The decryption rsa algorithm information technology essay

Technology, Information Technology

Introduction

1. 1 Introduction

The title of the project is Implement Encryption algorithms in SAC (Single Assignment C). To evaluate Single Assignment C by implementing encryption algorithms Implement the same encryption algorithm in C and evaluate encryption algorithms. As well as do the performance of both algorithms. Using the two measurement technique first is LOC (Line Of Code) and second is Time Complexity. To understand the RSA algorithm we need to understand first what is the encryption? Cryptography is the study of techniques for communication in the secure manner in the presence of the third parties. Also in modern era it is a process of converting ordinary information into unintelligible gibberish and decryption is the reverse, in other words, moving from the unintelligible cipher text to plaintext. A cipher is a couple of algorithms that create the encryption and the reversing into decryption. The operation of a cipher is controlled by the algorithm and by the key. In the first instance we will describe few terms: The original message is called the plaintext and the coded message is known as a cipher text. The process of converting a plaintext to cipher text is called encryption while restoring the plain text from the cipher text is called decryption. Algorithm is a process which generates output form input language by steps of process. Secret Key: It is a step in process of encryption algorithm and on that base encryption algorithm performs a specific operation on plain text and generates a cipher text. The cipher text can be different on the bases of the secret key. The process of converting plain text into cipher text by the use of algorithms is

called encryption algorithms. An algorithm which accepts cipher text and secret key and generate an original plain text by processing few steps of procedure, called a Decryption algorithm. SAC (Single Assignment C) is a functional array of processing language which is mainly used for the numerical applications i. e. Sudoku solver, Cryptography, Image processing, etc. The simple form of cryptography is Symmetric. The Symmetric encryption is a conventional encryption in which the encryption and decryption are performed by using the same key or the password. These algorithms are very simple and easy to operate but if third party intercepts the key then they will be able to decrypt the messages. The researchers have to develop so many different kinds of encryption algorithms because of the trustworthy e-commerce and computer file security issues. The other type is Asymmetric (public key) cryptography which as name suggest for the use of more than two persons to decrypt the message. Encryption can protect the data files on computers and storage devices. The use of Cryptography was developed because of the secure communication. In the figure below the basic cryptography communication scenario has been described. The person X and person Y how will communicate in a secure way by the using secure communication. The person Z wants to read the message of person X and Y. Figure 1. 1 secure communicationThe series of steps involved in a secure communication between X and Y. Choose the secret key for the encryption and decryptionInput the plain text and secret key into algorithm which is encryption algorithm and generate a cipher textPerson X will send the cipher text to person YPut in a secret key and plain text into a decryption algorithm which will generate the plain textNow

in the case of person Z wants to read the message, He will only see the cipher text which is generated by the encryption algorithm. If person Z wants to unlock or read the cipher text, he has to have a secret key to unlock or decrypt the cipher text into plain text. But the person Z doesn't have the key so he cannot decrypt the message. So this communication is secure between the person X and Y. Encryption Algorithms define the rules of the encryption of the plain text and decryption of the cipher text. The cipher text of algorithms depends on the plain text and key. Now C is cipher text, K is key, P is original plain text, E is set of encryption algorithm, we can say thatC = E (K, P)There are three main types of the encryption algorithms: Substitution cipher, stream cipher and block cipherThe substitution cipher is an encryption method by which the units of plain text will be replaced by the cipher text. Deterministic algorithm is operating on groups of bits which are called Blocks or Block Cipher that have Symmetric key (similar transformation). In stream cipher, plain text digits are used in combination with a cipher digit stream. In stream cipher which is also known as a state cipher, each plain text digit is encrypted with one at a time with the digit of the key stream. The difference between this two is, Block ciphers are operated on the large blocks of digits with a fixed and same transformation. There are some classical types of cipher also exists such as caser cipher and Playfair cipher. Caser Cipher: Caser cipher is invented by Julius Caesar which is substitution cipher and in this encryption algorithm letter is replaced by another letter with the help of fixed number. Suppose we have fixed number 7, we can replace alphabets such asPlain Text : A B CDEFGHIJK……ZCipher Text: G H I J K L M N O P Q……. FPlayfair Cipher: Algorithm was first used by

the British army in World War I and later on it was also used by the US Army and other armed forces in World War II. Algorithm is based on 5 x 5 matrix or box. This matrix is filled with the letters according to the key or key word. We have key " UNITED" Following the matrix which is generated according to the key word. UNITEDABCFGHJK/LMOPQRSVWXYZTable: Playfair cipher matrixHere, the keyword is UNITED. The matrix is created by filling in the letter of keyword from left hand side to right hand side and from up to down and then filling in the remainder of the matrix with the remain letters in alphabetical order. The count letters K and L as one letter. Plaintext can be encrypted two letters as same time. The above matrix table and by knowing some rules we can cipher the plain text. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as X, so that the balloon would be treated as ba 1X 1o on. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example , ar is encrypted as RMTwo plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example , mu is encrypted as CM. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM. Hill Cipher: Hill cipher was invented in 1929 by Lester Hill and it is based on the linear algebra. In these cipher method we have the key M. We can compute encryption and decryption of any plain text. The key M is in (3 X 3) matrix, C is cipher text and P is plain text, C = PM mod 26

