✍️ Project Overview: POC

Target release	October 25
Epic	1. POC-10: Repository Setup DONE 2. POC-20: Module I: UX/UI IN REVIEW 3. POC-30: Module II: Encrypt Variables IN REVIEW 4. POC-40: Module III: Process Encrypted Variables TO DO 5. POC-50: Module CI: Send Variables from MI to MII CODE COMPLETE 6. POC-60: Module CII: Send Encrypted Variables from MII to MIII (Backend) CODE COMPLETE 7. POC-70: Module CIII: Send Final Result from MIII to MI CODE COMPLETE
Project status	Repository Setup Complete. Starting Work on UX/UI.
Document owner	@jramirez
Designer	@jramirez
Tech lead	
Technical writers	
Project Manager	@jramirez
Document Editors	@jramirez

🎯 Objective

The objective of constructing a functional and basic Proof of Concept is to rapidly showcase our innovative technology to investors, collaborators and potential clients, without having to build a robust and complex system. The POC is intended to be developed in a short period and is only supposed to superficially demonstrate the functionality of our Homomorphic Encryption Scheme.

We will deploy a webpage where the Client can enter their financial information into dialog boxes. This data is to be encrypted and THEN sent to the Bank (us) for processing. The Client will not have to download or execute any software.

When we receive the encrypted data, we will apply a mathematical function to it and obtain the final credit score. Then we send that number to the Client.

📊 Success metrics

We will have deployed a successful POC when the general public is able to execute and verify the security standard of our POC from any smart device.

Goal	Metric
Demonstrate the functionality of the Homomorphic Encryption Scheme.	Maintain data precision and evaluate time results.
Allow Client to verify security levels of our scheme.	The Client must be able to verify the Data shared is Encrypted.
Achieve established time limit.	6 week time limit.

Problem Background

One of the most basic issues in cryptography that has not been satisfactorily resolved is the problem of encryption during processing. There are currently cryptography standards for the encryption of data at rest and in transit (end-to-end encryption). However, if the data is to be processed or manipulated with mathematical operations, it will have to be decrypted. The data is vulnerable when it is decrypted for processing. For example, there may be instances where outsourcing AI training to a data center is the best option. However, in many cases the data required for AI training is sensitive and this can make the option unfeasible because there are laws that prevent the sharing of certain sensitive data.

A more everyday scenario happens when a Client goes to their Bank to request a loan. The Client must provide personal data to the Bank so that it can evaluate the Client's financial situation and thus be able to make a decision Fig. 1.

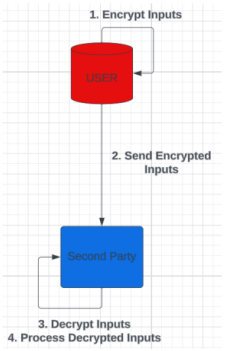


Fig. 1

Homomorphic Encryption is a general method under development that seeks to solve this problem. It was first proposed as a conceptual solution in the 1970s, to process data without reading the data. The first FHE scheme was developed in 2009 by Craig Gentry (See: References>CS>Security>Homomorphic Encryption). The general idea behind HE is that the order of data processing and decryption is reversed. In the case of traditional encryption, the data is first decrypted and then processed. In the case of HE, it is first processed and then decrypted Fig. 2.

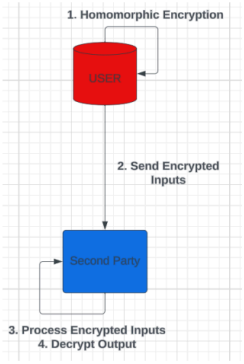


Fig. 2

Advantages of HE:

- Hides plaintext Input from Second Party.
- Second Party can be granted or denied access to the decrypted Output with a numeric key.

Disadvantages of HE:

- Long Delays
- Energy Intensive Processing and Decryption
- Noise
- Low Precision

Our Proposal

Although the concept of HE has been proposed for several decades, the solutions found to date are unsatisfactory due to time and energy consumption factors that make it an unfeasible implementation in most of the most important cases. A new method for HE is proposed that can improve the performance and efficiency of HE.

