Multi-Threaded Basic File Encryptor and Decryptor Program

•••

Steven Stefanovski and Wendy Fogland
Electrical and Computer Engineering Department

ECE 5772
Fall 2023

Application Overview

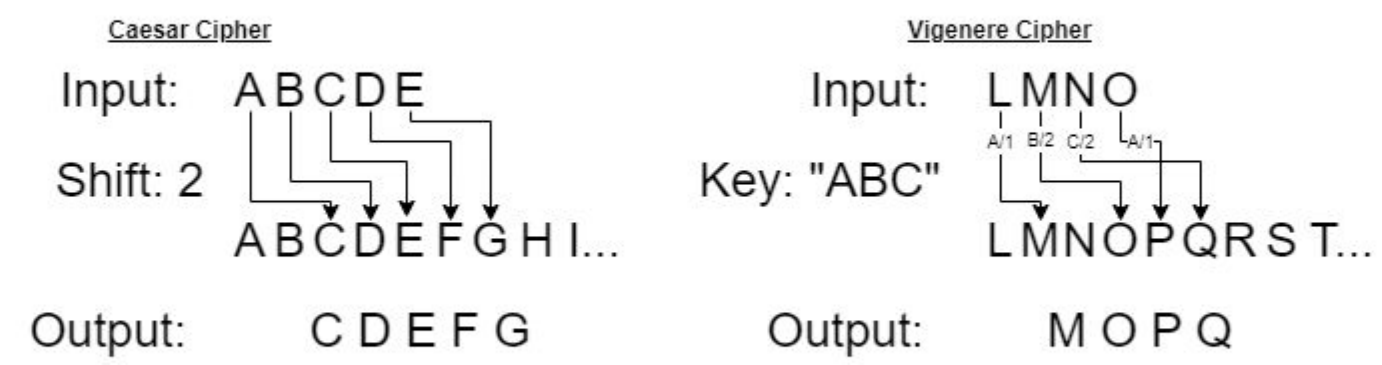
Main Goal: Implement a file encryptor/decryptor program utilizing different execution methods.

Sub-Goal:

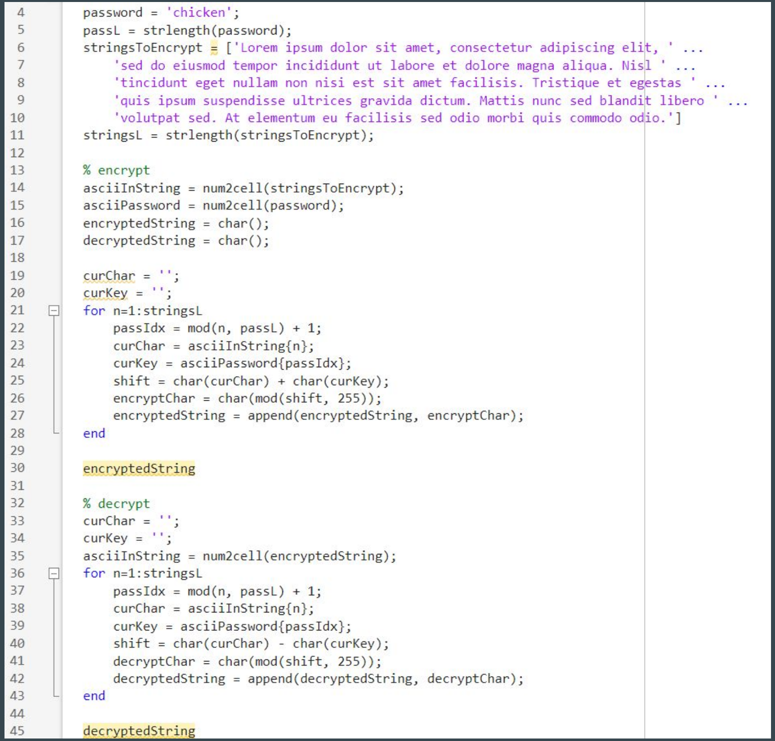
- Prove that a higher performance parallel method is possible compared to sequential execution

