

De-identification of personal information for use in
software testing to ensure compliance with the
Protection of Personal Information Act

Stephen John Mark

March 16, 2018

Submitted in partial fulfillment

of the requirements of the degree of

MASTER OF SCIENCE

of Rhodes University

Grahamstown, South Africa

Abstract

Encryption of Personally Identifiable Information stored in a Structured Query Language Database has been difficult for a long time. This is owing to block-cipher encryption algorithms changing the length and type of the input data when encrypted, which cannot subsequently be stored in the database without altering its structure.

As the enactment of the South African Protection of Personal Information Act, No 4 of 2013 (POPI), was set in motion with the appointment of the Information Regulators Office in December 2016, South African companies are intensely focused on implementing compliance strategies and processes. The legislation, promulgated in 2013, encompasses the processing and storage of personally identifiable information (PII), ensuring that corporations act responsibly when collecting, storing and using individuals' personal data. The Act comprises eight broad conditions that will become legislation once the new Information Regulator's office is fully equipped to carry out their duties. POPI requires that individuals' data should be kept confidential from all but those who specifically have permission to access the data. This means that not all members of IT teams should have access to the data unless it has been de-identified.

This study tests an implementation of the Fixed Feistel 1 algorithm from the National Institute of Standards and Technology (NIST) "Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation : Methods for Format-Preserving Encryption" using the LibFFX Python library. The Python scripting language was used for the experiments.

The research shows that it is indeed possible to encrypt data in a Structured Query Language Database without changing the database schema using the new Format-Preserving encryption technique from NIST800-38G. Quality Assurance software testers can then run their full set of tests on the encrypted database. There is no reduction of encryption strength when using the FF1 encryption technique, compared to the underlying AES-128 encryption algorithm. It further shows that the utility of the data is not lost once it is encrypted.

Acknowledgments and thanks: Professor George Wells - for being my supervisor throughout this research. This work was much improved as a result of his attention to detail.

Ann Mark and my children - for putting up with the long hours of research.

Bayport Financial Services - for supplying the data for the experiments.

Karen Julius - for QA testing Experiment-1 and Experiment-2.

Kevin Dyer [23] - for supplying the LibFTE python library on Github.

Qualica Technologies: Francis Viviers, Fanie Health, Phillip Smith - for assistance with the Experiment-2 Python script.

Arno Rossouw - for assistance with the Experiment-1 Python script.

Contents

1	Introduction	1
1.1	Research Aim	3
1.2	Objective	4
1.3	Approach	4
1.4	Assumptions	5
2	Literature Review	7
2.1	Introduction	7
2.2	Legislative Requirements	8
2.2.1	South Africa's Protection of Personal Information Act	8
2.2.2	Data Protection Act of the United Kingdom 1998	11
2.2.3	Data Protection in Botswana	11
2.2.4	Data Protection in Colombia	12
2.2.5	General Data Protection Regulation	12
2.3	Possible Encryption Solutions	14
2.3.1	Hashing	15
2.3.2	Tokenisation	16
2.3.3	Data Encryption Standard	16
2.3.4	Advanced Encryption Standard	17
2.3.5	Extract, Transform and Load	17
2.4	Format-Preserving Encryption	18
2.4.1	Black and Rogaway (2002)	19
2.4.2	Bellare, Ristenpart, Rogaway and Stegers (2009)	25
2.4.3	Bellare, Rogaway and Spies (2010)	27
2.4.4	NIST Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation (2016)	27
2.4.5	Weiss, Rozenberg and Barham (2015)	34
2.4.6	New Analysis of FF3 (2017)	36
2.5	Conclusion	36
3	Experimental Design	37
3.1	Introduction	37

- 3.2 Tool Selection 38
 - 3.2.1 Commercial Software Review 38
 - 3.2.2 Evaluation of Available Open-source Software Tools 40
- 3.3 Implementation 42
- 3.4 Experiment-1: Colombia 44
 - 3.4.1 Procedure for running the de-identification script 45
 - 3.4.2 Output from the first run of the de-identification script 47
 - 3.4.3 Challenges 51
 - 3.4.4 Program statistics 54
 - 3.4.5 QA department testing 54
- 3.5 Experiment-2: Botswana Novalend 54
 - 3.5.1 Procedure for running the de-identification script 55
 - 3.5.2 Output from the first run of the de-identification script 55
 - 3.5.3 Challenges 61
 - 3.5.4 Program statistics 61
 - 3.5.5 QA department testing 62
- 4 Analysis and Discussion of Results 63**
 - 4.1 Introduction 63
 - 4.1.1 Encrypted block-size 64
 - 4.1.2 Encrypting email addresses 67
 - 4.1.3 Special characters, accented characters, or other odd characters found
in names 68
 - 4.1.4 Usernames 70
 - 4.1.5 Search for Client by Name 70
 - 4.1.6 Dates 70
 - 4.1.7 Integer de-identification 71
 - 4.1.8 On-the-Fly de-identification 71
 - 4.1.9 Key principles in encrypting data for de-identification 72
 - 4.1.10 Current practice 72
 - 4.1.11 Future practice 72
 - 4.2 Test results - Experiment-1 73
 - 4.2.1 Tests Performed: 73

4.2.2	Login to System:	73
4.2.3	View system inbox	74
4.2.4	Search for Client by Name	74
4.2.5	Search for Client by Customer Number	74
4.2.6	View Client Details	74
4.2.7	Bank Details	74
4.2.8	Add / Edit Address	75
4.2.9	View Account Statement for a Client	75
4.2.10	Print Disbursement Details for a Client	75
4.2.11	Capture Documentation	75
4.2.12	Verify Employer Contacts	75
4.3	Test results - Experiment-2	77
4.3.1	Tests Performed:	77
4.3.2	Login Section	77
4.3.3	Inbox Section	77
4.3.4	Dashboard	79
4.3.5	Launch Process	80
4.3.6	Find a Client	82
4.3.7	Search First Name	83
4.3.8	New Loan	83
4.3.9	Edit Client	83
4.3.10	Document Upload	83
4.4	Summary	83
5	Conclusion	85
5.1	Introduction	85
5.2	Chapter Summary	86
5.3	Research Goals	86
5.4	Significance	87
5.5	Future Work	88
5.6	Limitations of the Research	89

List of Figures

1 Feistel Network from Mat Green [34] 25

2 Feistel Rounds from Rogaway [84] 31

3 Process of de-identifying the PII data 43

4 QA staff access to QA Data store 45

5 Experiment-1: Entity Relationship Diagram 48

6 Experiment-1: Initial Encryption 48

7 Experiment-1: null values 49

8 Experiment-1: Employer Referral 50

9 Experiment-1: All Customer Data Encrypted 52

10 Experiment-2: Masterdata Schema ERD 56

11 Experiment-2: First Run 57

12 Experiment-2: All PII de-identified 58

13 Experiment-2: Client Details 59

14 Experiment-2: Manage Users 60

15 Experiment-2: Dashboard de-identified 61

16 Experiment-1: Search for Client by Customer Number 65

17 Experiment-1: View Client Details 65

18 Experiment-2: Disbursement Client Details 66

19 Experiment-1: Address 68

20 Experiment-1: View Account Statement 69

21 Experiment-1: Verify Employer Contacts 76

22 Experiment-2: Dashboard - Your Most Recent Clients 79

23 Experiment-2 - Launch Process - Loan Quote 81

24 Experiment-2: Find a Client -All Details de-identified 82

List of Tables

1 Experiment-2: Encrypted Names 67

2 Experiment-2: Encrypted Names with Capitalisation and numbers removed . . 67

3 Tests performed on Experiment-1 dataset 73

4 Tests performed on Experiment-2 dataset 77

List of Algorithms

1	Prefix Cipher algorithm from Black and Rogaway [8]	21
2	Cycle Walking Enciphering Algorithm from Black [8]	22
3	Encipher using a Feistel Network from Black [8]	24
4	FF1.Encrypt(K, T, X) Algorithm 1 taken from NIST800-38G [21]	35
5	Experiment-1: Python Code to perform the encryption	46
6	Experiment-1: Performing the encryption on a SQL table	47

1 Introduction

In many countries in the world the right to privacy is considered a fundamental human right. Storing personal information in databases has been common practice from the onset of information technology and the consequent processing of information about individuals. As a result, a fundamental element of privacy rights is information (or data) privacy, which encompasses an individual's ability to protect their personal information across the process of collection, use, disclosure and deletion of their personal information by third parties. To ensure responsible custodianship of personal information data protection laws were introduced and have been evolving world-wide over the last 30 years.

In South Africa the right to privacy is enshrined in the constitution, and the Protection of Personal Information (POPI) Act [83] of South Africa gives effect to this right. POPI was signed in November 2013 but is not yet effective as the date of commencement has not been set. The purpose of the Act is to protect privacy and ensure that companies behave ethically and responsibly when collecting, processing, storing, sharing and deleting personal information. The POPI Act reflects the principles in the European Union Data Privacy Directive, which informs the legislation in many European countries, and the Organisation for Economic Co-operation and Development (OECD) Guidelines. The POPI Act is similar to the United Kingdom's Data Protection Act (DPA) [33] that was enacted in 1998 and the new General Data Protection Regulation (GDPR) in the European Union (E.U.) that will be a requirement in the E.U. from March 2018.

POPI particularly affects organisations that hold 'valuable' personal information such as the Financial Services, Healthcare and Marketing sectors. The ramifications of the Act also impact the IT sector, in activities related to the development, testing and quality control of personal information databases. To comply with POPI, companies need to implement adequate measures and controls to prevent unauthorized people accessing personal information. "Unauthorized people" extends to employees within the company who do not specifically require access to a customer's personal data. This means that employees such as Development and Quality Assurance (QA) or testing teams will not be able to use databases containing personal information in their activities. According to the Act, de-identified or anonymised data is no longer personal information. Therefore, when processing any personal information that has been de-identified,

POPI does not apply. This makes anonymising or de-identifying data key to the activities of certain IT functions.

Development and QA / testing teams often show resistance when asked to de-identify databases (DB) containing personally identifiable information (PII). There are several reasons for this. In the past they have had access to this data and until they understand that they are potentially in contravention of the POPI Act they feel entitled to use the data as is. Once educated about POPI they accept that de-identification must be performed. More pertinently, reluctance to de-identifying information stems from perceived threats to the successful outcome of their tasks or responsibilities as a result of encrypting the data. Challenges emanating from the changes in the data schema to incorporate the de-identification process or the possibility of de-identified data sets being rendered unusable by the very nature of the de-identification process are real risks to consider, stemming in part, from historic encryption methods. The encryption methods used to de-identity the data must allow the Development and QA / testing teams to continue to perform their duties unhindered.

The feasibility of running a hash function on the fields that need to be de-identified was explored initially but rejected because this created a fixed length binary output that was significantly longer than the original field in the Structured Query Language (SQL) database table and was a different data type (i.e. binary data). As a result, the database schema would need to be changed, which was not an acceptable solution. In addition, hashes were not a human-friendly format as they were not easy for people to interpret. The popular MD5 algorithm produced a 128-bit output string but suffered from several vulnerabilities, which have led to the National Institute of Standards and Technology (NIST) recommending that it was no longer used. Further, rainbow tables can be used to effectively reverse the output from many popular hash algorithms. Security awareness project, Crackstation.net [18] has lookups for almost 15 billion MD5 and SHA1 hashes. Their database encompasses almost all known hashes, including: “LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin))” [18].

Format-preserving encryption (FPE) is a useful mechanism for encrypting a piece of data while maintaining the structure of the field with regard to the field length and the type of data.

In addition, FPE creates a unique identifier if the same key and tweak are used each time the encryption is performed. This means that an encrypted FPE field can be used as an index or a link in a database and also that encrypted data can be decrypted using the original key and tweak. FPE works well with credit card numbers because a 16 digit number is encoded to a new 16 digit number, compared to an MD5 hash that would create a 128-bit binary string. FPE typically uses a pre-defined secret key to ensure the fast generation of ciphertext. This offers a more viable solution to the de-identification of PII.

HPE Voltage [36] has developed a commercially available software solution using FPE that uses the NIST 800-38G standard encryption with both the FF1 Advanced Encryption Standard (AES) encryption technique and the FF3 AES encryption technique to encrypt data in a way that does not alter the data format. Researchers at Voltage developed the FF1 AES encryption techniques that was later ratified by NIST in the NIST800-38G standard. Unfortunately, the FF3 algorithm was withdrawn by NIST in 2017.

The result is a strong encryption scheme that allows data to be encrypted with minimal modifications to the way that existing applications and databases operate [36]. A possible problem with this is that the encrypted data appears to be random, which could make it difficult to use. This could be a problem with data like names, as users expect these to be easy to read.

1.1 Research Aim

This thesis aims to show that data can be de-identified without damaging the database schema and be used by the QA / test teams to perform their tasks as normal.

It will be verified that the QA / test teams' full set of tests can be run successfully on a de-identified data set.

A potential issue may occur in general support where a developer needs to look at a personal record to fix a specific problem, but this could be resolved by using an internal customer number, rather than a customer surname or identity number. The POPI Act allows for this information to be used as long as all POPI requirements are met.

1.2 Objective

The primary objective of the research is to test the FPE method of data de-identification on a production data set. The de-identified data set will be loaded into a test environment to ensure that the QA staff can run their pre-deployment test cases against the data with no issues. If the QA team are able to run all their pre-deployment tests on a de-identified dataset, then the de-identification process will have been successful.

A problem that may occur is that when FPE encrypts the data, it may make it difficult for QA staff to use. The Voltage solution [36] creates random encryption output of customer names in the database. For example, if encrypting the name “Bob Alice”, the encrypted output could be “Sdw Edrtv”, which QA staff may find difficult to work with.

It was anticipated that the author may have needed to further investigate a pseudo-random encryption of customer names, as this should be easier to use than random characters for customer names. The Anonymisation Decision-Making Framework [25] suggests that risk should be factored into the anonymisation process such that, if the risk of re-identification is low, a lesser form of encryption could be used. As it happens, the QA team were content to use the de-identified form of names, so pseudo-random encryption was not necessary.

1.3 Approach

The data used in this research comes from a financial services group that operates in emerging markets providing financial products and credit solutions. There are two sets of data, from different countries - Botswana and Colombia. The company’s operations in each country use completely different versions of the group’s proprietary software, providing access to two distinct databases and test teams with which to work. The software that the company uses at the remainder of its operations is the same as that used in Botswana, so the results there would most likely be the same. If the research is successful, the company will implement the solution across its operations in all the countries in which it operates, but the implementation will be beyond

the scope of this research.

The software is bespoke software developed by an outsourced partner contracted to the company. For completeness the names of the software packages are as follows:

- Colombia - Flexifin
- Botswana - Novalend

In doing this research, the privacy legislation in the different jurisdictions needs to be considered.

A de-identified dataset will be created and used to run standard QA test cases to ensure they work in the same way as the original test cases.

However, test cases only need to be run with the de-identified data to verify the hypothesis.

1.4 Assumptions

The author's preference was to use open-source software to perform this research as there is no financial investment and it can easily be incorporated into standard operating procedures for de-identification during the QA process. The company has a development team that develops the Lending system, and can easily make step-changes to the software and database used in the QA test processes.

Since there are many open-source FPE libraries to choose from, a number will be evaluated and one will be chosen that fits the research purpose, as opposed to writing one from scratch.

NIST800-38G [21] discusses two modes of operation for the de-identification of data, both of which use the underlying AES algorithm. The two techniques rely on a Feistel network to perform the encryption, so they are known as Feistel Format 1 (FF1) and Feistel Format 3 (FF3). Feistel networks are discussed on more detail in paragraph 2.4.1 on page 22. The research focuses on the implementation of the FPE FF1 primarily and does not go into the detail of the mathematics of FPE or AES. These are covered in NIST800-38G [21] and "Announcing the Advanced Encryption Standard" [35], respectively.

An underlying AES 128-bit block cipher is used for the FF1 encryption.

The de-identification of PII stored in a graphical format is excluded as this is a binary data type rather than a string or integer data type. This will form part of a future project.

Encrypted fields will not be used as database indexes, although this should work effectively as the encryption is always the same provided the same passphrase and tweak are used throughout the database encryption.

Re-identification was not necessary for the databases used in these experiments. The de-identified database was used purely for QA testing, so, when a new database was required it was generated using a de-identified production database. If re-identification were to become necessary, the LibFFX library would need to be modified to support special characters.

2 Literature Review

2.1 Introduction

This research investigated various encryption solutions to find the best method of de-identification, using data from a financial services company with operations across a number of jurisdictions. It was necessary to review the following:

1. Legislative requirements
2. Possible encryption solutions
3. Format-Preserving Encryption

As the data used was processed in South Africa, complying with the requirements of the POPI Act [83] of South Africa of 2013 was a key focus. In addition, consideration of global best practice was prudent. As such the international standards below were taken into account.

The Information Commissioner's Office (ICO) [94] in the United Kingdom (U.K.) provides guidelines regarding the requirements for de-identification, but not the implementation of such. In 2012, the ICO published the "Anonymisation: managing data protection risk code of practice" [42], which considers some key anonymisation techniques, such as: data masking, pseudonymisation, aggregation, and derived data items and banding.

The Anonymisation Framework [25] discusses de-identification and anonymisation in some detail. However it focuses primarily on data that would likely be released to researchers rather than being used internally by different teams within an organisation. It sets out a risk framework that outlines a process to ascertain when it is safe to allow data to be released to certain groups, particularly researchers. This risk framework addresses the possibility of reducing the security controls required to keep data safe, depending on the intended audience of the data. It also examines several mechanisms that might be used by a threat-actor to re-identify data that was previously thought to be properly anonymised.

In the United States of America (U.S.A.), the National Institute of Standards and Technology (NIST), a non-regulatory agency of the United States Department of Commerce, produced a special publication NIST800-38G [21] in 2016, that set forth "Methods for Format-Preserving Encryption". This standardises a new methodology of encrypting PII in any format, while maintaining the format of the inputted data.

Little academic research exists on the implementation and testing of de-identification of data, although there is some research on de-identification of private health information (PHI) [60] and big-data data protection [43, 42], but these did not provide solutions that work without making database schema changes.

2.2 Legislative Requirements

Before investigating possible encryption solutions, it was necessary to review the legislative framework applicable to the data. This included data protection legislation in the countries where the data originated - Colombia and Botswana, as well as requirements in the countries where the data is processed - South Africa and the United Kingdom. A summary of the relevant legislation is set out below, highlighting possible challenges that may arise as a result of the legislation and how these challenges could be addressed.

