AES Image Encryption with CUDA C

Camden Landis

CMDA 4984 Final Presentation
Wednesday, December 08, 2021

Goal of Encryption

- The purpose of encryption is *confidentiality*—concealing the content of data by translating it into a code.
- Advanced Encryption Standard (AES) is one of a multitude of symmetric block ciphers.
  - Currently, the US government uses AES to protect classified information.

Mathematical Background for AES

- Fields and Finite Fields
  - Mathematical set in which operations of arithmetic are defined and satisfy certain basic rules
  - Rijndael's (rain-dahl or rhine-dahl) Finite Field
  - Byte Polynomial Representations
- Modular Arithmetic
- Polynomial Arithmetic

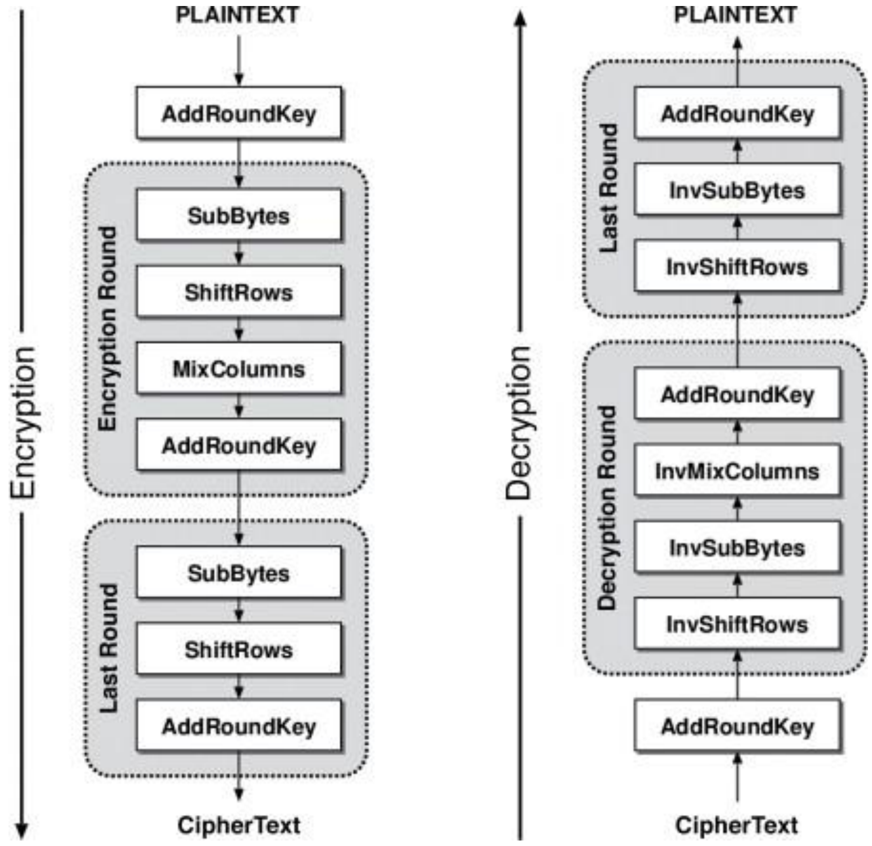
AES Overview

- When DES was originally proposed as a standard, one of the criticisms was the relatively short size of the key space.
- In 2002, NIST selected Rijndael (developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen) to be the AES.
- AES has a block length of 128 bits, and it allows key lengths of 128, 192, and 256 bits.