=

P = CK-1 mod 26= -1mod 26Rail Fence: Rail fence cipher is a very simple technique of a cipher. This technique through easy to crack cipher. In this technique is plain text message writing in a rectangle formats. For example we have one sentence " I will come to the university tomorrow" , now we will write in a special way " rails" suppose we have taken key here is 3. i . . . l . . . e . . . h . . . i . . . s . . . t . . . r . . .. w . l . c . m . t . t . e . n . v . r . i . y . o . o . r . w. . i . . . o . . . o . . . u . . . e . . . t . . . m . . . o . Table 1. 2 Rail FenceThe table 1. 2 rail fence is read the above message row by row. Here , given the key is 3. So, now the ciphertext message is :" ilehistrwlcmttenvriyoorwioouetmo". DES (Data Encryption Standard) and RSA are both types of block cipher algorithms. DES is developed in early 1970s at IBM and is based in the earlier design by Horst feistel. It is highly influenced by the advancement of modern cryptography in academic field. RSA was invented by Ron Rivest, Adi Shamir and Leonard Adleman in 1977 at MIT and it is named on the base of their surnames first letters. We can more discuss about RSA algorithm next chapter.

1. 2 Structure Of Report

If you are doing any project than this section is very important for the reader. Because of this section can give you an overview of this report structure. Section 1 is an introduction to project. In this section I have discussed about basic of this project like encryption algorithm definition and SAC definition. As well as describing cryptography and shortly describe encryption algorithm . In section 2 I have discussed about the RSA algorithm.

This section is introduction of RSA, description of RSA. Basic step to generate the RSA algorithm like key generation, Encryption and decryption. Also I have given a simple example and as well as the security of RSA algorithm. Section 3 is SAC (Single Assignment C). This section I have discussed about SAC. Installation of SAC, How to run SAC file and Basic operation of SAC. Lastly, I have explained similarity and difference of SAC and C through one example program. The next section is Measurement and Performance Testing. In this section I have discussed about measure the size of RSA algorithm with LOC (Line Of Code) technique. Also I have did measurement using Time complexity and as well as did the graph of measurement complexity. After that I have done discussion section. In this section I have written description 512-bit RSA in SAC and 64-bit RSA in C language. The next section is the conclusion in this section I have written project conclusion. After that I have done bibliography in this section write project references. The last section is appendix in this section code 512-bit RSA algorithm in SAC and 64-bit of the RSA algorithm in C.

3. RSA Algorithm

3. 1 Introduction:

The RSA algorithm is the most popular cryptography algorithm. RSA algorithm is developed in 1977 by a this three man named called Ron Rivest, Adi Shamir and Len Adlemen. It is published for the first time in 1978. This algorithm for public-key cryptography. The RSA name comes from the name of this three man Rivest-Shamir-Adeleman (RSA). This algorithm most commonly recognized and implemented a general-purpose methodology to

public-key encryption (Stallings, 2011). The RSA algorithm is very easy to understand. But calculation was very difficult and also time taking. In RSA algorithm public key and private key are based on large prime numbers. This algorithm was depended on mathematical fact.

3. 2 Description of RSA algorithm:

The RSA algorithm is generally used in developing e-commerce and e-business system for the making digital signature and digital envelope (Doroeviae et al., 2002). Encryption is the process of converting plain text into cipher text. Assume a person A want to establish secure communication with person B using RSA algorithm. Person A and person B decide the value of n. Person A see the value of e and person B see the value of d. RSA algorithm Encryption and decryption are working on following equation. C = Me mod N (encryption algorithm)M = Cd mod N (decryption algorithm)Now, see the above equation C it is called cipher text, and M it is called original Plain text. This equation e and n are mentioned public key of PU={e, n}, and d and n are mentioned private key of PB={d, n} (Stallings, 2011).

3. 2. 1 Different step for Key generation of RSA algorithm

Here, summarizes the different step involve for RSA algorithm in key generation. Step 1: Randomly select any of two large prime number p and q. Step 2: Calculate value of n= p*q. Step 3: Calculate the function φ (n)= (p-1)*(q-1). Step 4: select any value of e, here value of e must be between 1 to φ(n). such that GCD(e, φ(n))= 1. Step 5: compute the value of d such that d*e = mod φ (n) .(Sahu, 2011). Below step are the encryption and decryption of RSA algorithm.

3. 2. 2 Encryption RSA algorithm

Step 1: Assume the person A want to send message to person B securely.

Step 2: Now, e has person B's public key. Here e is public so A can access to e. Step 4: Here, equation cipher text C= Me mod n, here n is the modulus.

3. 2. 3 Decryption RSA algorithm

Step 1: Now, C has received cipher text from A. Step 2: Compute message M= Ce mod n, here d is persion B of the private key and n is modulus (Dhakar et al., 2012).