2.2.1 South Africa's Protection of Personal Information Act

Implementation of the Act The Protection of Personal Information Act (POPI) [83] was promulgated in 2013, with the enforcement date currently anticipated to be mid-2019. Initially law firms expected the enforcement date to be mid-2017 [65, 61]. There has been much conjecture with regards to this date and the timeframe has been extended several times. Michalsons Attorneys [61] most recently expected the commencement date to be December 2017. One of the primary requirements for the commencement of POPI was the establishment of an Information Regulator [98]. In October 2016, Pansy Tlakula was appointed as chairperson of the South Africa Information Regulator and, in line with the Act, she appointed four additional members in December 2016. The POPI Act draft regulations [96], were published in September 2017, but did not make any substantial changes to POPI. The regulations cover the administrative aspects of the legislation, such as the processes and procedures related to implementing POPI and the responsibilities of appointed roles. Since the POPI regulations were published on 10 September 2017 [96], it is likely the commencement date will be late 2017 or early 2018. When the Information Regulator announces the POPI commencement date, South African companies will have a one year grace period to become fully compliant with POPI. During this year companies are required to make the changes to their systems and processes necessary to comply with the Act. This represents a large body of work, so it is in companies' interests to start the compliance programme timeously. The consequence of non-compliance with POPI, once the

enforcement date is reached, is a fine up to R10 million or a prison sentence of up to ten years for the company Chief Executive Officer [62]. Non-compliance could also lead to reputational damage and a loss of customers.

An Overview of POPI According to Tlakula [67], POPI requires strict security controls for the use and protection of PII by private and public companies.

According to POPI [83], the definition of personal information (PI), otherwise referred to as PII is as follows:

“information relating to an identifiable, living, natural person, and where it is applicable, an identifiable, existing juristic person, including, but not limited to:

(a) information relating to the race, gender, sex, pregnancy, marital status, national, ethnic or social origin, colour, sexual orientation, age, physical or mental health, well-being, disability, religion, conscience, belief, culture, language and birth of the person

(b) information relating to the education or the medical, financial, criminal or employment history of the person

(c) any identifying number, symbol, e-mail address, physical address, telephone number, location information, online identifier or other particular assignment to the person

(d) the biometric information of the person

(e) the personal opinions, views or preferences of the person

(f) correspondence sent by the person that is implicitly or explicitly of a private or confidential nature or further correspondence that would reveal the contents of the original correspondence

(g) the views or opinions of another individual about the person, and

(h) the name of the person if it appears with other personal information relating to the person or if the disclosure of the name itself would reveal information about the person”.

According to POPI [83] the definition of de-identify is as follows:

“in relation to personal information of a data subject, means to delete any information that:

(a) identifies the data subject

(b) can be used or manipulated by a reasonably foreseeable method to identify the data subject

(c) can be linked by a reasonably foreseeable method to other information that identifies the data subject, and “de-identified” has a corresponding meaning.”

The POPI Act establishes eight general conditions to ensure reasonable protections when processing personal information, as follows:

1. Accountability: PII processors must comply with the principles of POPI
2. Processing limitation: PII processors must process PII for lawful reasons and in a manner that does not infringe privacy
3. Purpose specification: PII processors must only obtain and retain PII for a specific purpose
4. Further processing limitation: PII processors must only perform further processing of PII in cases which are compatible with the purpose for which the PII was collected
5. Information quality: PII processors must ensure that PII is complete and accurate
6. Openness: PII processors must inform individuals that their PII has been obtained and the purpose thereof
7. Security safeguards: PII processors must ensure that the integrity of PII must be secured using reasonable technical and organisational measures
8. Data subject participation: PII processors must allow an individual to request whether the organisation holds their PII. The individual may then request that the information is deleted or corrected if it is incorrect.

The bulk of the regulation is only five pages long, with the remaining pages providing many forms to facilitate the communication around POPI between data subjects and data processors.

Implementing a system of compliance with POPI The Information Officer is the key person who must ensure compliance within an organisation [83]. The Act requires that individuals who do not have a specific requirement to view customer data should not be able to do so. This requirement means that Development and Quality Assurance (QA) staff / testing teams should not perform their jobs using databases of customer data that have not been de-identified.

According to clause 6 of POPI, the Act does not apply if PII has been de-identified such that it cannot be re-identified. Therefore, POPI does not apply to datasets that have been encrypted or de-identified. All employees that do not work directly with the customer should, therefore, perform their tasks using a de-identified data set. In this research, the employees are QA / test staff who test software for the company.

There are other teams that do not need to view customer PII to complete their job functions, such as Application Development teams. These were excluded from this research as getting time with Developers was more challenging than getting time with QA staff, and proving that the methodology works with the QA team should be sufficient to start testing with the Developers.

Since it is difficult to create a random dataset that has the richness of actual data, this financial services company, like many other companies, chose to de-identify customer datasets for QA testing.

All people or systems that process PII in South Africa will be required to comply with POPI, so even though the data used for testing comes from other African countries and Latin American countries, POPI still applies.

2.2.2 Data Protection Act of the United Kingdom 1998

POPI is very similar the Data Protection Act (DPA) of the United Kingdom (U.K.) of 1998 [33], which also includes eight principles for the protection of personal data in the U.K. The U.K. DPA was enacted in 1998 as a result of the 1995 European Union (E.U.) Data Protection Directive [92] of the European Parliament and of the Council of 24 October 1995 on the “protection of individuals with regard to the processing of personal data and on the free movement of such data”.

The DPA in the U.K. is enforced primarily by the Information Commissioners Office (ICO) [94, 95], with the backing of the Financial Conduct Authority [27] (FCA) and the Prudential Regulation Authority [3] and has a long history of serving large fines [11] for breaches of the Act.

2.2.3 Data Protection in Botswana

In the Botswana 2011 “Proceedings of the E-Government Strategy Stakeholders Conference” [76] it was noted that twenty two laws were to be formulated and amended. One of the key laws identified was that of Data Protection.

In the 2012 “Development of ICT Legal And Regulatory Framework In Botswana” [46], Advocate Keetshabe noted that Botswana needed to develop policy and legislation to deal with the protection of personal privacy, “particularly in the context of cross-border data flow, health care and financial services and transactions.” According to Bakibinga-Gaswaga [2] in 2016, Botswana had not enacted comprehensive data protection laws.

Rautenbach notes that it is quite likely that Botswana will enact its DPA once South Africa’s POPI Act is fully commenced [82] since Botswana regularly cites South African case law in its court cases. Dr. Tobin concurs with this statement [98].

2.2.4 Data Protection in Colombia

Colombia does not have a comprehensive law or a data privacy law equivalent to the E.U. Data Protection Directive. However, the protection of privacy and personal data as well as the regulation of databases in Colombia is governed by Law 1581 of 2012 and Law 1266 of 2008 [32]. The Constitution and specific court rulings have been issued to regulate the use of personal data to some extent.

In 2011, Gomez Pinzon-Zuleta issued a statement that the Colombian DPA would soon be signed by the President which would apply to the collection and processing of personal information in Colombia. This law would also ensure that cross-border transfer of personal information and processing would only be permitted in countries with similar legal protections to those of Colombia [32].

2.2.5 General Data Protection Regulation

The new data protection protocol known as the General Data Protection Regulation (GDPR) was adopted on 27 April 2016 in the European Union (E.U.) and will be fully in force from 25 May 2018 [93]. This regulation aims to strengthen and unify data protection for individuals in the E.U., significantly more so than its predecessor the Data Protection Directive. It will give citizens more control over the use of their personal data and will simplify the regulations for international companies working with E.U. citizens’ information as there will only be one regulation, rather than a different law in each E.U. country, as is currently the case. With the old Data Protection Directive, all E.U. countries were required to enact law enabling data protection.

The penalties are significantly higher than the Data Protection Directive and the DPA, as

they can be as much as 4% of global turnover for a second breach and 2% for a first breach. Levied fines could be an additional 2% of global turnover for each GDPR rule breached. "The Register" has calculated that the fines delivered in the U.K. in 2016 would have been 79 times higher under the GDPR [50], when compared to the fines levied under the DPA.

The GDPR is both a requirement for the processing of data of E.U. citizens and a requirement for companies that have offices in the E.U.

The financial service company from which the data used for the research comes, has offices in the E.U., so the GDPR was also in scope. The GDPR is the the European Parliament's new regulation aimed at strengthening and unifying data protection for all individuals within the E.U. The GDPR commenced on 24 May 2016 with a two year grace period set to end on 25 May 2018. Since the POPI effective date will be later than January 2019, POPI's grace period will end after the GDPR's grace period [61]. It is therefore essential that the GDPR requirements are considered as well as POPI.

A new requirement is that Data Controllers must be able to prove consent for any interaction with a data subject [93].

Hintz [40] notes that de-identification techniques can be used to help comply with the GDPR requirements as information risks are significantly reduced when de-identification is used on a dataset. He notes that certain techniques can reduce risk to almost zero, but these can potentially cause the information to lose its utility or value. He concludes that different levels are appropriate in different situations and that a risk analysis should inform the technique that is used. Should there be a breach of de-identified data, it is unlikely that the regulator would require that the data subjects are notified as the information is anonymous. The Anonymisation Decision-Making Framework agrees with this risk-based approach to data de-identification [25]. According to Hintz [40], the GDPR provides the basis to recognise a much more complete spectrum of de-identification than the 1995 Directive. The GDPR adds an explicit recognition of de-identification with the concept of pseudonymous and anonymous data.

The GDPR offers more specifics regarding the sometimes vague concept of what is meant by personal data [9] by defining a data subject as a natural person and PII including: "an iden-

tification number, location data, online identifier or one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that person” [9].

The GDPR introduces a new concept that did not previously exist under the Data Protection Directive. This new concept is pseudonymization, which is the “processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information” [10]. Pseudonymization can ensure that an individual’s PII can no longer be used to identify the individual, thus allowing the dataset to be shared. There is, however, some concern that companies may use pseudonymization to avoid other PII controls.

The GDPR further defines anonymised data as “data rendered anonymous in such a way that the data subject is not or no longer identifiable” [105]. This definition emphasises that anonymised data must be stripped of any identifiable information, making it impossible to derive insights on a discreet individual, even by the party that is responsible for the anonymisation, or with the use of external data sources.

As such, de-identification of all PII in a dataset, when done properly, using the FF1 algorithm from NIST800-38G, matches both the higher standard of “anonymisation”, as well as “pseudonymization”, as per the GDPR, which places the processing and storage of de-identified personal data outside the scope of the GDPR [25, 24].

The GDPR makes it clear that Data Protection Officers (DPO) are not personally responsible if their company is not compliant with the GDPR [16]. Compliance is the responsibility of the controller or the processor.

Hyams [41] indicates that the GDPR provides individuals with the “right to be forgotten” especially when PII is “inaccurate, inadequate, irrelevant or excessive”.

2.3 Possible Encryption Solutions

Traditional de-identification involves encryption or hashing. Hashing and encryption change the size of the data. If the data was stored in a Structured Query Language (SQL) database table it would not fit in the original structured database without field or schema changes in respect of length, at the very least, and possibly even changes to the data-type. Many hash functions can also be reverse engineered using rainbow tables if they are not salted, according

to Crackstation.net[18].

A solution widely employed in the Payment Card Industry is tokenisation, however, the size and complexity of these tokenisation databases negate the effectiveness of this as a practical solution because they need large lookup tables that would not scale for the many input fields needed for PII. The Data Encryption Standard and the Advanced Encryption Standard are the traditional approaches to encryption but they only encrypt fixed sized blocks that are binary, so they do not work with shorter non-binary input.

Another key concept when manipulating data is that of Extract, Transform and Load (ETL), which does not encrypt data, but offers a solution to load, manipulate and encrypt data, then output encrypted data to a new database.

Combining ETL with format-preserving encryption (FPE), provides a solution to the problem of QA staff testing using actual customer data.

Each of these concepts is reviewed below, in more detail. This review points us to FPE, as the most suitable solution. FPE has evolved, since its inception more than 25 years ago, and recently there have been several developments, that have made this type of encryption more robust. These developments include using an additional password called a “tweak” to give small encryption domains additional security. A vital development or key turning point that makes this the best solution and the one with the strongest framework, is the introduction of the National Institute of Standards and Technology Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation (2016), which ratifies FPE as “best practice” in the security realm.

2.3.1 Hashing

Hash functions, otherwise known as “one-way hash functions” are generally used for authentication and identification [57]. A hash function is an algorithm for mapping arbitrary binary strings of any length to binary strings of a fixed length [57]. The best known hash function is “MD5”, which outputs a 128 bit hash. Hash functions are designed to not be reversible, however, flaws in the algorithms and advances in computing power allow new attacks to be successful. Another well known hash function is “SHA-1”, which outputs a 160 bit binary hash. Hash functions are not as secure as they were once thought to be, as MD5 was cracked in 1996.

Also, Crackstation.net [18] has created rainbow-tables lookups for almost 15 billion hashes

for both MD5 and SHA-1, which shows that hashes are not an ideal mechanism for encrypting data. Crackstation has advised that the security of hashes can be dramatically increased by using a salt.

Since hashes change the length of the data, they cannot be used to encrypt data in a SQL table without first changing the length and possibly the datatype of the SQL table fields.

2.3.2 Tokenisation

Tokenisation is widely used for Payment Card Industry Data Security Standard (PCIDSS) [77] compliance initiatives to secure credit card details.

Tokenisation has been implemented in some of the main commercial FPE software solutions, such as Protegrity [49] and HPE Secure Data [37].

Tokenisation is a process whereby a piece of sensitive data is replaced with a token, of the same type and length. The token is retrieved from a lookup database table so that the correct token can replace each instance of the original data. The token is then used in place of the sensitive data. The tokens are useless by themselves without access to the original lookup table, however the lookup table can become quite large as more tokens are created. It is critical that the lookup table is kept secure to avoid a PII or a PCIDSS breach. This also means that lookups become slower as the lookup table grows in size. Since a large amount of data will be encrypted in this research, this is not a feasible solution. The effort involved in securing a large lookup table is considerably more difficult than managing a single passphrase / key.

It is likely that using a tokenisation approach to encrypt first names and surnames to create more human-friendly output would be possible. This would have been feasible based on a risk assessment of the PII data and the QA user group, as well as the risk of re-identification as per the Anonymisation Decision-Making Framework [25].

2.3.3 Data Encryption Standard

The Data Encryption Standard (DES) is the most well-known symmetric-key block cipher [58]. It set a precedent in the mid-1970's as the first commercial-grade modern encryption algorithm with the algorithm being fully open and implementation details published for open access. It is defined by the American standard FIPS 46-3 [69]. DES was superseded by Triple DES (TDES) or (3DES) [72], which involved running the encryption through three iterations of DES. DES

and 3DES were ultimately replaced by the the block-cipher encryption algorithm, Advanced Encryption Standard.

2.3.4 Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic information established by the U.S.A. National Institute of Standards and Technology (NIST) in 2001. The AES cipher was developed and submitted by two Belgian cryptographers: Joan Daemen and Vincent Rijmen and was originally known as Rijndael. Four other algorithms were included in the final selection process for AES, but Rijndael was selected. AES can be used with key lengths of 128, 192 or 256 bits, but always uses a block size of 128 bits. AES uses a symmetric-key, which means that the same keys are used for encrypting and decrypting the information. AES was first published as FIPS PUB 197 [73] in 2001 [26] and was subsequently included in the ISO/IEC 18033-3 standard [44]. Some attacks on AES have been issued, but none have been proved to be computationally feasible in terms of a successful attack taking a reasonable amount of time [106].

The block size for AES is 128 bits, so using this to encrypt data results in a 128-bit output block, hence the resulting ciphertext of AES encryption running in 128-bit block mode cannot fit into the space of a 16 digit credit card number [34].

NIST published SP 800-38A [20] in 2001 which defined five confidentiality modes of operation, or algorithms, for use with an underlying symmetric key block cipher algorithm, such as AES. These modes of operation are: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR). These modes are transformations on binary data, such that the inputs and outputs are binary strings. These AES modes cannot operate on sequences of non-binary symbols and cannot create an output block smaller than 128 binary characters.

2.3.5 Extract, Transform and Load

Extract, Transform and Load (ETL) [17] is a process used in data warehousing which is responsible for extracting data from one or more source systems, transforming it and then loading it into a data warehouse or new database. An ETL process was performed to de-identify / encrypt the source data in the experiment and load into a new QA database.

When performing an ETL process on the database the structure of the database fields are known so a rank-then-encipher [5] approach is simple. This is discussed in more detail in section 2.4.2 on page 26.

2.4 Format-Preserving Encryption

Format-Preserving Encryption (FPE) [107] is a mechanism for encrypting a piece of data while maintaining the structure of the original data. In cryptography, the term "format-preserving encryption" refers to encrypting information in such a way that the output, or ciphertext, retains the format of the input, or plaintext [107]. Typically only finite domains are considered for FPE, so a 13 digit South African I.D. number would be encrypted to a new 13 digit number, or a 16 digit VISA credit card number would encrypt back to a 16 digit number and an English word would be encrypted to a new English word [34]. A finite domain as small as two characters can be encrypted with FPE.

Evolution of FPE The origins of FPE date back 25 years. In 1981, the U.S.A. National Bureau of Standards, which later became NIST, published FIPS 74 [75] describing an approach for enciphering arbitrary strings over an arbitrary alphabet. Rogaway [84] later showed this approach to be completely flawed.

Brightwell and Smith [56] coined the phrase "datatype-preserving encryption" in 1997, seemingly the first researchers to describe a potentially workable solution to preserving data format when encrypted. However, they did not provide definitions or proofs for their proposed solution [5].

Black and Rogaway [8] addressed the issue of preserving data format with the cryptographic community, once again, in 2002, providing the requisite definitions and a number of solutions for FPE, which they tested and showed to be usable.

Their proven solution, FPE, allows encryption of the data in a database field without changing the schema of the database to accommodate the output.

Terence Spies, Voltage Security's CTO [87] gave FPE its currently used name i.e. format-

preserving encryption around 2003 [84]. Voltage Security was one of the first companies to create a software product to implement FPE. They were subsequently bought by Hewlett Packard Enterprise Security [101]. Spies was the first to submit an FPE algorithm to NIST, using cycle-walking and an AES balanced Feistel network [5].

HPE Voltage [36] developed a commercially available software solution using FPE that uses standard encryption with the Feistel Format 1 (FF1) AES encryption algorithm to encrypt data in a database field. HPE Voltage also claim to have filed a patent on the technology [38].

Meystre et al. published “Automatic de-identification of textual documents in electronic health records: a review of recent research” [60] in 2010, in which they discuss suitable de-identification methods as per the Health Insurance Portability and Accountability Act (HIPAA) [99]. The HIPAA “Safe Harbour” technique requires that 18 Protected Health Information (PHI) data elements be removed. They noted that HIPAA de-identification was often performed manually and required significant resources. They further concluded that dictionaries performed better with uncommon PHI de-identification and that machine learning performed better with PHI that was not covered by dictionaries. They further noted that the data could easily lose its utility if it was over-scrubbed.

Black and Rogaway’s ongoing research, which is discussed below, contributed greatly to developing FPE as a solution that could be used practically with any alphabet.

2.4.1 Black and Rogaway (2002)

Implementing FPE in a secure fashion, such that the FPE algorithm was cryptographically as secure as that of the underlying block cipher, in this case, AES, was first undertaken in a paper published by cryptographers John Black and Phillip Rogaway [8, 107] in 2002, which described three ways to perform FPE:

- FPE from a prefix cipher
- FPE from cycle walking
- FPE from a Feistel network

Each of these techniques is described briefly below:

Prefix Cipher An integer k is chosen and M is the set $[0, k - 1]$.

