Homomorphic Encryption Within The NHS Using Paillier

Adam Richard Jones
Student ID: 1553590
Supervisor: Dr. David Galindo

University of Birmingham
School of Computer Science

A thesis submitted for the degree of

BSci Computer Science and Mathematics

2018

Acknowledgements

I would like to thank my project supervisor, David Galindo, for his guidance throughout my project. I would also like to thank my family, friends and girlfriend for their support through my degree.

Abstract

The system described herein has been developed with the aim of supporting the NHS and Government. It collects, collates and analyses data in a secure format to assist, health professionals, Government officials and politicians to improve patient care and get most value for money from available budgets. The final solution aims to assist health professionals carry out their day to day job, providing information they can use to improve the patient care they provide in terms of quality and effectiveness and permitting homomorphic computations to be performed on the nations encrypted medical records. It also aims to enable critical analysis to settle debates on policy decisions. The system has been developed by focusing on two current political issues in Wales to demonstrate its application to both a wide and more specific application. The areas used are the decision to provide free prescriptions to all residents in Wales and the use of prescriptions for non-medical treatment to implement the Welsh Government Book Prescription Wales scheme. The system solution has been achieved through the application of Pailliers' cryptosystem to address the requirement for homomorphic addition. This has enabled a solution to be identified which grants the option of analysing medical records without compromising patient privacy.

Contents

1 Introduction **1**
1.1 Report structure 2

2 **4**
2.1 Further background material 4
2.1.1 RSA 5
2.1.2 ElGamal 5
2.1.3 Exponential ElGamal 6
2.1.4 Paillier 6
2.1.5 Fully homomorphic encryption 7
2.2 Analysis and Specification 9
2.2.1 System requirements 11
2.2.1.1 Functional requirements 11
2.2.1.2 Non-functional requirements 13
2.3 Design 15
2.3.1 Database 15
2.3.2 System 18
2.3.3 Decisions 23
2.4 Implementing and testing 25
2.5 User interface 32
2.6 Project management 33
2.7 Results and evaluation 35
2.7.1 Functionality 35
2.7.2 User interface 35
2.7.3 Performance 35
2.7.4 Robustness 35

3 **40**

3.1 Discussion 40

3.2 Conclusion 43

A Additional reading on encryption schemes **44**

B Planning **46**

C Running the system **47**

List of Figures

2.1 NHS expenditure 10
2.2 Structure of applications table 15
2.3 Structure of BPW table 15
2.4 Structure of drugs table 16
2.5 Structure of employee table 16
2.6 Structure of filters table 17
2.7 Structure of messages table 17
2.8 Structure of patients table 17
2.9 Navigational flow of system developed 18
2.10 Layout of login form 18
2.11 layout of forgotten password form 19
2.12 Layout of new organisation form 19
2.13 Layout of prescription form 20
2.14 Layout of patient information form 20
2.15 Layout of BPW analysis form 21
2.16 Layout of doctors inbox 21
2.17 Layout of health expert inbox 21
2.18 Layout of total drug cost form 22
2.19 Layout of application inbox 22
2.20 Example simulation test for analysis of BPW 26
2.21 Test for submission of a rating 26
2.22 Test of statistical operations 27
2.23 Test of homomorphic addition 27
2.24 Generation of a large prime 28
2.25 Generation of a valid generator g 28
2.26 Calculating λ 29
2.27 Calculating μ 29
2.28 Encryption of an integer 30

2.29 Decryption of an integer 30
2.30 Encryption of a string 31
2.31 Decryption of a string 31
2.32 Ciphertext generated for plaintext 16 36
2.33 Ciphertext generated for plaintext 43 36
2.34 Ciphertext generated through multiplying figures 2.32 and 2.33 36
2.35 Ciphertext 1 37
2.36 Ciphertext 2 37
2.37 Ciphertext 3 37
2.38 Deconstruction and encryption of a string. 38
2.39 Base64 encoder applied to return a string which can be stored in database. 38
2.40 Decryption and reconstruction of a string. 38
2.41 Output generated showing rating for books selected. 39
2.42 Output generated showing a mixture of statistical analysis 39

B.1 Time management 46

Chapter 1

Introduction

The purpose of the work included within this Report is to provide the National Health Service (NHS) Wales a means for storing patient data in a secure manner, but in such a way as to ensure data analysis was not prohibited. We are in an age today, where the analysis of big data by healthcare experts, researchers and other practitioners can be used to address problems in the healthcare System, (Wang et al. 2015) such as:

- The variance in drugs prescribed to patients with the same disease(s) which create issues such as the 'post code lottery' of care;
- provide definitive answers as to which drugs are most effective thus ensuring patients receive the drug which is generally most effective first, generating significant public funding savings and improving quality of life;
- good evidence to support new policy or justify the continuation of existing policy such as free drug prescriptions in Wales, where this expensive policy could perhaps be modified or stopped without affecting the health of the public;
- the effectiveness of other non-drug treatments on prescription such as the recently implemented Book Prescription Wales(BPW) scheme within Wales, where people with mild to moderate mental health issues are prescribed a self-help therapy book(selected as the most effective books by psychiatrists and psychologists.

The analysis system developed, should also be capable of:

- assisting a doctor with their day to day job, while simultaneously gathering an ever growing volume of anonymous medical data;
- storing the patients' personal data in an encrypted format to be semantically secure as to ensure anonymity of the patient; and

- 3^{rd} party organisations once granted permission, should be able to perform computations on the encrypted data in a homomorphic manner, in order to achieve accurate analysis.

In 2013, the paper (Groves et al. 2013) revealed that since 2010 over 200(we can expect this number to be even greater now) new businesses have developed a range of techniques to improve the use of medical data which is readily available in this day and age. As technological capabilities advance we can expect new ideas for analysis of big data to emerge, some of which could substantially reduce the ever increasing cost of healthcare. In order to bring this ideological big data analysis into practice, the current healthcare system needs to undergo some essential modifications to its current structure, it is integral to protect the identity of those who's data is undergoing analysis, where the results will inevitably end up being publicly available.(Wang et al. 2015)

My solution will comprise of two fundamental aspects. Firstly the gathering and retrieval of all medical data which must be handled in a secure manner, it is of paramount importance that when the data is at rest (in the national database) it is encrypted so patient privacy is guaranteed. Secondly, the data which could potentially provide key analysis for the future development of healthcare, needs to be encrypted in such a way that homomorphic operations may be computed, without altering the integrity of the plaintext or revealing the identity of those involved.

1.1 Report structure

The remainder of this paper will be structured as follows.

Further background material
Here we will define some of the fundamental concepts which are integrated into the solution. Additionally we will analyse the different forms of homomorphic encryption available.

Analysis and Specification
How I analysed the problem. This section will also include the system requirements.

Design
An overview of the structure of the solution. Additionally, we will inspect some of the influential algorithms embedded in the system.

Implementation and testing

Here we will see the testing methods carried out to ensure the functionality of the system is both correct and purposeful.

User interface

An overview of the user interface and decisions behind its design.

Project management

An insight into the timeline designed to keep myself on track throughout the project in addition to how i overcame unforeseen obstacles.

Results and evaluation

Evaluation of the system as a whole. Here we will scrutinise the system, looking at its overall robustness, performance and practicality of application to the initial problem.

Discussion

An overall summary of both the achievements and deficiencies of the project as a whole, also an insight into where this system could be taken if I had more time or was developed further.

Conclusion

A conclusive paragraph on the solution provided, to the problem given at the beginning of the Report, ending with a list of references and appendices.

Chapter 2

2.1 Further background material

Encryption is the process of converting data from something intelligible into something unintelligible. The main purpose of this is to prevent unauthorised personnel from viewing this data. Homomorphic encryption (HE) is a form of encryption where the application of an algebraic operation on a given ciphertext results in an algebraic operation being performed on the plaintext(not necessarily the same operation as we will discover later). There are two types of HE, namely, fully homomorphic encryption(FHE) and partially homomorphic encryption(PHE). A FHE scheme means that taking an arbitrary function f and wishing to obtain $f(m_1,..., m_n)$, the user is instead able to compute on the encrypted values of these inputs, $c_1,..., c_n$. The resulting ciphertext can then be decrypted to obtain $f(m_1,..., m_n)$. (Armknecht et al. 2015) A partially homomorphic encryption(PHE) scheme is one in which it provides only one of either additive or multiplicative homomorphic properties, but strictly not both.

An encryption algorithm can be either asymmetric or symmetric, which involve the scheme having both a public and private key or just a private key respectively. In an asymmetric system, the public key can be accessed by anyone allowing them to encrypt a message, the resulting ciphertext can only be decrypted(feasibly) by the owner of the private key. While in a symmetric system there is only a secret key, which is used for both the encryption and decryption of a message.(Paillier 1999)

In practice, it should be computationally infeasible to calculate one key from the other in an asymmetric system(Simmons 1979). There have been a number of PHE algorithms published over recent years. ElGamal, Exponential ElGamal, Paillier and RSA are some of the most commonly used and readily available systems which could be used in the solution to the given problem.

2.1.1 RSA

RSA is the first example of a PHE we will look at. The homomorphic property which it possesses is that it permits multiplicative computation of given ciphertexts. RSA is asymmetric and thus relies on two keys to operate. The proof of correctness and key generation for RSA can be found in appendix 1. The robustness of RSA rests on the intractability of integer factorisation for large values of n, because of this, once both keys have been created it is *imperative* that the values of p and q are destroyed or kept secret.

An application of RSA encryption shown in (Tebaa et al. 2012) could be in Cloud security, where for example a client sends the pair (c_1, c_2) where $c_1 = m_1^e \ mod \ n$ and $c_2 = m_2^e \ mod \ n$ to the Cloud, the server performs the computation requested and returns the encrypted result (c_1, c_2) back to the client. As stated in the paper(Tebaa et al. 2012), there is a potential flaw in this form of encryption, which occurs if an attacker intercepts the two ciphers (c_1, c_2). Encrypted with the same private key, the attacker could be able to decrypt all messages exchanged between the Cloud and the client due to the product of the ciphers equalling the cipher obtained from the product of the plaintext. This potential flaw can only be utilised due to the bijection of plaintext to ciphertext which occurs through deterministic encryption. However, this problem can be overcome by the introduction of random padding, which ensures a plaintext can have multiple different ciphertexts which, once decrypted all return the original value, this is called probabilistic. (Coppersmith 1997)

2.1.2 ElGamal

The second algorithm we will look at is ElGamal. Similar to RSA, it also possesses the multiplicative homomorphic property. The security of ElGamal is said to rest on the difficulty of taking discrete logs in a finite field. It is believed that while taking exponents is tractable, that the inverse operation is intractable.(of Mount Union n.d.) For the proof of correctness and key generation of ElGamal the reader is referred to appendix 1.

One example where ElGamal can be used is demonstrated in the paper (DÜLGERLER & SARISAKAL 2003), where an ElGamal cryptosystem is used to bolster the security of an email application. To do this, the text is split into blocks. Characters are then scanned one at a time and for each character a random k is chosen. This step is essential to make sure the system is semantically secure and is an advantage ElGamal

has over the 'textbook' algorithm of RSA. The values y_1 and y_2(the two parameters of the ciphertext), must be modulated to the length of the alphabet after encryption.