Below are described the elements and steps necessary for the implementation of a communication channel between Client and Bank. The Client requests a loan by filling out a form in a web application that allows the bank to process the data, without reading the data. The Client will provide personal financial data such as income, expenses, debts, assets, etc. This data will be encrypted through a “change of variables”, defined mathematically as an unknown transformation with security parameters (keys). After being encrypted, the data is sent to the bank. Traditionally, the bank would have to possess a decryption key to decrypt the data and calculate the customer's credit score and make a decision. Or, with a HE scheme, the Bank processes the data and then decrypts it. But, in either situation, the Bank requires a decryption key.

We are going to implement a scheme such that 1) the Bank does not require a Decryption Key to find the plaintext credit score, 2) the data can only be used for its intended purpose of calculating the credit score, and 3) data privacy is maintained even during processing. This happens because in our proposed method, the processing function is also the decryption key to obtain the final result. Once we have received the encrypted variables, a transformation is applied to them, resulting in a numerical score between 0 and 10. It will be important that the client can verify in the payloads that the data sent to the Bank is encrypted and we do not receive the plaintext inputs that the Client enters in the Interface, as in Fig. 3.

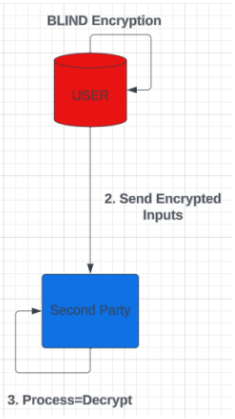


Fig. 3

Advantages of Keyless Decryption over HE

- Lower Time Delay
- Energy Efficiency
- Cutting Steps. Merges the processing and decryption steps of HE.
- The encrypted data can only be used for the intended purpose.
- The encryption of the inputs is the application of a family of continuous invertible functions. This enables other benefits such as:
  - Arbitrary floating-point precision. This simulation is a basic model with a medium level precision for managing 32-bit inputs. By medium level precision it is meant that this scale exhibits more precision than current standards of credit scores, but more precise application-specific versions can be constructed.
  - Noise is manageable.
  - Relatively fast computability, compared with other HE schemes.
- Flexibility. Some applications may require the decrypted Output to be known only by the User, other applications may require the decrypted Output to be known only by the Second Party, and yet other applications may require both parties to know the decrypted output. Keyless Decryption can accommodate different scenarios.

- Custom credit score. This math model can be modified to operate the functions, Parameters and Variables that the financial institution may prefer. By modifying the Parameters that are built into the model and adding variables we can consider various types of loans with different rates, periods, amounts, etc.
- Fast Derivative

Disadvantages of Keyless Decryption:

- Only works for a subclass of functions. Good News: Its limited to a pretty big and useful class of functions.
- Does not work when the final Output has to remain hidden to the Second Party. Good news: There are plenty of applications where the Second Party has to see ciphertext inputs and plaintext output.

Detailed Description of POC

The mathematical functions for encrypting and processing the variables are already defined and the Developers are not required to have previous knowledge of encryption standards. Below we detail the elements and parts of the proposed solution. The description is divided into Data Types, Modules and Steps. Our POC will simulate the communication between a Client and Bank, where the Client submits their personal information to the Bank for a loan approval. In reality, we will be communicating between the Client's terminal and the Browser.

Data Types

There wil be a total of six (6) Data Types.

I) INPUTS

First we have 6 (six) plaintext INPUTS representing the Client's data, in the form of 32-bit unsigned integers. The Inputs are collected in a round of six (6) questions (six dialog boxes).

1. **Net Income** (Yearly Income after taxes): Yearly income such as payouts, dividends, salaries, allowances, earnings, etc.
2. **Expenses**: Year-over-Year expenses and financial obligations such as personal expenses, spouse/child support, wages, home mortgage, vehicle loans, other recurring payments such as insurance payments, service to existing debt, credit cards, etc.
3. **Capital:** Total Liquid Assets such as personal wealth, cash, liquid bank accounts, bonds, personal funds, etc.
4. **Assets:** Real Estate/Land properties, vehicle titles, other non-liquid properties such as jewelry, art, etc.
5. **Current Debt:** Total Current Debt such as loans, mortgage, credit cards payments, etc. The Current Debt excludes the Newly Incurred Debt.
6. **Loan Amount:** The Client has to provide the Loan Amount which will also impact the credit score.