The goal is to build a cipher with domain M .

The approach is a simple, practical method for small values of k .

This cipher is named P_x .

The P_x cipher uses a known block cipher E , such as AES, with key-space K , and the domain is a superset of M .

The key space for P_x is K .

To compute $P_{xK}(m)$ for some $m \in M$ and $k \in K$ the tuple

$I = (E_K(0) E_K(1) \cdots E_K(k - 1))$ is first computed.

Since each element of I is a distinct string, each element in I may be replaced with the ordinal position (starting from zero) to produce tuple J .

To encipher any $m \in M$, $P_{xK}(m)$ is computed as the m^{th} component of J (again counting from zero).

The enciphering algorithm is shown in Algorithm 1.

Example from Black and Rogaway [8]:

To encipher the domain $M = \{0, 1, 2, 3, 4\}$.

A random key K is chosen for some block cipher E .

If E is assumed to be an 8-bit ideal block cipher, then E_K is a uniformly chosen random permutation on $[0, 255]$.

Next each element of M is enciphered to create I .

I is the set of each element of M .

If $E_K(0) = 166$, $E_K(1) = 6$, $E_K(2) = 130$, $E_K(3) = 201$, and $E_K(4) = 78$.

The tuple I is then (166, 6, 130, 201, 78).

And our tuple J is (3, 0, 2, 4, 1).

Any $m \in M$ is now ready to be enciphered.

The m^{th} element from J can be returned by counting from zero.

For example, from J , we encipher 0 as 3, 1 as 0, 2 as 2, 3 as 4 and 4 as 1.

If $m = 3$, the output is 4.

Analysis from Black and Rogaway [8]:

Assuming that the underlying block cipher E is ideal, J is equally likely to be any of the

Algorithm 1 Prefix Cipher algorithm from Black and Rogaway [8]

```

Init  $P_{x_K}$ 
for  $j \leftarrow 0$  to  $k - 1$  do  $I_j \leftarrow E_K(j)$ 
for  $j \leftarrow 0$  to  $k - 1$  do  $J_j \leftarrow \text{Ord}(I_j, \{I_j\}_{j \in [0, k-1]})$ 
for  $j \leftarrow 0$  to  $k - 1$  do  $L_{J_j} \leftarrow j$ 

```

permutations on M . The method remains valid when E is secure in the sense of a pseudorandom permutation (*PRP*) such as AES.

Practical Considerations:

Enciphering and deciphering are constant-time operations. The computational cost here is $O(k)$ time and the space used in the initialisation step. This means that this method is practical only for small values of k , as the cost increases as k increases, especially time.

A further practical consideration is that, although this initialisation is a one-time cost, it results in a table of sensitive data, which must be stored securely.

Cycle walking Cycle walking uses a block cipher with a domain larger than M and handles the cases where a point is out of range.

Choose an integer k , then let M be the set $[0, k - 1]$, and use the AES algorithm to encrypt M .

Let N be the smallest power of 2 larger than or equal to k .

Let $n = \lg N$.

Let $E_k(\cdot)$ be the AES encryption algorithm (our n -bit block cipher).

Construct the block cipher function Cy_K on the set M by computing $c = E_k(m)$ and iterating to find whether c is, or is not, in the set M : $c \notin M$.

The enciphering algorithm is shown in Algorithm 2.

Decryption is the reverse of this procedure.

Example from Black and Rogaway [8]:

Let $M = [0, 10^6]$, then $N = 2^{20}$, so $n = 20$.

Use AES to build a 20-bit block cipher $E_k(\cdot)$ on the set $T = [0, 2^{20} - 1]$.

If one wishes to encipher the point $m = 314159$, one computes

$c_1 = E_k(314159)$ which yields some number in T , say 1040401.

Algorithm 2 Cycle Walking Enciphering Algorithm from Black [8]

```

 $Cy_K(m)$ 
 $c \leftarrow E_K(m)$ 
if  $c \in M$  return  $c$ 
else return  $Cy_K(c)$ 

```

Since $c_1 \notin M$, we iterate by computing $c_2 = E_k(1040401)$ which is, say, 1729.

Since $c_2 \in M$, we output 1729 as $Cy_K(314159)$.

Analysis from Black and Rogaway [8]:

If the permutation $E_k(\cdot)$ is viewed as a family of cycles: any point $m \in M$ lies on some cycle, and repeated applications of $E_k(\cdot)$ can be viewed as walking along the cycle, starting at m . In fact, the construction can be considered as follows: to encipher any point $m \in M$ walk along the cycle containing m until one encounters some point $c \in M$. Then $c = Cy_K(m)$. This method assumes that one can efficiently test for membership in M . This is simple in this case when $M = [0, k - 1]$, but might not be quite as simple for other, more complicated, sets. It is now evident that $Cy_K(\cdot)$ is well-defined. Given any point $m \in M$, if $E_k(\cdot)$ is applied enough times, one will eventually arrive at a valid point in the domain M , rather than a point that is always outside M . This is because walking on m 's cycle must eventually arrive back at some point in M , even if that point is m itself.

Black and Rogaway [8] noted that no security is lost in deriving this permutation

The theorem proving this is shown in Black and Rogaway 2002 [8].

Feistel network The third method tested and discussed by Black and Rogaway [8] works as follows. The numbers are split into M pairs of same-sized numbers and then the Feistel construction [89] is applied to produce a cipher.

Again an integer k , is chosen and M is the set $[0, k - 1]$, and a method to encipher M is devised.

The cipher is called $Fe[r, a, b]$ where r is the number of rounds used in our Feistel network and a and b are positive numbers such that $ab \geq k$.

a and b are used to break any $m \in M$ into two numbers for use as the inputs into the Feistel network.

Within the network r random functions F_1, \dots, F_r whose ranges contain M are used.

The algorithm to encipher is shown in Algorithm 3.

If using the Feistel construction results in a number that is not in M , it is iterated again just as with the Cycle-Walking Cipher.

Example from Black and Rogaway [8]:

In order to specify some particular $Fe[r, a, b]_K(\cdot)$ the numbers a and b , the number of Feistel rounds r , and the choice of underlying functions F_1, \dots, F_r must be specified.

As an example, choose $k = 2^{35}$, $r = 3$, $a = 185360$ and $b = 185368$.

Note that $ab \geq k$ as required.

Since ab is larger than k by 74112, the Feistel construction will be on the set $M^1 = [0, (2^{35} - 1) + 74112]$, meaning there are 74112 values, which are in $M^1 - M$, will have to be iterated (just as was done for the Cycle-Walking Cipher).

If the DES algorithm is used with independent keys as the underlying pseudo random function (PRF), a 64-bit cipher with a 56-bit key results. The 64-bit strings on which DES operates are regarded as integers in the range $[0, 2^{64} - 1]$.

Three (PRF)'s are needed so the key $K = K_1 || K_2 || K_3$ will be $3 \times 56 = 168$ bits.

To compute $Fe[3, 185360, 185368](m)$ one computes $L = m \bmod 185360$, and $R = \lfloor m/185360 \rfloor$, and then performs three rounds of Feistel using $DES_{K_1}(\cdot)$, $DES_{K_2}(\cdot)$ and $DES_{K_3}(\cdot)$ as the underlying (PRF)'s.

The first round results in $L \leftarrow \lfloor m/185360 \rfloor$ and

$R \leftarrow (m \bmod 185360 + DES_{K_1}(\lfloor m/185360 \rfloor)) \bmod 185360$.

Analysis from Black and Rogaway [8]:

$Fe[r, a, b](\cdot)$ is a permutation. It is well-known that the Feistel construction produces a permutation, and it was previously shown that iterating any permutation yields a new a permutation.

Black and Rogaway's Theorem 2, which shows the security of a Generalized-Feistel Cipher is published in [8].

The AES cipher is usually used as the random function. Any other good block cipher algorithm could be used instead of AES. DES was used for the Feistel network example by Black and Rogaway in 2002. DES has now been withdrawn by NIST.

Black and Rogaway used game simulations to test their theories and prove them to be cor-

Algorithm 3 Encipher using a Feistel Network from Black [8]

Algorithm

 $Fe[r, a, b]_K(m)$ $c \leftarrow fe[r, a, b]_k(m)$ if $c \in M$ return c else return $Fe[r, a, b]_K(c)$

Algorithm

 $fe[r, a, b]_K(m)$ $L \leftarrow m \bmod a; R \leftarrow \lfloor m/a \rfloor$ for $j \leftarrow 1$ to r do if (j is odd) then $tmp \leftarrow (L + F_j(R)) \bmod a$ else $tmp \leftarrow (L + F_j(R)) \bmod b$ $L \leftarrow R; R \leftarrow tmp$ if (r is odd) then return $aL + R$ else return $aR + L$

rect.

Black and Rogaway's 2002 paper [8] proved that there were three mechanisms for FPE to achieve the same level of security as the underlying block cipher. This meant that if the AES algorithm was used as the block cipher to create an FPE algorithm, then the resulting FPE algorithm was just as secure as AES because an adversary capable of defeating the FPE algorithm would also be able to defeat the AES algorithm. So far, no adversaries have defeated the AES algorithm, therefore, since AES is considered secure, the FPE algorithms constructed from AES must also be considered as secure as AES.

Input data needs to be preserved for a wide range of applications, as Black and Rogaway [8] observed, in their research. They demonstrated that this could be achieved, securely and efficiently, using standard ciphers like AES. They proposed building a cipher using a Feistel network and breaking the input into two halves and then performing encryption on each half, separately using multiple rounds [34], rather than using the prefix-cipher or cycle walking to perform FPE.

The Feistel network is illustrated in Figure 1.

For example, to encrypt a 64-bit block, the input block is divided into two halves. The resulting 32-bit halves are called $L0$ and $R0$. $L0$ and $R0$ are then pushed through a Feistel network for ten rounds to perform the encryption.

A block cipher, such as AES, with truncation of the results, and XOR is all that is required

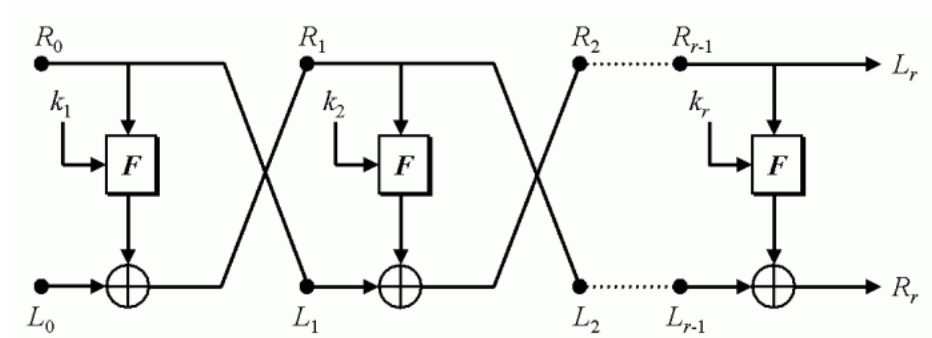


Figure 1: Feistel Network from Mat Green [34]

to create the Feistel network. As a block passes through the left side of the Feistel network, the block is XOR'ed with the right-hand side.

In cases where the output from the algorithm is too big to fit in the same space as the original data, Black and Rogaway [8] devised a solution which they called cycle-walking [34], discussed in section 2.4.1 on page 21. If the output from the encryption algorithm was too big, they re-encrypt the output again using their block cipher and then tested again to check if the output was the right size. This process was repeated with the new output until the encrypted block was small enough to fit into the required 16-bit block.

2.4.2 Bellare, Ristenpart, Rogaway and Stegers (2009)

Bellare, Ristenpart, Rogaway and Stegers [5] wrote a paper on FPE in 2009. They were the first to formally define the security goals for FPE [66]. Spies [87] gave FPE its name in 2003.

They introduced the concept of a tweak-able block cipher to the FPE construction, which was especially useful when encrypting small domains to enhance security and resist dictionary attacks.

As well as using a key, or password, for the encryption technique, each algorithm uses an additional variable, or password, known as the “tweak”. The tweak is regarded as an additional component of the key, which does not have to be kept secret and can vary for each input. For instance, the tweak can be read in a separate column from a database table when performing encryption on a database table. The tweak is especially important for small alphabets, as it is easier to guess the input in these cases. The tweak is similar to using a salt with a hash function. The tweak significantly increases the size of the dictionary required to attack the cipher in the case of a dictionary attack.

A block-cipher can be viewed as a limited form of FPE, where the length of the block is fixed i.e. 64 or 128 bits.

Rank-then-Encipher The Rank-then-Encipher (RtE) method suggested by Bellare et al. [5] reduces the task of designing a FPE scheme for format F to the task of designing a FPE scheme for an integral domain. In particular, the RtE framework allows one to apply FPE to domains as small as two characters securely.

Bellare et al. [5] introduce the RtE method for encrypting numeric message spaces of an arbitrary finite set, of the form $(X = [N])$, or string message spaces of the form $(X = \Sigma^n)$.

To illustrate the idea with an example: To encipher a 13-digit South African identity number string with a zero Luhn checksum [55, 1], the final digit in the identity number is a checksum calculated with the Luhn algorithm.

First select its first 12 Digits: X' .

Encipher X' as a 12-digit string to get a 12-digit ciphertext Y' .

Calculate the final digit necessary to produce the correct Luhn checksum for Y' .

Then convert this 12-digit string Y' to a valid 13-digit South African identity number Y by appending the checksum to Y' .

The RtE approach works as follows: If there is a finite message space X to be encrypted with FPE and it is already possible to perform FPE on some other message space X' which has the same number of points as X . To encipher a point $Z \in X$, a ranking function maps it to a corresponding point $Z' \in X'$. Then that point in X' is enciphered to get a ciphertext $Y' \in X'$, then Y' is mapped to its corresponding point Y in X . The mapping between X and X' must be bijective, i.e. each point in X is matched up with one and only one point in X' .

The set X' is usually of the form $X' = [N]$, or something similar in which case the map from X to X' is a ranking of the points in X . RtE encryption is considered to be a “meta-technique” for building an FPE scheme, because it turns an FPE scheme on one message space into an FPE scheme on another, similar one.

The RtE approach enables FPE schemes to be created for the kinds of message spaces that arise in practice. This approach could be used to build an algorithm that would encrypt special characters, as long as one can ensure that each point in X is matched up with one and only one point in X' .

Creating ranking and un-ranking functions that map between X and X' is not a cryptographic problem, so there is no cryptographic significance as to how it is performed [84]. The simplest and most efficient method available can be chosen.

2.4.3 Bellare, Rogaway and Spies (2010)

In 2010, Bellare, Rogaway and Spies published: “The FFX mode of operation for format-preserving encryption” [7], which was their submission to NIST for the FFX mode of operation for the the AES block-cipher to perform FPE. This was tested by NIST and finally released as the FF1 mode of operation for FPE in NIST800-38G [21]. In 2010, Bellare, Rogaway and Spies also published the “Addendum to The FFX Mode of Operation for format-preserving Encryption” [6]. The FFX scheme is more open-ended than BPS since it has many parameters that must be set to make into a complete FFX mode. As such, it is more like a framework than an AES mode. There are a total of nine parameters that may be set.

Bellare et al. suggest using parameter collections A2 and A10 primarily. Parameter collection A2 encrypts binary strings of 8–128 bits and parameter collection A10 encrypts decimal strings of of 4–36 digits. Both parameter collections use 12 rounds of Feistel for long strings, but the number of rounds increase as the strings get shorter, as high as 24 rounds (for FFX-A10) or 36 rounds (for FFX-A2) for very short strings. This is to increase the strength of the encrypted output for shorter strings, but not waste time with unnecessary rounds of encryption for longer strings, where there is a bigger alphabet and subsequently, less risk of a successful dictionary attack.

2.4.4 NIST Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation (2016)

Key Information For NIST800-38G NIST800-38G was published by the Information Technology Laboratory (ITL) at NIST, which provides technical leadership for the United States’ measurement and standards infrastructure. One of their primary outputs is the SP 800 series of documents, which provides NIST’s information security guidelines. ITL are viewed as thought leaders for information security standards worldwide.

Symbols used in NIST 800-38G [21]:

- ceiling: $\lceil \rceil$
- Concatenate: $\|$
- floor: $\lfloor \rfloor$
- XOR: \oplus

Definitions used in NIST 800-38G [21]:

- Bit string: “An ordered sequence of bits“
- Feistel structure: “A framework for constructing an encryption mode. The framework consists of several iterations, called rounds, in which a keyed function, called the round function, is applied to one part of the data to modify the other part of the data. The roles of the two parts are swapped for the next round”
- PRF: Pseudo random function
- Ceiling: Given a real number x , $\lceil c \rceil$ is the least integer that is not less than x
- Floor: Given a real number x , $\lfloor f \rfloor$ is the greatest integer that is less than or equal to x

Formulas used in NIST 800-38G [21]: There are several component functions used in NIST800-38G which in the FF1 technique are called:

- $x = Num_{radix}(X)$ converts a numeral string (X) to a number (x) using the base radix
- $Num(X)$ converts a byte string (X) to an integer
- $X = STR_{radix}^m(x)$ converts an integer (x) to a numeral string (X) of length (m) in base radix
- $Y = REV(X)$ reverses the order of a numeral string (X) to a numeral string (Y)
- $Y = REVB(X)$ reverses the order of a byte string (X) to a byte string (Y)
- $Y = CIPH_k(X)$ is the designated cipher function of an approved 128-bit block-cipher, such as AES128-CBC, on block string input (X) using key (k) to output block string (Y)

- $Y = PRF(X)$ invokes the $CIPH_k(X)$ function i.e. $PRF(X) = CIPH_k(X)$. $PRF(X)$ is known as the pseudo-random function and invokes the AES-CBC technique on the input to generate the ciphertext. This function is the core function of the FF1 encryption algorithm.
- $REVB(X)$, and $PRF(X)$ are represented as bit strings, as AES uses bit strings

Overview Of NIST800-38G The NIST published the: “Draft Recommendation for Block Cipher Modes of Operation: Methods for Format- Preserving Encryption” in 2013 [74] for public comment.

The draft recommendation included three methods to perform FPE using an underlying block-cipher. These included: FF1, FF2, and FF3. Each is a format-preserving, Feistel-based mode of operation of the AES block cipher. FF1 and the FFX framework was submitted to NIST by Bellare, Rogaway and Spies under the name FFX[Radix] [7], FF2 was submitted to NIST by Vance under the name VAES3 [100] and FF3 is the main component of the BPS mechanism that was submitted to NIST by Brier, Peyrin, and Stern [12]. The submission documents are available at [74].

According to Bellare [5], Spies went on to submit a proposed mechanism, FFSEM, to NIST that combines cycle walking and an AES-based balanced Feistel network.

In March 2016, NIST in the U.S.A. published the final version of “NIST Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption” (NIST 800-38G) [21].