3. 3 Example of RSA algorithm:

First of all Choose 2 prime numbers. p = 557, q = 821. Calculate the modulus for the public/private key ' n'. Here, n is the number that is the same value in both keys, n =(p*q)= (7*17) = 457297Than Calculate the O of n. O(n) = (p - 1) * (q - 1). O(119) = (7 - 1) * (17 - 1). This equals 455920. Now, we have got the these values: p = 557q = 821n = 457297O(n) = 455920In this step Choose a value for eThis number is a bit harder than the others. This number has to be between 1 and n, it can also coprime to n. This basically means that the greatest common divisor of both numbers is 1. If you choose a prime number for e, so what you need to do now is check that e isn't a divisor of n. Here, I have choose the number: e = 17We will Calculating the modular multiplicative inverse of e * (mod O(n))((e * x - 1) mod (O(n)) = 0) = 160913Now, we can get all the value those we required in to implement algorithm Putting it all togetherp = 557q = 821n = 457297O(n) = 455920e = 17d = 160913we are got the public-key is now: e = 17, n = 457297Here, Private-key is now: d = 160913, n = 457297Using

this private and public-key data can be encrypted: Here, encrypt the value 107m = 107Now, we will encrypt with the public key, take m to the power of e (in the public-key) mod nm ^ e mod n107 ^ 17 mod 457297 = 323453Here, Our encrypted value is c = 323453It can only be decrypted with the private-key. To decrypt it, you can take c to the power of d (in the private-key) mod nc ^ d mod n323453 ^ 160913 mod 457297 = 107 which is the decrypted value.

3. 4 Security of the RSA algorithm

However, security is very important for the RSA algorithm. In RSA algorithm security and message encryption by the algorithm depends on the difficulty of factoring the value of n. If the value of n can be easily factored into the consistent value of p and q, At that time you can find valued easily. Suppose, Person A, B and C. Person A had sent messages to person B using public key encryption procedure EA. If person C want to see that message. They can be able to see that message in the ciphertext C= EA (M) format without the decryption key d. Person C can’t see the original message. The RSA algorithm security is described by following possible approaches to attacking the RSA algorithm (Kelly, 2009). There are three approaches to attacking RSA algorithm.

Mathematical attacks:

Mathematical attacks are based on factoring the product of large primes. Such as factor n into its prime factors p and q respectively then computing φ, which has opportunity, enables determination of d = e-1 mod φ (Sharma et al., 2011). Here, currently identified algorithms determining d given e and n

look like to be at least as time consuming as the factoring problem. Now, we can use factoring performance as a benchmark compared to calculating the security of RSA (Stallings, 2011).

Timing attacks:

Implement cryptographic algorithm it is very important for execution on a non-constant time because of performance optimization. Timing attacks is if any one required another message it is very difficult for security of cryptographic algorithm. A timing attack idea is firstly purposed a cryptographic consultant Paul Kocher. They decide private key through track how long a computer task deciphers messages. Timing attacks are not a relevant only for the RSA but they other public key cryptographic system. This attack is only for two reasons, firstly it comes from an unexpected direction and secondly attacks only ciphertext (Stallings, 2011).

Chosen ciphertext attacks (CCA):

The RSA algorithm is at risk for a chosen ciphertext attack. Chosen ciphertext attack is a basically based on theoretical assumptions. Here, A public key cryptosystem is at risk for a Chosen Ciphertext attack, so frequently is measured to only a theoretical weakness. This attack needs more decryptions with every candidate key to identify the expected clear text data. The public key cryptosystem it does know the victim's public key, because an attacker can create own required clear text / cipher text pairs. The main problem an applying this method to the RSA algorithm is that all modular exponentiations is very costly, and its time complexity raises cubically with the size N of the modules (Salah et al., 2006).

4. Single Assignment C (SAC)

Single Assignment C (SAC) is a functional array processing language. SAC is supporting n-dimensional arrays as major data structure. A fact of SAC constitutes a no-frills functional variant of C. So, It has some benefits over commanding programming languages, like referential transparency. The all functional programming languages appear to be mostly apt for these demands. The absence of side special effects agrees complex code optimizations to be applied more simply since all data need to explicit. They allow the functional paradigm for an upper level of abstraction which commonly improves the overview of programs and so make simpler reuse and maintenance of existing programs. However, The programming languages dedicated to array processing powerful. A specifying array operation in an abstract manner has been developed. Starting out from a mathematical notation for arrays. Below basic design principles of SAC. The first principle is call by value semantics. In a SAC language is supported directly n-dimensional arrays. Their support for shape-invariant array comprehensions. SAC memory management is based on reference counting. They have the aggressive high-level optimization to complete competitive runtimes (Grelck and Scholz, 2006). SAC is designed to keep it closer to C language and on the other hand base it on the principle of context free substitutions. C Language attracts application programmers while SAC is designed for extensive compile time optimization as well as non-sequential execution. It uses reference counting techniques to overcome runtime and space efficiency whenever any single element of an array is modified, in other programming language any change in the element of an array needs

to modify the entire array to be copied. This technique also helps programmers from the burden to think about memory issues entirely, as no memory needs to be allocated, arrays passed as arguments to functions can never be affected by such calls, and the programmer does not need to copy arrays in order to preserve a certain state. The programmer defines various arrays and relies on the compiler and the runtime system to ensure that the memory is being allocated and reused as often as possible (Scholz, 2003). SAC supports shape invariant programming on the Meta level, it enables the programmer to design asset of shape-invariant operators to the particular requirements of a given set of applications. These operations can be stored within a library, which can be shared for the entire sets of application. An array on SAC represented by a two vectors, First one is a data vector in this vector all the elements in linear order and another one is shape vector they defines structural properties of the array (Scholz, 2003).

4. 1 SAC Installation

If you want to do any program in any language it is necessary to install a language on your system. Below I am going to write about SAC available on following operating system. Solaris (Sparc, x86)Linux (x86 32bit, x86 64bit)DEC (alpha), Mac OS X (ppc. X86), netBSD (under preparation). SAC can install above five operating systems. SAC is currently not available on windows platform.