II) PARAMETERS

The Inputs will be processed, along with a set of weights and parameters that can be modified to calibrate the mathematical model to different custom settings and configurations for evaluating different types of loans. The situation presented here will measure the User's ability to repay the loan in the established payback terms. We simulate one scenario for a personal loan to be paid back in 12 monthly payments over a period of 12 months and with 15% yearly interest rate. Under the current settings, 5 points is the minimum suggested score for loan approval. Any score below 5 points should be considered risky. In the current setting a 2:1 Income/Expense ratio will give 5 points, so the loan is considered safe regardless of capital and debt. However, if the Income/Expense ratio is 1:1, then User requires a high Capital/Debt ratio to reach a score of 5 points.

- **Weight on the Income/Expense Measure.** The current settings are calibrated to attribute more importance to the Income/Expense Measure.
- **Weight on Expense.** Correction on the center of income/expense ratio. Under the current configuration, a 1:1 ratio on Income/Expense requires a perfect Capital/Debt ratio of 1:0 for acceptance.
- **Weight on Capital/Debt Measure.** The current model gives less importance to Capital/Liability than Income/Expense. Certain types of scores can be made to take Capital/Liability more or less into account.

- **Weight on Debt.** Correction on the center of Capital/Debt. The current model will give a quarter 0.25 points for a 1:1 Capital/Debt ratio. A ratio of 4:1 gives 0.9 points, while a ratio of 10:1 gives 1.5 points.
- **Weight on Assets.** Capital and assets are not necessarily valued equally. For example capital can be considered to be liquid assets and can therefore be valued close to 100% of their value, while other assets can be hard to liquidate. A parameter is included whereby a given type of assets can be given a weight. Here we allow input of non-liquid assets that are valued at below 50% of market value.
- **Interest Rate.** The yearly interest rate used in this model is 15% for a one-year-loan to be paid in 12 monthly payments. This model can be generalized for longer loan periods and different interest rates but this is only an illustrative example and it will be kept in simple relations to allow the User to analyze the behavior in one specific scenario.

III) VARIABLES

We will reduce the six (6) Inputs to a total of four (4) numeric values we will call VARIABLES. The four (4) Variables are Net Income (NI), Total Expenses (TE), Worth (W) and Total Debt (TD). The Score will be a numerical measure of the client's ability to repay the solicited Loan, based on four (4) financial Variables derived from their response to questions 1-6. Before calculating the Variables, we first have to find to other values. The loan is given for a default period of one year and it is calculated at 15% APR. This simply means

- *Newly Incurred Debt* = 1.15 * Loan Amount
- *Service to Debt* = *Newly Incurred Debt* / 12

The four (4) Variables of the mathematical model are:

- Net Income
- Total Expenses = Expenses + *Service to Debt*
- Worth = Capital + (*P5* * Assets)
- Total Debt = Current Debt + *Newly Incurred Debt*

P5 = 0.5 is a weight factor on the assets.

IV) KEYS

The Keys are random long integers. The Variables are then encrypted using the Parameters and Keys. The number of Keys will vary depending on the encryption function. We will use a set of five (5) encryption functions, and a set of five (5) corresponding processing functions. Each encryption function has to be processed by its respective processing function.

V) ENCRYPTED VARIABLES

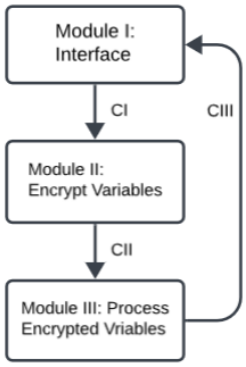
Encrypting the Variables produces new ENCRYPTED VARIABLES with 256-bit length. The Encrypted Variables are sent to the Bank for processing. The Bank processes the Encrypted Variables together with another set of Parameters that fine tune and adjust the mathematical model for different cases. For example, Parameters can adjust interest rate, loan period, weights on debt or on different classes of assets, etc.