This recommendation specifies two methods for format-preserving encryption called FF1 and FF3, both of which use an underlying NIST approved symmetric-key block cipher algorithm to encrypt data. The underlying symmetric-key block cipher algorithm for both FF1 and FF3 is the AES [35]. By default, AES only encrypts binary information. FPE can be used to encrypt data that is not binary, which is unusual for encryption algorithms. As such, if a set of symbols like decimal numbers needs to be encrypted, the FPE encryption techniques encrypt the data in such a way that the length and structure of the encrypted data are the same as the original data.

For example, a South African identity number (ID) consists of thirteen decimal numerals, which is an integer that is less than 10 trillion. Using older encryption techniques, the integer

would be converted to a bit string as input, then encrypted and converted back to an integer, however, when this occurs, the integer may be higher than 10 trillion. This new integer would be too long for an ID field in a database, which is configured to only accept thirteen numeric digits and may not work correctly with software that is expecting an ID as input. The FPE algorithm would encrypt the same South African ID such that the output was a new thirteen digit number. FPE also allows for the integrity of the checksum to be preserved.

The algorithm used by NIST800-38G FF1 is based on the work of Bellare, Rogaway and Spies [7]. The algorithm used by NIST800-38G FF3 is based on the work of Brier, Peyrin and Stern [13].

The NIST800-38 series of special publications is a series of seven documents regarding modes of operation, or techniques, of block cipher algorithms. Before NIST800-38G, all NIST block-cipher encryption techniques were algorithms that operated only on binary data. NIST did not provide an algorithm that worked on non-binary data, without first converting to binary data. The FPE method works on both binary and decimal numbers. It does, in theory, work on sequences created from any set of symbols, known as the base radix. FF2 was removed from the final version of NIST800-38G as it was found to be lacking the required 128 bits of cryptographic strength [22]. The FPE techniques are named to indicate that they are format-preserving Feistel encryption modes. The Feistel structure is also used in the DES and 3DES algorithms [59, 72].

Dworkin, the author of NIST800-38G [21] noted that one should be careful when de-identifying PII to ensure that it cannot be re-identified by alternative means. The concept of re-identification and the required risk-calculations are discussed at length in the “Anonymisation Decision-Making Framework” [25] published by the ICO in the U.K.

Implementing The Recommendations The FF1 algorithm uses an average of ten rounds to complete the encryption, although this is variable, so it is marginally slower than FF3, which uses only eight rounds, however, FF1 supports longer inputs than FF3.

FF1 converts character strings to numeral strings before it starts the encryption process. As an example: “a” would be represented by 0, “k” by 10, and “z” by 25. Uppercase characters, special characters and decimal numbers would all be transposed to a numeric string. After the

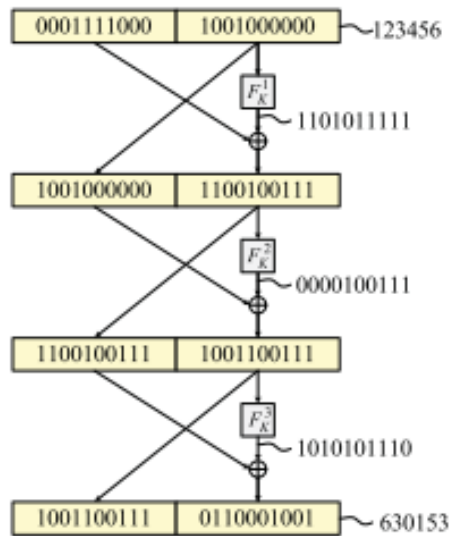


Figure 2: Feistel Rounds from Rogaway [84]

encryption is performed, the output from the encryption is transposed back to the original alphabet. A South African ID number would be represented as a character string using an alphabet of decimal numbers. FF1 interprets numeral strings as numbers by representing the numbers with decreasing order of significance [21], which is slightly different from FF3's implementation. Both FF1 and FF3 use round functions to split the original data into equal halves to be encrypted, which they then encrypt using AES. FF1 uses a "tweak", which is an additional password used to gain additional security for short input strings. Since the tweak does not necessarily have to be secret, it can be read from a separate database field when using a database table. The secret key and the tweak are both used to encrypt the input, likewise they are both used to decrypt the encrypted string, if necessary. The radix in FF1 denotes the base alphabet, which could be any set of symbols, but for this research will be $[0..9]$ or $[a..Z]$.

Figure 2 shows three rounds of a classical Feistel network from Rogaway [84] on a 20-bit string. In this example, 0001111000 1001000000 gets enciphered to 1001100111 0110001001 using the round function F_K^r . Its subscript is the key and its superscript is the round number. The input is split into left and right halves. The right half is XOR'ed with the left half after the round function is run and the right half is copied to the left side.

For a key (K) the underlying block cipher of the encryption technique may be a permutation, i.e. a transformation on bit strings of a predetermined length, in this case, 128 bits. The

forward transformation and the reverse transformations are generally referred to as the encrypt and decrypt functions of the block cipher.

The designated block cipher techniques: FF1 or FF3 consist of an encryption and a decryption function utilising the key (K).

The encryption algorithm takes as inputs the plaintext, represented as a numeral string (X), a byte string, called the tweak, denoted by (T), and the key (K).

FF1 algorithm The output is a numeral string Y , which is the same length as X . The algorithm refers to the underlying AES encryption cipher as $CIPH_k$. A function called the round function is applied to the first part of the data to modify the second part of the data. The roles of the two parts are then swapped for the next round as per Figure 2. The input data (and output data) for each round are two strings of characters which will be converted to numerals for FF1.

During Round 1, the round function (F_K) is applied to the first input string (Bi) using length (n), the tweak (T), and the round number (i) as additional inputs. The round function varies the lengths of its input and output strings when the round number (i) is odd or even.

When using the FF1 technique, the left-most numeral is the most significant.

The encryption technique also requires the base alphabet (*radix*) and the minimum and maximum range of lengths for the numeral string.

For each technique, the 128-bit input and output blocks of the designated block cipher, $CIPH_k(X)$, are represented as strings of 16 bytes.

Description of the FF1.Encrypt(K, T, X) algorithm The full algorithm is shown in Algorithm 4.

K is the secret key, T is the tweak, X is the input to be encrypted.

X is represented as a numeral string between *minlength* and *maxlength*.

Algorithm steps 1 to 5

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$
2. Let $A = X[1..u]$; $B = X[u + 1..n]$
3. Let $b = \lceil \lceil v \cdot \text{LOG}(\text{radix}) \rceil / 8 \rceil$

$$4. \text{ Let } d = 4\lceil b/4 \rceil + 4$$

$$5. \text{ Let } P = [1]^1 \parallel [2]^1 \parallel [1]^1 \parallel [\text{radix}]^3 \parallel [10]^1 \parallel [u.\text{mod}256]^1 \parallel [n]^4 \parallel [t]^4$$

Discussion

In Step 1 and Step 2: X is split into two substrings: A and B . If n is not even, $LEN(A)$ is equal to $LEN(B - 1)$.

Step 3 defines the byte length of (b) . This is used in Step 6a.

Step 4 defines the byte length of (d) . This is used in Step 6c.

Step 5 defines the fixed block (P) as the initial block for the invocation of the PRF function in Step 6ii.

Step 6 is the iteration loop for the Feistel rounds, which is executed 10 times and consists of nine individual steps: 6i to 6ix as shown in Algorithm 4:

Algorithm step 6i to ix (iteration loop)

6. For i from 0 to 9:

$$(a) \text{ Let } Q = T \parallel [0]^{(-t-b-1)\text{mod}16} \parallel [i]^1 \parallel [NUM_{\text{radix}}(B)]^b$$

$$(b) \text{ Let } R = PRF(P \parallel Q)$$

$$(c) R \parallel CIPH_k(R \oplus [1]^{16}) \parallel CIPH_k(R \oplus [2]^{16}) \dots CIPH_k(R \oplus [\lceil d/16 \rceil - 1]^{16})$$

$$(d) y = NUM(S)$$

(e) If i is even, let $m = u$; else, let $m = v$

$$(f) \text{ Let } c = (NUM_{\text{radix}}(A) + y) \text{mod. radix}^m$$

$$(g) \text{ Let } C = STR_{\text{radix}}^m(c)$$

$$(h) \text{ Let } A = B$$

$$(i) \text{ Let } B = C$$

Algorithm step 7

7. Return $A \parallel B$

Discussion

In Step 6a, the tweak, T , the substring, B , and the round number, i , are encoded as a binary string Q . When B is encoded as a byte string, the variable b is the number of bytes that are output.

In Step 6b, the *PRF* function is applied to the concatenation of *P* and *Q* to produce a block, *R*.

In Step 6c, *R* is truncated or padded to a byte string *S* so that the output length matches in *d* bytes. *S* represents the string of the first *d* bytes of the following string of $\lceil d/16 \rceil$ blocks.

In Step 6d, *S* is converted to a number, *y*.

In Step 6e the length, *m*, for this Feistel round is determined.

In Step 6f *y* is added to the number represented by the substring *A*, and the result is reduced modulo the m^{th} power of radix, yielding the number *c*.

In Step 6g the output from step 6vi is converted to a numeral *C*.

In Steps 6h and 6i, the roles of *A* and *B* are swapped for the next round.

In Step 6h the substring *B* is renamed as the substring *A*.

In Step 6i, *C*, which is the modified *A* is renamed as *B*.

This completes one round of the Feistel structure for FF1.

Step 7 After the tenth round of the Feistel structure, the concatenation of *A* and *B* is returned as the output to Step 7 and encrypting the input string using the FF1 encryption algorithm is completed.

NIST have published all their encryption algorithms at [68] with example input and output values for each algorithm, which can be used to test software implementations of the NIST800-38G FF1 algorithm to verify correct output of our implementation. Conformance testing for implementations of the functions that are specified in this NIST800-38G were conducted within the framework of the Cryptographic Algorithm Validation Program (CAVP) and the Cryptographic Module Validation Program (CMVP) of NIST.

2.4.5 Weiss, Rozenberg and Barham (2015)

Weiss, Rozenberg and Barham [104] state that format-preserving encryption (FPE) schemes encrypt plaintext into a ciphertext while preserving its format (e.g. a legitimate identity number is encoded into a new but valid identity number), which allows the encrypted data to be stored and used in the same manner as the unencrypted data, such as being stored in a structured SQL table.

Weiss et al. believes that current FPE schemes are inefficient and insecure, but shows that a combination of the cycle-walking and the Rank-then-Encipher method yields an efficient en-

Algorithm 4 FF1.Encrypt(K, T, X) Algorithm 1 taken from NIST800-38G [21]

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$
 2. Let $A = X[1..u]$; $B = X[u + 1..n]$
 3. Let $b = \lceil \lceil v \cdot \text{LOG}(\text{radix}) \rceil / 8 \rceil$
 4. Let $d = 4 \lceil b/4 \rceil + 4$
 5. Let $P = [1]^1 \parallel [2]^1 \parallel [1]^1 \parallel [\text{radix}]^3 \parallel [10]^1 \parallel [u \bmod 256]^1 \parallel [n]^4 \parallel [t]^4$
 6. For i from 0 to 9:
 - (a) Let $Q = T \parallel [0]^{(-t-b-1) \bmod 16} \parallel [i]^1 \parallel [\text{NUM}_{\text{radix}}(B)]^b$
 - (b) Let $R = \text{PRF}(P \parallel Q)$
 - (c) $R \parallel \text{CIPH}_k(R \oplus [1]^{16}) \parallel \text{CIPH}_k(R \oplus [2]^{16}) \dots \text{CIPH}_k(R \oplus [\lceil d/16 \rceil - 1]^{16})$
 - (d) $y = \text{NUM}(S)$
 - (e) If i is even, let $m = u$; else, let $m = v$
 - (f) Let $c = (\text{NUM}_{\text{radix}}(A) + y) \bmod \text{radix}^m$
 - (g) Let $C = \text{STR}_{\text{radix}}^m(c)$
 - (h) Let $A = B$
 - (i) Let $B = C$
 7. Return $A \parallel B$
-

encryption algorithm that only encrypts format-specific properties. They only used format partitioning when a message is too large to be encrypted in its absence. They note that their encryption algorithm and libFTE, have the same security guarantees, since the underlying encryption scheme is the same in both.

2.4.6 New Analysis of FF3 (2017)

In April 2017, NIST announced that two researchers, Betl Durak and Serge Vaudenay had given NIST notification that they had successfully attacked the FF3 technique for FPE [19, 70]. NIST has subsequently withdrawn support for FF3 as technique for FPE.

The attack seems to be relatively impractical as the attacker has to have access to a significant number of encrypted outputs for known input data as well as know the 8-byte tweak for two sets of input data. A typical attacker would only have access to the encrypted output but not to multiple sets of input or tweaks.

2.5 Conclusion

FPE, when using FF1, is a compelling solution to de-identifying PII without damaging the database schema so that it can be used by the QA / test teams with ease and without compromising the integrity of the database, the integrity of the application, or compliance with POPI. The strength of employing FPE as an encryption solution is further bolstered by the finalization in 2016 of the highly respected NIST's Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption, which provides approved methods of FPE. This still holds true for the FF1 algorithm, which was used in this research.

3 Experimental Design

3.1 Introduction

In addition to Commercial Off-The-Shelf (COTS) software tools that implement FPE, to facilitate the de-identification of PII, there are several open-source software libraries. Both open-source and commercial offerings were reviewed for use in the experiments. However, since the open-source solutions are very cost effective to implement and easier to integrate with all IT systems, the chief objective was to find an open-source solution that worked effectively and could be implemented with any of the databases used in the experiments. As the primary failing with encryption techniques can usually be traced back to the implementation, rather than the algorithm, the software and implementation had to fully support NIST800-38G, to ensure the algorithm was secure.

As the open-source software libraries purport to support FPE as per NIST 800-38G [21], their fitness for purpose was evaluated and one selected. Consequently, it was not necessary to write the encryption module from scratch. A further objective in selecting tools was to find an open-source tool that could be used effectively with a scripting language such as PERL or Python, as these languages are easy to build into an ETL process, and are the author's preference (rather than C or Java).

After the tool selection was completed, the de-identification script was created and then run and tested on the Lending software deployed in Colombia and Botswana. Entirely different versions of the Lending software were implemented in each country. The Colombian version was first deployed in 2014 and then discontinued in November 2016 and the Botswana version was first implemented in September 2017. As such, the database schema of each was completely different. This enabled more granular testing of the solution in the experiment, and ensured that the solution met the company's needs.

POPI compliance was a requirement, as the Lending software was developed in South Africa (S.A.) and the QA testing was performed in S.A. and the commencement date for POPI will be announced soon [63]. De-identifying the dataset used by the QA team ensured that the POPI compliance requirements were less onerous in terms of reporting, and the risk of a PII breach was reduced, which is a reportable event under POPI.

3.2 Tool Selection

3.2.1 Commercial Software Review

Reviewing the COTS software tools available highlighted the strengths and weaknesses of the tools available and provided context for evaluating open source solutions.

HPE Voltage HPE acquired Voltage Security in 2015 [101] and added it to its HPE Security - Data Security suite software [37]. The company is currently developing new software in the FPE and Identity-Based Encryption (IBE) space. HPE Security - Data Security has been awarded several patents for encryption technology by the U.S.A. Patent Office [39], but the company have issued a letter relating to its patent rights, such that HPE grant access to a non-exclusive license for companies wishing to implement the FPE encryption technology [47]. The HPE FPE patent covers two techniques for FPE defined in NIST 800-38G [21]: FF1, also known as FFX mode, and FF3, also known as BPS mode. Both modes make use of a Feistel network to perform the encryption. HPE SecureData includes FPE [21], Secure Stateless Tokenisation (SST), Stateless Key Management, and data masking and supports the pseudonymisation guidance of the GDPR.

Semtek Innovative Solutions Semtek Innovative Solutions was acquired by Verifone in 2010 [15]. The company was involved in the FF2 or VAES scheme for FFX, which was included in the draft NIST 800-38G guidance, but was removed before final publication [107]. Further details are not available as the old Semtek homepage <http://www.semtek.com> is no longer available [51]. Bellare, Rogaway, and Spies wrote an additional paper [6] discussing why VAES / FF2 should be included in the final NIST report, which is under active review by NIST. Nevertheless it was not included in the published version of NIST 800-38G. Verifone used the Semtek and Safenet solutions for point-to-point encryption [29] and have since moved on to electronic payment transactions. It is likely that it will be included in a later version of NIST800-38G.

Protegrity Protegrity [79][49] provides a variety of data security methods for protecting data in database environments including “big data” databases. Protegrity also supports patented Protegrity Vault-less Tokenisation, data-masking, strong encryption and data-type preserving

encryption. Protegrity has a strong bias towards tokenisation rather than encryption. Tokenisation is a process whereby a piece of sensitive data is replaced with a token, which is then used in place of the sensitive data. The tokens are useless by themselves without access to the original lookup table. This is one of the first methods used to protect credit card numbers (CCN) in databases. Protegrity have released a number of articles arguing that tokenisation is superior technology than FPE [80], even though tokenisation solutions require far more disk space to operate as each token needs to be stored in a lookup table. FPE uses far less space, as it only requires that the secret key and tweak are retained to enable encryption and decryption.

Protegrity helps enforce Separation of Duties (SoD) by placing an obfuscation layer between the database and all users, including data base administrators (DBA). This means that the data that a privileged user sees is also encrypted or tokenised before they receive it.

Prime Factors EncryptRIGHT Prime Factors EncryptRIGHT [78] provides an enterprise-wide encryption suite, which can encrypt or tokenise data and manage all keys from a central console. It runs on all major operating systems, including mainframes. Prime Factors EncryptRIGHT focuses more on tokenisation than FPE [78].

Vormetric Vormetric (<http://www.vormetric.com>) [103] was purchased by Thales in 2016 [97]. Vormetric Orchestrator implements protection of personal data through pseudonymisation and encryption [102]. The solution allow users to choose between AES or FPE, therefore avoiding database schema changes.

Gemalto SafeNet SafeNet was one of the early creators of FPE software, but was acquired by by Gemalto [30] in 2015 and the SafeNet website (<http://www.safenet-inc.com>) was absorbed into Gemalto's website [28]. Gemalto sell FPE software that can encrypt both structured and unstructured data [31].

MIRACL MIRACL [64] is an enterprise encryption scheme that secures the people and applications needed to run a digital business. It implements BPS in C/C++ and offers Affero General Public Licensing or commercial licensing.

IRI FieldShield This is primarily data masking software [90] for data warehouse, business information reporting, and big data manipulation, but also has a module that performs FPE [91].

3.2.2 Evaluation of Available Open-source Software Tools

From the outset of this research, using open source software tools was identified as a preferential solution.

LibFFX: The FFX Mode of Operation for Format-Preserving Encryption This software library was tested first, as the author's preference was to use a scripting language for the database conversion in which he has extensive experience. The library worked sufficiently well for the needs of this research, although it was not able to de-identify special characters, so some enhancements were made to it.