Encryption/Cipher Method

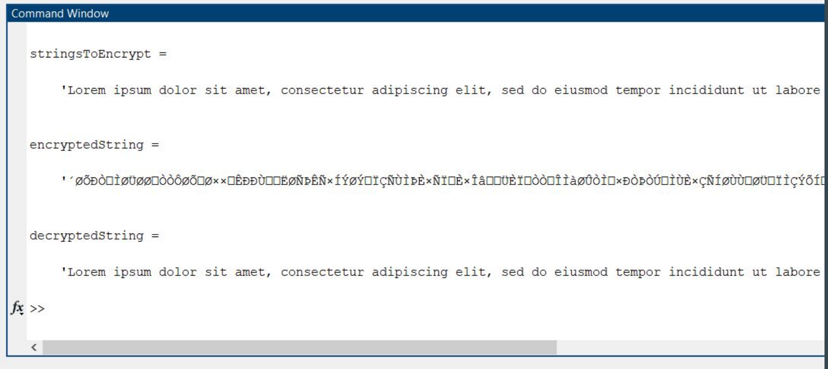
- Vigenère Cipher and Caesar Cipher used to encrypt/decrypt the provided text
- Caesar Cipher “shifts” each letter by a given number 1-26
  - A Caesar Cipher with a shift of 2 would turn the text “ABC” into “CDE”
- Vigenère Cipher uses a key to Caesar Cipher each character by a different shift
  - The letters of the key determine how much each character in the input text will shift
  - The letters of the key map A-Z to 1-26
  - The first letter of the input text is shifted by the mapped value of the first letter of the key
    - Second letter of the input is shifted by the value of the 2nd letter of the key, and so on
  - Start from the beginning of the key, if you run out of characters of the key but still have characters or the input text



Proof of Concept with MATLAB



```matlab
password = 'chicken';
passL = strlength(password);
stringsToEncrypt = ['Lorem ipsum dolor sit amet, consectetur adipiscing elit, ' ...
    'sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nisl ' ...
    'tincidunt eget nullam non nisi est sit amet facilisis. Tristique et egestas ' ...
    'quis ipsum suspendisse ultrices gravida dictum. Mattis nunc sed blandit libero ' ...
    'volutpat sed. At elementum eu facilisis sed odio morbi quis commodo odio.']
stringsL = strlength(stringsToEncrypt);

% encrypt
asciiInString = num2cell(stringsToEncrypt);
asciiPassword = num2cell(password);
encryptedString = char();
decryptedString = char();

curChar = '';
curKey = '';
for n=1:stringsL
    passIdx = mod(n, passL) + 1;
    curChar = asciiInString{n};
    curKey = asciiPassword{passIdx};
    shift = char(curChar) + char(curKey);
    encryptChar = char(mod(shift, 255));
    encryptedString = append(encryptedString, encryptChar);
end

encryptedString

% decrypt
curChar = '';
curKey = '';
asciiInString = num2cell(encryptedString);
for n=1:stringsL
    passIdx = mod(n, passL) + 1;
    curChar = asciiInString{n};
    curKey = asciiPassword{passIdx};
    shift = char(curChar) - char(curKey);
    decryptChar = char(mod(shift, 255));
    decryptedString = append(decryptedString, decryptChar);
end

decryptedString
```



```
stringsToEncrypt =

    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore


encryptedString =

    '´ØÕÐÒ□ÌØÜØØ□ÒÒÔØÕ□Ø××□ÊÐÐÙ□□ËØÑÞÊÑ×ÍÝØÝ□ÏÇÑÙÌÞÈ×ÑÏ□È×Îâ□□ÜÈÏ□ÒÒ□ÎÌàØÛÒÌ□×ÐÒÞÒÚ□ÌÙÈ×ÇÑÍØÙÙ□ØÜ□ÏÌÇÝÕÍ□


decryptedString =

    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

fx >>
```

Parallelization Methods

Pipeline Method

- Expect data to come as 1D-Char Array
- Stage 1
  - Create partitions of whole data array to break up (pipeline) chunks of data
- Stage 2
  - Encrypt/Decrypt partitions passed through from Stage 1
- Stage 3
  - Concat all partitions back into one char array of the same size as original data char array