VI) FINAL OUTPUT

The sixth data type is the FINAL OUTPUT which is the final Credit Score resulting from the Bank processing the Encrypted Variables. The output is a final numeric score between 0 and 10. Any score between 0 and 5 is considered risky, while a score over 5 is considered safe.

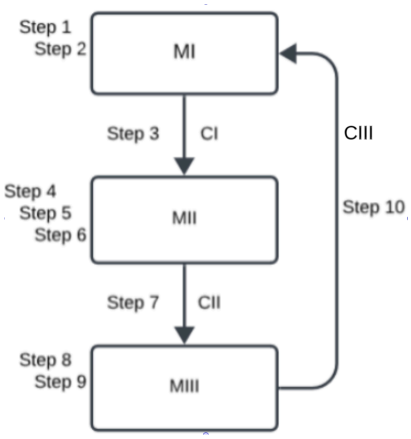
Modules

6 (six) Modules. The communications and internal operations for this scheme will be divided into six (6) MODULES, three (3) of which are for processes MI,MII,MIII and three (3) for communication between the first three Modules CI,CII,CIII. We will only be simulating communications between a Client and a Bank, so we will only be executing scripts from the browser using JAVA. Module I receives the Inputs and applies basic arithmetic to the Inputs, to find the Variables. Module II will encrypt the Variables. Module III receives and processes the Encrytped Variables.



1. MODULE I: UX/UI for INPUT/OUTPUT. The Client Interface is divided into three tasks.
   a. Receive Inputs
   b. Calculate Variables
   c. Display Information
      - Input: Six (6) 32-bit Integers representing Inputs of financial and loan data.
      - Output: Four (4) 32-bit Integers representing the Variables: Net Income (NI), Total Expenses (TE), Worth (W), Total Debt (TD).
2. MODULE II: Encrypt Variables. This module receives the Variables from Module I, and encrypts them. It is also divided into three tasks.
   a. Random Select Encryption Function. The encryption module has a library of encryption functions and each time the routine is executed, a random encryption function of the library is selected.
   b. Generate Keys. Once the encryption function has been randomly selected, we generate the corresponding keys for that function.
   c. Encrypt Variables. The encryption function executed uses the Variables and Keys as inputs, and outputs Encrypted Variables. Each encryption function of the library of encryption funcitons returns a different number of Encrypted Variables.
      - Input: Four (4) Variables
      - Output: Encrypted Variables
3. MODULE III: Process Encrypted Variables. for PROCESSING ENCRYPTED VARIABLES. This module will be run by the Bank, to calculate the final Credit Score using Encrypted Variables and Parameters, and it must support arbitrarily long arithmetic (512-bit arithmetic for operating on 256-bit numbers). Although it does not have an explicit decryption step, the output of this module is the decrypted Final Output (Credit Score).
   a. Read Encryption Function Index
   b. Process Encypted Variables
      - Input: Encrypted Variables (Output of Mod. II)
      - Output: Final Credit Score in plaintext (decrypted Credit Score)
4. MODULE CI: SEND VARIABLES from MI to MII. This module sends the Output of Module I (Variables) from the Client to the Browser.
   SEND: VARIABLES
   from: Module I
   to: Module II
5. MODULE CII: SEND ENCRYPTED VARIABLES from MII to MIII. This module communicates the Output of Module II (Encrypted Variables) from the Client to the Browser for processing. The Client will have the ability to verify that the Payload consists of the Encrypted Variables.
   SEND: ENCRYPTED VARIABLES
   from: Module II
   to: Module III
6. MODULE CIII: SEND FINAL OUTPUT from MIII to MI. This module is used for communicating the final Credit Score to the Client. The site displays the Final Output and other information in Module I (Interface).
   SEND: FINAL OUTPUT (Credit Score)
   from: Module III
   to: Module I

Steps



Ten (10) Steps are sufficient to complete the data path from the Client's Inputs, to the Bank's final decision.

STEP 1: INPUT DATA at MODULE I. The six Inputs are 32-bit unsigned integers representing Net Income, Expenses, Capital, Assets, Current Debt and Loan Amount.

STEP 2: CALCULATE VARIABLES at MODULE I. Use Inputs to find the four (4) Variables Net Income (NI),Total Expense (TE), Worth (W), Total Debt (TD). Variables are a simple transformation of the Inputs.