2.1.3 Exponential ElGamal

A PHE named Exponential ElGamal can be derived by making a slight change to the ElGamal encryption scheme. Exponential ElGamal is again asymmetric and probabilistic but differs in its homomorphic property, which is additive homomorphic as opposed to multiplicative. This alteration is done by encrypting α^m rather than m, giving us:

$$c := (y_1, y_2) := (\alpha^r, \alpha^m \cdot \beta^r) \; mod \; p$$

Proof of correctness for homomorphic addition:

- Let $c_1 := (y_1, y_2) := (\alpha^{r_1}, \alpha^{m_1} \cdot \beta^{r_1}) \; mod \; p$
- Let $c_2 := (y_3, y_4) := (\alpha^{r_2}, \alpha^{m_2} \cdot \beta^{r_2}) \; mod \; p$
- Compute $c_3 := c_1 \cdot c_2$
- $c_3 := (y_5, y_6) := (\alpha^{r_1} \cdot \alpha^{r_2}, \alpha^{m_1} \cdot \alpha^{m_2} \cdot \beta^{r_1} \cdot \beta^{r_2}) \; mod \; p$
- $c_3 := (y_5, y_6) := (\alpha^{r_1 + r_2}, \alpha^{m_1 + m_2} \cdot \beta^{r_1 + r_2}) \; mod \; p$
- Let $r' := r_1 + r_2$ and $M := m_1 + m_2$
- Then $c_3 := (y_5, y_6) := (\alpha^{r'}, \alpha^{M} \cdot \beta^{r'}) \; mod \; p$

A disadvantage to Exponential ElGamal is that we have to limit the plaintext available in order to ensure that decryption is tractable. If we leave the plaintext space with no restrictions, then discrete logarithms would be needed to recover the original plaintext if the value is considered too large, which by construction is assumed to be intractable. (Ryan 2008)

2.1.4 Paillier

The penultimate homomorphic scheme we will look at is Paillier. This is another example there the homomorphic property possessed is additive. Paillier is known to be a probabilistic encryption scheme due to the randomness implemented in its encryption algorithm. Which as shown by (Ryan 2008) can be done in the following steps:

- Calculate $n := p \cdot q$ where p and q are two large primes
- Calculate $\lambda := lcm(p-1, q-1)$
- Find a generator $g \epsilon \mathbb{Z}^*_{n^2}$ such that $g = 1 \; mod \; n$
- Calculate $\mu := L(g^{\lambda}(mod n^2)) \; mod \; n$, where we define $L(x) := (x-1)/n$

- $Pub(n, g)$ *and* $Priv(\lambda, \mu)$

The security of the Paillier encryption system is believed to rest on the intractability of factorising large numbers, due to easily being able to calculate the private key once you have the primes p and q. The encryption and decryption of a message m as described by (Ryan 2008), is shown below:

$$c := g^m \cdot r^n \; mod \; n^2$$
$$m := L(c^\lambda \; mod \; n^2) \cdot \mu \; mod \; n$$

Another example of where Paillier has been used comes in the form of electronic voting systems. As explained in (Damgård & Jurik 2001), there is a scheme implemented by Cramer, Gennaro and Schoenmakers, which in laymans's terms, is where voters each broadcast an encryption of their vote, with the required proof that their vote is valid. All votes provided with a valid proof are then combined to form an encryption of the result of votes presented. This will be done using the homomorphic property of Paillier. Finally, a set of trustees who own the private key are able to decrypt the result and publish the findings. The potential uses are not limited to voting systems. Due to the durability of Paillier, other applications where it has also been used in papers include, watermarking, threshold cryptosystems and secret sharing schemes(Paillier 1999). The proof of correctness for the homomorphic feature of Paillier is as follows:

- $c_1 := g^{m_1} \cdot r_1^n \; mod \; n^2$
- $c_2 := g^{m_2} \cdot r_2^n \; mod \; n^2$
- $c_3 := c_1 \cdot c_2 = g^{m_1+m_2} \cdot (r_1 \cdot r_2)^n \; mod \; n^2$
- $M := m_1 + m_2, R := r_1 \cdot r_2$ resulting in $c_3 := g^M \cdot R^n \; mod \; n^2$

Where c_3 is in the standard form of decryption when using Paillier, this results in the following identities:

- $\forall \; m_1, m_2 \; \epsilon \; \mathbb{Z}$ and $k \; \epsilon \; \mathbb{N}$
- $D(E(m_1)E(m_2) \; mod \; n^2) = (m_1 + m_2) \; mod \; n$
- $D(E(m_1)^k \; mod \; n^2 = (k \cdot m) \; mod \; n$
- $D(E(m_1) \cdot g^{m_2} \; mod \; n^2 = (m_1 + m_2) \; mod \; n$
- $D(E(m_1)^{m_2} \; mod \; n^2 = (m_1 \cdot m_2) \; mod \; n = D(E(m_2)^{m_1}) \; mod \; n^2$

2.1.5 Fully homomorphic encryption

From (Gentry 2009), I will try to summarise the steps needed to create a FHE following Gentry's research. The method comprises of 3 main steps. Firstly, a "somewhat

homomorphic scheme" (SHS) is needed. This is a HE scheme which allows both additive and multiplicative computations on ciphertexts but, only a limited number of times. It is necessary that we make this SHS 'bootstrappable', which involves making the depth of decryption less than what the scheme can actually handle (Chapter 8 and 10 of (Gentry 2009) for details on how this is done). The second step involves designing an initial construction using ideal lattices. Through the use of ring operations, practically an unlimited number of additions are allowed. Additionally, this method is homomorphic for circuits whose multiplicative depth is greater. The final step of this complex procedure is to try and 'squash' the decryption circuit in a way that would allow bootstrapping. This procedure would be too complex to successfully apply within the parameters of which, I have assessed are appropriate to deliver the desired outcome of this project. It is however, recognised that Gentry's method could also be applied to possibly achieve even greater results.

In the U.K. patient health records are held by the NHS, which can be accessed in various places such as your GP, any hospital you have received treatment, your dental practice e.t.c. There are strict laws in place to ensure confidentiality of our personal data, namely the Data Protection Act(1998), one of these requirements is that the data held must be kept secure. Currently, the NHS adopts a scheme called "Encryption Good Practice Guide (GPG)" to determine how encrypted data should be evaluated, for the protection of data on a number of mediums such as laptops, removable media and emails. Encryption is used to ensure the NHS has confidence that data in transit and at rest is protected.(NHS 2017)

From collating my findings I believe Paillier is the most appropriate HE scheme available for my application. I have come to this conclusion because of the need for homomorphic addition throughout my system, thus eliminating ElGamal and RSA, despite their individual advantages over Paillier in other possible solutions. Both Paillier and Exponential ElGamal have the advantage of being probabilistic cryptosystems. As mentioned earlier one of the benefits of using Paillier over Exponential ElGamal is that the plaintext space will not need to be limited. This is another reason on why I feel Paillier is more applicable than exponential ElGamal in this scenario, as it means we need not restrict inputs. I believe as a HE scheme Paillier ticks all boxes of what is required for my system.

2.2 Analysis and Specification

When researching the problem established previously, there are a number of important solutions which are needed to ensure the system is successful in terms of providing an acceptable overall solution to the problem. In times when public funding is already restricted, the ever increasing costs to the NHS of new more expensive drugs, an ageing population and increased patient expectations, is a particular challenge to meet.

The policy in Wales has been to protect and increase the NHS budget even though it already receives significantly more funding then all other policy areas. The most efficient use of funding is therefore an imperative so, the political commitment to provide 100% free prescriptions in Wales is under constant consideration by the Welsh Government. The dilemma is then, if the funding were to be saved by charging for prescriptions, should the funding released be used to meet other NHS costs or should it be used in other policy areas such as social services, the economy or the environment etc. Such decisions must be made for the benefit of the nation and should only be made if there is such evidence available that demonstrates that either the previous policy was ineffective or can be achieved by other means. Whichever the case, data needs to be available to provide that evidence.

There is high level data collected and published by StatsWales but the level of detail needed to provide sufficient evidence is compromised due to restrictions to ensure patient confidentiality and lack of an analysis system that can collect more in-depth data within the security restrictions imposed. Through the implementation of homomorphic analysis it should be possible to offer a greater insight into this ongoing saga, providing useful results to finalising the correct decision on this policy and spend more time addressing other issues which affect the nation. The improved analysis capability would provide numerous additional benefits to justify evidence for policy decisions such as learning more about the general health of the nation with more substantive evidence then the current self diagnosis that is used in completion of health surveys.

There are also numerous opportunities of sharing and learning from good practice for example, if there is a spike in frequency of a specific disease in the population of the nation, it would be in everyone's interest if this could be detected (through regular analysis of medication given, duration, quantity, geographic location etc). Following the detection of a spike in occurrence of a disease a few paths could be taken. Firstly, health experts may delve deeper into why there is this increase and

try to raise awareness over how lifestyle changes can be used to help reduce the chance of being effected. Secondly, there may need to be an increase in research into the disease, looking for better treatment methods. Depending on the severity of the disease, charities may be granted funding to further their research, supporting NHS research and jointly delivering better treatments.

A smaller example of how this analysis system could be applied is as a tool to provide evidence of effectiveness to the NHS Wales adopted policy. Where they provide health expert advice via prescription, on their Book Prescription Wales scheme where, expert advice has been collated on which are the best and most effective, of the plethora of self help books for mild to moderate mental health conditions. They then make those best books available for free access via the library service across Wales. If there is evidence that the scheme is both effective and value for money then that helps decisions to continue and progress the scheme further. Wales is already considered a world leader in this field but such policies must continually be evaluated, which can only be done if the prescription data is collated and analysed.

Mental health issues effect 1 in 4 of us at some time in our lives and the cost to the NHS and the economy is huge (Government 2017). Targeting improvements to how we identify and treat such conditions, in any way, is therefore of significant importance to Wales, the UK and the world. Better data analysis will help raise awareness of mental health along with other major health conditions, shifting the focus to prevention and promoting further research into medication for some of these issues.

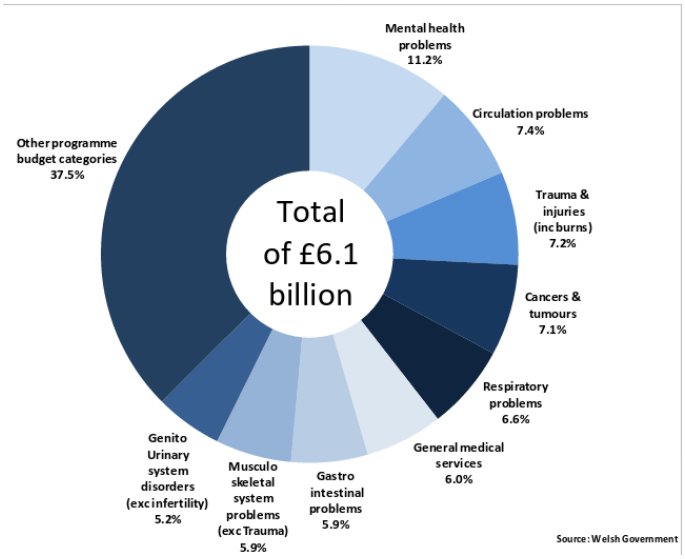

Figure 2.1: NHS expenditure

Information collected in a medical check such as family histories, allergies, current medication and past surgeries can help a doctor make a more accurate diagnosis or perform the necessary tests to diagnose diseases linked with the patients symptoms and medical history. This can be the case when a patient is not always aware of

some information. This may arise for example if the patient has lost contact with a side of their family or is not aware they are adopted, where in this scenario they will have a different family history to their guardians. Another potential case is if a patient is unconscious in a hospital, in this case they are not able to offer any medical history. To provide an effective solution to these problems, this information can be stored in their medical record, eliminating the two example issues, which can arise when a patient is unconscious or does not know their medical history but, health professionals can access it without breaking any confidentiality laws.

