Fast Software Implementation: Bit Slicing

By: Necko Bailey, Ryan Fazal, & Daouda Gaye

Breakdown of presentation

- What is bit slicing?
- Origin of bit slicing
- AES encryption
- The Four Levels of AES encryption
- The Security Faults of AES

What is Bit Slicing

- First concepted by Eli Biham.
- a technique for implementing cryptographic algorithms was proposed to improve the software performance of DES [3]
-
- It takes the form of a hardware program, that is implemented within software.
- When it comes to AES, bit slicing takes the link layer, and or S boxes, and creates bit-logical directives.
- Emilia Kasper and Peter Schwabe claim that bit slicing is 25% faster than standard AES encryption.

Eli Biham [4]

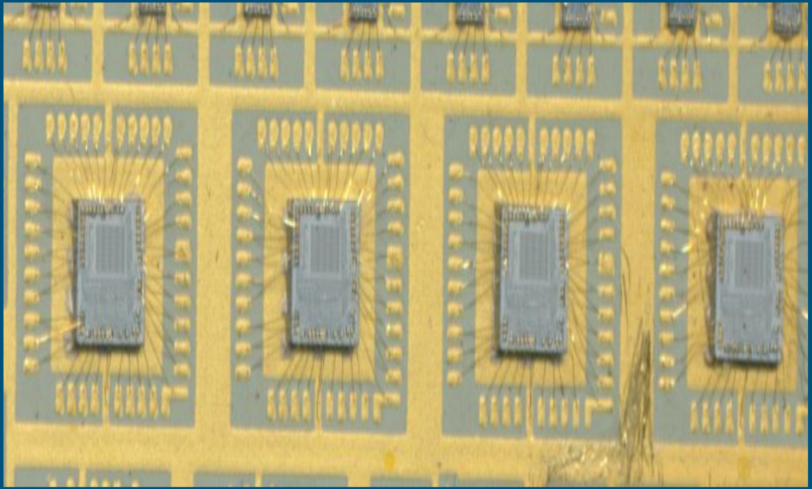
Peter Schwabe [5]

What is Bit Slicing and why is it important

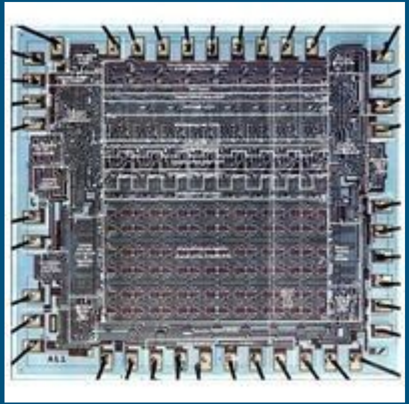
- bit -slicing is a method of combining processor modules to multiply the word length.
- Bit-slicing was common with early processors, notably the AMD(Advanced Micro Devices) 2900 series in 1975
- In a bit slicing processor, each modules contains an ALU (Arithmetic-logic Unit) usually capable of handling a 4-bit field.
- By combining two or more identical modules, it is possible to build a processor that can handle 8 bits ,12 bits, 16 bits, etc..
- Each module is called a slice
- The control lines for all the slices are connected effectively in parallel to share the processing”work” equally.
- Prior to about 1980, experimenters used bit-slicing to create more powerful computers. 64 bits processors were built.
- More recently, bit slicing has been used in specialized encryption methods such as block ciphers.

FIRST PROCESSOR

4x AMD AM2900 B



AL-1 8-bit computer processor slice.



The Origin

- First referred to as “Rijndael Cipher” after its developers.
- It was part of a 1997 competition to discover the newest encryption method.
- Due to its victory, it was later renamed AES in 2001.
- AES was short was Advanced Encryption Standard.