STEP 3: SEND VARIABLES from MODULE I to MODULE II using Module CI. After the Variables are calculated in Module I, they are sent to Module II for encrypting.

STEP 4: RANDOM SELECT ENCRYPTION FUNCTION at MODULE II. An encryption function is randomly selected from a library of encryption functions.

STEP 5: GENERATE KEYS at MODULE II. Keys are generated for encrypting the Variables using the selected encryption function.

STEP 6: ENCRYPT VARIABLES at MODULE II. Variables are encrypted using Module II.

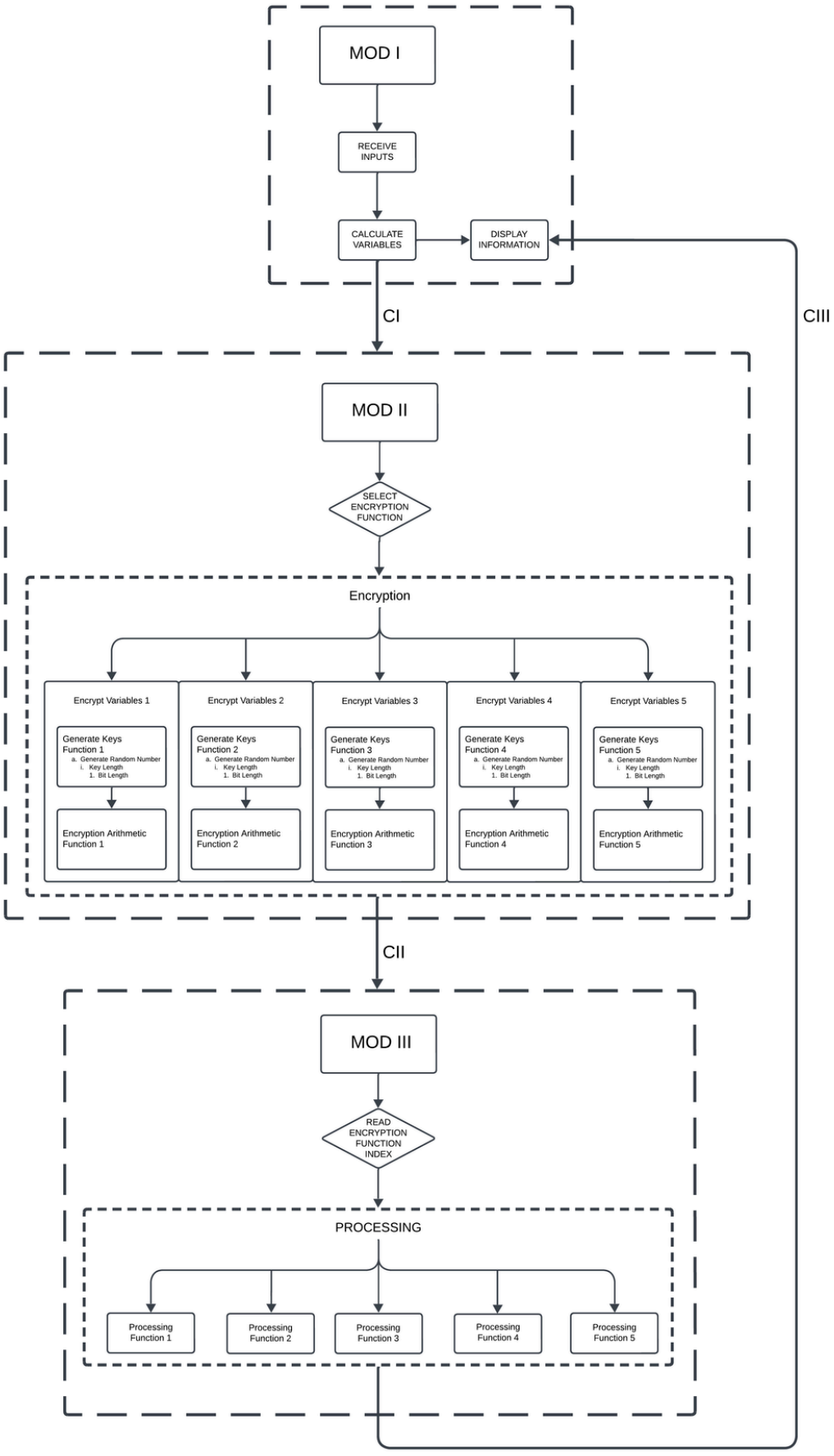
STEP 7: SEND ENCRYPTED VARIABLES from MODULE II to MODULE III using MODULE CII. Once the Variables have been encrypted in Module II, they are sent to Module III for processing.