How to Compile and Run SAC program

To compile the SAC program, First of all we need save the file on right place, To save the file we will need to open a terminal windows and login as super

user. After that open the right directory and then type sac2c . sac . Now, Type command called ls we can see the three different file on our folder like a. out, a. out. c and . sac . Last of all we can type command called a. out this file contains the results of our program (Scholz et al., 2010).

4. 2 Basic Operation of SAC

In this section we are going to see about basic operation available on Single Assignment C (SAC).

genarray:

genarray function is used for a define a new array and they set a default value for a each element. Syntax: Example: genarray ([1, 2, 3] , 2)Now, we can see above example show the 3 dimensional array and set value of each element for a 2.

reshape:

The reshape function can be used for a reshape the parameter from an array. Syntax: reshape ([dimension], [array])Example: reshape ([2, 2],[1, 2, 3, 4])Here, we can see in the above example that is a 2 dimensional array and inner array can identical shape.

dim:

dim function shows the deimensity of the argument array.

shape:

Shape function is shown the vector argument an array.

rotate:

This function is used to rotate the array in leftmost axes and rightmost axes.

take:

They take certain element from an array.

drop:

They drop certain element from an array.

cat:

Cat returns the concatenation of two arrays.

4. 3 SAC and C Comparison

In this section we are comparing SAC and C through one example programme. A program to find a minimum and maximum number from an array.

Find a minimum and maximum number in SAC

use StdIO: all; use Array: all;//A program to find maximum and minimum numbers from an array. int main()

{

a=[1, 2, 0, 0, 4, 5, 6, 9, 9, 17];// arrayprint(a);// print arrayprintf(" Max is : ");

print(maxval(a));// print maximum numberprintf(" Min is : ");

print(minval(a));// print minimum numberreturn(0);

}

Here, Below figure 4. 1 shows the output from above example in SAC. Figure 4. 1 Output of SAC ExampleTo implement a simple program in any language, it is necessary to first learn basic of that language and program. SAC is an array processing language. So I have implemented a simple array example program, a program to find a minimum and maximum number from an array. In the simple program example. To implement the program, I have use following function. Firstly, I have defined an array from line 8. After that in line 10 prints an array. Line 13 finds maximum number of values in array and line 16 find the minimum number of values in array. Now compile the program in SAC. Go to the terminal emulator and go to the same directory where the SAC program is saved. Then type the command: Here, the program name is the file name in that I have written in SAC. After the completion of compiling programs generate a. out file. This file contains the output. So, type the command in a terminal window.. /a. OutNow figure 4. 1 generates the output of the program. Here we can see that the dimension shows the array dimension i. e. 1D, 2D, 3D. Second is shape, show the number of elements in an array. Last we can see the result that shows the maximum number of array and minimum number of arrays.

Find a minimum and maximum number in C

// a program to find minumum and maximum from an arrayvoid printarray(int a[], int length)

{

// A function to print array. int i; printf(""); for(i= 0; i } int findmax(int a[], int length)

{

// A function to find maximum from an arrayint i, res; res= a[0]; for(i= 1; i

{ res= a[i];

}

return(res);

}

int findmin(int a[], int length)

{

// A function to find minimum from an arrayint i, res; res= a[0]; for(i= 1; i

{ res= a[i];

}

return(res);

}

void main()

{

int a[10]={1, 2, 0, 0, 4, 5, 6, 9, 9, 17};// Define Arrayprintarray(a, 10);// Print Arrayprintf(" Max Value is : %d ", findmax(a, 10)); // Print Maximum

Numberprintf(" Min Value is : %d", findmin(a, 10)); // Print Minimum Numberprintf("");

}

Here, Below figure 4. 2 shows the output from above example in C. Figure 4. 2 Output of C exampleIn comparing to SAC and C it is necessary to implement the same program in C. So I have implemented the same program in C. An example shows the find a maximum and minimum number from an array in C. In C language I have used four functions in coding. Firstly, I have used void printarray () this function can use print array. Secondly, void findmax () this function was written to find the maximum value from an array. After that void findmin () this function can be used to find the minimum number from an array. Then the last function void main (), In this function array is defined, print the array, print a maximum and minimum number from an array. After we need to compile this program, open terminal emulator and go to the directory then type a command: gcc . cHere, the file name is written the file name of the program. Same here gcc compiler generates the one file called a. out. These files contain the output of the program. So we need to run that file type command on terminal emulator.. /a. OutFigure 4. 2 shows, The output of above example . Here, See the first row shows number of elements available from an array. Next, generate Output and print maximum number from an array. Last they print the minimum number from an array.

4. 4 Difference and Similarity of SAC and C

In this section we are going to see what a similarity between SAC and C is and what the difference between SAC and C. SAC and C similarity we have seen above example. The programming syntax is same in a SAC and C language. As well as they generate same output from programs. In at both languages all the line ended with ; (semi-columns). If we can do the coding in SAC or C language they must require main function. Another similarity was if you want to comment the one line or more than one line we can use // or /*…….*/. In both languages use of function In the same like if …. else, while, for etc. that’s the similarity of SAC and C. Now, we are discussing about what is a difference between SAC and C. The first difference is we have seen in the example program number of line code only 13 lines in SAC and other side in C language 39 line of code. We are not considering blank line and commented line in both languages. Secondly, the SAC have inbuilt function called print () this function print the array dynamically. In C language we have created a user defined function. Another difference is SAC has memory allocation is dynamically to the array. Other side C has if we are using a pointer, so we have to allocate memory to the pointer.