V.Rijmen (left) & J.Daemen (right) [1]

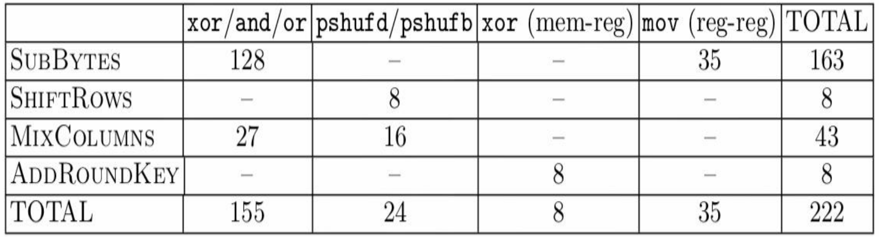
AES Encryption

- Encryption refers to a method of scrambling data so that is is unreadable to others it is not intended for.
- Allows for secure storage and transportation over a network.
- Uses a key to scramble data. The same key must be used for its decryption.
- Symmetric key algorithm.

Block Cipher

- It is able to encrypt 128 bits, the equivalent of 16 bytes, at a single time. Data larger than 128 bits is divided into 128 bits each.
- Each block of 128 bits is individually encrypted.
- If the data is not divisible by 16 bytes then the remainder is the number of bytes of padding needed.

AES Vs. Bit Slicing Implementations



	xor/and/or	pshufd/pshufb	xor (mem-reg)	mov (reg-reg)	TOTAL
SubBytes	128	–	–	35	163
ShiftRows	–	8	–	–	8
MixColumns	27	16	–	–	43
AddRoundKey	–	–	8	–	8
TOTAL	155	24	8	35	222

Table 1. Instruction count for one AES round

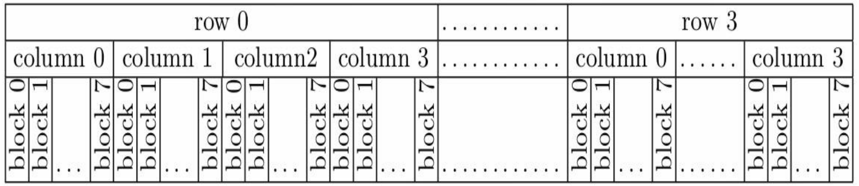
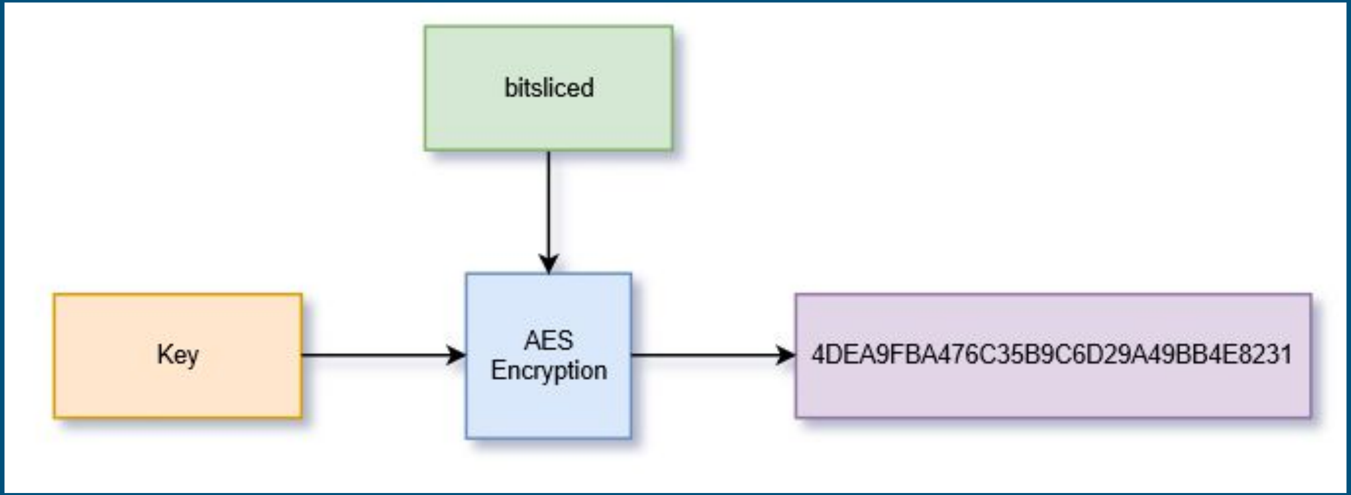