AES Cipher

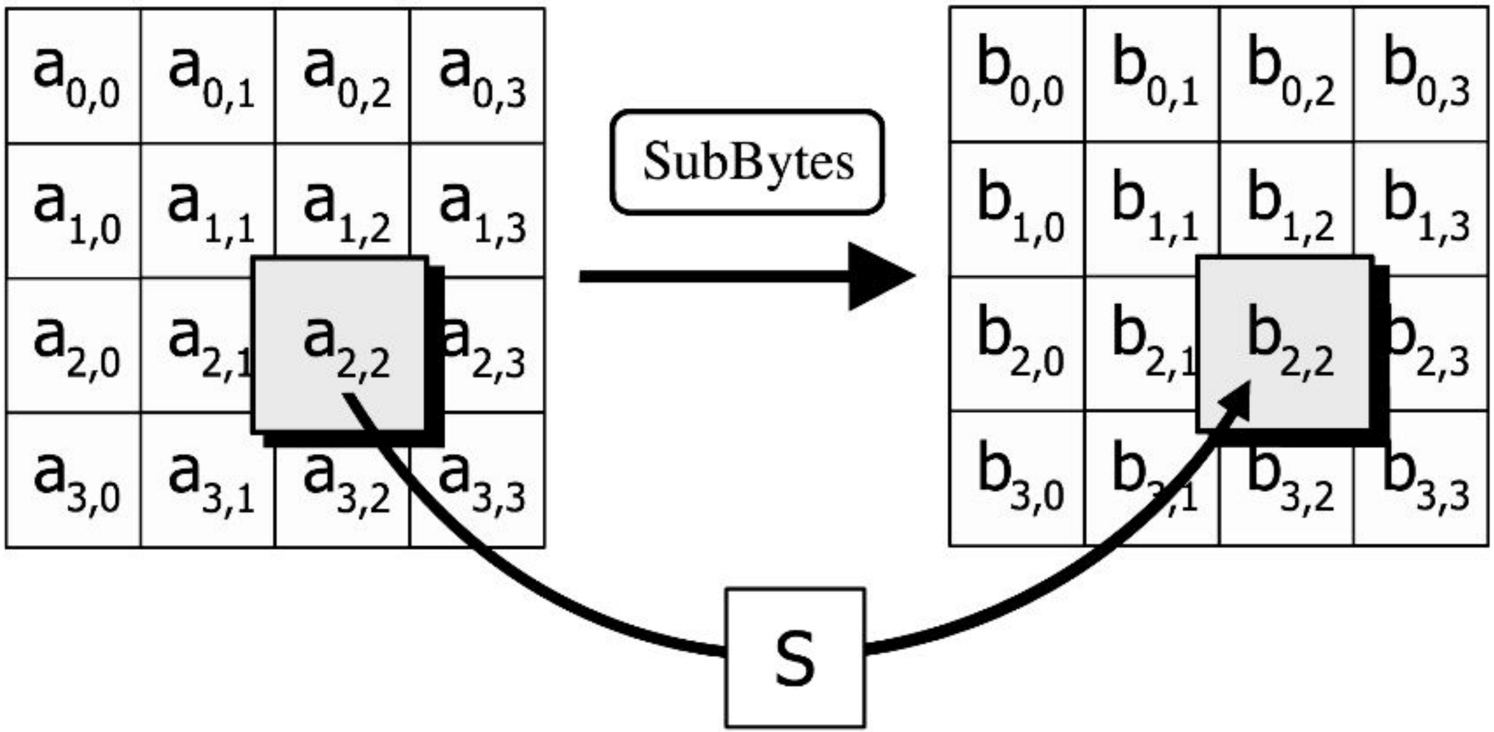
- Four layers used to form each round of the cipher:
  - SubBytes – A non-linear S-box substitution
  - ShiftRows – An index permutation
  - MixColumns – A linear “hybrid” transformation
  - AddRoundKey – Standard key mixing

AES Cipher



Encryption
PLAINTEXT
AddRoundKey
Encryption Round
SubBytes
ShiftRows
MixColumns
AddRoundKey
Last Round
SubBytes
ShiftRows
AddRoundKey
CipherText
Decryption
PLAINTEXT
Last Round
AddRoundKey
InvSubBytes
InvShiftRows
Decryption Round
AddRoundKey
InvMixColumns
InvSubBytes
InvShiftRows
AddRoundKey
CipherText