Parallel_For Method

- TBB library does most of the heavy lifting
- Optimizes the amount of threads to use based on amount of data
- Works similar to a regular for loop

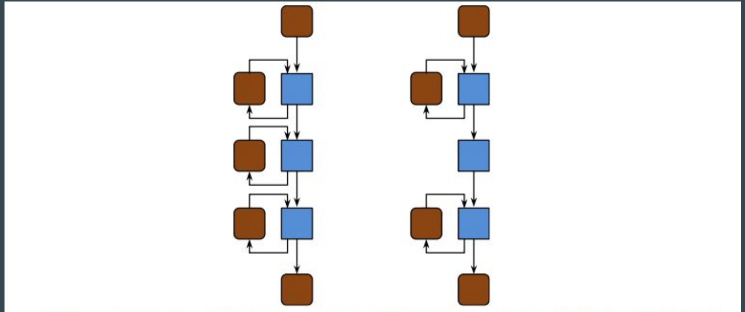


Figure 15. (a) Serial pipeline. Each stage can maintain its state so that later outputs can depend on earlier ones. (b) Parallel pipeline. The parallel stage is stateless; thus, multiple invocations of it can run in parallel.

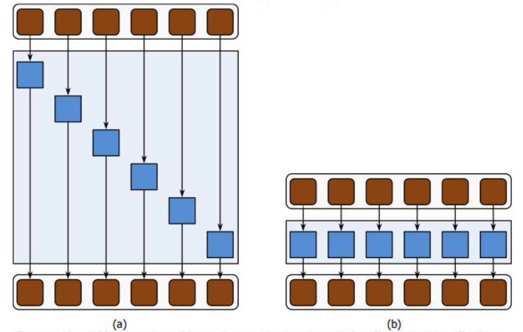
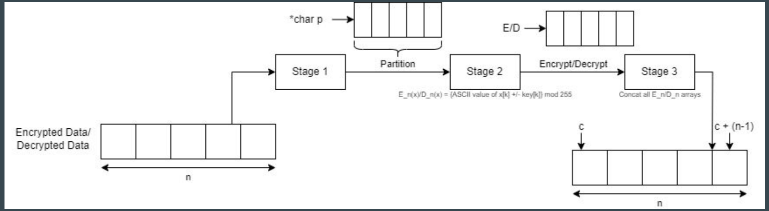
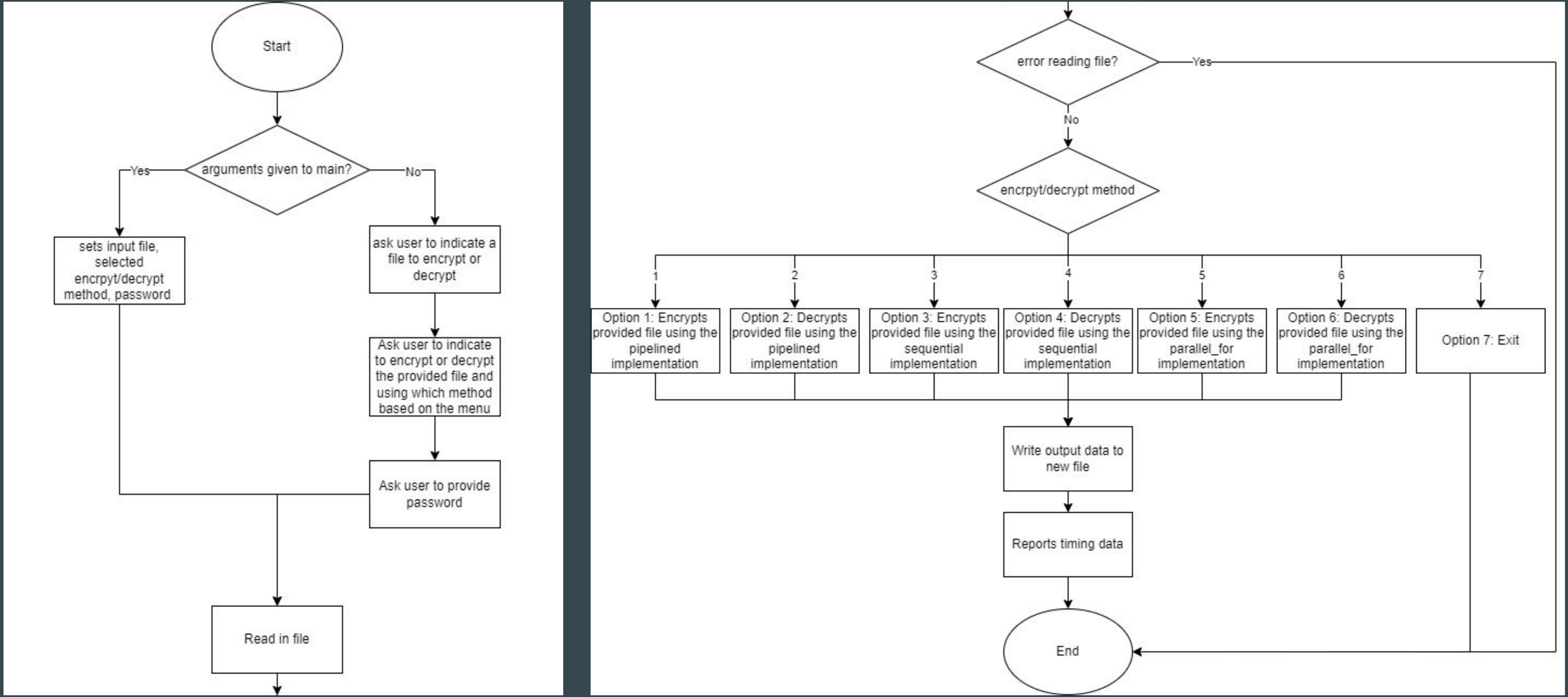