5. Measurement and Performance Testing

Measurements are very important if you are producing software algorithm. Measurement through we can control, evaluate and improve the process of software. To define measurement According to Lord Kelvin (1924-1904) " when you can measure what you are speaking about and express it in number, you know something about it. Otherwise your knowledge is a

meagre and unsatisfactory kind; it may be the beginning of knowledge but you have scarcely in thought advanced to the stage of science" (Lim and Ahmed, 2000). To do measurement there are many measurement techniques available like LOC, Function point, Time complexity, etc. Here, We are used to measure RSA algorithm LOC and Time complexity techniques.

5. 1 LOC (Line Of Code)

LOC stands for a Line Of Code. The LOC is used to measure the size of software and algorithm. The LOC is another way to write KLOC. LOC is a very easy technique. We have here used LOC technique and measure the size of RSA algorithm. LOC can be measured in three ways, Firstly kLOC (k= 1000) means a 1000 line of code. kLOC can count the all the line including commented line of code. Second one is a NCLOC (Non Comment Line Of Code) can be counted only the non-commented line of code. Lastly, CLOC (Comment Line Of Code) is counted only comment line of code. We can take a simple example of a C program.//#include int main(int m, char *argv[])

{

printf(" how are you."); return(0);

}

Now we can measurement of above example with LOC technique LOC= 7 , NCLOC= 6 and CLOC= 1. Below Table 5. 1 shows the measured size of implementing the RSA algorithm thought LOC technique.

CODE

LOC

NCLOC

CLOC

512-Bit RSA in SAC5505341664-Bit RSA in C15114407Table 5. 1 Measurement size of RSA algorithm SAC and CTable 5. 1 shows the measured size of the RSA algorithm in SAC and C. Here, column 1 is a code written in a SAC and C. Column 2 is represents the Line Of Code. Column 3 represents Non-Comment Line Of Code and last column represent the Comment Line Of Code. Normally this measurement technique can be used on a big project.

5. 2 Time complexity

Time complexity is a very important measurement for a this algorithm. Complexity can be expressed use of big-O notation. The time complexity is generally estimated by counting the number of fundamental operations performed by the algorithm. Here, A fundamental operation takes a fixed amount of time to get output. A computer can perform different operations in deferment speeds. Some system have addition is very fast and other side division is very slow. The system can also be differ in size and processor, memory and disk speeds. The time complexity has most interesting property of an algorithm. We can measure the RSA algorithm using time command. We are going to measure this way the time complexity of execution of algorithm. For Measure time we can use of command time available on Linux. Write a command in command windowThe time command is given the

output is real time, user time and sys time(system time). They give you a how much CPU time used by your process and how much CPU time used by the system by behalf of you. This command is not use for a system time. This command is for a only a specific task how long they task. Below Table 5. 2 shows the performance of RSA algorithm key generation Time in second.

Key Size (Bit)

Key Generation Time (Sec)

320. 007640. 1181282. 00225622. 358512420. 455Table 5. 2 performance of RSA algorithm Key generation timeTable 5. 2 is a performance of RSA algorithm key generation. Here, first column shows the key size in a number of Bit and the second column shows a key generation time in a second. Now, we can describe all the given data using line graphs. Below figure 5. 1 shows the chart of an RSA algorithm key generation time. Figure 5. 1 Performance chart of RSA key generationAbove figure 5. 1 shows the key generation time of RSA algorithm. Here, X-Axis represents the data size in a number of bit and Y-Axis is a Time of key generation in a second. Here , we can see on a graph data size 32-bit to 128-bit key generation time has slightly increased. After that 128-bit to 256-bit is a increased almost 10 times by 128-bit. Lastly , data size is 256-bit to 512-bit time was suddenly rise time is 420. 255 second. Table 5. 3 shows the performance of RSA algorithm with 32-bit key size in SAC.

Data Size (Bit)

Encryption Time (Sec)

Decryption Time (Sec)

320. 0140. 067640. 0210. 1681280. 0420. 2572560. 0820. 5125120. 1551. 03310240. 2922. 057Table 5. 3 Performance testing of RSA 32-bit key sizeAbove table 5. 3 column 1 is a data size in bits, column 2 represents a Encryption time in a second and last column is a Decryption time in a second. Below figure 5. 2 chart shows the RSA 32-bit key size comparison of encryption and decryption time. Figure 5. 2 performance chart of RSA 32-bit key sizeAbove figure 5. 2, X-Axis represents data size in a number of bit and Y-Axis represents time in a second. First of all see the encryption time is starting to end increases gradually as we increase data size. But in a other side decryption time is increases sharply as we increase data size. Table 5. 4shows the performance of RSA algorithm with 64-bit key size in SAC.

Data Size (Bit)

Encryption Time (Sec)

Decryption Time(sec)

1280. 0250. 2322560. 0480. 4365120. 0951. 12510240. 1651. 8720480. 343. 64440960. 6797. 454Table 5. 4 Performance testing of RSA 64-bit key sizeAbove table 5. 4 column 1 is a data size in bits, column 2 represents a Encryption time in a second and last column is a Decryption time in a second. Below, figure 5. 3 is shows performance chart of RSA 64-bit key size. Figure 5. 3 performance chart of RSA 64-bit key sizeFigure 5. 3 shows a comparison chart of encryption and decryption time of RSA 64-bit key size.