AES SubBytes



AES SubBytes

Example: Suppose that we begin with the byte 01010011, which is **53** in hexadecimal, that represents the field element $y = \mathbf{x}^6 + \mathbf{x}^4 + \mathbf{x} + 1$.
One can verify that the multiplicative inverse in the field F is $y^{-1} = \mathbf{x}^7 + \mathbf{x}^6 + \mathbf{x}^3 + \mathbf{x}$. Therefore, in binary notation, we have $y^{-1} = 11001010$. Then

$$\left.\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\} \pi_S(\mathbf{53}) = \mathbf{ED}$$

AES SubBytes S-Box

This computation can be checked by verifying that the entry in row 5 and column 3 of the table is ED.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

AES ShiftRows

ShiftRow: Let the output of SubByte be:

$$B = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

For $0 \leq i \leq 3$, the i-th row of the matrix is left-shifted cyclically to yield:

$$\begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} = C$$

C is the output of the ShiftRow.

AES MixColumns

MixColumn: Still viewing a byte as an element of F, we perform the following matrix multiplication to get:

$$D = \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix}$$

D is the output of the MixColumn Layer.
Note: $02 = 00000010$ corresponds to x, $03 = 00000011$ corresponds to $x+1$ and $01 = 00000001$ corresponds to 1.

AES AddRoundKey

AddRoundKey: A round key is generated from the given 128 bit key, and bitwise XOR-ed with output D of MixColumn:

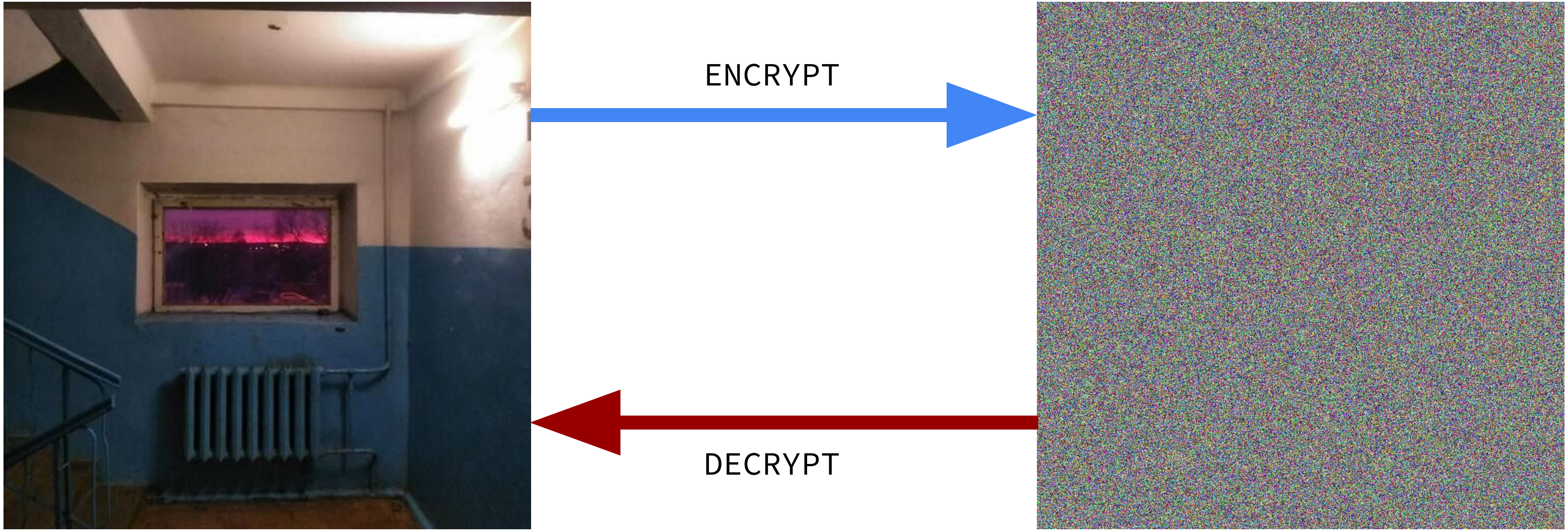
$$E = \begin{bmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

