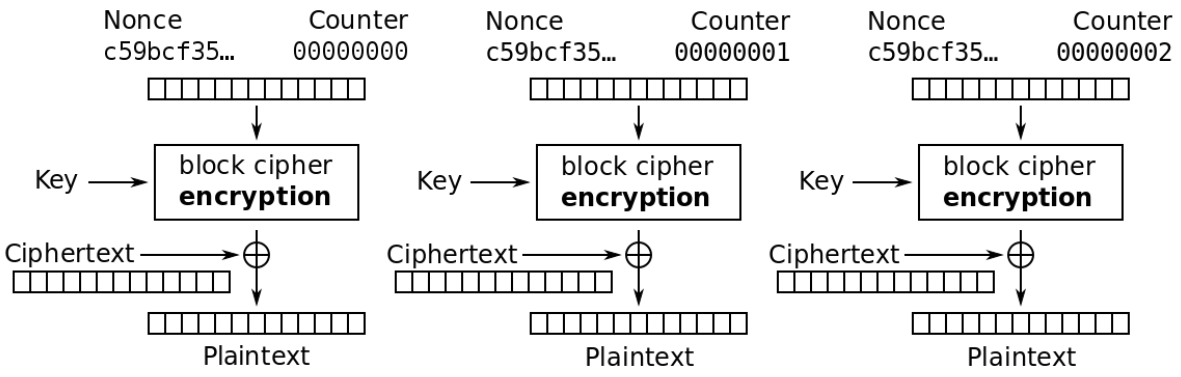
Like Output Feedback Mode, Counter mode turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a "counter". The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual increment-by-one counter is the simplest and most popular. The usage of a simple deterministic input function used to be controversial; critics argued that "deliberately exposing a cryptosystem to a known systematic input represents an unnecessary risk." However, today CTR mode is widely accepted and any problems are considered a weakness of the underlying block cipher, which is expected to be secure regardless of systemic bias in its input. Along with CBC, CTR mode is one of two block cipher modes recommended by Niels Ferguson and Bruce Schneier.



Counter (CTR) mode encryption

If the IV/nonce is random, then they can be combined together with the counter using any lossless operation (concatenation, addition, or XOR) to produce the actual unique counter block for encryption. In case of a non-random nonce

(such as a packet counter), the nonce and counter should be concatenated (e.g., storing the nonce in the upper 64 bits and the counter in the lower 64 bits of a 128-bit counter block). Simply adding or XORing the nonce and counter into a single value would break the security under a chosen-plaintext attack in many cases, since the attacker may be able to manipulate the entire IV-counter pair to cause a collision. Once an attacker controls the IV-counter pair and plaintext, XOR of the ciphertext with the known plaintext would yield a value that, when XORed with the ciphertext of the other block sharing the same IV-counter pair, would decrypt that block.



Counter (CTR) mode decryption

Methodology

It can be considered as a counter-based version of Cipher Feedback mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

Steps involved are as follows:

~ Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.

~ Encrypt the contents of the counter with the key and place the result in the bottom register.

~ Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.

~ Continue in this manner until the last plaintext block has been encrypted.

~ The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.

Algorithm

```
Set ctr base value
For j = 0 to 10
{
   Substitute(key)
   For i = 0 to 31
   {
      Key[i] = (key[i] + (ctr % 256)) %256
      Circular_shift(ctr, 8)
   }
   Ctr value warp
   Transposition(key)
   Shuffle array a[32] with values from 0 to 31
   For i = 0 to 31
   {
      Temp[i] = key[a[i]]
      Key[a[i]] = key[i]
      Key[i] = temp[i]
   }
}
```