This library is written in Python by Kevin Dyer [23] and implements the FF1 and FF3 modes of operation for FPE. It does not implement the FF2 mode of operation as this was excluded from the published version of NIST 800-38G [21]. The implementation was tested by the author to work with message sizes between 2 and 128 characters and with radix values between 2 and 36 inclusive. LibFFX implements a maximally-balanced Feistel network with a constant number of ten rounds, independent of message sizes, as per Bellare's paper [6]. Bruce Schneier [86] recommended that more rounds be implemented, however, Bellare and Rogaway refuted this [7, 6] and in this experiment it was not necessary, as the de-identified data was not being supplied to a high-risk group [25].

Pyffx Pyffx is a Python implementation of the FFX2 technique of encryption, also known as VAES. FFX2 was included in the first draft of NIST800-38G but was removed from the final publication as it did not offer the required 128 bits of encryption. The researchers submitted a revised version of the algorithm, but NIST did not have time to test and include it, in the final publication. This is well documented and very easy to use. If NIST do allow the FFX2 algorithm to be used in a future special publication, Pyffx would be a preferable library to use.

LibFTE Created in 2014, the LibFTE [54] encryption library is a hybrid of C, C++, Python, Flex, and Bison. One of the authors, Kevin Dyer, wrote LibFFX. The authors created a custom regular expression parser to ensure they could implement their nondeterministic finite-state automata relaxed-ranking algorithm. They provide interfaces in C++, Python, and JavaScript. The library can be downloaded from GitHub [48].

Format-Preserving Encryption Library for Java This library was written by Kai Johnson and distributed by Sourceforge.net [45]. The package implements the FF1 and FF3 methods of FPE discussed in NIST Special Publication 800-38G [21]. It also implements the A2 and A10 parameters for FF1, which are implementations of "The FFX Mode of Operation for Format-Preserving Encryption" by Mihir Bellare, Phillip Rogaway, and Terence Spies [7]. The library has been tested with the test databases provided by NIST to prove that it works correctly [71]. The interface that was provided was written in Java. The library was very promising, but uses C and Java. This could have been integrated into a Ruby de-identification script using the JRuby module.

Botan Botan [52] is a comprehensive cryptographic library written in C++ and Python, with a Python application programming interface. Jack Lloyd was the primary developer. The FPE functions are only found in a C header file, however, so are not available in the Python API. The build program was written in Python and C. FPE encrypts data such that the ciphertext has the same format as the plaintext. Botan can be used to encrypt credit card numbers while maintaining a valid checksum such that the ciphertext was also a credit card number with a valid checksum, or similarly for identity numbers or names and addresses. The scheme implemented in Botan was FF1, as described in the paper "Format Preserving Encryption" by Bellare, Ristenpart, Rogaway, and Stegers [5]. Further development is continuing with Botan and a Python API will be included in the future.

Fp_ruby_rc4.rb Fp_ruby_rc4 is an FPE library [88] implemented in Ruby but it uses the RC4 algorithm, which was cracked over ten years ago. It also required that the key length was always a power of 2.

Prismacloud - Privacy and Security Maintaining Services in the Cloud The E.U. horizon 2020 Prismacloud research project [85] is a project that intends to provide an encryption module for all cloud-based data storage. The library exposes an interface which uses Perl-compliant regular expressions to identify the source data structure and then encrypt it using the same data structure, thereby relieving the programmer of the burden of making the correct encryption algorithm choice and potentially making a mistake. It is an auspicious project but will only be complete in 2020 [53].

FFX codec FFX codec [14] only encrypts unsigned integers, but it increases the size of an integer to 32-bit or 64-bit integer, therefore is not really a format-preserving encryption scheme.

3.3 Implementation

The data used in the implementation of the experiments came from Bayport Financial Services (Bayport) [4], a financial service provider based in South Africa. Bayport offers payroll loans to customers in six countries in Africa and two countries in Latin America. The company also offers retail loans to customers in South Africa. The financial software was developed and tested in S.A.. As such, it is a requirement of POPI that Bayport either comply with all the clauses of POPI or de-identify PII, where possible, so that POPI is taken out of scope. The data used in this research is taken from the Botswana and Colombia implementations as they use different software packages and database schemas for the handling of customer loan information and PII.

LibFFX [23] was chosen for testing as it provided a Python interface. PyFFX [81] also uses a Python interface and was briefly tested and found easier to use than LibFFX, but only supported the FFX2 technique, which was excluded from the final draft of NIST800-38G. The data was loaded from the production database using a Python script and all the PII fields were encrypted using the LibFFX library, as shown in Figure 3. The de-identified data was then loaded in the QA database.

Considerable data cleaning was necessary to ensure the data was in the correct format. It was evident that the database was not well designed. As a result, data that should have been stored as integers was often stored in VARCHAR fields. This meant that users were able to store any alphanumeric data or special characters in those fields. There was also no input validation in the application that prevented users from inputting incorrect data. As the algorithm worked differently depending on whether the input was integer or string, the data had to be converted to the correct type before performing the de-identification.

In some cases, customer names were stored in fields that should have been reserved for integers. These were removed as they were clearly typographical errors.

There were instances of integers that were prepended with a “-” sign, which were also typographical errors, so the “-” sign was stripped.

Where integers were stored as “none”, these were converted to 0.

Integers that were stored as empty or “null”, were converted to 0.

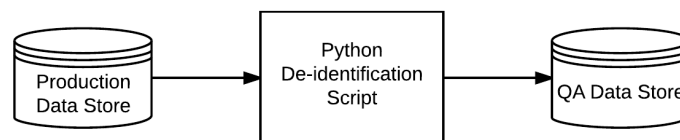


Figure 3: Process of de-identifying the PII data

Most of the data-cleansing issues could have been resolved by enforcing input filters in the application that ensured that only the correct type of data could be captured. This would help ensure that a lengthy data-cleansing exercise would not be needed prior to a POPI de-identification exercise. Input filters were used in the application used in Experiment-2, so no data-cleansing was needed with the Experiment-2 dataset.

The LibFFX library does not support special characters, so these had to be removed or handled in an alternative manner. Email addresses presented a problem as they included the “@” symbol. These were resolved by splitting the username and the domain name, encrypting each part separately and then joining the encrypted username and encrypted domain name with the “@” sign in the middle. A “.com” was appended to the end so that the encrypted form of the email address still looked like an email address. The encrypted username and encrypted domain name were encrypted as a 10 character block, so the resulting string did not look much like an email address without the “.com”, for example it was “wedlydmpud@qactpunfem” before appending the .com.

The FF1 algorithm is designed to work in either string mode or integer mode. If the input is a string, then the output is a string. If the input is an integer, then the output is an integer.

The original database was 10GB for Experiment-1, and the conversion took 17 minutes to run. The new database was then made available to QA staff for testing.

For Experiment-2, the database was 12GB and the conversion took 30 minutes to run.

The following PII fields were identified and encrypted:

name, surname, maiden name, physical address, postal address, email address, postal code, identity number, passport number, telephone numbers, next of kin details, bank account number, and birth date.

Birth date is not specifically a personal identifier, but it could potentially be used with other information to re-identify PII [25]. There is discussing of birth data encryption in section 4.1.6 on page 70.

No credit card numbers (CCN) were processed in this database, although FF1 and FF3 were designed to encrypt CCN's securely to comply with PCI-DSS. Therefore, this implementation would be sufficiently secure to comply with the Payment Card Industry Data Security Standards' requirements [77], as long as the secret key was properly secured.

Figure 4 shows the high-level architecture of the application. In production, the production database was running on the database server in the database layer. A separate instance of the application was built for QA testing. For the QA testing instance of the application the de-identified database was loaded on the database server in the database layer.

NIST have published all their encryption algorithms at [68] with example input and output values for each algorithm, which can be used to test software implementations of the NIST800-38G FF1 algorithm.

3.4 Experiment-1: Colombia

Experiment-1 was performed against the database for the implementation of the company's lending software known as Flexifin. This was first deployed in Colombia in 2014. The QA component of the software development lifecycle (SDLC) was undertaken in S.A., which is

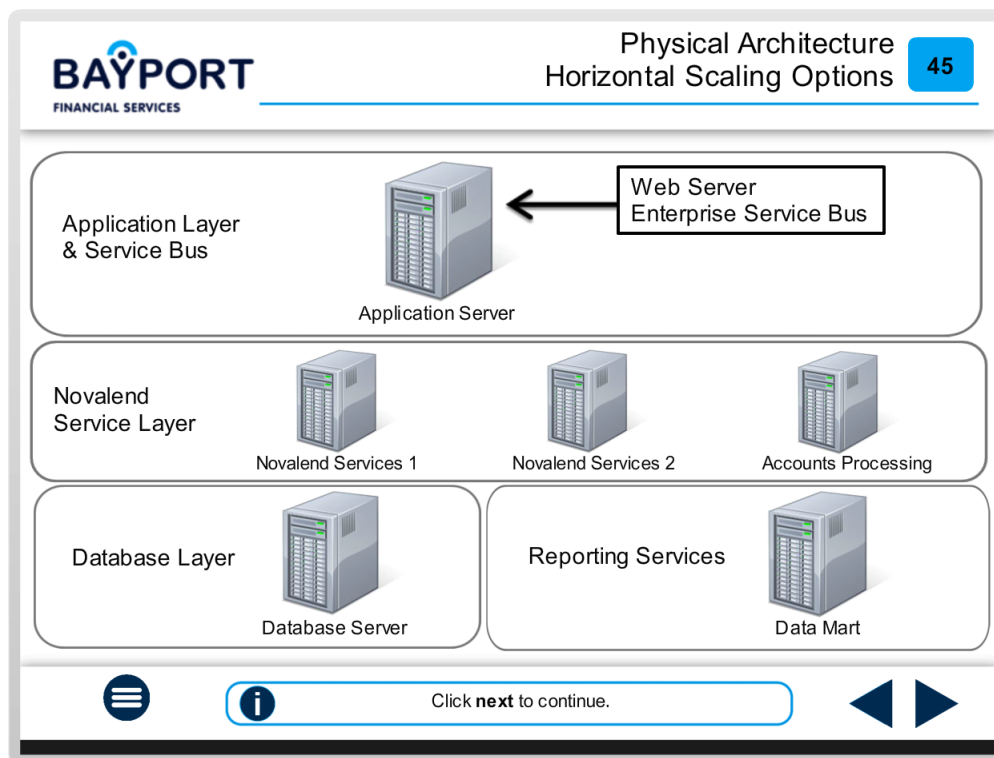


Figure 4: QA staff access to QA Data store

why it was important that the company comply with POPI. This software is now defunct as it was replaced in 2017. As a result, getting access to people who understood the database schema and the testing process was quite difficult.

3.4.1 Procedure for running the de-identification script

The required libraries to run LibFFX with Python were installed. Algorithm 6 shows the Python code written to call the LibFFX library and run the FFXInteger encryption function. The output from FFXInteger was a type(FFXInteger), so this had to be converted back to a string to be stored in a VARCHAR field in the database. The integer data in the database needed a lot of cleaning, as they were generally stored in VARCHAR fields and alphabet characters were often stored therein. This posed a problem for the FFXInteger function as the correct alphabet had to be chosen so that the outputted data had a similar format to the inputted data. The author wrote a function to convert all “null” data, minus-signs, alphanumeric characters or spaces to integers. This precluded the data from being re-identified correctly, but as these were data input-errors, re-identification was not important.

Algorithm 6 shows the code to load the data from the database table, encrypt it and then upload to the QA database.

Algorithm 5 Experiment-1: Python Code to perform the encryption

```

1  #!/usr/bin/env python2.7
2  from __future__ import print_function
3
4  import pymysql
5  import datetime
6  import ffx
7  from ffx import FFXInteger
8  from datetime import datetime
9
10 key1 = '36-random-alphanumeric-characters'
11 private_key = '16-random-alphanumeric-characters'
12 time = str(datetime.now())
13 print("start", time)
14 starttime = datetime.now()
15     C = None
16     radix=36
17     try:
18         if row == "":
19             C = ""
20             exit
21     elif type(row) == type('a'): #alphanumeric
22         unique_chars    = list(set(row))
23         string_len      = len(row)
24         radix = 36 #alphabet
25         K = FFXInteger(key1, radix=radix, blocksize=128) #was 16
26         T = FFXInteger(private_key, radix=radix, blocksize=16) #tweak
27         #tweak must be same character set as input
28         M1 = FFXInteger(row, radix=radix, blocksize=32)
29         #input ffx_obj = ffx.new(K.to_bytes(16), radix)
30         C = ffx_obj.encrypt(T, M1)
31     elif type(row) == type(100): #numeric
32         string_len      = len(str(row))
33         radix = 10 #alphabet
34         K = FFXInteger(key1, radix=radix, blocksize=32)
35         T = FFXInteger(private_key, radix=radix, blocksize=16) #tweak
36         M1 = FFXInteger(row, radix=radix, blocksize=16)
37         #input ffx_obj = ffx.new(K.to_bytes(16), radix)
38         C = ffx_obj.encrypt(T, M1)
39     elif row == "None":
40         C = "None"
41 return C #— end encrypt function

```

Algorithm 6 Experiment-1: Performing the encryption on a SQL table

```

1  #--1 Stephen 0.4 accounting.restructuring_batch_line
2  conn = pymysql.connect(host='127.0.0.1',
3  port=3306,
4  user='*****',
5  passwd='*****',
6  db='ff1')
7
8  cur = conn.cursor()
9  cur.execute("SELECT_id,first_name,identity_number,last_name,passport_number
   _from_accounting.restructuring_batch_line_where_id_is_not_null")
10 data = []
11 for row in cur:
12     first_name = encrypt(row[1],private_key)
13     identity_number = encrypt(row[2], private_key)
14     last_name = encrypt(row[3], private_key)
15     passport_number = encrypt(row[4], private_key)
16     data.append([first_name,identity_number,last_name,passport_number,
17     row[0]])
18 cur.executemany("UPDATE_accounting.restructuring_batch_line_SET_first_name
   =%s,identity_number=%s,last_name=%s,passport_number=%s_where_id=%s",
   data)
19 conn.commit()
20 cur.close()
21 conn.close()

```

Figure 5 on the following page shows the Entity Relationship Diagram (ERD) between the tables containing PII in the Experiment-1 database. Some of the tables and fields have been excluded from the ERD.

3.4.2 Output from the first run of the de-identification script

Figure 6 shows the results of the first connection to the application after the initial encryption. During the QA and debugging process, additional SQL tables were discovered that contained PII and these were subsequently de-identified.

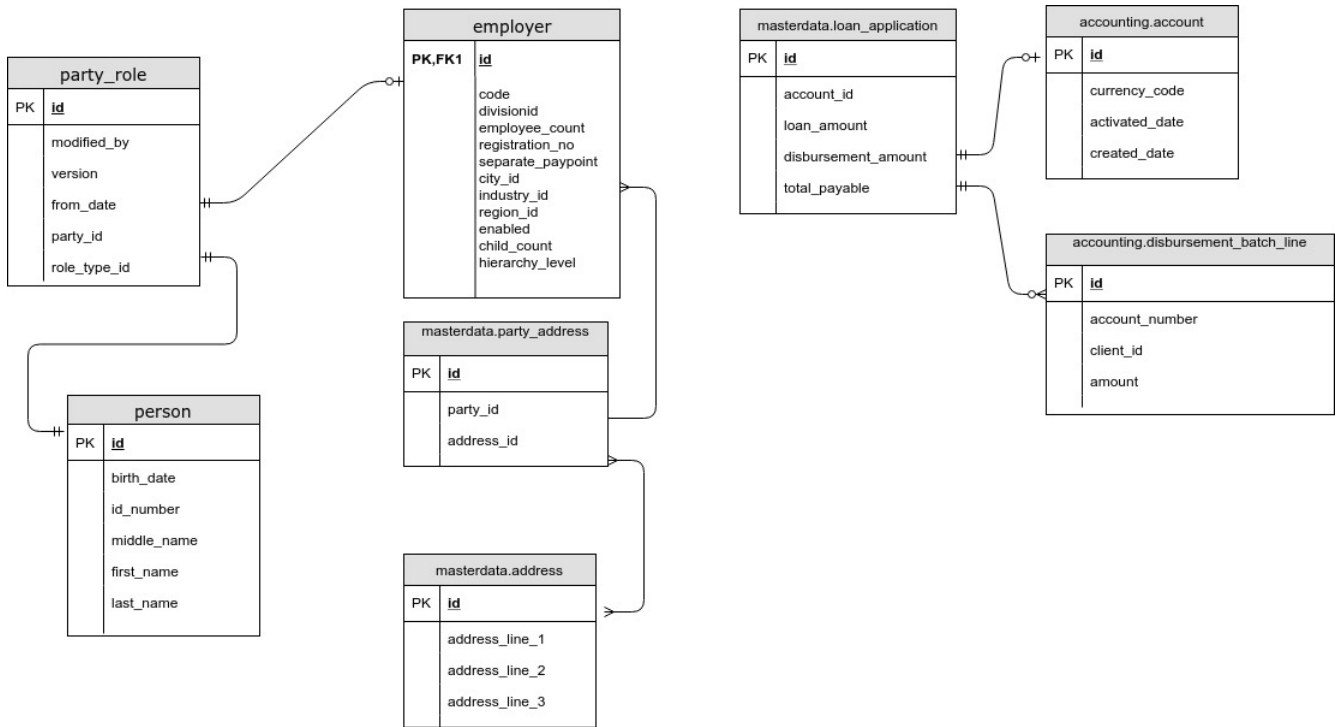


Figure 5: Experiment-1: Entity Relationship Diagram

Flexi Fin

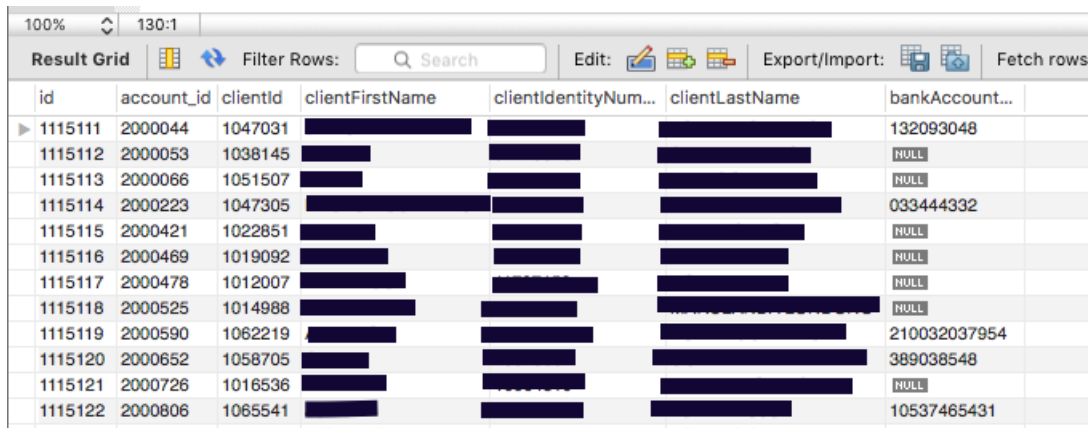
MY WORK FIND A CLIENT SUBMISSIONS COLLECTIONS SETTINGS

Agent Management

Manage Agent List

First Name	Last Name	Role	Home Branch
00bj3k6e22c9d1351aebliu	onb2z0wxijwm2q3ty9uu8rnjvmw	Asesor Externo	BOGOTA - CENTRO
00bj3k6e22c9d1351aebliu	wqe5yx0k9zks72svoggwjggu57	Asesor Externo	GIRARDOT
00bj3k6e22c9d1351aebliu	81ybdh1f3hstz05ojd4f1bu89hklr	Asesor Externo	PEREIRA
00bj3k6e22c9d1351aebliu	ga2q9skg4khe2pdg91urun9s47q	Asesor Externo	BOGOTA - CENTRO
00bj3k6e22c9d1351aebliu	xg4zq7jwrjrslgk1wpk45yclaa25c	Asesor Externo	BOGOTA - CENTRO
00bj3k6e22c9d1351aebliu	ara0m9kok49f0asw05w0sj7ab6d	Asesor Externo	BOGOTA - CENTRO
00bj3k6e22c9d1351aebliu	453x4v6iux9a1omxvvgf6cwhrh7	Asesor Externo	BOGOTA - CENTRO
0110mg9y8yhonz2w16r5rkl	tmbdw6k4m7avybjkd3mouxsknt	Asesor Externo	CALI
0110mg9y8yhonz2w16r5rkl	s5wk0b0lhq90fcv7sekaqj7d29v5i	Asesor Externo	BARRANCABERMEJA
02ef821bzbz5hs0o796ipon	x3g0rvrznbtw17xztanuinmn47	Asesor Externo	IBAGUE
02kin9n1v4tchbcmwhczc	mfh9x93lwyjtrit3io7uc77ymwf9k	Asesor Externo	POPAYAN
032vib23i5ozvn0kslud8iq	4g5v3olyeepkw6uij2y99c7gj0s7k	Asesor Externo	VILLAVICENCIO
032vib23i5ozvn0kslud8iq	uutriu7sdn4vwhzpktjk19d6dv5g	Asesor Externo	SINCELEJO
03iv8uka6ghfz557jmsvli7	t7k4nxtl1hw2b5kyjvuyvtxs4wdc	Asesor Externo	QUIBDO
03nndi3ysyjn0g2i8rlykfr	i38ubumv748kqahm7glrgy3th9rr	Asesor Externo	BOGOTA - CENTRO
03Edaaptdehazk340r8l	96ubal9eapcubek1at6lkdum	Asesor Externo	MONTERIA