STEP 8: DETERMINE PROCESSING FUNCTION at MODULE III. The Bank will receive Encrypted Variables and an index indicating the processing function that must be used.

STEP 9: PROCESS ENCRYPTED VARIABLES at MODULE III. The Encrypted Variables are processed.

STEP 10: DISPLAY INFORMATION at MODULE I. The results will be displayed in the Interface.

Functions


MOD I
RECEIVE INPUTS
CALCULATE VARIABLES
DISPLAY INFORMATION
CI
CIII
MOD II
SELECT ENCRYPTION FUNCTION
Encryption
Encrypt Variables 1
Encrypt Variables 2
Encrypt Variables 3
Encrypt Variables 4
Encrypt Variables 5
Generate Keys Function 1
a. Generate Random Number
i. Key Length
1. Bit Length
Generate Keys Function 2
Generate Keys Function 3
Generate Keys Function 4
Generate Keys Function 5
Encryption Arithmetic Function 1
Encryption Arithmetic Function 2
Encryption Arithmetic Function 3
Encryption Arithmetic Function 4
Encryption Arithmetic Function 5
CII
MOD III
READ ENCRYPTION FUNCTION INDEX
PROCESSING
Processing Function 1
Processing Function 2
Processing Function 3
Processing Function 4
Processing Function 5

The functions will be defined, declared/named and called on, as per the following convention.

I) Module I

We receive six (6) 32-bit Inputs Net Income, Expenses, Capital, Assets, Current Debt, Loan Amount in an online environment developed with JavaScript. These Inputs will be processed to obtain the four (4) Variables which are 32-bit integers NI,TE,W,TD. Finding the Variables requires using 64-bit integer arithmetic including addition, multiplication and division.

II) Module CI

The four Variables NI,TE,W,TD will be sent to Module II.

III) Module II

This Module receives the four (4) Variables “NI,TE,W,TD” and transforms them into a vector of Encrypted Variables. We define a function “Enc(NI,TE,W,TD) = (EV1,EV2,…)“ that takes in four (4) 32-bit integers and outputs a vector of Encrypted Variables.

To encrypt the variables, one of the five (5) different encryption functions will be executed. That is to say, there are five (5) encryption methods and only one is randomly selected and executed. The first subroutine executed in 'Enc' is this random selection function “SelectEncryptionFunction( ) = Int[1,5]”.

Once an integer in the interval [1,5] is randomly chosen, we need to run the corresponding encryption function. For this, we need to give each of the five (5) encryption functions an address/index associated to its integer in [1,5]. The second subroutine, of 'Enc', is function “RunEncryption(Int[1,5]) = (EV1,EV2,…)“ that receives the randomly selected integer and proceeds to execute the corresponding encryption function. The five (5) encryption functions are

1. EncryptVars1(NI,TE,W,TD) = (EV1,EV2,…,EV8)
2. EncryptVars2(NI,TE,W,TD) = (EV1,EV2,…,EV6)
3. EncryptVars3(NI,TE,W,TD) = (EV1,EV2,…,EV6)
4. EncryptVars4(NI,TE,W,TD) = (EV1,EV2,…,EV10)
5. EncryptVars5(NI,TE,W,TD) = (EV1,EV2,EV3,EV4)

Each one of these encryption processes will consist of two main subroutines. First, a set of Keys is generated, and then the Variables are encrypted.