Here, X-Axis represents the data size in a number of bit and Y-Axis represents time in a second. We can say that here encryption time is 128-bit to 4096-bit increase very slowly. Other side decryption time is increases regularly as we increases data size. Table 5. 5 shows the performance of RSA algorithm with 128-bit key size in SAC.

Data Size (Bit)

Encryption Time (Sec)

Decryption Time (Sec)

1280. 0881. 6112560. 133. 2985120. 2657. 58110240. 51813. 37820481. 03427. 45840962. 00552. 346Table 5. 5 Performance testing of RSA 128-bit key sizeTable 5. 5 is a performance of RSA algorithm 128-bit column 1 is represent the data size in number of bit column 2 represent encryption time in a second and column 3 represent decryption time in a second. Figure 5. 4 chart shows the performance of RSA algorithm of 128-bit key size. Figure 5. 4 performance chart of RSA 128-bit key sizeAbove figure 5. 4 chart show the performance of RSA algorithm with 128-bit key size. Here, X-Axis represents the data size and Y-Axis represent time. We can say that encryption time is increasing gradually and decryption time is increased step by step. Table 5. 6 shows the performance of RSA algorithm with 256-bit key size in SAC.

Data Size (Bit)

Encryption Time (Sec)

Decryption Time (Sec)

1280. 19310. 0352560. 37120. 2485120. 73240. 2610241. 45780. 54620482. 804160. 21140965. 547318. 533Table 5. 6 Performance testing of RSA 256-bit key sizeAbove table 5. 6 shows data of RSA algorithm with 256-bit key size. The first column represents the data size. Second and third column represents the encryption and decryption time respectively. Figure 5. 5 shows the comparison chart of RSA algorithm with 256-bit key size. Figure 5. 5 performance chart of RSA 256-bit key sizeFigure 5. 5 shows the comparison of encryption and decryption time of RSA algorithm. Here, X-Axis represents the data size and Y-Axis represent the time. In this chart encryption time is increasing sharply to the 128-bit to 4096-bit. The decryption time increasing ratio is almost double if we use 128-bit data size then takes time is 10. 035 and if we use 256-bit data size than take time is 20. 248 if we increase data size. Table 5. 7 shows the performance of RSA algorithm with 512-bit key size in SAC.

Data Size (Bit)

Encryption Time (Sec)

Decryption Time (Sec)

2561. 253133. 6485122. 434265. 67810243. 615397. 70820484. 796529. 73840965. 977661. 768Table 5. 7 Performance testing of RSA 512-bit key sizeAbove table 5. 7 shows the performance of RSA algorithm with 512-bit key. In table 5. 7 column 1 represents the data size in a bit. Column 2 and

column 3 represents the Encryption and decryption time in a second respectively. Figure 5. 6 shows the comparison chart of RSA algorithm with 512-bit key size. Figure 5. 6 performance chart of RSA 512-bit key sizeFigure 5. 6 shows the comparison of encryption and decryption time of RSA algorithm with 512-bit key. X-Axis represents the data size and Y-Axis represent the time. However, encryption time is increased step by step. On the other side decryption is suddenly rise as we increase data size. We have seen above all the chart figure 5. 2, 5. 3, 5. 4, 5. 5 and 5. 6 so we can say that, if we increase the data size or key size then increase time of encryption and decryption.

6. Discussion

In this section we are going to discuss about implemented RSA algorithm in SAC and C. First of all discuss about 512-bit RSA algorithm in SAC. Then after discuss about 64-bit RSA algorithm in C using long long data type.

6. 1 512-bit RSA algorithm in SAC

How we can implement RSA algorithm with 512 Bit key size? It is very easy to implement RSA algorithm with 32 bit key size and 64 bit key size. We can implement this algorithm with short key size in C and in SAC also by using arithmetic operation such as mod, power, etc. To achieve the result for 512 bit RSA algorithm, we have think differently, because if we use int or long long int, they support only 32 bit and 64 bit respectively. To solve this problem, we have to implement functions that can support more than 64 bits. To implement functions that supports more than 64 bits, we have to take our data type in binary array. Each element of an array represents bit 0

or 1. E. g. Message = [1, 0, 1, 0, 0, 0, 1]. Here in this example message size is 7 bit and message is 81. In RSA, most important functions are ency= me mod n and decy= encyd mod n. It is very easy to implement these functions with decimal arithmetic. We can perform these operations with fewer steps. But, if we want to implement these functions in binary data type, we have to implement different functions, because there are no supporting library that supports binary operation. How we can implement the binary arithmetic? To do arithmetic in binary, we have to implement some functions such as binary addition, binary subtraction, binary multiplication, etc. In this project I have implemented some functions that perform binary addition. We will discuss basic logic to implement these functions one by one by taking an example. Binary addition: Suppose we have two array of binary a=[1, 0, 0, 1] and b=[1, 0, 1, 1] . Now we want c= a+ b. We have to take both arrays in reverse order. E. g. last bit comes first and first bit comes last. For bit to bit we have to do addition. Rules for the addition are: 1 + 1 = 0 and Parity = 11 + 0 = 0 and parity = 01 + 1 + 1 = 1 and parity = 1