AES Decryption

- In order to decrypt, one needs to perform all the operations in the reverse order and use the key schedule in reverse order.
- The operations ShiftRows, SubBytes, MixColumns, and AddRoundKey must be replaced by their inverse operations.
    - Note: AddRoundKey is its own inverse.
- The AES is secure against all known attacks.
    - There are apparently no known attacks on AES that are faster than brute force exhaustive search which could take billions of years on current hardware.

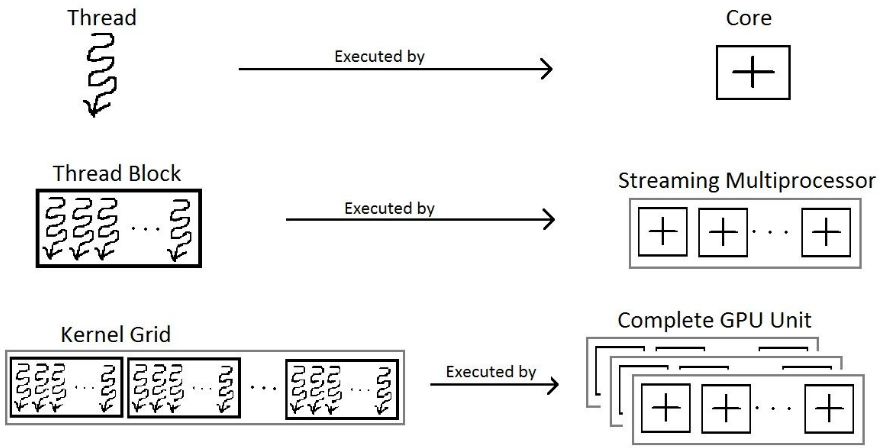
Image Encryption with AES



bitmap (.bmp) image

Parallel Implementation

- Since AES is a block cipher which operates on 128-bit of input at a single iteration, we can divide the image into 16 pixel block sizes.
- Currently, the threads per block is set at 512 threads.
- Ideally, this means that 16 pixels will be operated on by a single thread and 8,192 pixels will be processed by each thread-block.