Figure 6: Experiment-1: Initial Encryption



The screenshot shows a database result grid with the following columns: id, account_id, clientId, clientFirstName, clientIdentityNum..., clientLastName, and bankAccount... The grid contains 13 rows of data. The first row has a value of 132093048 in the bankAccountNo field. The second, third, fourth, fifth, sixth, seventh, eighth, ninth, and tenth rows have NULL values in the bankAccountNo field. The eleventh row has a value of 210032037954, the twelfth row has 389038548, and the thirteenth row has 10537465431.

id	account_id	clientId	clientFirstName	clientIdentityNum...	clientLastName	bankAccount...
1115111	2000044	1047031				132093048
1115112	2000053	1038145				NULL
1115113	2000066	1051507				NULL
1115114	2000223	1047305				033444332
1115115	2000421	1022851				NULL
1115116	2000469	1019092				NULL
1115117	2000478	1012007				NULL
1115118	2000525	1014988				NULL
1115119	2000590	1062219				210032037954
1115120	2000652	1058705				389038548
1115121	2000726	1016536				NULL
1115122	2000806	1065541				10537465431

Figure 7: Experiment-1: null values

Figure 7 shows some of the integer “NULL” values in the bankAccountNo field.

Review Employer Referral

Name: * Industry: *

Reg No: Employer Type:

City: Founding Date: *

Employers Code: Deducting Trading Entity: *

Labour environment:

Average employment period:

Anticipated retrenchments in ne...

Full time employees: Employee reduction over last year:

Employees with less than 1 year ...

Undefined worker count:

Contract workers: Pensioned worker count:

Employees on existing loan sche...

Temporary worker count:

Key Contacts

Name	Position	
EEAAAANEI GCGNA	LÍDER PROYECTO DE NOMIN	View
DDAD RISSR	DIRECTOR DE NÁMINA	View

Lending Trading Entity

Name	
Bayport	
CoopMicrocredito	

Note:

Figure 8: Experiment-1: Employer Referral

Figure 8 shows the Employer Referral screen with all PII de-identified.

Figure 9 shows all customer PII data encrypted after the final run of the Experiment-1 de-identification code.

3.4.3 Challenges

The software that was tested in Experiment-1 was discontinued in 2016 and many of the development, QA and DBA staff that worked on the software no longer worked for the company. As such, finding all the PII in the database was somewhat challenging and rather protracted. Several debugging and QA cycles were needed to identify all the PII.

The database design dictated that there were thirteen individual SQL tables containing PII, thus there were thirteen SQL select statements required to isolate and de-identify all the PII information.

Getting time with the few remaining QA staff and especially with Development staff was a challenge especially with a variable software release schedule for the new Lending software, which occupied much of their time. Thus it was decided to focus testing efforts on the QA team.

One significant issue was that staff usernames were stored in the same table as customer usernames, so after the first run all staff usernames were encrypted and staff users were prevented from logging in to the application. This was overcome by identifying staff identities and ensuring these were not de-identified.

One of the standard operating procedures was for QA staff to search for customers by their first name or surname, which was not possible with a de-identified dataset. This was highlighted before testing began. The change to the protocol required that QA staff search for clients by their client ID number, which was a seven-digit number. Since this was a POPI compliance issue and had to be performed, QA staff indicated that this change was not too much trouble and that they were comfortable with the new process. QA staff were supplied with a list of customer numbers to make their testing simpler. This was a change from the usual testing process of searching for users and customers by first name.

Examples of client ID numbers:

- 1007257

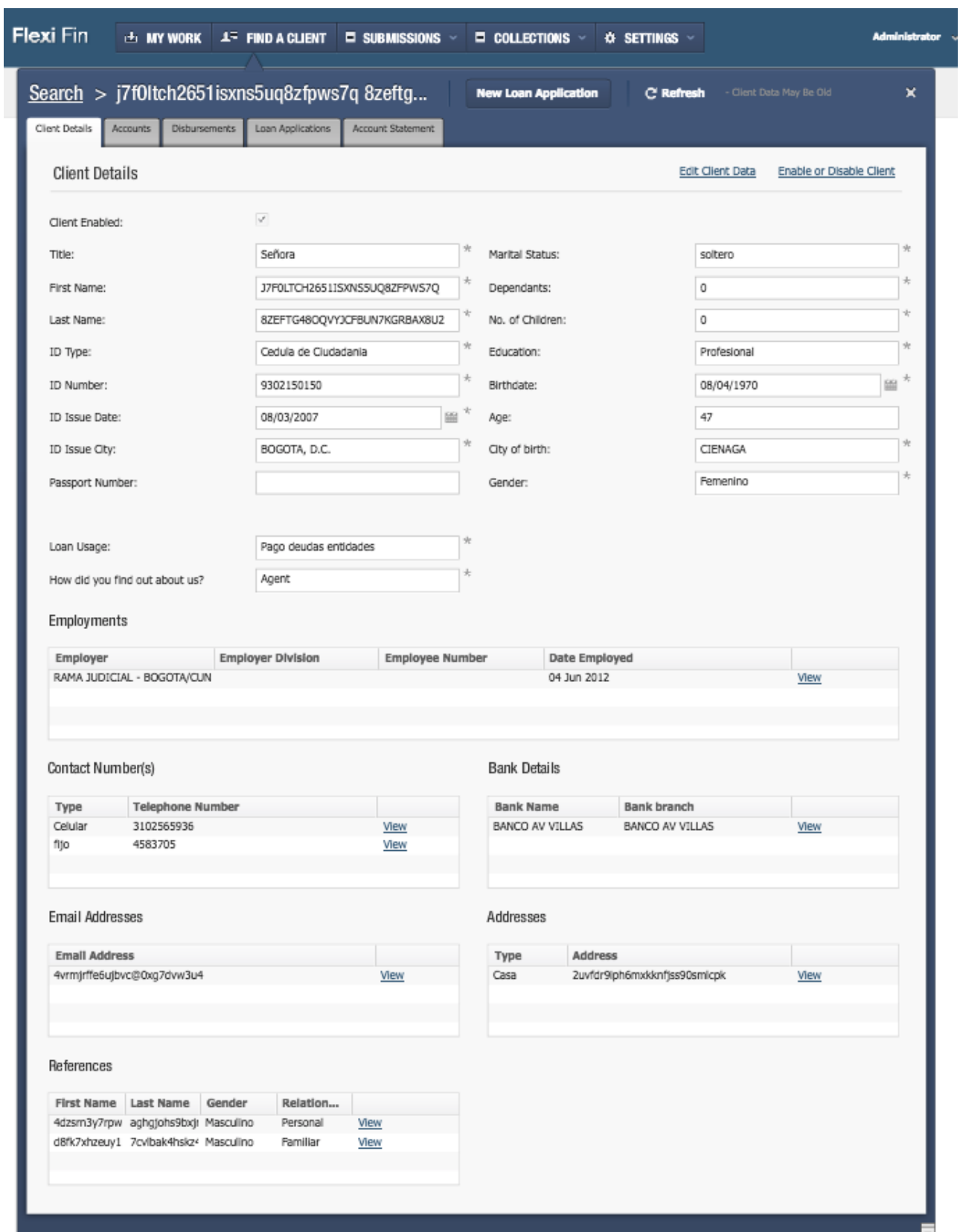


Figure 9: Experiment-1: All Customer Data Encrypted

- 1009217
- 1006925

Special characters and accented characters were not de-identifiable using the LibFFX module. A work-around was found by stripping all special characters out before encrypting. This did not impact the usefulness of the de-identified data, as it was encrypted anyway. It was not a requirement that this data would ever need to be re-identified so this was also not a problem.

The problem with email address encryption was discussed in section 3.3 on page 42.

It was anticipated that the QA team may complain about the clients' first names and surnames being an encrypted block of thirty two random alphanumeric characters, but none of the QA testers were unhappy with the output. It was expected to have to use a tokenisation approach to encrypt first names and surnames to make them a more human-friendly output. This would have been feasible based on a risk assessment of the risk of the QA user group and the risk of re-identification as per the Anonymisation Decision-Making Framework [25]. It was decided to set strings to 28 characters and integers to 10 numbers for Experiment-1, but this was changed to match the length of the input for Experiment-2 as that seemed like it would be a better solution.

There was no simple way to encrypt dates while retaining the age checking that the software required to originate loans. If the year was isolated, the potential outputs were 0000 to 9999. Valid year dates would be 1952 to 1999 based on the decision criteria regarding the age of the lender, since near-retirees did not qualify for loans. For the purpose of the experiment the date was set to "1970-01-01" as this verified that the de-identification mechanism worked correctly. For the production version of the de-identification code a more robust algorithm to de-identify the date would be needed, as hard-coding 1970 would eventually result in the lender being too old to qualify for a loan.

The author could have chosen to leave the year unchanged and to only de-identify the month and day, as this would adequately de-identify the date without breaking the Lending system's internal data integrity rules.

3.4.4 Program statistics

On the final run of the de-identification script for Experiment-1:

1,7 Million encryption operations were performed in 17 minutes.

501,363 SQL rows were processed.

Thirteen individual SQL tables included PII that needed to be de-identified.

The script loaded and encrypted the PII in a 10 GB database and then wrote to a new database.

This was acceptable in terms of the computing time required to de-identify data to become POPI compliant.

3.4.5 QA department testing

To test the Lending software, the QA team logged into a de-identified version of the software and ran through a series of pre-defined tests as shown in Table 3. The testing was structured such that a QA staff member stepped through the various sections in the Lending software to ensure that the software was working correctly and that all PII was de-identified. The sections are shown in the table, along with the corresponding test detail.

As issues were picked up in the first tests, the de-identification script was modified and re-run against the production database to create a new de-identified dataset.

There is discussion and output from the QA process shown in section 4.2.1 on page 73.

An "unstructured interview" was used to review the acceptance testing component of the experiment with the QA team.

3.5 Experiment-2: Botswana Novalend

Experiment-2 was performed against the database for the implementation of the company's new software known as Novalend. This was deployed in Botswana in September 2017. The QA component of the software development lifecycle (SDLC) was done in S.A., which is why it was important that POPI was complied with. The de-identification script used was similar to Experiment-1, except that the database structure was somewhat different so the SQL select and update statements were different.

Figure 10 shows the entity relationship diagram for the Masterdata schema for Experiment-2. The Management Information System, Batching, Accounting and Operational Data Store schemas have been excluded from this diagram.

3.5.1 Procedure for running the de-identification script

The procedure for the preparing the environment for the running of the de-identification script and running the script was the same as with Experiment-1. Some small improvements were made to the script for Experiment-2, but these were not material. The database structure was different for Experiment-2.

3.5.2 Output from the first run of the de-identification script

Figure 11 shows the first run of the QA instance of the de-identified dataset for Experiment-2, with the “Client Full Name” correctly de-identified.

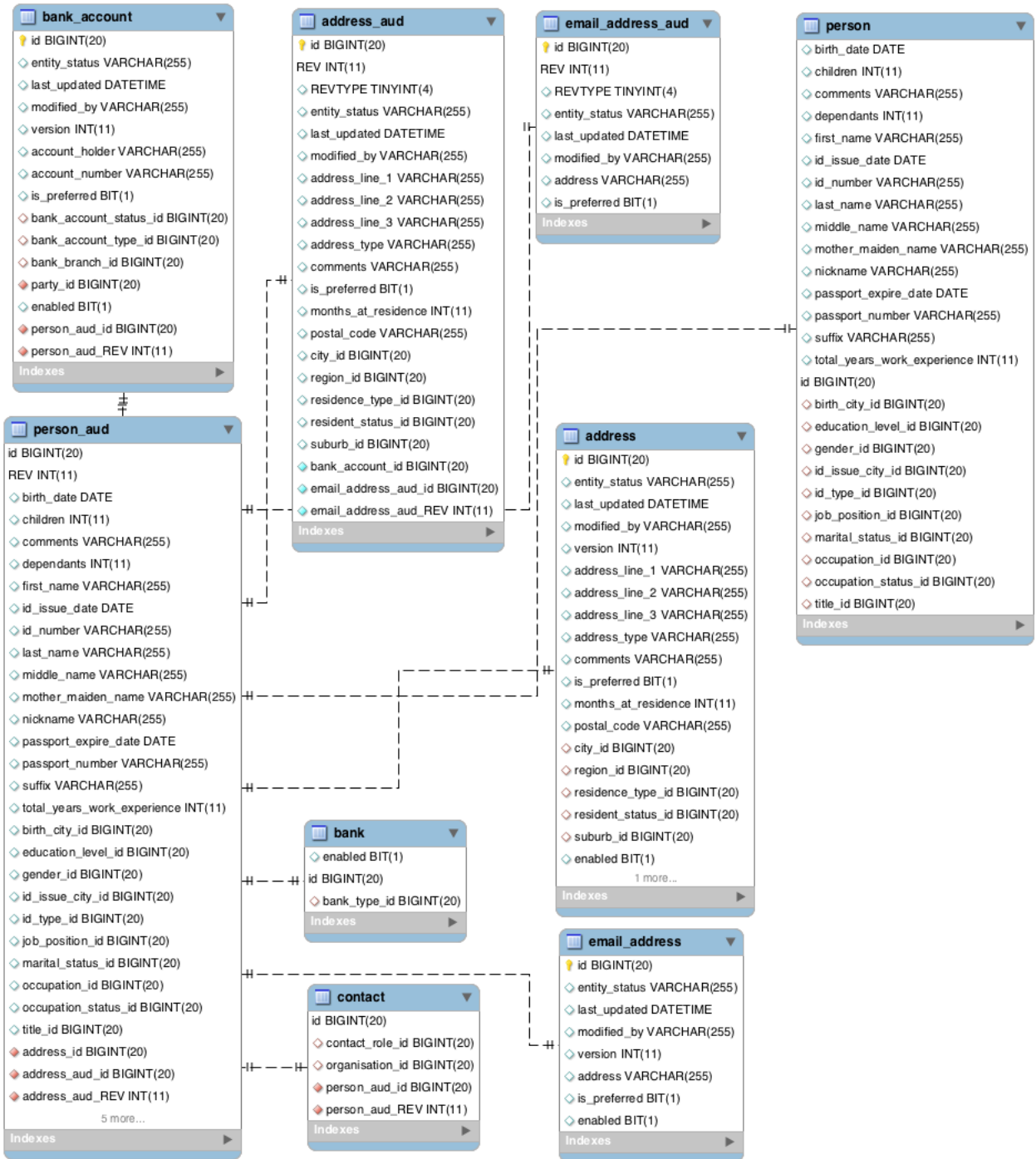


Figure 10: Experiment-2: Masterdata Schema ERD

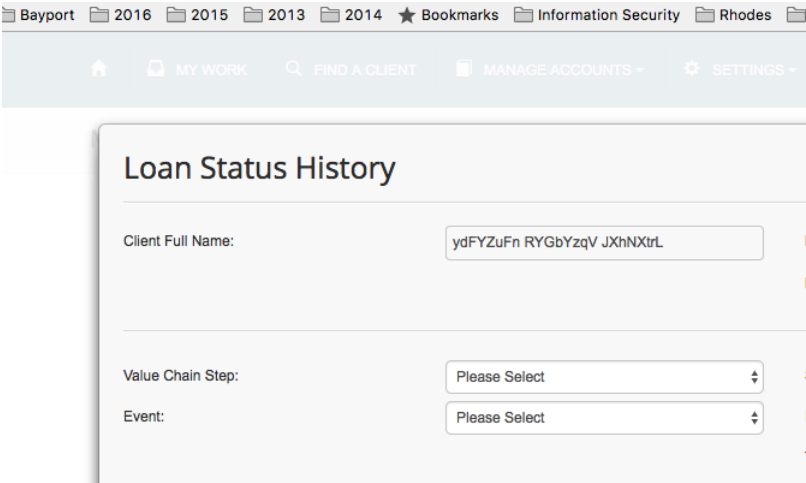


Figure 11: Experiment-2: First Run

The screenshot shows a web application interface for 'Manage Submissions'. At the top, there is a navigation bar with links for 'MY WORK', 'FIND A CLIENT', 'MANAGE ACCOUNTS', 'SETTINGS', 'REPORTING / BI', and 'ONLINE HELP'. Below the navigation bar, there is a 'Client Search' form with various input fields for searching clients. The search criteria include 'Find Client By' (set to 'Personal'), 'First Name', 'Last Name', 'National ID', 'Passport Number', 'Client Number', 'Bank', and 'Bank Account Number'. A 'Search' button is present, along with a 'Clear' button and a 'Show 50 results' dropdown. Below the search form, there is a table of search results with columns for 'First Name', 'Last Name', 'Client Number', 'ID Number', 'Passport Number', 'Employee Number', 'Employer', and 'Account Number'. Each row in the table contains de-identified data, with 'Edit' and 'View' links for each entry.