Fig. 1. Bit ordering in one 128-bit vector of the bitsliced state

- 4 processes for encryption
- 128 bits for each process
- Only 4 possible variations for a 128 bit encryption process

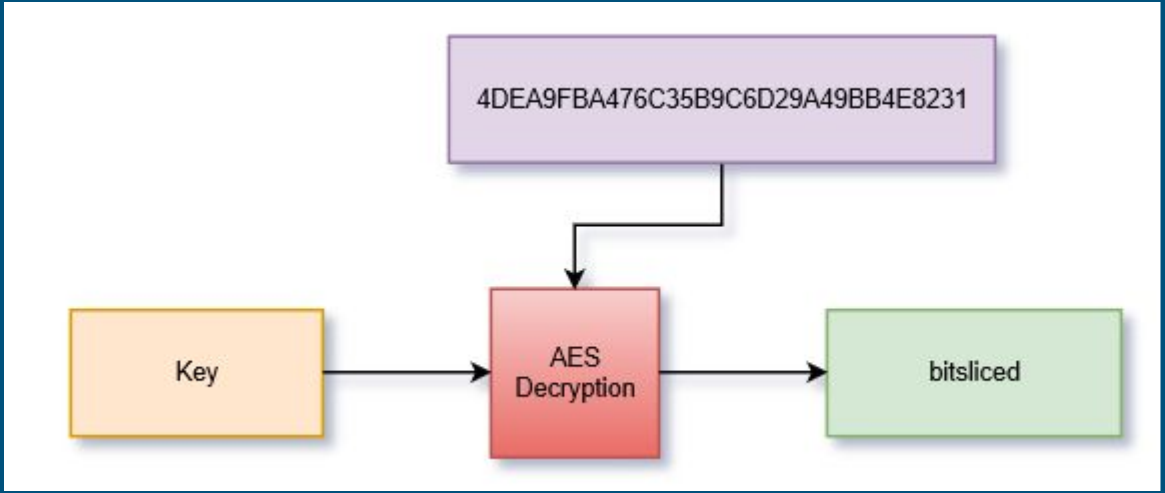
- Each block contains 8 16 bytes encryption
- For every block, the message is broken down in 8 separate parts
- Since each block is parallel to one another, that helps for an efficient flow of data processing,

First level



- In this level the programmer gives the data/message with the key needed.
- Both are given to the AES Encryption which, in return, gives back a scrambled, unrecognisable message.