Another potential benefit of improving prescription data analysis could be taking the information gathered and applying it to improving other processes within the healthcare system. For example, the amount of time a GP can spend with each patient is constantly under increasing pressure across both Wales and the United Kingdom. GPs therefore need to use their time as efficiently as possible. If there are improved electronic processes in place to diagnose, determine most efficient and effective medication and dosage for the GP to prescribe there would be increased potential for not only increased quantity of patients that GPs can see but also there could be an increase of GP time available to spend on improving the quality of patient care. This in turn means that the GP can spend more time providing more personalised time with the patient and potentially assessing whether there are any other underlying issues that are causing the health problem. The goal could therefore be to create a process whereby the system, using historic analysed data,can create prescriptions for the patient in a simple manner, filling out the patients details and medication automatically for consideration and completion by the GP.

2.2.1 System requirements

2.2.1.1 Functional requirements

1. Login form
   (a) Desired language can be changed.
   (b) Providing a valid login the system should navigate the user to the correct main menu dependant on their privileges.
2. Forgotten password form
   (a) Answering the security question correctly for the selected user and providing two matching password, the password will be changed.

3. New organisation form
    (a) Providing all fields are filled correctly, application will be sent to the NHS for inspection.
4. Patient's medical record
    (a) Submission of matching patient ID and name. Patient's decrypted medical record is presented to the doctor.
    (b) Modifications to patient's record can be updated.
    (c) If any drugs are prescribed the filter table will need to be updated.
5. BPW form
    (a) An available book may be prescribed to the patient.
    (b) A new rating, for a book may be submitted. The new average is now recalculated.
6. Prescription form
    (a) Submission of a patient's ID will generate a partially filled out prescription form.
    (b) Prescription can be printed.
7. Inbox(Doctor)
    (a) Message can be decrypted and returned to the sender.
    (b) Message can be deleted.
8. Inbox(NHS)
    (a) Application can be accepted, storing their login details in employees table.
    (b) Application can be declined, removing the record from the system.
9. Inbox(Analyst)
    (a) Result can be opened in full. If result is of a graph, generate and display graphs in separate windows.
    (b) Result can deleted.

10. Total drug cost form

    (a) Drug information can be retrieved from database.

    (b) User can select the amount of filters applied.

    (c) Filters can be applied, gender and smoking habits.

    (d) Cost per unit can be modified(give insight into future costs).

    (e) Total cost for the drug and filters selected, at the price set is calculated, homomorphically.

11. BPW analysis form

    (a) A range of statistical operations can be chosen.

    (b) Choice over what the user would like to analyse, average rating or the number of ratings.

    (c) Select the number of books the user wishes to analyse.

    (d) The result is computed homomorphically.

2.2.1.2 Non-functional requirements

Reliability

The data stored within the database *must* be encrypted and stored correctly, failure to this could lead to the loss of data integrity. The method for encrypting data will be tested vigorously, giving confidence that this will not be possible.

Usability

The system should be designed in such a way that its use is self explanatory. The system should be simple to use, ensuring users have no difficulty with navigating through the system or making full use of all of the functions available.

Performance

The speed at which the system can operate is of high importance. An efficiently running system can assist with health checks of patients, providing faster and more accurate diagnoses. This can lead to both an overall improved healthcare and reduced waiting times for patients.

Security

The vast majority of data stored in the database is patient's medical records. Obviously, it is paramount that this is both stored and handled securely. To be certain that the data stored is secure, all patient data is encrypted using a probabilistic cryptosystem.

Portability

The software will be developed to work on all operating systems, providing they have an up to date Java Runtime Environment(JRE) and Java Development Kit(JDK).

2.3 Design

The design of my solution is centred around two key aspects, the database and the system itself. The database comprises of seven tables (Applications, BPW, Drugs, Employees, Filters, Messages, Patients). Each of which are explained where and when they are applied, a sample of each structure is shown with an example record(shown as plaintext).

2.3.1 Database

Applications

The applications table is used to store any pending requests from organisations who wish to analyse the encrypted data. These will be pending until an employee of the NHS confirms the organisations intentions and providing sufficient reasoning to allow them access. An example of this situation could be where the British Heart Foundation are looking for correlations in data between effectiveness of different treatments of heart diseases. From this, further development can be made to create even more effective drugs.

Company	Username	First_name	Surname	Childhood_bestfriend	Mothers_maiden_name	Place_of_birth	Contact_number	Password
British Heart foundation	BFS2018	John	Smith	Thomas	Smith	Birmingham	01215966845	Password1

Figure 2.2: Structure of applications table

BPW

This table holds the data regarding the NHS Book Prescription Wales scheme, in addition to storing the data of each book, the table additionally keeps a record of its average rating and number of times the book has been prescribed. The data which populates this table is taken from (NHS 2012). The table would be utilised in the circumstance where homomorphic analysis on the effectiveness of the scheme is questioned and when a patient is to be prescribed a book for support of a mental health issue.

Book_id	Book_title	Problem	Author	Year	Publisher	ISBN	Rating	Number_of_ratings
1	Overcoming Anger and Irritability	Anger	Davies, William	2009	Oxford Press	1255788492	4.25	123

Figure 2.3: Structure of BPW table

Drugs

The Drugs table contains information on the medication the NHS provides for the public, hence being vital in the political decision over the continuation of free prescriptions in Wales. Overall cost of individual drugs will be able to be calculated through the manipulation of data stored in this table. This can then be compared to the cost had patients purchased the drugs themselves, from a pharmacy or supermarket(commonly used 'over the counter' medications such as painkillers, cough medicine etc). Further analysis could be made into the effectiveness of drugs against different diseases if multiple drugs can be used as treatment for the same illness.

Drug_id	Drug_name	Drug_cost	Disease_treated
1	Hydrocodone	1.73	Mild to severe pain

Figure 2.4: Structure of drugs table

Employees

The data contained within this table is multi-purposeful, it is used to keep track of what users should have access to the system i.e a registered doctor or an approved organisation. Depending on the privileges the user has should determine where they are 'placed' within the system whether it be on the doctors side or the side for homomorphic analysis.

Username	First_name	Surname	Company	Admin	Place_of_birth	Mothers_maiden_name	Childhood_bestfriend	Password
arj590	Adam	Jones	GP	1	Cardiff	Jones	Cea-Jay	Password123

Figure 2.5: Structure of employee table

Filters

This table is created to keep a tally of the number of times a drug is prescribed to different filter groups. This is linked closely with the drugs table as when combined, can provide analysis of correlations between specific filters and medication taken.

Filter	Hydrocodone
Male, Smoker	126
Male, Non-Smoker	109
Female, Smoker	145
Female, Non-Smoker	77

Figure 2.6: Structure of filters table

Messages

Once a ciphertext has been obtained, following the application of homomorphic operations on the encrypted data, the result needs to be decrypted by the owner of the private key, in this case the GP/hospital. The messages table stores the 'message' consisting of a ciphertext and the context in which it was generated. When the GP/hospital confirm the intentions are fair then the decrypted result will be returned to the initial party.

Message	From	To
The total cost of the drug Generic_Zocor is (ciphertext) applied filters.	GP	NHS

Figure 2.7: Structure of messages table

Patient

Arguably the most important table in the database, is that used to hold the patients data. For the purpose of this project it was filled with false data for 1,000 patients using random numbers to determine the information selected for each field of each patient. The patients data is needed to be accessed by doctors in their day to day job when a patient comes to the GP or hospital. The patients table needs to be both updated to ensure data available is up to date and accurate but also accessible to be analysed.

Patient_id	First name	Middle name	Surname	Date of Birth	Address	City	Postcode	Smoker	Current medication	Allergies	Family history	Past Surgeries	Gender
1	John	Edward	Smith	01/01/2000	123 coronation road	Cardiff	CF64 4HH	No	N/a	Dust	Heart disease	N/a	Male

Figure 2.8: Structure of patients table

2.3.2 System

The system is split into two main functionalities, for use by doctors and use by health experts for analysis. Each side of the system has unique functions for each of the forms accessible to the user. The navigational options through the system can be seen below.

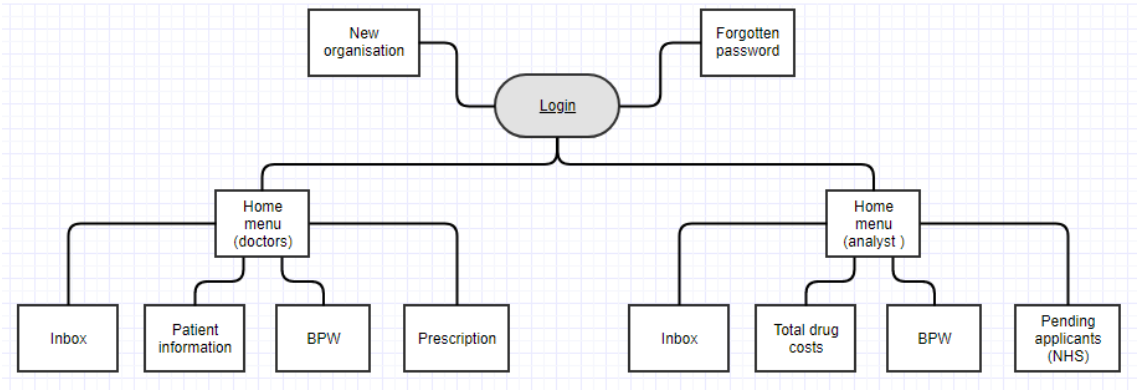

Figure 2.9: Navigational flow of system developed

Digging deeper into the system we can see the role each component plays in the system. Due to the limited purpose of both menu pages other than for navigation through the system, they are omitted from the structure breakdown.

Login

Welcoming page to the system. From here the user has 4 options, retrieve a forgotten password by answering security questions linked to their account, registration page for new organisations, log in as a doctor or an organisation and change the language of the system.

The login form is the first page which the user will see upon loading the software. Its design initialises the colour scheme, text font, and a general layout idea. As this system was designed for primary use within the health sector in Wales, there is a language option for translating the system from English(default) to Welsh. This is to ensure that the system is compliant

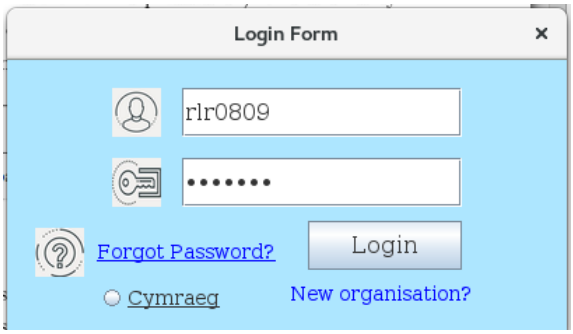

Figure 2.10: Layout of login form

with the Welsh Language Act. As the most common function on this form will be to log in, the log in button is a clear option, with the links to the registration and password recovery pages designed as just a label so as not to take away focus from the main function. To log in the user simply enters an existing username followed by the matching password.