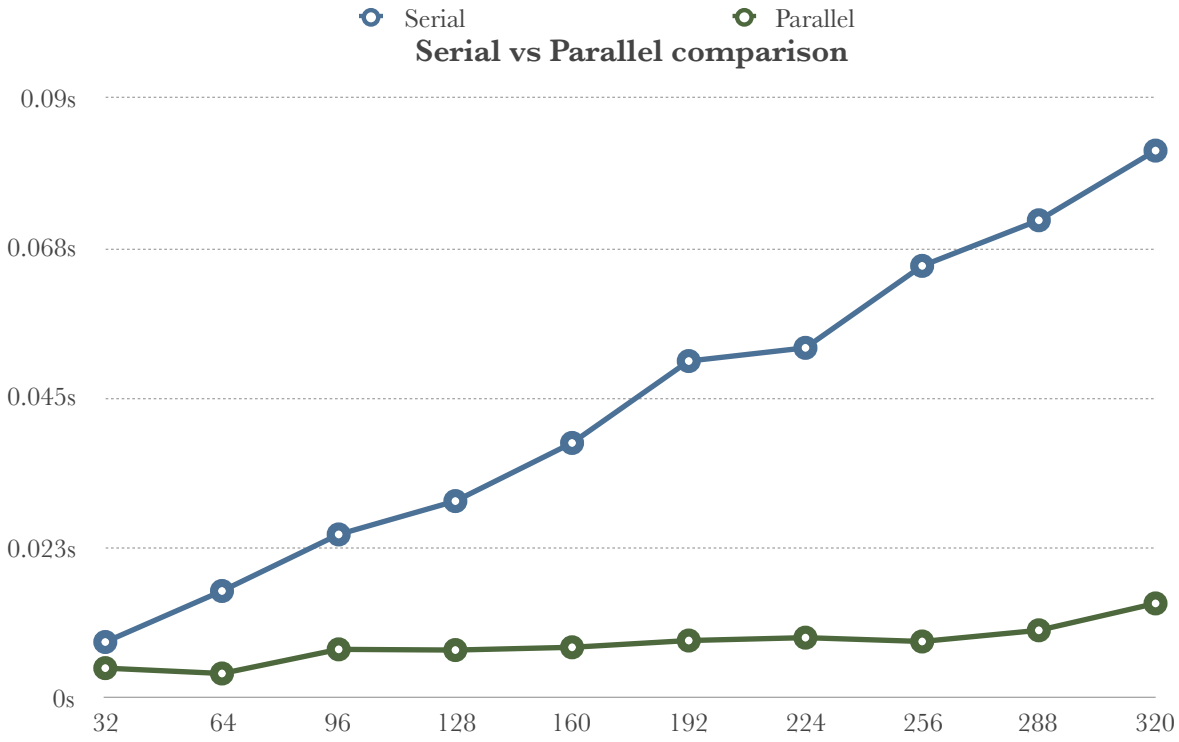
Results

The parallel implementation appears to be about 5 times faster on average when compared to the serial implementation of the same.

The speedup is a result of encrypting blocks of the data in parallel in separate threads. The time taken in parallel implementation is found to be directly proportional to the block size taken for the kernel.

Input size (kb)	Serial time (s)	Parallel time (s)	Times Faster
32	0.008441	0.004521	1.8671
64	0.016099	0.003715	4.3335
96	0.024587	0.007337	3.3511
128	0.029594	0.007232	4.0921
160	0.038320	0.007646	5.0118
192	0.050615	0.008662	5.8433
224	0.052596	0.009094	5.7836
256	0.064900	0.008518	7.6192
288	0.071746	0.010197	7.036
320	0.082203	0.014232	5.7759
		Average	5.07136

For an input set of file sizes from 32kb to 320kb, maximum speedup achieved was 7.6192 times for the file of size 256kb. This is due to evenly split block sizes for the kernel, allowing maximum efficiency.



Limitations and Possible Improvements

Currently we are using fixed size blocks of 32 bytes. The algorithm can be made more efficient by calculating the block size taking the file size and the number of threads available for processing. This will allow simultaneous execution of all the blocks, eliminating any wait time caused by irregular blocks.

Conclusion

Parallelization improves upon the execution time. The tradeoff between complexity and execution improves in favor of the parallel implementation as the input size increases.

References

- http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html
- http://cryptosmith.com/2008/05/31/stream-reuse/
- http://airccse.org/journal/nsa/0113nsa02.pdf
- http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38761
- http://www.iks-jena.de/mitarb/lutz/security/cryptfaq/q84.html