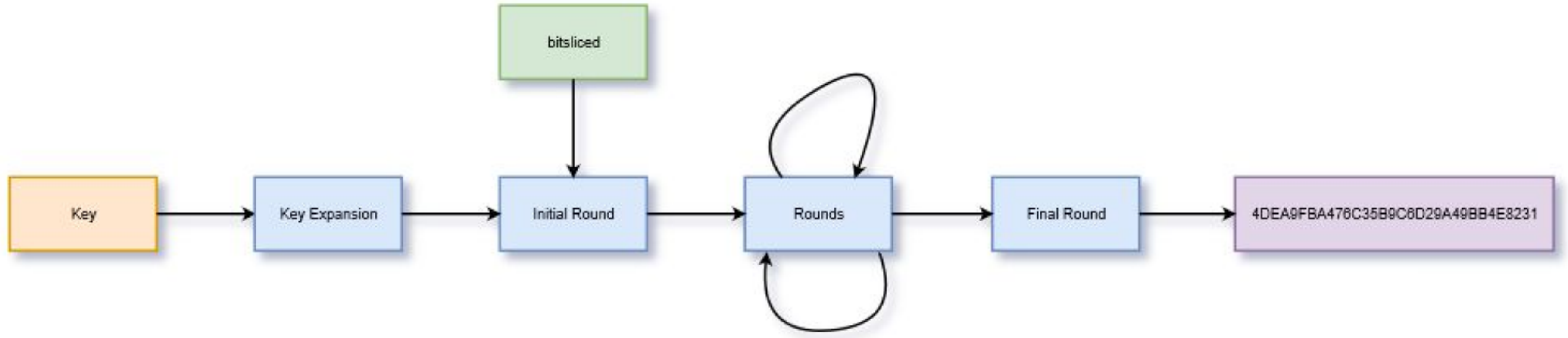
Decryption



- When decrypting data, the encrypted message along with its key is supplied to the AES Decryption.
- In return, this should output the original, decoded message (aka plaintext).

Second Level

- Uses a series of rounds in order to get the encrypted message.
- The number of rounds performed is a result of the size of the key.
- The last round is the simpler one.



Key Lengths

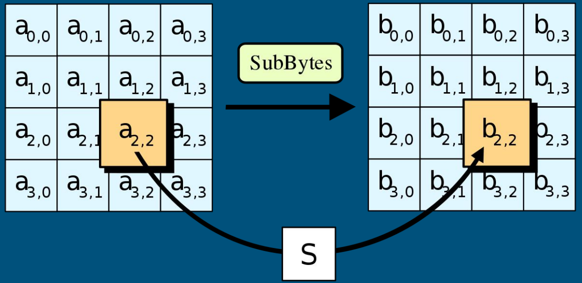
- Keys can range from 128, 192, and 256 bits.
- The original AES algorithm allows for 160 ad 224 bits as well.
- The larger the key in size the more secure because it goes through more rounds but has a slower encryption speed.
- The smaller the key size the less secure because it has less rounds but it has a faster encryption speed.
- 128 bit key are the fastest and most well know. However they are the least most secure
- 256 bit key is the most secure, however the slowest

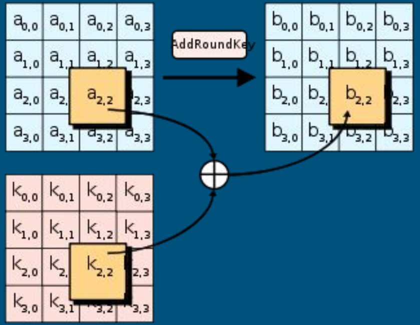
128 bit key	10 cycles
192 bit key	12 cycles
256 bit key	14 cycles

AES Key Expansion

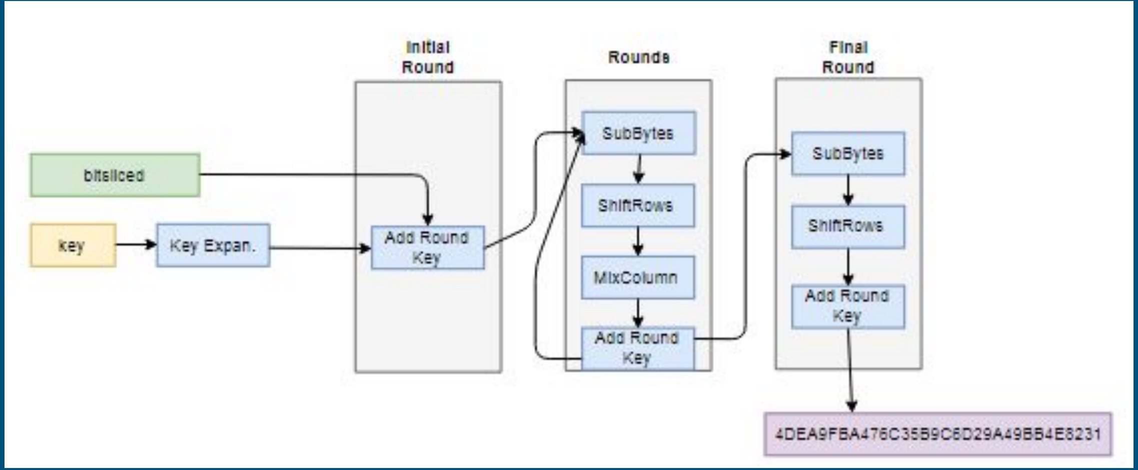
- Each round modifies the previous keys which are products of the original key.
- The modifications are AES Key Expansions.
- Changes the key along with the encrypted message.
- Each round results is a altered key meaning that the key is able to encrypt itself.