Forgotten password

This form is needed for the recovery of a password. For obvious security reasons, before this can be done the user must select one of three security questions and provide a matching answer to that given upon registration. If all fields are filled correctly then the users password will be changed.

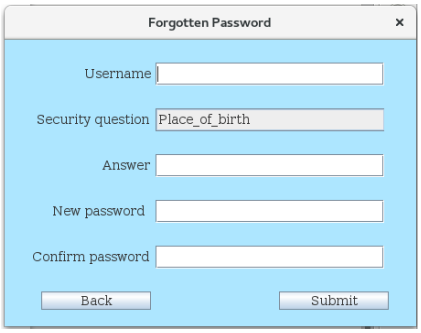

Figure 2.11: layout of forgotten password form

New organisation

To apply for access to the encrypted database an organisation must first fill in the form regarding the users potential account. The NHS, which acts as a body of trust between both parties, will establish contact with the organisation applying, after performing a gentle interrogation into the intentions of the organisation(reasons they wish to analyse data). They will then make a decision over the validity of their needs. If they believe this to be sufficient, the account will be approved and added to the 'Employees' table and so can be used to log in to the system.

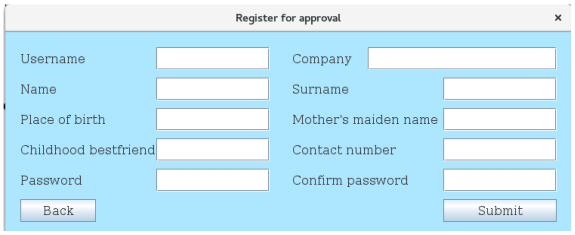

Figure 2.12: Layout of new organisation form

Prescriptions

Figure 2.13: Layout of prescription form

After a patient has been seen by a doctor, a prescription for a specific drug may need to be issued. In this scenario a doctor simply has to enter the desired patient ID number(equivalent to an EHIC number) to issue a pre-made form. The only details omitting from this is the doctors signature and the actual prescription details, such as amount and the frequency it should be taken. As described above, this could potentially be developed further to generate information automatically for the GP to consider.

Patient information

In order to ensure a more efficient and detailed medical history of a patient when they see a doctor, this form is designed to load up the relevant information regarding a patients health, providing they give a matching patient ID with name. Both credentials are needed to ensure only the correct patients information is decrypted to ensure patient privacy. The doctor is able to edit and update the information stored regarding the patient, for example, if they need to start taking new medication or previous treatment is stopped. Once an update is made to the system, the necessary amendments are made to the filter table in a homomorphic manner by adding to the count.

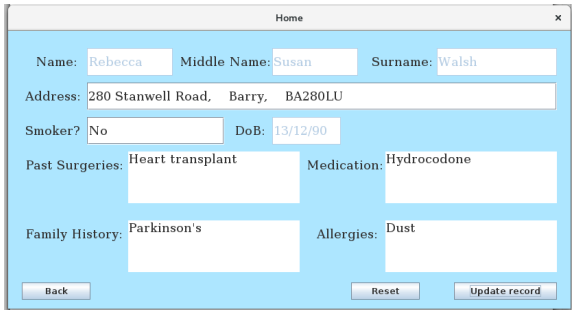

Figure 2.14: Layout of patient information form

BPW

This form is largely reliant upon the homomorphic operations available through Paillier. Through the manipulation of Pailliers formulas, solutions for the following statistical analysis functions can be derived, mean, median, mode, range, upper and lower quartile. These functions have been implemented into this form in

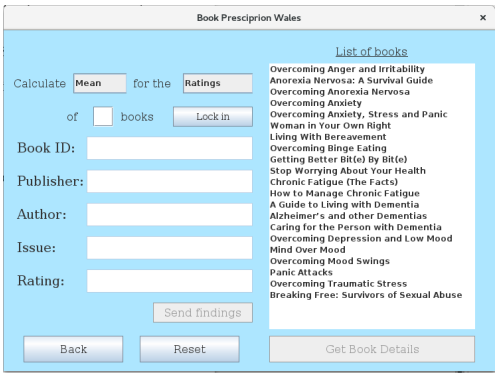

Figure 2.15: Layout of BPW analysis form

order to establish graphical representations of the average ratings given to a book prescribed or to see the uptake of books prescribed. The user is prompted to select the number of books they wish to analyse. They also need to choose which function they wish to apply be it mean, median mode, range or graph(showing a graph of all operations). In addition to whether they wish to look at the average rating for each book or at the number of ratings(approximately number prescribed).

Inbox

Shown on this form is the table of 'messages' containing ciphertexts following a users analysis of the encrypted data. If within the hospitals inbox then the message will be decrypted using the private key of which they have access to and the result will then be sent in a message back to the sender. To clarify the context this value had, there is a description attached to the message of how it was calculated. If however in the inbox of a user of the system outside of the hospital then the inbox will show a list of their decrypted findings. In order to ensure the form remains of suitable proportion in relation to the rest of the system, messages are condensed while in the inbox view. A message can be read in full, in a separate page by opening the message.

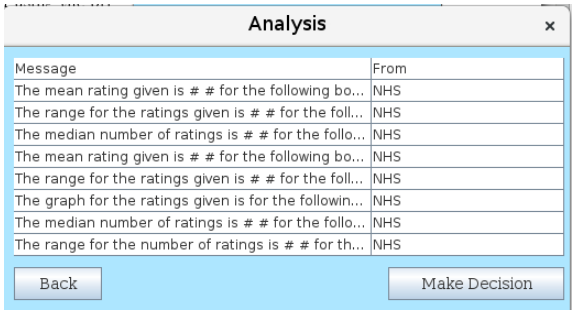

Figure 2.16: Layout of doctors inbox

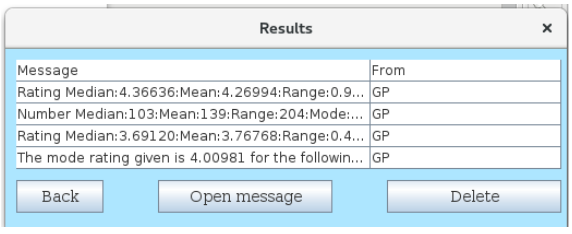

Figure 2.17: Layout of health expert inbox

An exception to this occurs when the result is a graph, as explained in the BPW form. In this scenario two graphs are generated, one showing the statistical operations, the other displaying the average rating/ number of ratings for each book, which is determined from the users previous choice.

Total drug cost

This form heavily relies on the use of homomorphic computations and is used to find the total cost of a certain drug. Upon loading the form, a list of available drugs is shown, generating the list from drugs currently being prescribed to patients. The user should select the desired drug and retrieve the relevant drug data, extracted from the database. Having selected the number of filters desired, the user will then choose which combination of filters they wish to apply for their computation. The system will take the appropriate ciphertext for each filter selected in the database and perform homomorphic addition to get the sum of the values chosen. After the total number of prescriptions for the chosen filters has been calculated, the drug cost entered is then rounded to the nearest integer (as seen earlier multiplication only works in Paillier through the use of integers in plaintext form). Using the appropriate formula shown ion page 7, the total cost of the chosen drug for the selected filters is displayed as a ciphertext and ready to be sent to someone with access to the private key, who can return the correct decrypted result.

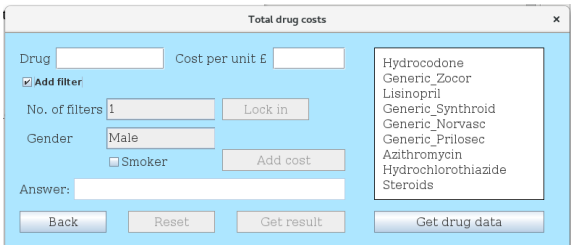

Figure 2.18: Layout of total drug cost form

Pending applications

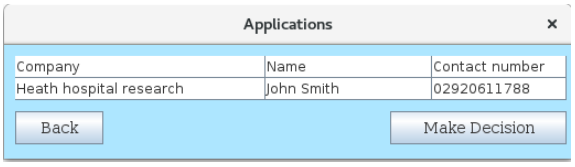

Figure 2.19: Layout of application inbox

Following the submission of an application to analyse the nations medical data, the NHS will need to view the application and follow up on whether they should be granted access to the database. All pending applications are placed in a table showing only the necessary information i.e. contact number, name and organisation. It is intended the NHS contact the enquiring organisation and follow up with any necessary questions or security checks. Following this, the application will either be accepted and added to the list of accepted logins or declined which will simply remove the application form.

2.3.3 Decisions

When designing the system, a number of executive decisions had to be made. One of these decisions came with the major issue for the storing of encrypted data, which was time consuming to overcome. This was initially caused when the cryptolibrary in use was being used on data types other than integers, for example, on a double. This meant it was not possible to encrypt the data as simply as usual. To get over this, the double was taken as a string. Once in string form, it is encoded, leaving the string in the form of a byte array. It was then possible to encrypt the byte array.

Having overcome this problem, it was now just needed to be stored in the database without losing the integrity of the data. Looking at this in more depth and in particular how others in the past had dealt with this problem. A solution that cropped up a number of times pointed towards storing the byte array in a Binary Large Object(BLOB) format within the database, which retains the integrity of binary arrays, although the most common need for this was through the storage of multimedia objects. Through reading all the responses, it was indicated it could be applied to this particular issue. After implementing all the steps to make this necessary, it became apparent that SQLite (being used to access and manipulate the database) can not handle this particular data form.

Upon further research being completed, a solution was found where, given the encrypted byte array and using Java's Base64 library, it was made possible to transform a byte array into a long string (which can be easily stored in the database). The original byte array can easily be retrieved, decrypted and decoded to return the original string which in this case was in decimal form.

Another key decision came with the analysis of the volume of drugs prescribed to patients who fall under different categories such as gender and smoking habits. The initial idea was to decrypt the necessary fields of each patient, checking first if they were currently taking the medication under analysis. Providing they were, the applied filters would then be decrypted and if a match was found then increment a counter. However, it was soon discovered this would take too long to decipher for a large database. The decision was made to run through the system for each filter and each drug currently being prescribed and then store the encrypted count for each result in a filters table. This would take time to establish, but once put in place it is easy to manage by homomorphically incrementing the count by one, once a patients record is updated given that their current medication shows they are being prescribed a drug. Now, when a total drug cost is needed the correct fields are found for the each filter applied, homomorphically summated to obtain a ciphertext for the total

number of prescriptions. The final step is to multiply the total count by the cost per unit (raising the ciphertext to the power of the integer value of the cost). Decrypting the resulting ciphertext will reveal the total cost of the drug selected for the applied filters.