Figure 1. (a) serial loop execution. (b) map pattern: a function is applied to all elements of a collection. Rounded rectangles represent data. Rectangles represent tasks.



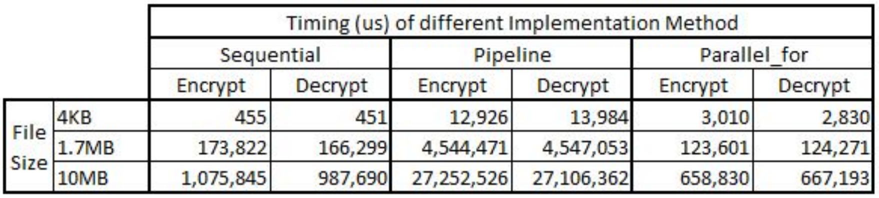
Flowchart of Software



Start
arguments given to main?
Yes
No
sets input file, selected encrpyt/decrypt method, password
ask user to indicate a file to encrypt or decrypt
Ask user to indicate to encrypt or decrypt the provided file and using which method based on the menu
Ask user to provide password
Read in file
error reading file?
Yes
No
encrpyt/decrypt method
1
2
3
4
5
6
7
Option 1: Encrypts provided file using the pipelined implementation
Option 2: Decrypts provided file using the pipelined implementation
Option 3: Encrypts provided file using the sequential implementation
Option 4: Decrypts provided file using the sequential implementation
Option 5: Encrypts provided file using the parallel_for implementation
Option 6: Decrypts provided file using the parallel_for implementation
Option 7: Exit
Write output data to new file
Reports timing data
End