Third Level

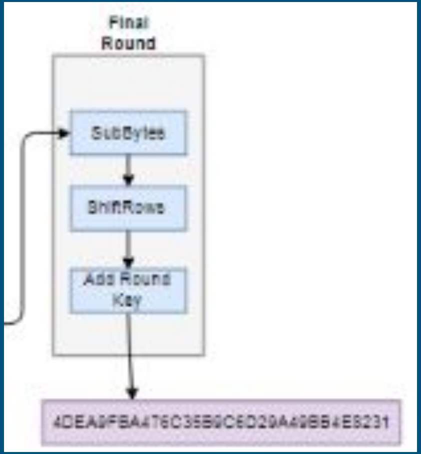




- The user supplies a message and the key which expands into how many keys are needed to the initial round.
- Moves onto the rounds where the data is shifted and mixed and the rounds are repeated.
- Data mixing involves substituting bytes, shifting rows, and adding round key.
- Moves onto the final round of shifting then the encrypted message is the result.



Fourth Level



- SubBytes
  - Replaces each byte with another based on the key.
  - The Rijndael S-box is comprised of 256 bit substitutions in a 16 by 16 grid form.
- Shift Rows
  - The first row is not shifted.
  - Each row after is shifted a byte to the left. The second is by 1, the third is by 2 and the last is by 3.
  - As bytes shift they rotate to the right.
- Add Round Key
  - Used in the initial round and looped rounds.
  - Adds the state and round keys.