The final design decision we will review is that used to reduce waiting time when decrypting medical records. Initially, when the system would begin to decrypt a record, following the submission of a patients name and ID. The form would appear to freeze to the user while the necessary computations took place, this gave a negative view of the system while it is unresponsive to user input. To counter this threads were implemented, which allow multiple functions to be performed simultaneously. Applying two threads to this case, one for decrypting the record and one for displaying a loading screen, which when full the patient form will be displayed. This feature gives comfort to the user that the system is still functioning fully.

2.4 Implementing and testing

To ensure that the system was behaving in the correct way and was returning the expected results I decided to use JUnit testing. This allowed test cases to be written to certify that the majority of the system was flawless with regards to its behaviour under different circumstances. This mainly comprised of two things, providing the system with what we would consider to be realistic inputs and also observing how the system handles data which would not be expected. We can assume the system is both correct and robust in the way that it operates, if it gives both the expected output for realistic input but also if it handles any 'bad' input in a secure manner and does not cause the system to malfunction.

It is important that as much coverage is achieved as possible, through the test cases, due to the need of patient privacy being delicately handled. It is essential that the use of homomorphic encryption and operations are semantically correct. If the wrong numbers are returned then incorrect analysis will be made, which could have catastrophic repercussions for the healthcare of the nation. Through the simulation of both good and bad inputs, buttons pressed and selection of items from tables, a maximum coverage of 75.2% was achieved. There is a very minor fluctuation of coverage which occurs depending on whether the first message in the inbox is a graph(higher coverage) or of a different form (returns a coverage of roughly 73%). Some functions which are contained in the cryptolibrary implemented in the system are not used in the system and thus are not tested which further reduces the coverage.

Despite this, a large proportion of the system is tested (All the code for tests carried out can be located in the LoginTest.java and TotalDrugCostTest.java classes.) and returns a very high coverage rate for being semantically correct, which is what was needed. In particular, all homomorphic functions are tested vigorously with a near 100% coverage (some catch statements are not reached) for behaving in the correct manner. This gives undoubted reassurance that any analysis obtained by health experts on the nations medical data is correct and can be used to makes strides in development of further research.

To avoid repetition we will just look at the test for calculating the mode rating of the 3^{rd} 4^{th} and 5^{th} books in the list provided. The system selects "Mode" from the combo box, leaving the source of analysis combo box to the default "Rating". The system then simulates a mouse click on the lock button, selecting the 3^{rd} option in the list, followed by simulating a click on the "Get book details" button. After repeating the previous step 3 times, the "Get results" button is clicked. The result is then sent to be decrypted by the private key.

```
@Test
public void NHSBPWTest() {
        NhsBPW b = new NhsBPW();
        assertEquals(true,b.validInput("4"));
        assertEquals(false,b.validInput("-2"));
        assertEquals(false,b.validInput("24"));
        assertEquals(false,b.validInput("String"));
        NhsBPW b2 = new NhsBPW();
        b2.oper.setSelectedItem("Mode");
        b2.rating.setText("3");
        b2.lock.doClick(200);
        b2.jlist.setSelectedIndex(3);
        b2.bookbtn.doClick(200);
        b2.jlist.setSelectedIndex(3);
        b2.bookbtn.doClick(200);
        b2.jlist.setSelectedIndex(3);
        b2.bookbtn.doClick(200);
        b2.prescribe.doClick(200);
```

Figure 2.20: Example simulation test for analysis of BPW

Additionally, it is needed to test the possible inputs of a user and ensure they are handled correctly. To calculate any of the statistical functions available the system requires a number x, such that $2 <= x <= 21$ of books to be analysed. Currently there are 21 books being used so this is the limit. To ensure the system only accepts values in this range it is tested for a string, negative number and a number within the range. The way in which the system handles these cases is by checking if the input string is an integer, if it is not then the error will be caught and a messagebox will be produced instructing the user an integer between 2 and 21 is needed. A similar methodological approach is used throughout the system when accepting input from the user, this can be further seen when a doctor gives a rating of a BPW prescribed book on behalf of a patient. As standard procedure, ratings range from zero (bad) to five (great). To ensure the mean rating of a book always fall within this range (inclusively) the system must not permit a rating to fall outside of the boundaries set.

```
@Test
public void BPWTest() {
        BPW b = new BPW();
        assertEquals(true,b.validInput("4"));
        assertEquals(true,b.validInput("5"));
        assertEquals(true,b.validInput("0"));
        assertEquals(false,b.validInput("6"));
        assertEquals(false,b.validInput("String"));
        assertEquals(false,b.validInput("-2"));
}
```

Figure 2.21: Test for submission of a rating

```
@Test
public void GPITest() {
    Login.comp = "GP";
    GpInbox gp = new GpInbox();
    gp.selected = true;
    assertEquals(true, gp.match("The mode for the num", "The mode for the num"));
    assertEquals(false, gp.match("", "String"));
    int[] ints = {5,7,6};
    int[] sorted = {7,6,5};
    double[] ds = {5,7,6};
    double[] sortedd = {7,6,5};
    assertArrayEquals(sorted, gp.BubbleSort(ints));
    assertTrue(Arrays.equals(sortedd, gp.BubbleSortd(ds)));

    String[] a = {"AGuHbjustmPlpmDB9tfaBOR56XInIgNoWAqT0dvCJJf/cv95e2jTaX7hE2boMJlV1qtxibmLj/gsbl/MU/LzrzQiHOVV7kYCjioTI4(
    String[] aa = {"AGuHbjustmPlpmDB9tfaBOR56XInIgNoWAqT0dvCJJf/cv95e2jTaX7hE2boMJlV1qtxibmLj/gsbl/MU/LzrzQiHOVV7kYCjioTI4
    String[] aaa = {"AGuHbjustmPlpmDB9tfaBOR56XInIgNoWAqT0dvCJJf/cv95e2jTaX7hE2boMJlV1qtxibmLj/gsbl/MU/LzrzQiHOVV7kYCjioTI
    assertEquals(new BigDecimal("2.00000"), gp.meand(a));
    assertEquals(new BigDecimal("2.00000"), gp.mediand(a));
    assertEquals(new BigDecimal("2.00000"), gp.mediand(aa));
    assertEquals(new BigDecimal("2.00000"), gp.moded(a));
    assertArrayEquals(a, gp.splitud(a));
    assertArrayEquals(a, gp.splitud(aa));
    assertArrayEquals(a, gp.splitld(a));
    assertArrayEquals(a, gp.splitld(aa));
    assertEquals(new BigDecimal("2.00000"), gp.lqd(a));
    assertEquals(new BigDecimal("2.00000"), gp.uqd(a));
    assertEquals(new BigDecimal("2.00000"), gp.uqd(aaa));
    assertEquals(new BigDecimal("0.00000"), gp.ranged(a));
    byte[] RAW=new byte[100];
    RAND rng=new RAND();
    rng.clean();
    rng.seed(100,RAW);
```

Figure 2.22: Test of statistical operations

In figure 2.22 the tests for mean, median, mode, upper quartile, lower quartile and range can be seen. To test this a group of plaintext integers are encrypted in the exact way we would if it is to be stored in the database. By hand the result which should be returned is calculated (to 5 decimal places). Using the built in function "assertEquals", the test will only show as successful if the two parameters are equal. In this case, the parameters are the value calculated by hand (on the plaintext) and the result of the function in question which is passed the array of ciphertexts.

```
@Test
public void homomorphicTest(){
    Login l = new Login();
    int five = 16;
    int seven = 43;
    String expectedans = "59";
    FF enc = new FF(8);
    enc.set(0);
    enc = Palliers.encrypt(l.pub, five, enc);
    System.out.println(enc.toString());
    FF enc2 = new FF(8);
    enc2.set(0);
    enc2 = Palliers.encrypt(l.pub, seven, enc2);
    System.out.println(enc2.toString());
    FF ans = new FF(16);
    ans.set(0);
    ans.add(FF.mul(enc2, enc));
    FF n2 = new FF(16);
    n2.set(0);
    n2.add4(l.pub.n2);
    ans.mod(n2);
    assertEquals(new BigInteger(expectedans), Palliers.decryptFromString(l.priv, ans.toString()));
    System.out.println(ans.toString());
```

Figure 2.23: Test of homomorphic addition

Figure 2.23 shows the test for the multiplication of two ciphertexts. The result is decrpyted using the private key. The decrypted result is then checked for equality to the addition of the plaintexts. Providing the two results match then it can be

concluded the algorithm for homomorphic addition is correct.

Throughout the project numerous algorithms were implemented in order to reach the end solution. We will look at some of the fundamental algorithms in both how they were programmed into the system and what some of the key steps to ensure they worked as they were intended. The class FF will be mentioned several times. This is abbreviated meaning finite field size in bits and must be of the size 256. (mcarrickscott 2018). Firstly, we will look at the steps needed to generate a valid public key. Without a working public key messages would not be encrypted in a secure way which is likely to result in the loss of data.

As seen previously, to generate such a public key, we firstly need to find two *large* primes of which, if of equal length we can be assured that $gcd(pq, (p-1)(q-1)) = 1$. The first step in this process is to generate a random *large* number, confirm the last digit is a 3 and finally, until the value is prime continuously add 4. This method for finding large primes was inspired by the work in (mcarrickscott 2018), assign this prime value to p, and repeat to calculate a prime q.

```
int n=PUB.n.getlen()/2;
FF p1=new FF(n);
for (;;) // loop indefinitely until breaks
{
        PUB.p.random(rng);
        while (PUB.p.lastbits(2)!=3){
                PUB.p.inc(1);
        }
        while (!FF.prime(PUB.p,rng)){
                PUB.p.inc(4);
        }
        break;
}
```

Figure 2.24: Generation of a large prime

The first value of the public key, n can now be calculated by multiplying p and q. The final step is to calculate a generator g, it is known that given two large primes a and b the generator will be valid given $g = (a^n + 1) * b^n \ mod \ n^2$.

```
//generate g
FF n2 = new FF(FF.FFLEN);
n2 = FF.mul(PUB.n, PUB.n);
FF atemp = new FF(FF.FFLEN);
atemp.set(0);
atemp.add4(PUB.a);
atemp.mul(PUB.n);
atemp.inc(1);
FF btemp = new FF(FF.FFLEN);
btemp.set(0);
btemp.add4(PUB.b);
FF ntemp = new FF(FF.FFLEN);
ntemp.set(0);
ntemp.add2(PUB.n);
btemp.pow(ntemp, n2);
atemp.modsqr(btemp, n2);
PUB.g = atemp;
```

Figure 2.25: Generation of a valid generator g

Having generated a secure public key (n,g), integers can now be encrypted successfully, however without a way of retrieving the original value the system would be flawed, hence the need to generate a private key which operates as the mathematical inverse of the public key.

```
PRIV.p.dec(1);
PRIV.q.dec(1);
FF lambda = new FF(2);
lambda = FF.mul(PRIV.p, PRIV.q);
BigInteger gcd = new BigInteger(FF.gcd(PRIV.p, PRIV.q).toString(),16);
BigInteger biglambda = new BigInteger(lambda.toString(),16);
biglambda = biglambda.divide(gcd);
```

Figure 2.26: Calculating λ

The first half of the key λ is calculated as the $lcm(p-1, q-1)$ which can be calculated using the formula $((p-1)*(q-1))/(gcd(p-1, q-1))$. An incredibly efficient method used for calculating the gcd is the Euclidean algorithm. To calculate the gcd of two integers x and y in this way. It is first needed to check if y is equal to zero, if so, then return x as gcd otherwise, call the method again with parameters now (y, x%y).