Steps / bit position (i)

A[i]

B[i]

parity

C[i]= a[i]+b[i]+parity

311002011010011011 00Final resultA= B= 1C= 101000Table 6. 1 Binary additionBinary Subtraction: Suppose we have two arrays. A = [1, 0, 1, 1], B= [1, 0, 0, 1] and C= A-B. To, do substation of binary arrays, we have to

calculate 2’s complement of 2nd array. Then do the binary addition with 1st array. Keep in mind; we have to ignore final parity. How to do 2’s complement? Take the array in reverse order. Take the same bit until finding 1st occurrence of bit 1. For rest of all bits do, 1 = 00 = 12’s complement of array B= [1, 0, 0, 1] is [0, 1, 1, 1].

Steps / bit position (i)

A[i]

2’scomplementB[i]

parity

C[i]= a[i]+ 2’scomplementB [i]+parity

311002111110110010 10Final resultA= B= 1C= 0010Table 6. 2 Binary SubtractionBinary multiplicationTo multiple binary, this function can be used. In this session we will discuss the logic behind this function. Suppose we have two binary numbers, A= [1, 0, 0, 1], B= [1, 0, 1, 1] and C= A * B. To obtain a result we have to go through the following steps. Take an array result = 0P= 0i= (length (b) – 1) to i= 0if b[i]== 1take pre = resultresult= embed (p) times 0 in aresult= binary addition of pre and resultp= p+1i = i-1Return result

Step(i)

B[i]

Pre= result

Result= a+ (embedding zeros)

Result= pre+result

P

3101001100102110011001011011110201110111001000110001 13Final Result= 1100011Table 6. 3 Binary multiplicationBinary divisionBinary division is a function, can be used for division. Suppose we have two array A=[1, 1, 1] and B=[1, 1] and C= A/B. To perform this task we have to go through the following steps. Not that this function accepts two argument a and b. Res= 0Len= length (a)Lenb= length (b)Resdiv= 0For i= 0 to i bRes = res – bResdiv = add 1 after resdivRemove leading zero from resElseResdiv = add 0 after resdivElseRes = res – bRemove leading zero from resResdiv = add 1 after resdivi= i+1return resdiv

Steps (i)

A[i]

Res

Length(res)

Resdiv

B

Res

01110111111120111

-

211101011

-

Final Result= 10Table 6. 4 Binary divisionBinary modulusThis function is used to find Modulus of two binary numbers. This function is very important for this project. Suppose we have two array A=[1, 1, 1] and B=[1, 0] and C= A(modB). The logical steps are listed below. Keep in mind that this function is recursive and accepts two parameters a and b containing binary array. Lena= length(a)Lenb= length(b)Res= 0If lena A

B

Lena

Lenb

Div

Div= div-b

res

Div= div+elements from a

11110321111111110221Final Result= 1Table 6. 5 Binary modulusNow, we will discuss implementation of RSA algorithm with 512 bit key size. In below Table 6. 6 we will discuss how we have implemented this algorithm in SAC with all functions. In Table 6. 6, there are several functions mentioned, are used to implement RSA algorithm

Function name

Return type

Parameters

BinaddInt, intInt a, int b, int cBinary_multInt[.]Int[.] a, int[.] bBinary_mod_powerInt[.]Int[.] a, int[.] b and int[.] cInt2binInt[.]Int aKey256bitInt[.]Int keysizeCkeckprimeIntInt[.] pBinary_candidateInt[.]Int[.] nBinary_divInt[.]Int[.] a, int[.] bHex2bin2Int[*]Int aHex2bin3Int[*]Int [.] aHex2binInt[*]String sBin2hexCharInt[.] aHexdisplayVoidInt[.] aReshapearrayInt[.,.]Int[.] a, int nMainInt

-

Table 6. 6 Function used implements RSA algorithm in SACNote: Table 6. 6 [.] – represents a 1D array, [.,.]and [*] represents 2D array. Now, we will discuss all these functions in details. BinaddThis function accepts three parameters: a, b and c, all parameters are in INT. This function returns two variables in INT format. One is res and another is carry. First it sums a, b and c. Checks the result= a+b+c. If result= 0 then res= 0 and carry= 0If result= 1 then res= 1 and carry= 0If result= 2 then res= 0 and carry= 1If result= 3 then res= 1 and carry= 1This function is used while doing addition of two binary. Binary_additionThis function accepts two parameters: a and b: both are in int[.]. This function returns one variable in int[.] format, named res. This function is used to do addition of two binary array a and b. In this function binadd function is used. TwoscompThis function accepts one variable : a in int[.]This function returns a variable in int[.] format named a.. This function is used to do 2’s complement of binary variable a. This function