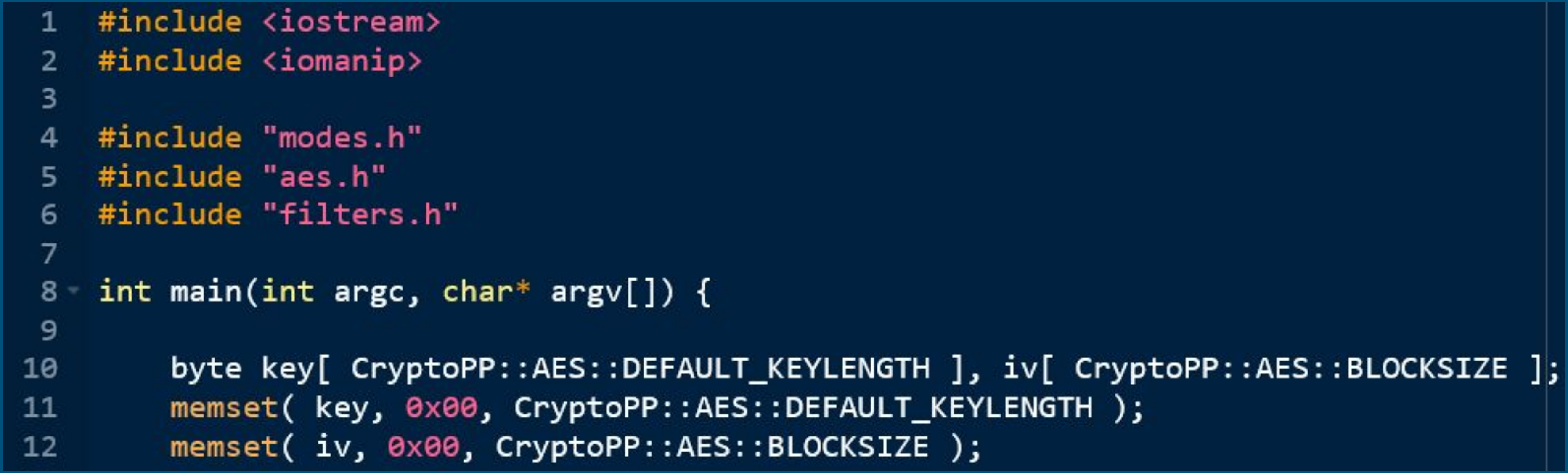
Security Faults of AES

- A typical implementation of AES uses precomputed lookup tables to implement the S-Box, opening up an opportunity for a cache-timing attack. [2]
- The xor of the plaintext and the first round key are all an attacker needs to gain access.
- An attacker can implement an attack on the key schedule. Here the secret key can be intercepted.

- With bit slicing, it takes the process of (XOR) and reduces its influence.

- If we XOR the polynomial to a result, which 9th bit is 00 (i.e. result does not overflow), then it will overflow since the irreducible polynomial in AES is 100011011100011011 (in bitform).[6]

Algorithm: Part 1

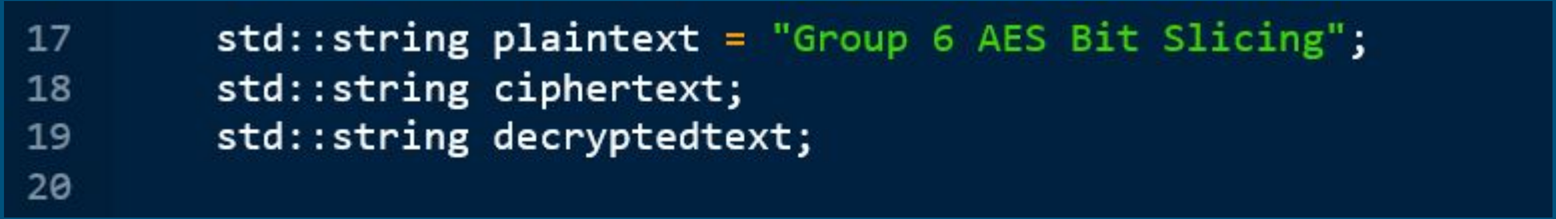


```
#include <iostream>
#include <iomanip>

#include "modes.h"
#include "aes.h"
#include "filters.h"

int main(int argc, char* argv[]) {

    byte key[ CryptoPP::AES::DEFAULT_KEYLENGTH ], iv[ CryptoPP::AES::BLOCKSIZE ];
    memset( key, 0x00, CryptoPP::AES::DEFAULT_KEYLENGTH );
    memset( iv, 0x00, CryptoPP::AES::BLOCKSIZE );
```

This is our setup for the key and IV. They are necessary for encryption. IV is used for our public information, while the key is needed for the amount of security we want for the plaintxt.
As we know, AES has a key of length (128, 196, 256 bit).
The key created above will be exchanged between ex. Alice/Bob before communication begins.
DEFAULT_KEYLENGTH = 16 bytes

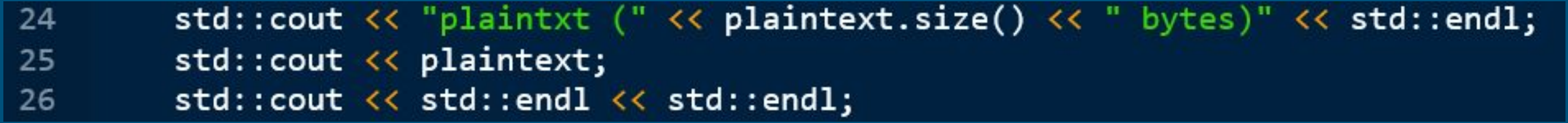
Algorithm: Part 2

Here “Group 6 AES Bit Slicing file is partitioned.String and sink setup is used to break apart the plain text into smaller bits of data