Secondly, the value of μ needs to be calculated . Creating two new FF variables of length 4 namely "l4" and "u" and assigning l4 the value of λ, u the value of g, taken from the public key. By performing $u^{l4}\ mod\ n^2$ (seen as "u.pow(l4,n2)") and subtracting one(assigning this value to the variable u). The remaining step is to divide by n to get the value of L(x) when x = $u^{l4}\ mod\ n^2$. As the cryptolibrary in use has no function defined for the division of FF by another FF object, it is necessary to take u as a string, defining a new BigInteger variable. As u was in the format of hexadecimal and BigInteger only accepts integer values, it is needed to convert the hexadecimal value to an integer value using the parameters "(s,16)", following the same procedure for converting the value of n into a BigInteger.

```
FF l4 = new FF(4);
l4.set(0);
l4.add2(lambda);
FF u = new FF(4);
u.set(0);
u.add(PUB.g);
u.pow(l4, n2);
u.dec(1);
String s = "";
s = u.toString();
BigInteger b = new BigInteger(s,16);
String s2 = PUB.n.toString();
BigInteger b2 = new BigInteger(s2,16);
b = b.divide(b2);
b = b.modInverse(b2);
BigInteger mu = new BigInteger("0");
mu = mu.add(b);
```

Figure 2.27: Calculating μ

```
int n = c.getlen();
FF gmodn2 = new FF(n/2);
gmodn2.set(0);
gmodn2.add(PUB.g);
gmodn2.power(m, PUB.n2);
FF tempn = new FF(n/2);
tempn.set(0);
tempn.add2(PUB.n);
FF rmodn2 = new FF(n/2);
Random ran = new Random();
int random = ran.nextInt(5000) +500;
rmodn2.set(random);
rmodn2.pow(tempn, PUB.n2);
gmodn2 = FF.mul(gmodn2, rmodn2);
FF tempn2 = new FF(n);
tempn2.set(0);
tempn2.add2(PUB.n2);
gmodn2.mod(tempn2);
String ans = gmodn2.toString();
BigInteger b = new BigInteger(ans,16);
c.add(gmodn2);
FF cshort = new FF(4);
cshort.set(0);
cshort.add12(c);
```

Figure 2.28: Encryption of an integer

It is now possible to divide these two values as desired. The final step to obtain μ is to find the modular inverse of $b \ mod \ n$, using Java's BigInteger class this is easily computable. The value of μ has now been found and a private key has been generated capable of decrypting any ciphertext calculated using the matching public key.

The next crucial step of the system is encrypting an integer, following the previously described formula on page 7, it is know to encrypt an integer a random value r must be generated and using the parameters of the public key n and g together with the integer m (value needed to be encrypted). Firstly, it is needed to create a temporary local FF variable storing the value of the public key composites, n^2 and generate a random integer r. Calculating $g^m \ mod \ n^2$, followed by computing $r^n \ mod \ n^2$. Penultimately multiplying these two results and calculating the new value $mod \ n^2$.

Having encrypted the integer, it is vital that the retrieval of the original integer is possible, providing the private key is known. Beginning in a similar manner to encryption, creating temporary local instances of the private key components. Start by calculating $c' := c^\lambda \ mod \ n^2$ and subtracting one, where c is the ciphertext to be decrypted, taking the string value of both c' and n, create two new BigInteger variables cbig and nbig, assigning them the values of these strings once converted from hexadecimal to decimal. It is now possible to divide cbig

```
FF ctemp = new FF(8);
ctemp.set(0);
ctemp.add(c);
FF templambda = new FF(8);
templambda.set(0);
templambda.add4(priv.lambda);
FF tempn2 = new FF(8);
tempn2.set(0);
tempn2.add2(priv.n2);
ctemp.pow(templambda, tempn2);
ctemp.dec(1);
String cstr = ctemp.toString();
BigInteger cbig = new BigInteger(cstr,16);
String nstr = priv.n1.toString();
BigInteger nbig = new BigInteger(nstr, 16);
cbig = cbig.divide(nbig);
cbig = cbig.multiply(priv.mu);
cbig = cbig.mod(nbig);
String cst = cbig.toString();
m = cbig;

return m;
```

Figure 2.29: Decryption of an integer

by nbig. Finally, multiplying the previous result by μ and finding the result modulo n will return the initial value of m, which is one of the key requirements of this project.

```
int sha=Palliers.HASH_TYPE;
RAND rng=new RAND();
rng.clean();
byte[] RAW=new byte[100];
for (int i=0;i<100;i++) RAW[i]=(byte)(i);
int RFS=Palliers.RFS;
byte[] C=new byte[RFS];
rng.seed(100,RAW);
byte[] Msg = message.getBytes();
byte[] EN=Palliers.OAEP_ENCODE(sha,Msg,rng,null);
Palliers.encrypttest(pub, EN, C);
String msg =Palliers.getDecodedString(C);
return msg;
```

Figure 2.30: Encryption of a string

The encryption of a string is a more complicated matter. As previously mentioned a message must first be converted into a byte array, following this, the array is then encoded for the purpose of making it easier to store. Encrypting this encoded byte array is performed in a very similar way as an integer, the only alteration is where the encoded byte array is placed into an FF class (as opposed to an integer in the method above). Following this, the application of a Base64 encoder is needed to generate a string which can now be stored within the database without losing the integrity of the data. The steps of decrypting a message, upon retrieval from the database can be performed in the following way. Firstly, through using a Base64 decoder to convert the encrypted string into a byte array. Secondly, decrypt this byte array using a similar approach to the decryption of an integer, with the same alterations as encrypting a string.

The result is a decrypted byte array as expected. Finally, through decoding the byte array a string is generated from the byte array, returning the initial message before any encoding or encryption took place.

```
int sha=Palliers.HASH_TYPE;
int RFS=Palliers.RFS;
byte[] C=new byte[RFS];
byte[] ML=new byte[RFS];
byte[] by = Palliers.getEncodedByteArray(message);
Palliers.decrypttest(priv,by,ML);
byte[] MSG=Palliers.OAEP_DECODE(sha,null,ML);
String mess=new String(MSG);

return mess;
```

Figure 2.31: Decryption of a string

2.5 User interface

When designing the user interface several things were considered to make it clear and concise, consistent in style, flow well and overall appear professional. The first decision was to find an inoffensive colour scheme, one which is not flamboyant. The options were narrowed down to were dark green and white, colours linked with doctors and healthcare or blue and white, colours of NHS Wales. In this scenario, although both are appropriate it is easier to see black writing on the light blue background so this was chosen as the colour scheme in the end.

In terms of consistency, despite each form having its own functionalities and in need of slightly different designs, where possible the spine of the system tries to be consistent, through the continued use of the same font (Serif, size 16). Moreover the layout is consistently structured in a way such that everything is both logical and familiar to the user. Examples of this are seen with the layout of navigational buttons, back button located at bottom left, proceed button located at bottom right.

Furthermore, the order of objectives for each page starts at the top and descents down the page. Where applicable symmetry and alignment of components is integrated in the systems design. This can be seen where related textfields, labels and buttons are aligned with one another. The use of images(royalty free) in areas where suitable, such as the login form or a user's inbox add to the design in a way such that JLabels are not needed all over a page.

2.6 Project management

Throughout the project I have tried to stay on top of my work to provide myself with enough time to complete the Report and system without adding to the abundance of stress which comes naturally with such an important piece of work. To assist myself with this, I designed a rough timescale of dates(Appendix B, Figure B.1) by which I would like to have completed certain aspects of the project. This was immensely helpful for splitting up work into a more manageable amount to do on a weekly or fortnightly basis, which in turn kept me motivated seeing progress being made.

There were occasions when an insufficient amount of contingency time was perhaps planned, for example the additional time needed over and above what was expected, to complete all the research required to be completely satisfied by the work produced. In particular, these problems arose in the weeks leading up to the Christmas break when the literature review was due, which clashed with deadlines in other modules. Part of this was down to myself not having read as much literature as I perhaps should have by this time. However, by re-evaluating the free time i had allocated myself allowed a comprehensive and extensive literature review to be written.

As always when programming, unexpected errors can crop up or the implementation of a complex algorithm can take longer than previously expected. This is something I am not unfamiliar with and because of this I planned to finish all programming two weeks before the demonstration, which would mean I would have two additional weeks. Where testing for any bugs could be made ,in addition to any extra time needed to solve complex issues which I thought could arise when developing the system. Before programming commenced, I broke the system down into the spine of what was needed to solve the problem faced, knowing that once the fundamental algorithms were in place it would be easier to implement additional features. I feel like this was an appropriate approach to the problem as it ensured, at the very least, the basics were there. Also having the opportunity for features which were not necessarily essential to the design but, added to the overall effect of being a potential system which could be integrated into reality.

The use of a journal allowed me to stay organised throughout the course of the project. It was enormously beneficial to note down thoughts on how to go about complex algorithms, layout of forms with the data flow. Planning in pseudo code made it much easier when it came to actually program the algorithm as I would already have all the steps thought out. Additionally, through maintaining a 'to do' list ensured I did not forget about ideas or little tweaks which were needed or would be

beneficial to change. Although at stages there seemed to be an overwhelming amount of jobs to do, through the aforementioned organisation with contingency times it was in the end manageable to complete everything to a level which I was happy with. Through the use of a journal it also allowed me to keep a note of any papers which I found useful or owed an intellectual debt to, which I would later need to place in an ongoing bibliography (continued from the literature review) at the of a day of working.

2.7 Results and evaluation

2.7.1 Functionality

The overall functionality of the system developed is considered excellent, it provides all of the functions which were described in the system requirements. It provides the foundation of a system which can easily be expanded in the future, to obtain more significant analysis. These features can be added through the manipulation of algorithms already designed and implementing them under new circumstances.

2.7.2 User interface

The primary requirement for the user interface was to produce a simple, effective and professional design. The forms displayed are all of a fixed size, they could have been designed differently, allowing resizing, which positions components in proportion to the form size. However, the executive decision of fixing the size of all frames and thus the distance between all components gave a professional appearance, which was one of the requirements.

2.7.3 Performance

Generally, the performance of the system is consistently of a high standard, radiating the feeling of a high quality, professional system. Upon running the software, navigation through the system is both smooth and efficient. The only notable change in performance is seen when a doctor decrypts a patients record which can take a few seconds to load. Despite this I do not feel like this has a negative impact on the overall image of the system.

2.7.4 Robustness

A vast and extensive range of tests were put in place in order to test the robustness of the software developed. Through the use of 'try catch' statements which are implemented throughout the system, enables the system to check the input and if it will cause an error to the system then 'catch' it. Once an error has been caught a helpful error message is produced by the system, it should be noted that the error message does not stop the system functioning.

A common security flaw when databases are used in conjunction with a system is SQL injection. This is when an SQL command is entered into an input field such as a textbox, with the intention of accessing or manipulating the data stored, usually with

malicious intent. To ensure the system developed is resistant to this form of attack, prepared statements are used. In this scenario, the SQL command intended to be computed is computed separately to the input from the user. Through the application of the two methods mentioned above, it is to the best of the authors knowledge that the system will not crash for any user input.