Program Execution Example

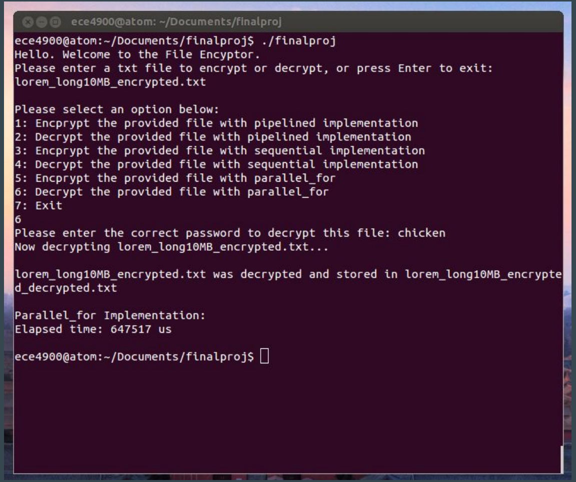
Results & Conclusion



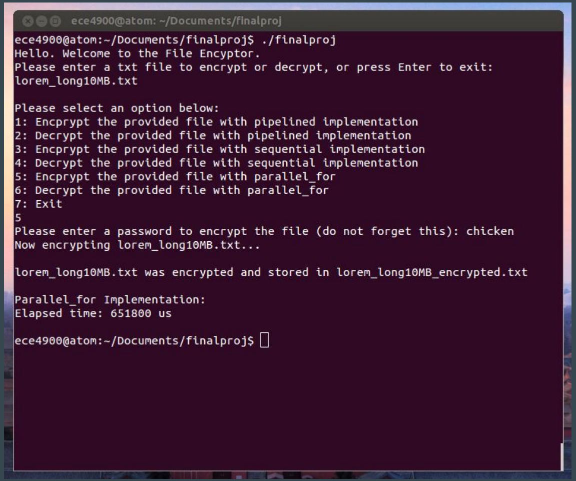
		Timing (us) of different Implementation Method					
		Sequential		Pipeline		Parallel_for	
		Encrypt	Decrypt	Encrypt	Decrypt	Encrypt	Decrypt
File Size	4KB	455	451	12,926	13,984	3,010	2,830
	1.7MB	173,822	166,299	4,544,471	4,547,053	123,601	124,271
	10MB	1,075,845	987,690	27,252,526	27,106,362	658,830	667,193

- Parallel_for performed best overall
  - Works great in applications like this, where there is the same, but independent, operation is performed on each element of the input
- Sequential implementation is still best for small datasets
- Pipeline implementation worst performance
  - Could be due to input data formatting
- Timings averaged over 5 executions

In conclusion, two different parallelization categories were implemented, but only one improved performance from the sequential implementation. Parallel_for was well suited for this application and greatly improved performance with larger file sizes, but pipelining seemed to struggle.



```
ece4900@atom:~/Documents/finalproj$ ./finalproj
Hello. Welcome to the File Encyptor.
Please enter a txt file to encrypt or decrypt, or press Enter to exit:
lorem_long10MB_encrypted.txt

Please select an option below:
1: Encprypt the provided file with pipelined implementation
2: Decrypt the provided file with pipelined implementation
3: Encprypt the provided file with sequential implementation
4: Decrypt the provided file with sequential implementation
5: Encprypt the provided file with parallel_for
6: Decrypt the provided file with parallel_for
7: Exit
6
Please enter the correct password to decrypt this file: chicken
Now decrypting lorem_long10MB_encrypted.txt...

lorem_long10MB_encrypted.txt was decrypted and stored in lorem_long10MB_encrypte
d_decrypted.txt

Parallel_for Implementation:
Elapsed time: 647517 us

ece4900@atom:~/Documents/finalproj$ 
```



```
ece4900@atom:~/Documents/finalproj$ ./finalproj
Hello. Welcome to the File Encyptor.
Please enter a txt file to encrypt or decrypt, or press Enter to exit:
lorem_long10MB.txt

Please select an option below:
1: Encprypt the provided file with pipelined implementation
2: Decrypt the provided file with pipelined implementation
3: Encprypt the provided file with sequential implementation
4: Decrypt the provided file with sequential implementation
5: Encprypt the provided file with parallel_for
6: Decrypt the provided file with parallel_for
7: Exit
5
Please enter a password to encrypt the file (do not forget this): chicken
Now encrypting lorem_long10MB.txt...

lorem_long10MB.txt was encrypted and stored in lorem_long10MB_encrypted.txt

Parallel_for Implementation:
Elapsed time: 651800 us

ece4900@atom:~/Documents/finalproj$ 
```

References

[1] https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

[2] https://www.secs.oakland.edu/~llamocca/emb_intel.html

[3] https://moodle.oakland.edu/pluginfile.php/8893205/mod_resource/content/1/Notes%20-%20Unit%204.pdf