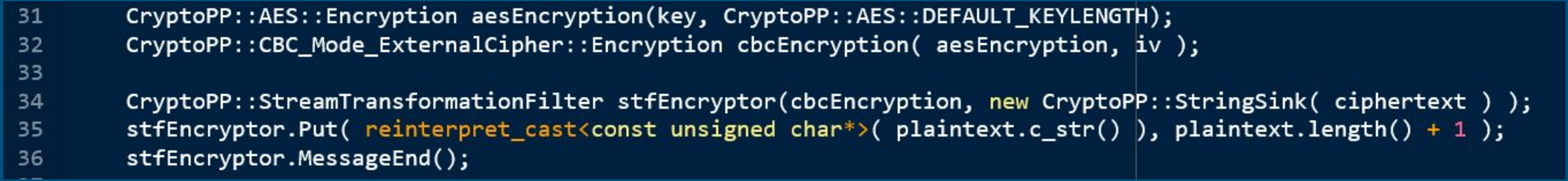
```cpp
    std::string plaintext = "Group 6 AES Bit Slicing";
    std::string ciphertext;
    std::string decryptedtext;

```

Dump plain text is used to create an archive for our plain text.



```cpp
    std::cout << "plaintxt (" << plaintext.size() << " bytes)" << std::endl;
    std::cout << plaintext;
    std::cout << std::endl << std::endl;
```

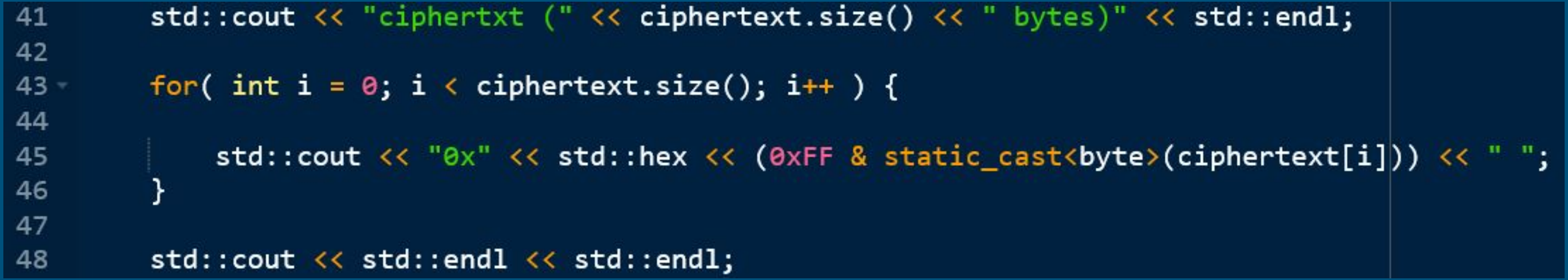
Creation of Cipher Text. In works with the IV, Cipher Block Chaining takes our specified data, and encrypts It into individual blocks.



```cpp
    CryptoPP::AES::Encryption aesEncryption(key, CryptoPP::AES::DEFAULT_KEYLENGTH);
    CryptoPP::CBC_Mode_ExternalCipher::Encryption cbcEncryption( aesEncryption, iv );

    CryptoPP::StreamTransformationFilter stfEncryptor(cbcEncryption, new CryptoPP::StringSink( ciphertext ) );
    stfEncryptor.Put( reinterpret_cast<const unsigned char*>( plaintext.c_str() ), plaintext.length() + 1 );
    stfEncryptor.MessageEnd();
```