can be used while doing binary subtraction. Binary_subThis function accepts two parameters : a and b, both are in int[.]. This function returns one variable in int[.] fromat, named res. This function is used to do binary subtraction of two binary variable a and b. This function, two's complement and binary addition function are used. PositiononeThis function accepts one parameter : a in int[.]This function returns one variable in int named res. This function is used to find the first occurrence of 1 in given array a. Binary_maxThis function accepts two parameters : a and b, both are in int[.]. This function returns one variable in int named res. This function is used to determine which binary number is greater than other. Binary_modThis function accepts two parameters: a and b, both are in int[.]This function returns one variable array int[.] named res. This function is used to calculate modulus of array a. e. g. a mod b. Binary_multThis function accepts two parameters: a and b, both are in int[.]. This functions returns one variable array int[.] named result. This function is used to calculate multiplication of two binary a and b. e. g. a X b. Binary_mod_powerThis function accepts three parameters: a, b and c. each of them is int. This function returns one variable array in int[.] named finalres. This function is used to calculate exponential modulus of a. e. g. ac mod b. Int2binThis function accepts one parameters a in int. This function returns one variable array in int[.] named res. This function is used to convert a decimal number into binary. Key256bitThis function accepts one parameters keysize in int. This function returns one variable array in int[.] named p. This function is used to generate up to 256 bit prime number. CheckprimeThis function accepts one parameters p in int. This function returns a variable int named prime. This function is used to

determine whether p is prime or not. It returns 1 if p is prime else returns 0. Binary_candidateThis function accepts one argument n in int[.]. This function returns a variable int[.] named res. This function is in key generating process. This function is used to calculate private exponential (d) in RSA. Binary_divThis function accepts two arguments a and b; both are in int[.]. This function returns a variable array in int[.] named resdiv. This function is used to calculate the division of two binary numbers. E. g. a/b. Hex2bin2This function accepts one argument a in int. This function returns variable a in int. This function is used to convert a ASCII into HEX value. Hex2bin3This function accepts one argument a in int[.]This function returns a variable array int[*] named res. This function is used to convert HEX value into binary equivalent. Hex2binThis function accepts one argument s in string. This function returns a variable array int[*] named intsb. This function is used to convert string into hex binary. HexdisplayThis function accepts one argument a in int[.]. This function is a void and has no return type. This function is used to display HEX data. ReshapearrayThis function accepts two argument a in int[.] and n in int. This function returns one variable in int[.,.] named mreshape. This function is used to reshape an array a into 2D array. How does program work? This program is written in SAC. There are four main steps of this program. Figure 6. 1 How programme workGenerate P and QKey256bitGenerate random number pIs P prime ? Returns PEndYesNoFigure 6. 2 Algorithm Generate P and QCalculate n, e and onStartN= p X qE= [1 0 0 0 1]On=(p-1) X (q-1)EndKey256bitFigure 6. 3 Algorithm calculate n, e and onCalculate private exponential d. Binary_candidateI= 1B= n X iB= b +1Res = b % 17Is Res= 0? Res = b /

17Return resI = I +1Figure 6. 4 Algorithm Calculate private exponential dEncryption and decryptionStartConvert Text into blockM= next block bitsEncy= binary_mod_power(m, n, e)Next Block? EndFigure 6. 5 Algorithm Encryption and decryption

6. 2 64-bit RSA algorithm in C

This section we can discuss about 64-bit RSA algorithm in C using long long data type. Long long data type can be used to handle very large number in C language. Basically, in C language number of 32-bits can’t exceed a specific value, So in order to handle a specific value handling 64-bit integers in C programing language. To print a long long data type display is different from other data types. The long long data type makes handling 64-bit integers easy. The long getrand method can create the series of random numbers and get a random number. isPrime function is used for a check given number is prime or not. If the number is divisible for only one and itself then generating number as a prime number. Now , we will discuss how we have implemented this algorithm, we have to implement some functions which are not inbuilt in C. Below in table 6. 7; I have mentioned functions used to implement this algorithm.

Function Name

Return Type

Parameters

GetrandLongLong toprangeIsprimeIntLong

numGenerateprimeVoidCandidatesLong longLong long qn, int

etoBinaryVoidLong long a, char *cExpmodLong longLong long a, char *e,

long long nMainIntTable 6. 7 List of function implement RSA algorithm in CWe will discuss above listed all functions in details. GetrandThis function accepts one parameter toprange in long format. This function returns a variable of long type. This function is useful in generating randomize numbers. isPrimethis function accepts one parameter num in long format. This function returns a variable of int type. This function is used whether input number is prime or not. This function returns 1 if num is prime else returns 0. GenerateprimeThis function does not accept any parameter. This function has type of void, so it has no return variable. This function is used to generate prime and stores the values of p and q. CandidatesThis function accepts two variable qn and e have data type long long and int respectively. This function returns a variable of type long long. This function is used to calculate private exponential (d). toBinayThis function accepts two variable long long a and char *c. This function is a type of void hence it has no return value. This function Is used to convert a into binary and stors value in c. ExpmodThis function accepts three variable long long a, char *e and long long n. This function returns a value in the long long format. This function is used to calculate encryption and decryption. E. g. Me(mon n).

Conclusion

In this project we have implemented the RSA algorithm in SAC as well as in C. C is a basic programming language and has accepted widely in the world and is the most popular language. On the other hand SAC is still in research and is not commercially accepted in the world. So, basically, SAC takes time to learn and SAC is also derived from C. In terms of performance, C takes a

less time than SAC to compile, because, when we compile the SAC file, it creates a C file and compile it. In terms of Memory Management, SAC is better than C. C does not provide dynamic memory on run time, while SAC provides dynamic memory on runtime. In C, once you have allocated memory, you cannot change memory, while in SAC, you can do this. So, for this kind of algorithms like RSA, etc., SAC is the better option than C.

Bibliography