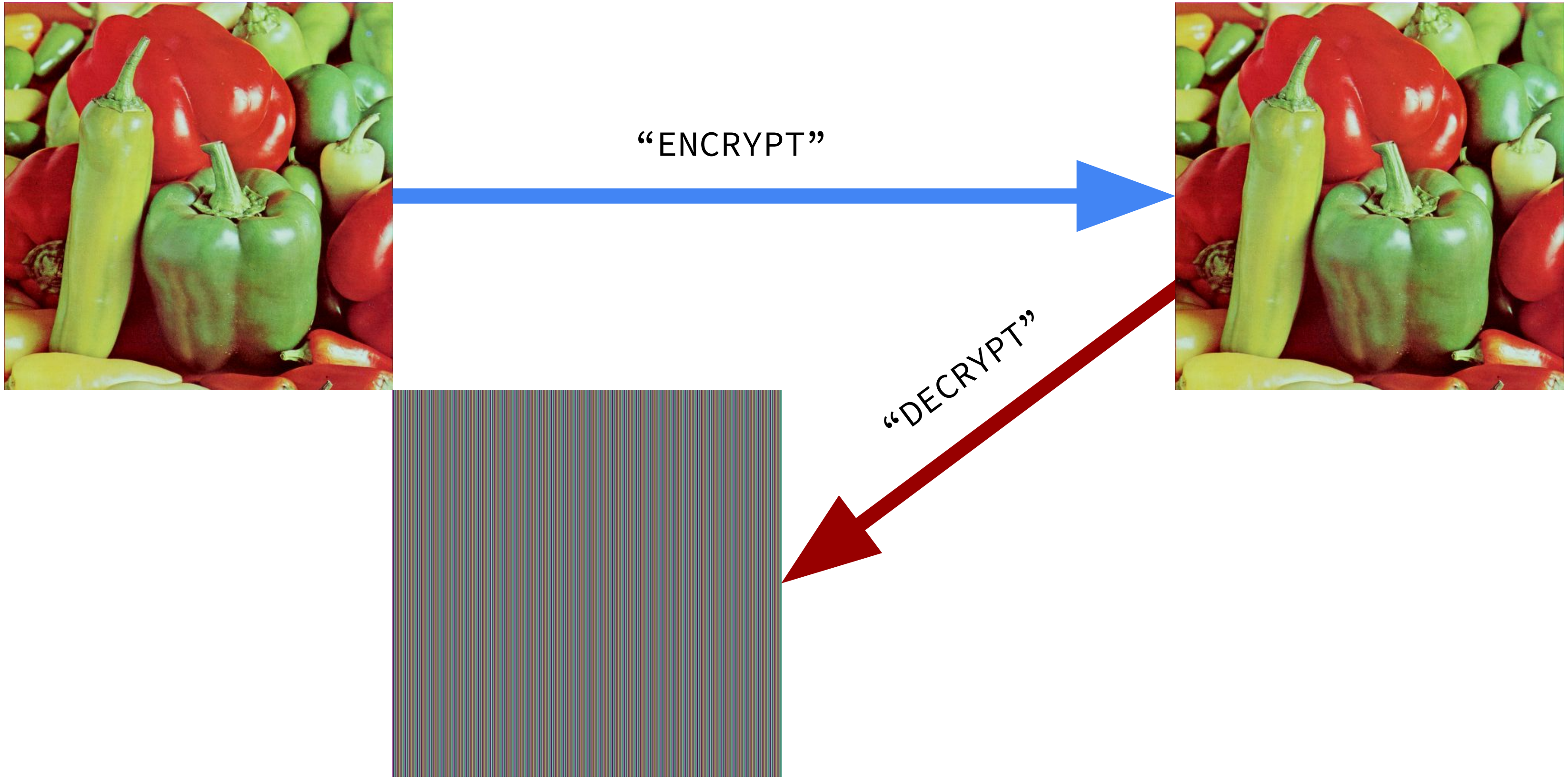


Shared Memory Functionality

```
int t = threadIdx.x;
...
// shared memory array for state data
__shared__ unsigned char s_state[THREAD_NUM * BLOCK_DIM];


// copying encrypted image array data to shared memory
for (int k = t * BLOCK_DIM; k < (t + 1) * BLOCK_DIM; k++) {
    int n_index = k + b * THREAD_NUM * BLOCK_DIM;
    if (n_index < size)
        s_state[k] = image[n_index];
}
```

Interesting Functionality (Indexing Issues)



DEVICE Architecture Issues

This is some example output from my home machine (GTX 1050 TI):

```
craine@telluric-ii:~/aes-image-encryption/device$ make dude
=== DEVICE Encryption/Decryption Results ===
Size of Input Image: 1474904 bytes
Dimensions of Image in Pixels (x,y): (701,701)

Encryption Time: 0.000000 sec
Encryption Throughput: 93909623621673227394813427726747172864.00 MB/s

Decryption Time: 0.000000 sec
Decryption Throughput: 1052526665481395521074745475018596470057074688.00 MB/s
```

DEVICE Architecture Issues

https://developer.nvidia.com/cuda-gpus#compute

NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1
GeForce GTX 1070 Ti	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 1050	6.1
GeForce GTX TITAN X	5.2
GeForce GTX TITAN Z	3.5

1050 TI

Highlight All Match Case Match Diacritics Whole Words Phrase not found

Notes on Kernel Comparisons

- Serial version is certainly not optimized (lacking MPI, OpenMP, etc.)
- ~~Is it really safe to compare encryption and decryption timings if the parallel implementation is (mostly) unsuccessful?~~ Fixed around 3:00 a.m.
- The plan is to plot comparison time and throughput data.
  - When considering the current implementations, the shared memory kernel is dominating.
- It is clear that even with a pathetic kernel, the GPU can encrypt and decrypt images faster and at a higher throughput when compared to purely serialized CPU code.