The results at each step for the addition of two ciphertexts whose values prior to encryption were 16 and 43:

1. Encrypt plaintexts using the public key(n,g)

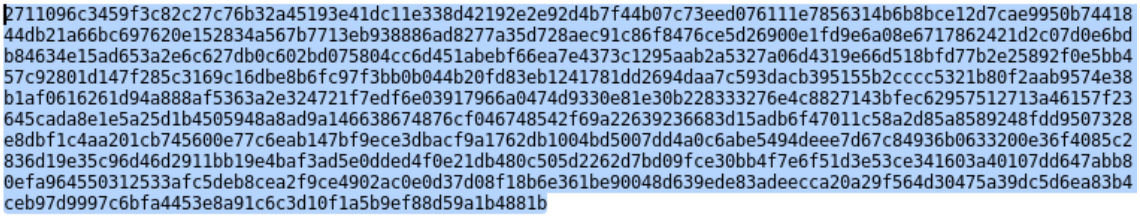

Figure 2.32: Ciphertext generated for plaintext 16

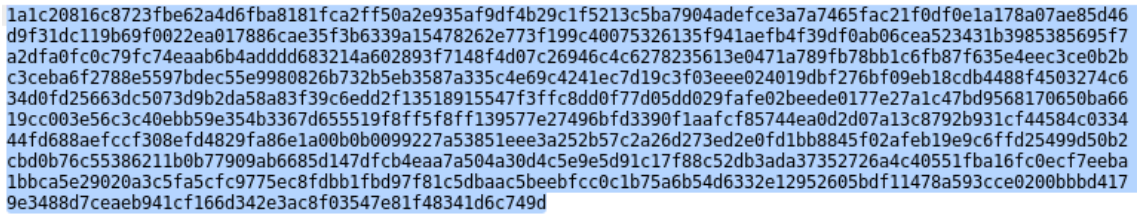

Figure 2.33: Ciphertext generated for plaintext 43

2. Multiplication of these two ciphertexts, assign this value c.

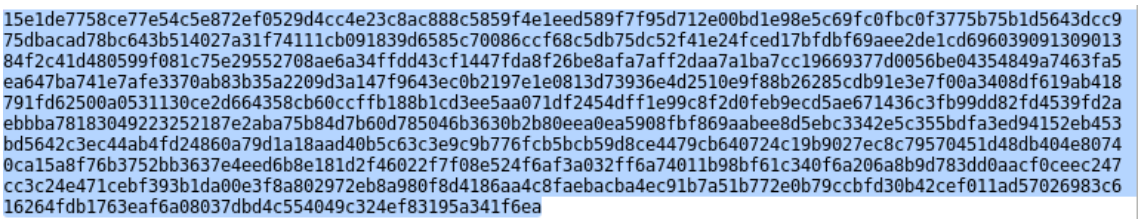

Figure 2.34: Ciphertext generated through multiplying figures 2.32 and 2.33

3. Upon decrypting c using the private key(λ, μ) returns the result 59, which is indeed the summation of the two original plaintexts as desired.

Evidence that the system developed is indeed indistinguishable can be seen in figures 2.35 - 2.37. Where, even when it is known the plaintexts of the three ciphertexts are 20, 20 and 3. There is no feasible way to be more than $33.\overline{3}\%$ certain that any one of them represents the value 3 i.e. a pure guess. In addition, we can see that the system is also probabilistic through the fact there is no feasible way to be more than $33.\overline{3}\%$ certain which two ciphertexts hold the value 20.

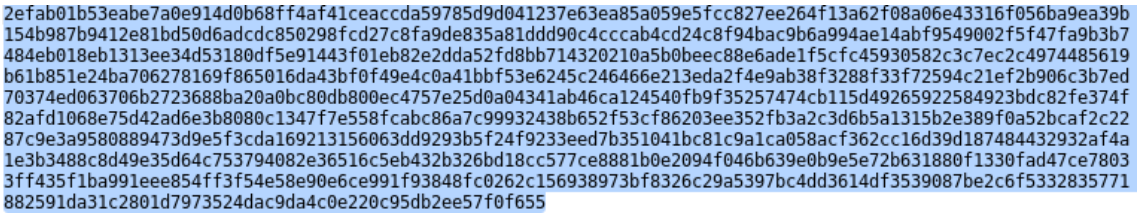

Figure 2.35: Ciphertext 1

Figure 2.36: Ciphertext 2

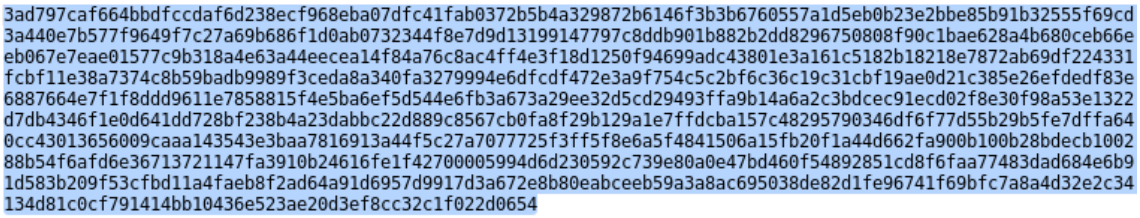

Figure 2.37: Ciphertext 3

The results at each stage of encrypting and decrypting the message "Hello World!" can be observed in figure 2.38-2.40.

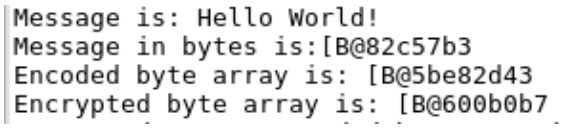

Figure 2.38: Deconstruction and encryption of a string.

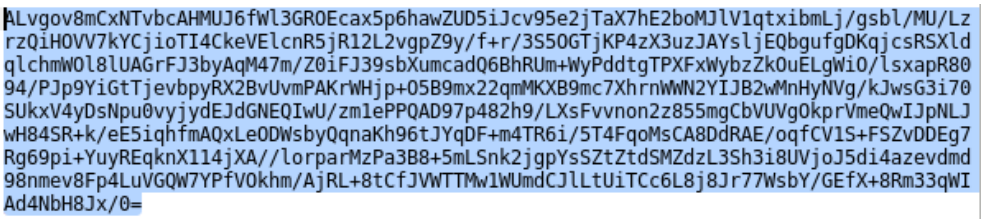

Figure 2.39: Base64 encoder applied to return a string which can be stored in database.

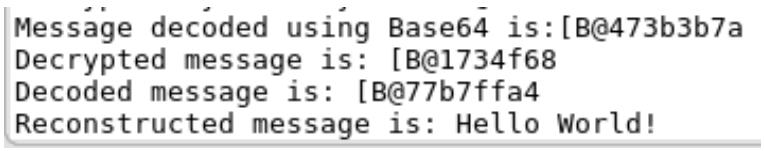

Figure 2.40: Decryption and reconstruction of a string.

When analysing the BPW scheme, whether it be the number of ratings or the rating of different support books, graphs can be produced by the system representing the results following decryption. An example of the graphs which can be produced by the system are seen in figures 2.41 and 2.42. I feel that a visual representation of the results in this scenario are much easier and cleaner to see and make better use of the results than just the numbers on their own. It should be noted that the use of a Java library to generate graphs could improve this further.

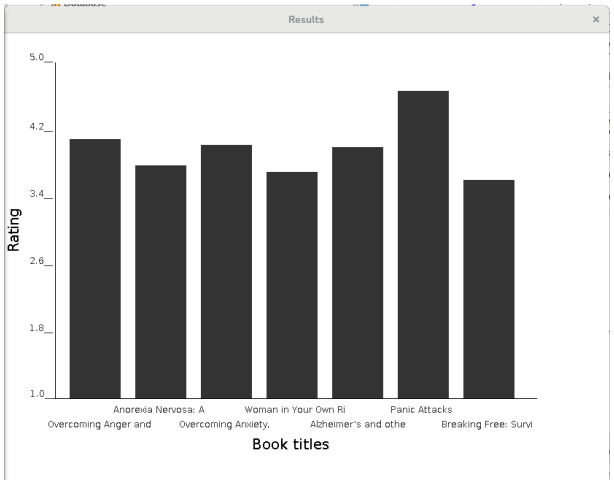

Figure 2.41: Output generated showing rating for books selected.

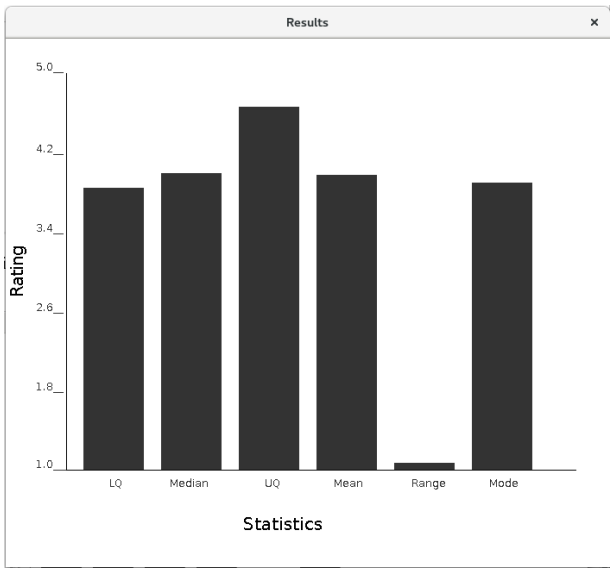

Figure 2.42: Output generated showing a mixture of statistical analysis

Chapter 3

3.1 Discussion

When looking back at the system as a whole, there are many great things which it accomplishes. There are also areas which it falls short or could be improved. Beginning with looking at some of it's flaws, when reviewing the structure of the system, in particular the database, there is a practicality issue when it comes to the filters table. When the table is for only two filters (gender and smoking habits) I can see no problems with how it handles this data and believe it to be practical and efficient for the current system and for showing the potential use of the data. Having said this there is an obvious problem which arises when more filters are added, if say we wish to analyse if the patient in question is over fifty and if their blood group was A-negative. In this scenario we would now need sixteen records in the database for four filters as opposed to just four records for two filters. This exponential growth would be detrimental to the efficiency and compactness of the system when in depth analysis is needed as there are a vast number of factors which could potentially be involved in the analysis. A potential solution to this problem, if the system was to be taken further in the future or if I had more time available, would be to remove the filters table and instead go through the list of patients medical records and analyse each one for the applied filters. This in turn would come with its own impracticality of taking an increased time when performed for the population of a country, instead of a sample of one thousand. This boils down to the executive decision of which is more important for the system, speed or size.

This leads on to a question about the efficiency of the system and how we go about storing data. Currently, the method for storing strings within the database is extensive and exhaustive on the system. Although it is necessary to encrypt certain values using Pailliers' it is not necessarily needed to convert a string to bytes and encrypt the byte array using Paillier, although we know this makes the resulting ciphertext

indistinguishable and is still very much manageable for the current structure. This would be an area which may need to adapt as the system expanded. A potential solution to this could be achieved by simply using a Java cryptolibrary to encrypt and decrypt strings to and from the system.