1. Function ‘EncryptVars1' will have a subroutine “GenKey1(NI,TE,W,TD) = (Key1,Key2,…,Key6)“ that outputs six (6) randomly generated integers, and a subroutine “EncryptionArithmetic1(NI,TE,W,TD,Key1,…,Key6) = (EV1,EV2,…,EV8)” that outputs eight (8) Encrypted Variables.
2. Function ‘EncryptVars2' will have a subroutine “GenKey2(NI,TE,W,TD) = (Key1,Key2,…,Key6)“ that outputs six (6) randomly generated integers, and a subroutine “EncryptionArithmetic2(NI,TE,W,TD,Key1,…,Key6) = (EV1,EV2,…,EV6)” that outputs six (6) Encrypted Variables.
3. Function ‘EncryptVars3' will have a subroutine “GenKey3(NI,TE,W,TD) = (Key1,Key2,Key3,Key4)“ that outputs four (4) randomly generated integers, and a subroutine “EncryptionArithmetic3(NI,TE,W,TD,Key1,…,Key4) = (EV1,EV2,…,EV6)” that outputs six (6) Encrypted Variables.
4. Function ‘EncryptVars4' will have a subroutine “GenKey4(NI,TE,W,TD) = (Key1,Key2,…,Key8)“ that outputs eight (8) randomly generated integers, and a subroutine “EncryptionArithmetic4(NI,TE,W,TD,Key1,…,Key8) = (EV1,EV2,…,EV10)” that outputs ten (10) Encrypted Variables.
5. Function ‘EncryptVars5' will have a subroutine “GenKey5(NI,TE,W,TD) = (Key1,Key2,Key3,Key4)“ that outputs four (4) randomly generated integers, and a subroutine “EncryptionArithmetic5(NI,TE,W,TD,Key1,…,Key4) = (EV1,EV2,EV3,EV4)” that outputs four (4) Encrypted Variables.

Each one of the five (5) encryption methods ‘EncryptVars' has its own arithmetical subprocesses to define its ‘GenKey' and 'EncryptionArithmetic’ functions, but there are also general functions that will be called on as subroutines of the 'GenKey’ functions. These general use functions are

- “GenRand(Int) = BigNum” that will generate a random number with bit size equal to the input (arbitrary bit size).
- “KeyLength(Int) = Int” that will take an integer as input, and outputs an integer.
- “BitLength(Int) = Int“ that will take an integer as input, and outputs the bit size of the input.

IV) Module CII

The Encrypted Variables and the “Address/Index”, which are the outputs of Module II, will be sent to Module III.

V) Module III

Module III will receive the Encrypted Variables and an Index/Address from Module II, and process them to produce the Final Output as 64-bit single-precision floating-point number. This Module contains a library of five (5) processing functions. One for each encryption method. Each encryption method will have to be processed using its corresponding processing function. The Module will be defined as a function “ProcessEncryptedVariables(EV1,EV2,…,Index) = FP64“ that will read the Address/Index and execute the corresponding processing function. The five (5) processing functions are

1. “Process1(EV1,EV2,…,EV8) = FP64“
2. “Process2(EV1,EV2,…,EV6) = FP64“
3. “Process3(EV1,EV2,…,EV6) = FP64“
4. “Process4(EV1,EV2,…,EV10) = FP64“
5. “Process5(EV1,EV2,EV3,EV4) = FP64“

VI) Module CIII

The Final Output will be sent from Module III to the Interface in Module I.

🗒️ Requirements

Requirement	User Story	Importance	Jira Issue	Notes
Installation: Client must be able to run and execute without any installation.	The user should be able to easily open a link to our credit score simulator and enter his information. After this, the results should be displayed on his screen.	HIGH		Sergio and Michael have both confirmed the best way to simultaneously achieve this requirement and the **“Verification“** requirement is by encrypting the Variables in the cloud (with the host).
Verification: Client must be able to verify the Data they share with us is Encrypted.	If the Client verifies the Payload in his browser, it should be verified the data	VERY HIGH		

	processed in Module III is encrypted.			
Benchmarking: Time evaluation of the encryption and processing steps, as well as the whole process.	The user should have some idea of the time lapse for the encryption and processing steps.	NORMAL		
Interface: Display a timeline showing the Data path.	The user should also have a display of the data path including “Inputs“, “Variables“, “Keys“, “Encrypted Variables“, “Credit Score“.	HIGH		

Open Questions

Question	Answer	Date Answered
Are we able to implement client-side encryption?	Yes.	August/12/2024