HOST v. DEVICE

```
cmda15@node03:~/aes-image-encryption/...
/host$ make peppers
=== HOST Encryption/Decryption Results ===
Size of Input Image:       786432 bytes
Dimensions of Image in Pixels (x,y):
                          (512,512)
Image Encryption Time:     0.265168 sec
Encryption Throughput:     2.97 MB/s
Image Decryption Time:     0.246127 sec
Decryption Throughput:     3.20 MB/s
```

```
cmda15@node03:~/aes-image-encryption/...
/device$ make peppers
=== DEVICE Encryption/Decryption Results ===
Size of Input Image:       786432 bytes
Dimensions of Image in Pixels (x,y):
                          (512,512)
Image Encryption Time:     0.000150 sec
Encryption Throughput:     5245.68 MB/s
Image Decryption Time:     0.000194 sec
Decryption Throughput:     4049.43 MB/s
```

Live Demo

- Source code repository can be found at the following link:

  https://code.vt.edu/craine/aes-image-encryption

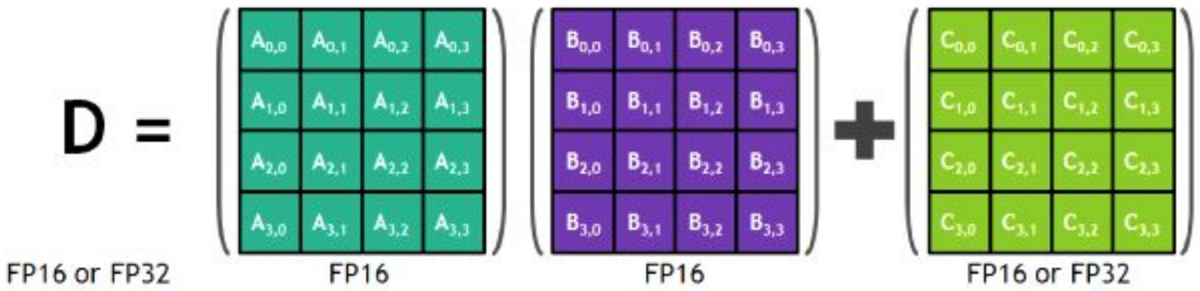
- Inside both `host/` and `device/` are makefiles.
  - Default call will compile the source:

    `$ make`
  - Check the makefile for different tests on images:

    `$ make {altitude, baboon, dude, scorn, etc.}`

Potential Code Improvements

- Utilize CUDA Tensor Cores
- Parallelize `bitmap.c` and possibly round functions
- Store S-BOX and KEY arrays in shared memory
- Implement a batch image kernel



Future Risks and AES

- Quantum computing poses risks to some *asymmetric* encryption algorithms; recall that AES is symmetric.
- Provided one uses sufficiently large key sizes, the symmetric key cryptographic systems like AES are claimed to be *resistant* to attack by a qubit system.
  - This is because AES is not affected by *Shor's algorithm*
  - However, *Grover's algorithm* suggests that quantum computing will weaken the AES-128 cryptographic system . . .

References

Daniel, T.R., Stratulat, M.: “AES on GPU using CUDA.” In: 2010 *European Conference for the Applied Mathematics & Informatics*. World Scientific and Engineering Academy and Society Press (2010).

Manoharan, Palanivel. “Advanced Encryption Standard (AES).” MATH 4175. Virginia Tech (2021).

Saxena, Aryan & Agrawal, Vatsal & Chakrabarty, Rajdeepa & Singh, Shubhjeet & Banu, J. “Accelerating Image Encryption with AES Using GPU: A Quantitative Analysis”. *Intelligent Systems Design and Applications* (2020).