	First Name	Last Name	Client Number	ID Number	Passport Number	Employee Number	Employer	Account Number
Edit View	AqkEupNU	XYfXkqEY	1001148	4520383828	0567039502			
Edit View	DBnEqXIL	aNmzoewS	1001084	8490514712	2728674941			
Edit View	ERObiHXW	wOvrvbPN	1001134	6481853882	9862057456	908215418	Ministry of Labour & Home Affairs-PPS-DO	1010899
Edit View	EnsHwBoa	AJQeZhiZ	1001124	1506387260	2313336957			
Edit View	FUILMqMf	GuoTKTng	1001102	2061896815	6257671071			
Edit View	HHDzarzE	RFVvAGse	1001114	7828555087	9115753386			
Edit View	HOoxQmQb	BlvgzXNn	1001104	9399336107	1497287381	142214008	Ministry of Environment, Wildlife and Tourism-IC-PD	1002658
Edit View	JDngelot	bPGTsCZI	1001108	4935664118	6370343812	001211006	Ministry of Finance and Development Planning-PPM-PD	1012203
Edit View	KSSymuij	JFYjRksl	1001095	2262035086	0938611562			
Edit View	LGMpQhC	GaQhfUQd	1001118	2964703084	2576808568			
Edit View	NdyOHxYa	yHDHYvVU	1001136	0303510832	4303096442	066716400	Ministry Of Defence Justice and Security-PPS-PD	1004264
Edit View	OZmPFyBJ	gpYFPpfw	1001122	6380677483	2594307962	810923803	State President The Parliament Office-IC-PD	1012511
Edit View	PXIRVrxy	fxiVrDma	1001125	8387315745	0931407488			
Edit View	RIJQofr	Kejdtmjl	1001106	8680972577	3732285435			
Edit View	RKoiVTKU	DmENPvgK	1001092	5340818902	9916532900			

Figure 12: Experiment-2: All PII de-identified

Figure 12 shows all PII de-identified after the final run of the de-identification code.

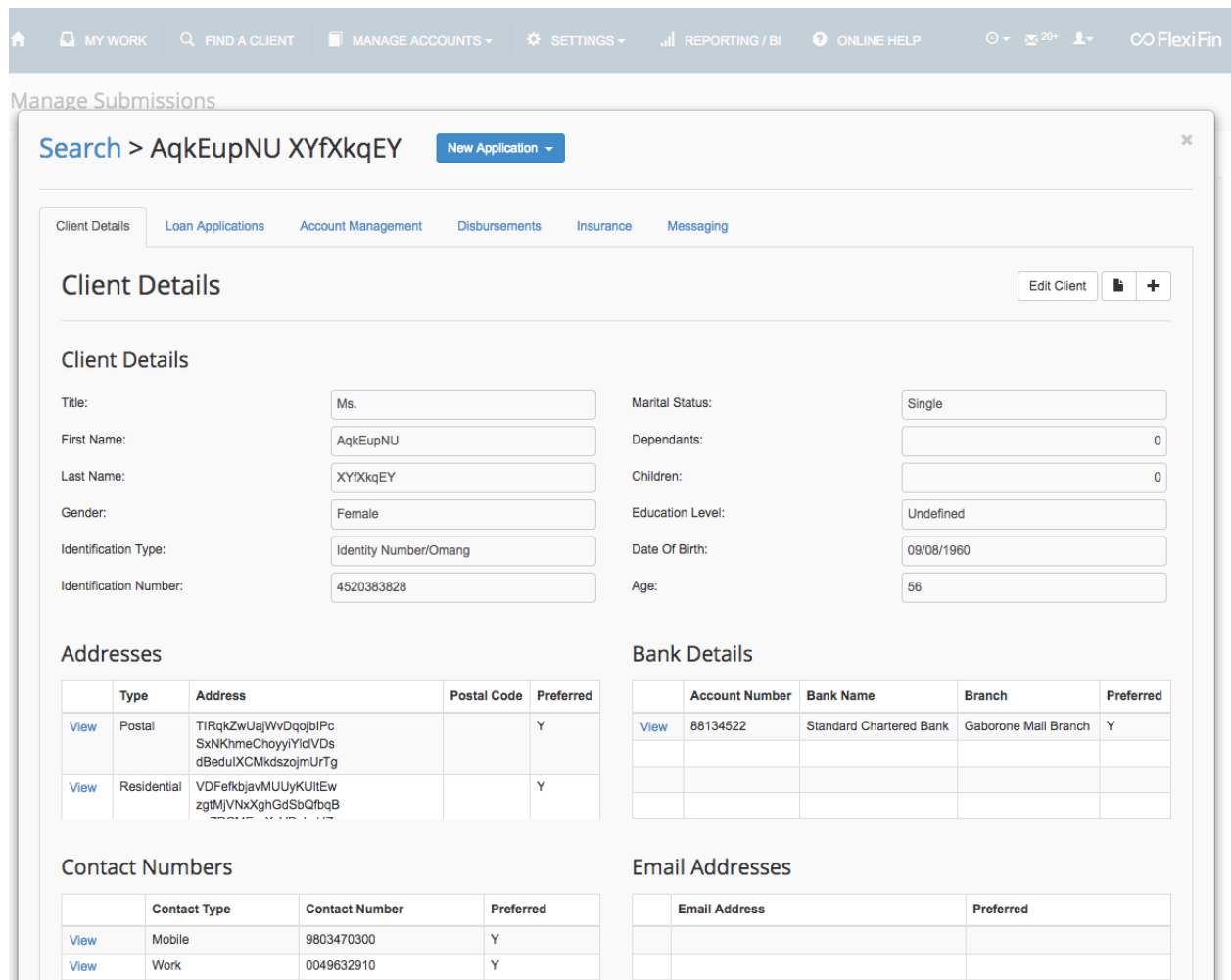


Figure 13: Experiment-2: Client Details

Figure 13 shows all client details de-identified.

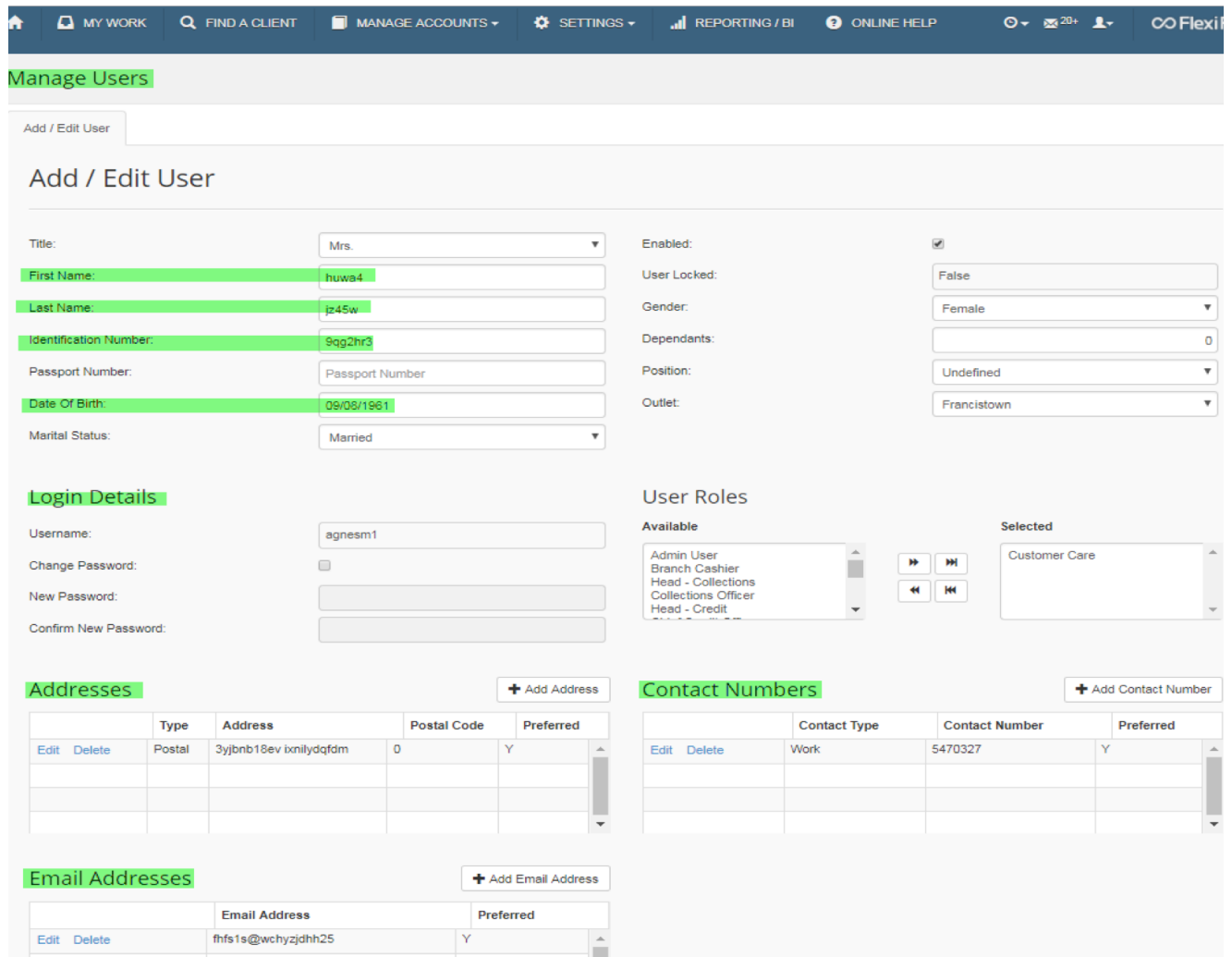


Figure 14: Experiment-2: Manage Users

Figure 14 shows all PII de-identified in Experiment-2 in the Manage Users section.

The Settings section shows the screens to Manage Users, Manage Third Parties, Manage Payroll Employees and Manage Outlets. All customer PII was correctly de-identified as shown in Figure 14 .

The screenshot shows the FlexiFin Dashboard with the following data:

Your Oldest Processes

ID	Process	Task(s)	Created
1117	System Data Import	Review Import	01/09/2017 - 10:44
2001	System Data Import	Select Import Type	02/09/2017 - 13:43
458	System Data Import	Upload	30/08/2017 - 09:19

Your Most Recent Clients

Title	Client Name	Identification Number
Mr.	85xsa0 twdafven	fh93g9zd6
Ms.	cv6myirjai oci4f	7s0403t5s
Mr.	hou341f22q8c6cd jz5k5	3s8wcioze
Ms.	5tdp4vfz6xk pu04oay	f9lad1uto
Ms.	e8se0 94gtwf	9eUsk6aja

Your Starred Notifications

Message	Source System	Date Sent
User initiated Reconciliation of Payroll batch (ID: 1000070) for employer: Barisma-DO h...	Post Reconciliation	21/09/2017 - 17:43
User initiated Reconciliation of Payroll batch (ID: 1000070) for employer: Barisma-DO h...	Post Reconciliation	21/09/2017 - 17:43
User initiated Reconciliation of Payroll batch (ID: 1000069) for employer: Barisma-DO h...	Post Reconciliation	21/09/2017 - 17:43

Navigation: Previous | 1 | 2 | 3 | 4 | 5 | ... | 124 | Next

Qualica Support | origin/flexifin-release-3.4.15 | b.172

Figure 15: Experiment-2: Dashboard de-identified

Figure 15 shows the dashboard de-identified.

3.5.3 Challenges

All the challenges were overcome in Experiment-1. There were no new challenges in Experiment-2.

3.5.4 Program statistics

On the final run of the de-identification script:

2,3 million encryption actions were performed.

221,684 SQL rows were processed

A 12 GB database was loaded and processed

Run time was 30 minutes

The script took longer to run than Experiment-1 partly because the SQL select statements were changed to select one field per table at a time rather than select all the fields at the same time. This was done to make processing of the different data types easier.

This was acceptable in terms of computing time required to de-identify data to become POPI compliant.

3.5.5 QA department testing

The QA testing process was the same for Experiment-2 as Experiment-1, except the software was somewhat different so there were different sections to test as per Table 4 on page 77.

An "unstructured interview" was used to review the acceptance testing component of the experiment with the QA team.

4 Analysis and Discussion of Results

4.1 Introduction

The experiments have shown that it is quite possible to de-identify PII easily without having to make changes to the database schema by using FPE as per NIST800-38G [21]. This was achieved, with relatively little programming to utilise the freely downloadable LibFFX library with Python to create a de-identification script to de-identify all PII in a SQL database. No changes to the SQL database schema or the application were necessary to provide a de-identified version for the QA test staff.

A NIST800-38G compliant encryption scheme was built to secure the PII in the QA version of the Lending Software for two separate versions of the Lending software, running, respectively in Botswana and Colombia. This was achieved by developing a Python script that read in all the PII from the Lending system's SQL database, invoking the LibFFX library to perform the encryption and then writing the encrypted data over the original SQL tables. The encrypted version of the database was supplied to the QA team for testing. A crucial requirement with the solution was that the key used to encrypt the data was kept private. The "tweak" [84, 21], or second password was used to help ensure that very short input strings were more secure when encrypted. The tweak does not necessarily have to be confidential and can vary for each encryption operation. It is used as an additional secret key for the encryption, similar to the salt, which is used with the MD5 algorithm, to achieve stronger encryption. The original text is effectively encrypted with two secret keys. The PII can only be re-identified using the correct NIST800-38G algorithm with the correct secret key and tweak.

Utilising the LibFFX library correctly was challenging, as it does not include any documentation. However, a working example was provided, which was used as a guideline to implement the library in the research. The library will encrypt either an integer or an alphanumeric string, depending on which mode is chosen.

When using the LibFFX library, the structure of the key has to match the structure of the input. Hence, an alphanumeric key was used for alphanumeric input and an integer key was used for integer input. The pyffx library was substantially easier to use but only supported the FFX2 technique of encryption, which was not supported by NIST800-38G. This was included in the

draft version of NIST800-38G, but was removed from the final version. The FFX2 technique is under review and may be included in a future version of NIST800-38G, in which case the pyffx library would be well worth considering.

In undertaking the experiment, a number of aspects were identified that could be refined to ensure that the encryption output was as user friendly as possible. Adjusting the size and form of the output so that it is looked similar to the input data encourages compliance from the QA team. Ensuring that the encrypted data was user-friendly helps ensure that the softer side of IT Change Management was easier to manage, since the encrypted data looks similar to the original data. Data that is difficult to read makes the QA staff members' jobs harder to perform.

It is not the responsibility of the QA staff to check the integrity of the customer data in the database. Their responsibility is only to verify that the Lending software works as expected. The in-country Application Specialists and Lending clerks play a part in ensuring that customer data is collected correctly and in updating it if it is found to be incorrect, thereby ensuring correct data-integrity.

4.1.1 Encrypted block-size

When running the encryption function, the size of the output block-cipher can be selected. Initially, a 128-bit output block was chosen, but this looked clumsy in practice as it filled the entire field in the on-screen text block as shown in Figure 16 on the following page, so it was reduced to a 28-bit output block, as shown in Figure 17 on the next page. The QA testers indicated that the 28-bit block output in Experiment-1 was more user friendly.

Even though the QA testers were happy with the 28 character output used in Experiment-1, this still looked quite long when compared to the original names, which were usually less than 10 characters. Therefore, in Experiment-2, the size of the output was selected to match the size of the input, so the encrypted block was always the same size as the input. FPE supports this method of encryption without losing any of its inherent security, especially when the tweak was used [7]. This is the default behaviour of the FF1 algorithm when using the rank-then-encipher (RtE) process of FF1 discussed in section 2.4.2 on page 25 [7]. This is shown in Figure 18 on page 66, where the first name and surname are different lengths.

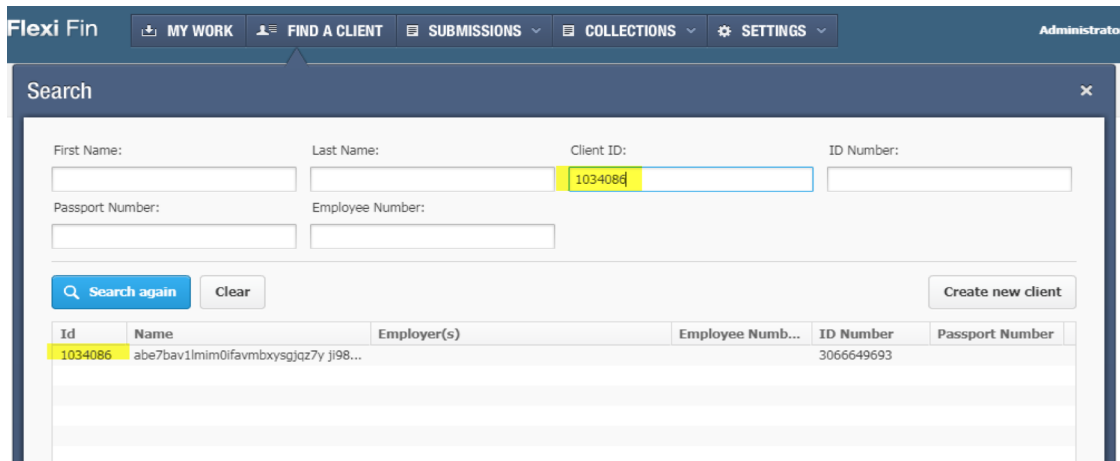


Figure 16: Experiment-1: Search for Client by Customer Number

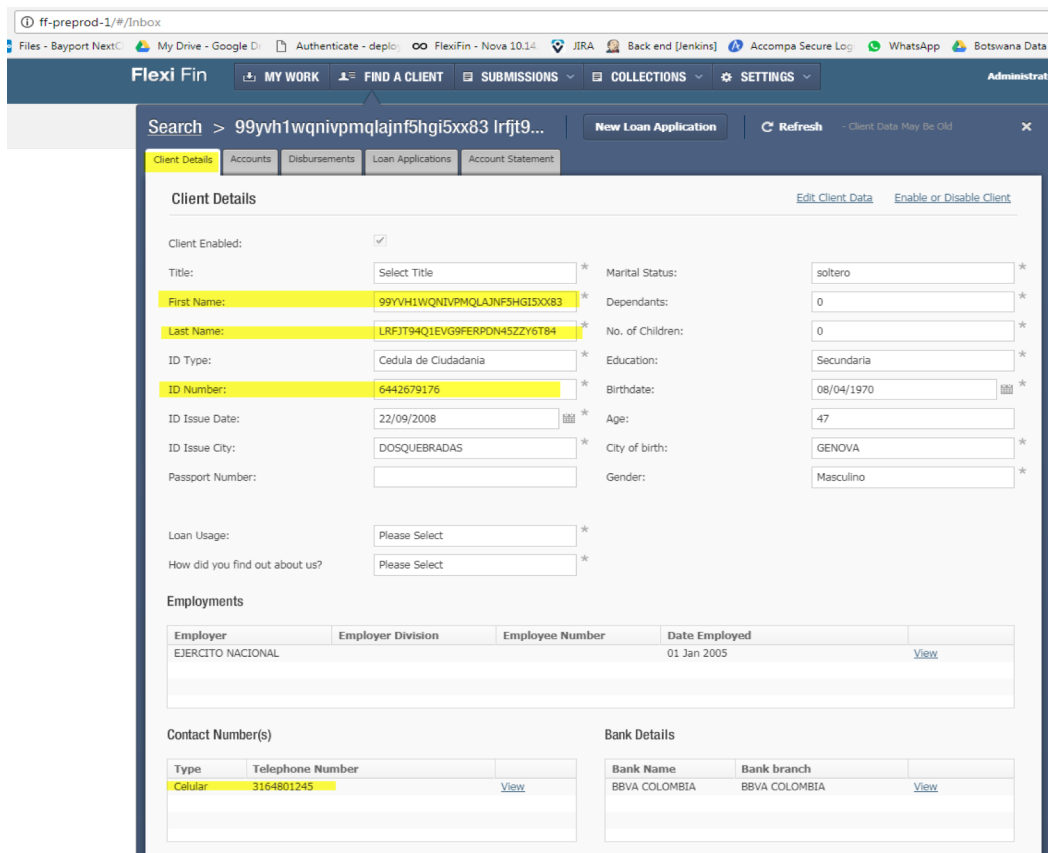


Figure 17: Experiment-1: View Client Details

Client Details

Title:

First Name:

Last Name:

Gender:

Identification Type:

Identification Number:

Marital Status:

Dependents:

Children:

Education Level:

Date Of Birth:

Age:

Addresses

	Type	Address	Postal Code	Preferred
View	Residential	cbja3lmzk8kg 8o8co86		Y
View	Postal	8rt9p2yb9 sf97py7wj9		Y

Bank Details

	Account Number	Bank Name	Branch	Preferred
View	9292321060750	Standard Chartered Bank	Gaborone Industrial Branch	Y

Contact Numbers

	Contact Type	Contact Number	Preferred
View	Mobile	44882877	Y
View	Work	1462239	Y

Email Addresses

	Email Address	Preferred
View		

Next of Kin

	First Name	Last Name	Gender	Relationship
View	bx1vr5bao	x5tyvlo	Male	Brother

Close

Figure 18: Experiment-2: Disbursement Client Details

Other adjustments that would improve the user experience were considered. These included capitalising the first letter of names so that they looked more like a actual names. The QA testers indicated that it was preferable to have capitalised names.