Additionally, with medicine being such a broad area there will always be room to integrate more features which can offer a deeper, more insightful analysis into the causes of a disease or improvement to treatments offered. Upon developing this system further, the implementation of analysis into other areas can be made, such as how successful treatments are with patients. Furthermore, some of the data collected is used only by doctors when they are inspecting a patient, having the data available it might be seen as wasteful to not look for correlations in data using this information, an example being is there a link between heart transplant and cardiac arrest.

Regardless of the odd drawback or area which could be improved in the system, the overall system does offer a solution to the initial problem set out. The first requirement was to provide a means of storing patient data in a secure manner, this has certainly been achieved, through the use of a semantically secure cryptosystem, Paillier to encrypt data and stored in a database. The data stored used a homomorphic cryptosystem and thus allows data analysis to be computed without revealing the underlying values. This can be evidenced through functions within the system where, for example, health experts can analyse the drugs and the volume to which they are being prescribed showing the overall cost to the NHS. This information could in turn assist in determining the best policy for both the Welsh Government (money saved can be spent in other areas such as education) and the public's health. Analysis can also be seen to provide evidence of the effectiveness of the BPW scheme which offers non-drug treatments and support to mental health suffers. This analysis could be integral to the expansion of this scheme across the rest of the U.K. providing additional support for people who need it most. On the doctors side of the system, it has many features which should be capable of assisting a doctor with a diagnosis of a patient through the use of retrieving a patients medical history in a more efficient way, this also removes the chance of a patient leaving out information which they do not think is relevant but in reality could be influential.

Another thing in which the system achieves well, is the implementation of many features which, although may go unnoticed for being small things add to the overall practicality of the system, some of these features include but are not limited to; the translator, being designed for primary use in Wales, I believe it was purposeful to include a translation for those who are not fully bilingual or even to promote the

use of Welsh speaking. This could also help patients feel more comfortable if they are able to speak in their first language; password recovery feature where the user must answer a security question which they choose upon registration in order to successfully change a password. Procedure which is familiar to the majority if not all of the population in this day and age; a range of statistical analysis can be computed, showing the homomorphic versatility of the system, having been implemented already it would be simple to re implement the same analysis in a different scenario. Given any mathematical analysis consisting of unlimited addition and multiplication of up to one plaintext integer, this operation can be computed from the functions already defined; high testing coverage shows a fully functioning system, which is essentially robust and handles errors in a secure manner such that the system does not crash from bad user input. For obvious reasons it was important to not have a system containing bugs, in particular for such a vital service for which this system is intended.

3.2 Conclusion

This Report provides written evidence of a proven, functioning solution to the need for a secure way of collating and analysing patient data, to provide evidence and information on effectiveness of policy decisions and prescribed treatment. It demonstrates how the software developed is capable of assisting health professionals with their day to day work, treatment and diagnosis of patients. It also shows how the software could be used as a tool for the Government to access evidence to support their policy decisions and make the best use of the funding they have available for the NHS. The expected success of implementing the solution identified can be evidenced through the approach taken for storing and displaying an up to date medical record for all patients and the fact that the system permits analysis of patient data without compromising anonymity or integrity, through the application of Pailliers' cryptosystem. Presently, the conclusions which can be drawn from the analysis are limited. However, the algorithms implemented can be reused, meaning future analytical features can be added easily. Additional features integrated into the system could provide more critical and useful analysis for more complex questions.

Appendix A

Additional reading on encryption schemes

The key generation algorithm for RSA, as shown in (Mohapatra 2000) is as follows:

- Select two large primes p and q, the prime is considered large if the size ranges from 2^{1024} up to 2^{2048} (Erra & Grenier 2014)
- Let n := $p \cdot q$
- $\phi(n) := (p-1) \cdot (q-1)$
- Calculate e, such that $0 < e < \phi(n)$
- Calculate d, such that d = $e^{-1} mod \phi(n)$. This can be calculated using the extended Euclidean algorithm in a way such that $d \cdot e \equiv 1 mod \phi(n)$
- pub(n, e), priv(n, d)

This process only works if e is coprime to $\Phi(n)$, as without this no multiplicative modular inverse of e exists. The encryption of a message m, is given by:

$$\text{c} := m^e \text{ mod n}$$

while the decryption of a ciphertext c, is given by:

$$\text{m} := c^d \text{ mod n}$$

The proof of correctness of RSA as proved in (Mohapatra 2000) is shown by:

- $e \cdot d = k \cdot \phi(n) + 1$ for some k
- = $(m^d)^e$mod n, as m = c^d mod n
- = m^{ed} mod n, by the laws of modulus
- = $m^{k \cdot \phi(n)+1}$ mod n, by substitution of $e \cdot d$
- = $(m^{\phi(n)})^k \cdot m$ mod n, by the laws of modulus
- = $1^k \cdot m$ mod n, two cases, 1) when m is coprime to n we use Eulers theorem, 2) m is not coprime to n, in which case the Chinese remainder theorem can be used.

- $=$ m mod n, as $1^k = 1$ for any k.

The key generation for ElGamal as described by (Ryan 2008) is performed as follows:

- Find large prime p.
- generator α of a large, prime subgroup G of $\mathbb{Z}_p^*$of order q, q|(p-1) (α is a generator of G if every element of G can be expressed as a power of α
- choose a random k ϵ $\mathbb{Z}_p$
- $\beta := \alpha^k$ mod p
- pub(p,α) and priv(β, k)

The encryption of a a message m yields a pair of terms, computed by:

$$c := (y_1, y_2) := (\alpha^r, m \cdot \beta^r) \text{ mod p}$$

where r is selected randomly from $\mathbb{Z}_p$ then the message can be decrypted by applying the private key such that:

$$m := y_2 / y_1^k \text{ mod p}$$

decryption scheme as proved in (of Mount Union n.d.) can be done in the following manner:

- y_2/y_1^k mod p
- $= M \cdot \beta^r (y_1^k)^{-1}$ mod p
- $= M \cdot \alpha^{rk} (\alpha^{kr})^{-1}$ mod p
- $=$ M

Proof of correctness for multiplying homomorphically:

- Let $c_1 := (y_1, y_2) := (\alpha^{r_1}, m_1 \cdot \beta^{r_1})$ mod p
- Let $c_2 := (y_3, y_4) := (\alpha^{r_2}, m_2 \cdot \beta^{r_2})$ mod p
- Compute $c_3 := c_1 \cdot c_2$
- $c_3 := (y_5, y_6) := (\alpha^{r_1} \cdot \alpha^{r_2}, m_1 \cdot m_2 \cdot \beta^{r_1} \cdot \beta^{r_2})$ mod p
- $c_3 := (y_5, y_6) := (\alpha^{r_1 + r_2}, m_1 \cdot m_2 \cdot \beta^{r_1 + r_2})$ mod p
- Let $r' := r_1 + r_2$ and $M := m_1 \cdot m_2$
- Then $c_3 := (y_5, y_6) := (\alpha^{r'}, M \cdot \beta^{r'})$ mod p

Which can be decrypted, as it is in the standard from for decrypting a ciphertext in ElGamal.

Appendix B

Planning

Objective	Duration	Start date	End date	Contingency time allowed
Background reading	14 days	18/09/17	01/10/17	0 days
Project proposal	7 days	02/10/17	08/10/17	3 days
Literature review	14 days	16/10/17	29/10/17	4 days
Ethical review	7 days	27/10/17	03/11/17	0 days
Inspection	14 days	10/11/17	24/11/17	2-7 days
Programming	56 days	08/01/18	04/03/18	7 days
Testing	7 days	05/03/18	11/03/18	7 days
Demonstration	7 days	12/03/18	18/03/18	0-5 days

Figure B.1: Time management

Appendix C

Running the system

Zip File Structure

- Electronic copy of Report
- Source code
- Literature review
- Address to Git Repository
- sqlite-jdbc.jar

Run From Eclipse

To run the system an external .jar is needed, sqlite-jdbc must be added to the build path. Once that is done:

- Import the project into Eclipse using the import wizard
- Ensure JRE, JUnit4 and sqlite-jdbc jars are all included in the build path
- Run the Login.java class

Existing users

Using the login credentials displayed below will allow the user to access both sides of the system and use all the available features.

- Doctor: Username: "arj590", Password: "jonesyy3"
- NHS: Username: "rlr0809", Password: "Rachel1"

Bibliography

Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C. A. & Strand, M. (2015), 'A guide to fully homomorphic encryption.', *IACR Cryptology ePrint Archive* **2015**, 1192.

Coppersmith, D. (1997), 'Small solutions to polynomial equations, and low exponent rsa vulnerabilities', *Journal of Cryptology* **10**(4), 233–260.

Damgård, I. & Jurik, M. (2001), A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system, *in* 'International Workshop on Public Key Cryptography', Springer, pp. 119–136.

DÜLGERLER, M. & SARISAKAL, M. N. (2003), 'A secure e-mail application using the elgamal algorithm: Md message controller', *IU-Journal of Electrical & Electronics Engineering* **3**(1), 801–806.

Erra, R. & Grenier, C. (2014), 'How to compute rsa keys'.

Gentry, C. (2009), *A fully homomorphic encryption scheme*, Stanford University.

Government, W. (2017), 'Nhs expenditure programme budgets: 2015-16'.
URL: *http://gov.wales/docs/statistics/2017/170426-nhs-expenditure-programme-budgets-2015-16-en.pdf*

Groves, P., Kayyali, B., Knott, D. & Van Kuiken, S. (2013), 'The 'big data'revolution in healthcare', *McKinsey Quarterly* **2**, 3.

mcarrickscott (2018), 'The apache milagro cryptographic library'.
URL: *https* : *//github.com/milagro − crypto/amcl*

Mohapatra, P. K. (2000), 'Public key cryptography', *Crossroads* **7**(1), 14–22.

NHS (2012), 'Booklist - books prescription wales 2011'.
URL: *http* : *//www.nhsdirect.wales.nhs.uk/pdfs/WG*%2016567%20*A*4%20*Booklist*%20*WEE*

NHS (2017), 'Encryption good practice guide'.

of Mount Union, U. (n.d.), 'The el gamal cryptosystem'.

Paillier, P. (1999), Public-key cryptosystems based on composite degree residuosity classes, *in* 'International Conference on the Theory and Applications of Cryptographic Techniques', Springer, pp. 223–238.

Ryan, P. Y. (2008), 'Prêt à voter with paillier encryption', *Mathematical and Computer Modelling* **48**(9-10), 1646–1662.

Simmons, G. J. (1979), 'Symmetric and asymmetric encryption', *ACM Computing Surveys (CSUR)* **11**(4), 305–330.

Tebaa, M., El Hajji, S. & El Ghazi, A. (2012), Homomorphic encryption applied to the cloud computing security, *in* 'Proceedings of the World Congress on Engineering', Vol. 1, pp. 4–6.

Wang, G., Lu, R. & Huang, C. (2015), Pslp: Privacy-preserving single-layer perceptron learning for e-healthcare, *in* 'Information, Communications and Signal Processing (ICICS), 2015 10th International Conference on', IEEE, pp. 1–5.

All images used within in the system are royalty free. The original image can be found at https://www.gettyimages.co.uk/license/810117358