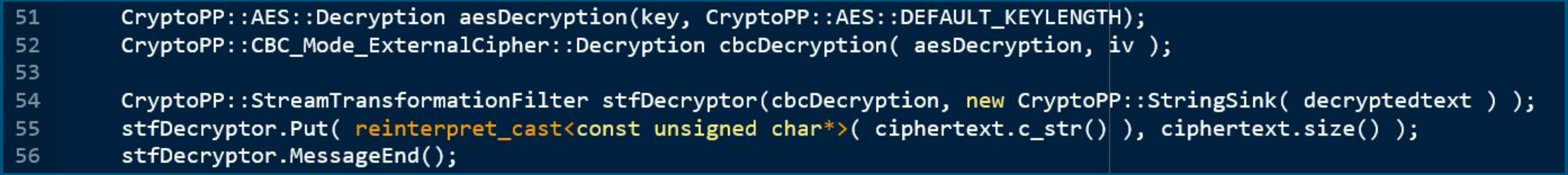
Algorithm: Part 3

Dump of Cipher Text is used to archive the files of the cipher text.



```
    std::cout << "ciphertxt (" << ciphertext.size() << " bytes)" << std::endl;

    for( int i = 0; i < ciphertext.size(); i++ ) {

        std::cout << "0x" << std::hex << (0xFF & static_cast<byte>(ciphertext[i])) << " ";
    }

    std::cout << std::endl << std::endl;
```

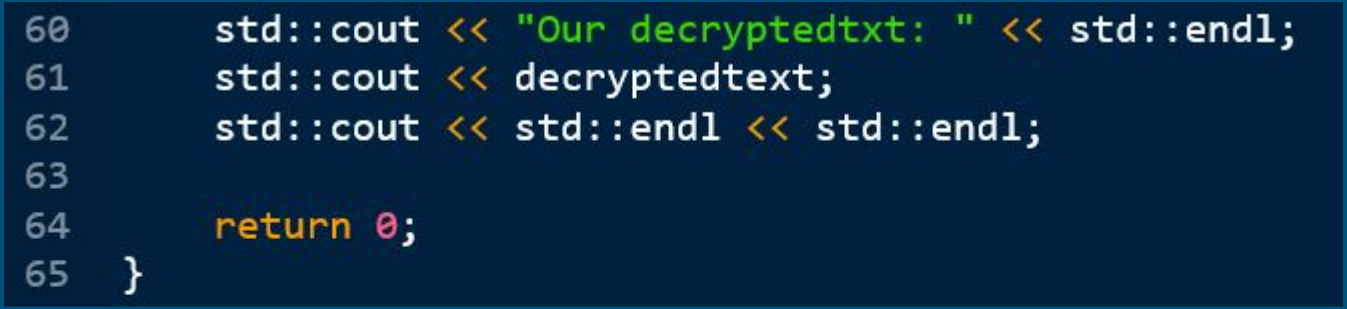
Decryption



```
    CryptoPP::AES::Decryption aesDecryption(key, CryptoPP::AES::DEFAULT_KEYLENGTH);
    CryptoPP::CBC_Mode_ExternalCipher::Decryption cbcDecryption( aesDecryption, iv );

    CryptoPP::StreamTransformationFilter stfDecryptor(cbcDecryption, new CryptoPP::StringSink( decryptedtext ) );
    stfDecryptor.Put( reinterpret_cast<const unsigned char*>( ciphertext.c_str() ), ciphertext.size() );
    stfDecryptor.MessageEnd();
```

Algorithm: Part 4

Decryption



```
    std::cout << "Our decryptedtxt: " << std::endl;
    std::cout << decryptedtext;
    std::cout << std::endl << std::endl;

    return 0;
}
```

Citations

[1] http://cs-exhibitions.uni-klu.ac.at/index.php?id=402
[2] https://eprint.iacr.org/2009/129.pdf
[3] https://eprint.iacr.org/2009/129.pdf
[4]http://www.cs.technion.ac.il/~biham/cv.html
[5]https://www.ru.nl/english/news-agenda/news/vm/icis/cyber-security/2017/peter-schwabe-dutch-prize-ict-research/
[6]https://crypto.stackexchange.com/questions/35132/how-can-bit-slicing-be-constant-time-when-mix-columns-is-in-the-cipher

Thank You!

Hope you Guys enjoyed :-)