In cases where the customer had two first names with a space between them, the space was removed and the names concatenated before encryption. The QA test team also preferred to work with this version of the encrypted data.

Table 1 shows an example of the encrypted “first name” and “last name” taken from the Experiment-2 database.

Capitalising the first letter of the encrypted names to make them more like actual names was possible but this was not tested during the experiments.

First_name	Last_name
y43poce	f3krp59
qd6v	3fvqqv2j
67fuq	wjleishsi
sqvv	5m26r
r4em7sq	pu04oay
8e41u	110heq9
bb2cndha	8njy3n
nkf0tmy3a	723zbynnm5
aofdiq	plegj4t5

Table 1: Experiment-2: Encrypted Names

First_name	Last_name
Ywsuytrr	Gewqpbd
Qddv	Ytrndiovj
Fefew	Redhncodu
Dews	Ednci
Wjleish	Rencfpo
Oepjr	Bfhdmsu
Bccwqasl	Mkgqid
Sindyelpv	Poejnvdtsm
Ourdmg	Xdjcimby

Table 2: Experiment-2: Encrypted Names with Capitalisation and numbers removed

Another solution was to adjust the alphabet, so that the encrypted form of the names only contained alphabet characters rather than alphanumeric characters. Table 2 shows the encrypted names using only alphabet characters with capitalisation.

4.1.2 Encrypting email addresses

The LibFFX library only supported alphanumeric characters and did not support special characters. This meant that special characters had to be removed from the input text or handled in a different way. Email addresses presented a problem as they included the “@” symbol. This was resolved by splitting the username and the domain name, encrypting each part separately and then joining the encrypted username and encrypted domain name with the “@” sign in the middle. A “.com” was appended to the end so that the encrypted form of the email address still looked like an email address. The encrypted username and encrypted domain name were encrypted as a ten character block in Experiment-1 so the resulting string did not look much like an email address without the “.com”, e.g. it looked like “wedlydmpud@qactpunfem” before

The screenshot shows a web form with a modal window titled "Add / Edit Address". The modal contains the following fields:

- Address Type: Casa
- Address Line 1: 6zifkseqpxfavf6jad6z2rvu8t88
- Address Line 2: (empty)
- Address Line 3: (empty)
- Postal Code: (empty)
- Country: Please Select
- Region: (empty)
- City: (empty)
- Suburb: (empty)
- Residence Type: Please Select
- Residence Status: Familiar
- Months At Residence: (empty)

The background form includes fields for Title, First Name, Last Name, ID Type, ID Number, ID Issue Date, ID Issue City, Passport Number, Loan Usage, How did you find out about us?, Employments (with a table for Employer and Telephone Number), and Contact Number(s) (with a table for Type and Telephone Number).

Figure 19: Experiment-1: Address

appending the .com, and “wedlydmpud@qactpunfem.com” after adding the “.com”. Figure 20 on page 69 shows a screen print of the encrypted email address.

Another option would be for email addresses that included full stops to be split at the full stop and each part encrypted independently as FF1 allows for encryption of domains as small as two characters. Security of this mechanism is further bolstered by using the additional password, or the “tweak”. For example: “alice.bob@campus.ru.ac.za” could be encrypted to “wepld.dyz@wlicm.wq.vp.me”.

The QA testers were content with the former method, but responded that they would prefer to use the latter method if given the choice.

If it is necessary to re-identify or decrypt the dataset the encryption process above could simply be reversed.

4.1.3 Special characters, accented characters, or other odd characters found in names

The LibFFX library uses a radix or alphabet of only the 36 english alphanumeric characters and does not support special characters or double-byte characters. Double-byte characters include the N accent character “ñ” in the Spanish alphabet, or apostrophe’s, dashes, or spaces in the English alphabet. These are commonly found in Spanish in names such as “Rafael Núñez” or the phrase “Cómo estás” as well as English names such as "O’Dowd" or "Smith-James".

The screenshot shows the Flexi Fin web application interface. The top navigation bar includes 'MY WORK', 'FIND A CLIENT', 'SUBMISSIONS', 'COLLECTIONS', and 'SETTINGS'. The user is logged in as 'Administrator'. The search bar contains the ID 'J7F0LTCH2651ISXNS5UQ8ZFPWS7Q 8...'. The main content area is titled 'Account Statement' and contains several sections:

- Account Statement:** A form with fields for Name (J7F0LTCH2651ISXNS5UQ8ZFPWS7Q 8), ID Number (9302150150), Client Id (1006925), Blacklisted (No), and Today's Date (22/09/2017).
- Contact Number(s):** A table with columns 'Type' and 'Telephone Number'. It lists 'fijo' (4583705) and 'Celular' (3102565936), each with a 'View' link.
- Email Addresses:** A table with columns 'Email Address' and 'View'. It lists '4vmjrf6e6ujbvc@0xg7dvw3u4.com' with a 'View' link.
- Addresses:** A table with columns 'Type' and 'Address'. It lists 'Casa' (2uvfdr9iph6mxdknfjss90smclpk) with a 'View' link.
- Account Details:** A table with columns: Account ID, Contr..., Empl..., Empl..., Lend..., Acco..., Reas..., Outlet, Agent, Loan..., Origi..., Rest..., Rem..., Loan..., Note. The first row shows: 2007335, RAMA JU, 3195, Bayport, FullySettl, BOGOTA, orp6e06i, 2016-02-, 19, 0, NO, View.
- Account status and transaction:** A table with columns: Account ID, Today's Balance/Arrears Amount(Total), Loan Transactions. The first row shows: 2007335, View, View.

Figure 20: Experiment-1: View Account Statement

This was resolved by stripping all special characters out before encrypting with the LibFFX module. As it was not necessary to re-identify this particular data set, this did not pose a problem. The encrypted form of the text would not differ much if these characters were encrypted, as the output would appear to be completely random.

If the library was improved to support these characters, or a different encryption library that supports these characters was used, the result would not be discernible from the chosen methodology.

Alternatively, special characters could have been stripped and re-inserted back into the correct place in the encrypted form of the name, but this would introduce the possibility that the person could be identified from the encrypted data, especially if they have an unusual character in their name such as a “” or a “-”. The Anonymisation Decision-Making Framework [25] recommends not performing this sort of action on the data, as this sort of customisation could lead to identification of an individual from the set of data. The RtE approach could have

been adopted to convert the special characters to a new alphabet and then encrypted. The same alphabet mapping would need to be used to re-identify the data, if needed.

4.1.4 Usernames

All staff login details had to be left unencrypted so that the staff could continue to login to the Lending system and use it correctly. The usernames were stored in the same table as customer names, so these were identified by their role privileges and excluded from de-identification. Staff PII is bound by the requirements of POPI so all staff PII except their usernames were de-identified. There is no risk of a PII breach, as the staff already know each others' names, but they do not need to know their colleagues' identity numbers or home addresses etc. to do their jobs.

4.1.5 Search for Client by Name

A change to the testing protocol was required in that QA staff could no longer search for customer using the customer's name, but had to search using their client ID, as shown in Figure 20 on the preceding page.

QA staff would generally search for a customer by name during their testing, but since this was de-identified, the client had to be searched for by their Client Number, which was an internal, unique identifier and was not PII. This was expected behavior and was discussed with the QA test team prior to their testing, and required that QA staff search for the client by client identity number instead of by client name.

Figure 17 on page 65 shows all client PII de-identified in the Search section. Figure 16 on page 65 shows searching for a client using the customer number "1034086".

Searching for "Abe" would have returned 22 customers before de-identification. Searching for "Abe" after de-identification would not have returned anything useful, as all names were de-identified.

4.1.6 Dates

There was no simple way to encrypt dates with FPE while retaining the age validation that the software required to originate loans. The software requires that the lessor was at least 18 years

old when the loan is originated and that the lessor was not within five years of retirement. If the year was isolated for encryption the potential outputs were 0000 to 9999. Valid year dates needed to be between 1953 to 2000 to pass age validation criterion. For the purpose of the experiments the date was set to “1970-01-01” as this verified that the de-identification mechanism worked correctly. For the production version of the de-identification code a more robust algorithm would be needed to de-identify the date, as hard-coding 1970 would eventually result in the lender being too old to qualify for a loan.

Alternatively, the year could have been left unchanged and only the month and day de-identified, as this would adequately de-identify the date without breaking the Lending system’s internal data integrity rules. This is quite easy using FPE, although not with the LibFFX library. There would be a slightly increased likelihood of identification if an attacker knew the birth year was not encrypted, since there are only 12 valid months to choose from and 31 valid days to choose from, however the set of customers was big enough to render this almost completely moot.

4.1.7 Integer de-identification

The FF1 algorithm was designed to work in either string mode or integer mode. If the input was a string, then the output was a string. If the input was an integer, then the output was an integer. If the incorrect input was supplied for the chosen mode, then the program would exit.

As such, care had to be taken to ensure that integer input was in fact integer input. The database allowed many integer fields to be stored in VARCHAR fields, which meant the data could be alphanumeric or even special characters. All of these spurious characters had to be stripped or converted to integers correctly.

4.1.8 On-the-Fly de-identification

No on-the-fly de-identification was performed while the QA staff were testing the de-identified data set. It was only de-identified prior to QA testing when the de-identification ETL process was run to create a QA database instance. QA staff would only enter made-up data when testing the functionality of the Lending system, so it was not necessary to perform any further encryption.

4.1.9 Key principles in encrypting data for de-identification

It is important to ensure that the de-identified data maintains a similar structure to the original data so that it is user friendly for the QA testers. This can easily be achieved with FPE.

The same methodology can be used with any SQL database, or any alphanumeric data, and for almost any user audience.

The LibFFX library would need to be upgraded to support special characters and accented characters if re-identifying the data was necessary, but it is possible to perform a one-way de-identification, which meets the requirements of this research.

4.1.10 Current practice

Until the release of a ratified FPE methodology with the publication NIST800-38G in 2016, it was not possible to de-identify PII stored in a database without first changing the database schema to support the encrypted output from a block cipher algorithm. This usually involved making the length of the field longer and changing it to binary. This was inconvenient in the real-world of application development.

This made it very difficult to comply with data protection legislation such as POPI, DPA and GDPR. In the case of POPI, it would be necessary to ensure that the entire testing process was POPI compliant as per the eight general conditions to ensure reasonable protections when processing personal information. The eight conditions are “Accountability, Processing limitation, Purpose specification, Further processing limitation, Information quality, Openness, Security safeguards, and Data subject participation” [83]. Additional information can be found in section 2.2.1 on page 10. These conditions add a significant burden to the process, so it is far simpler to de-identify PII, which takes the POPI conditions out of scope.

4.1.11 Future practice

It is now possible to use FPE to encrypt data and de-identify PII without making significant changes to the database.

Using de-identification means that since PII is anonymous, the eight POPI compliance conditions can be excluded. Additional information can be found in section 2.2.1 on page 10.

No.	Section	test Details
1	Login	test System Login
2	Inbox	View System Inbox
3	Find a Client	Search for client by Name
4	Find a Client	Search for client by Customer Number
5	View Client Details	View Client Details
6	View Client Details	Add / edit bank account
7	View Client Details	Add / edit address
8	View account statement	View account statement
9	Print disbursement	Print disbursement
10	Capture Documentation	Capture Documentation
11	Verify Employer Contacts	Verify Employer Contacts

Table 3: Tests performed on Experiment-1 dataset

4.2 Test results - Experiment-1

4.2.1 Tests Performed:

Table 3 shows the tests performed on the Experiment-1 dataset. The testing is structured such that a QA staff member steps through the various sections in the Lending software to ensure that the software was working correctly and that all PII was de-identified. The sections are shown in the table, along with the corresponding test detail.

To test the Lending software, the QA team log in to a de-identified version of the software and run through the various sections in the Lending software to ensure that the software is working correctly and that all PII is de-identified. The series of pre-defined tests is shown in Table 3.

As issues were picked up in the first tests, the de-identification script was modified and re-run against the production dataset to create a new de-identified dataset. The primary issues were that some PII was not encrypted, so this had to be located in the database and de-identified.

There is discussion and output from the QA process shown in section 4.3 on page 77.

The following screen-prints show the result of connecting to standard screens in the Lending system to verify that all PII has been successfully de-identified.

4.2.2 Login to System:

The QA staff test their login to the Lending system using a normal username and password.

Care had to be taken with the experiment to not encrypt the usernames of system users, since they were stored in the same table as customers, as this would stop them being able to log in.

4.2.3 View system inbox

The Inbox section is where all current work items were shown and a QA user could claim an item for processing. This screen does not show any customer PII but only the customer ID.

4.2.4 Search for Client by Name

The Search for Client by Name section shows the screens used to find and manage a client in the Lending software. Before de-identification, the QA user would enter a client's name to search for them, but this was not possible after de-identification.

As an example, searching for "Abe" would have returned 22 customers before de-identification, however this falls away once the database is de-identified.

4.2.5 Search for Client by Customer Number

The Search for Client by Customer Number section has screens to find and manage a client in the Lending software. QA staff would generally search for a customer by name, but since this was de-identified, the client was searched for by their Client ID, which was an internal, unique identifier and was not PII. Figure 16 on page 65 shows all client PII de-identified in the Search section when we search for customer: 1034086

4.2.6 View Client Details

The View Client Details section shows the screen to view client details in the Lending software. Figure 17 on page 65 shows all client PII, such as First Name, Last Name, ID Number, Contact Number, Birth Date and Telephone Number are encrypted.

4.2.7 Bank Details

The add / edit bank account section shows that the bank account number was encrypted.

4.2.8 Add / Edit Address

The add/ edit address section shows the customers' address was encrypted in Figure 19 on page 68.

4.2.9 View Account Statement for a Client

The View account statement section shows all PII was encrypted in Figure 20 on page 69, including all telephone numbers and email address.

Results were similarly encrypted for all clients viewed.

4.2.10 Print Disbursement Details for a Client

The Print disbursement section has a screen to show all PII was encrypted.

4.2.11 Capture Documentation

The Capture Documentation section has a screen to show all PII was encrypted.

4.2.12 Verify Employer Contacts

The Verify Employer Contacts section again shows all PII was encrypted in Figure 21, including the Key Contacts details.

ff-preprod-1/#/Process/47454774

is - Bayport NextC My Drive - Google D Authenticator - depl FlexiFin - Nova 10.14 JIRA Back end [Jenkins] Accompa Secure Log WhatsApp Botswana Data M

Flexi Fin MY WORK FIND A CLIENT SUBMISSIONS COLLECTIONS SETTINGS Administrator

Employer Application FONCEP - ADMINISTRATIVO

Capture Employer Data Capture Policies Create Condition

Capture Employer Data

Name: FONCEP - ADMINISTRATIVO * Industry: Servicios *
 Reg No: 860041163-8 Employer Type: Publica *
 City: BOGOTA, D.C. Founding Date: 18/12/1900 *
 Employers Code: Deducting Trading Entity: Bayport *

Labour environment:

Average employment period: 175 Anticipated retrenchments in next year:
 Full time employees: Employee reduction over last year:
 Employees with less than 1 year service: Undefined worker count:
 Contract workers: Pensioned worker count:
 Employees on existing loan scheme: Temporary worker count:

Key Contacts + Add Key Contact

Name	Position	
k6zvg33wvkg4ot0ipma5v	LÁDER GESTIÁ'N HUMANA	Edit Delete
amw4mh7qk5iv17yv9i0y	JEFE NÁ' MINA Y PAGADOR	Edit Delete

Lending Trading Entity + Add Lending Trading Entity

Name	
Bayport	Delete

Note: + Add Note

Note	Process	Task	User	Date	
Test field encryption	EmployerApplication	review employer referral	admin	18 Sep 2017 - 15:55:37	View
Field details	EmployerApplication	verify documentation	admin	18 Sep 2017 - 16:06:31	View

Figure 21: Experiment-1: Verify Employer Contacts

No	Section	test Details
1	Login	test System login
2	Inbox	Process Tasks (View Inbox, Claim Process, View Disbursement Details)
3	Dashboard	Your Most Recent Clients
4	Launch process	Task Details
5	Find a Client	Client Details
6	Search_First Name	Search Filter(s)
7	New Loan	Existing Functionality
8	Edit Client	Existing Functionality
9	Document upload	Existing Functionality
10	Settings	Set-up Details

Table 4: Tests performed on Experiment-2 dataset

4.3 Test results - Experiment-2

4.3.1 Tests Performed:

Table 4 shows the tests performed on the Experiment-2 dataset. The testing was structured such that a QA staff member stepped through the various sections in the Lending software to ensure that the software was working correctly with the de-identified PII data and that all PII was de-identified. The sections are shown in the table, along with the corresponding test detail.

The following screen-prints show the result of connecting to standard screens in the Lending system to verify that all PII has been successfully de-identified.

4.3.2 Login Section

This test tested a login to the Lending system using a normal username and password to verify that access privileges were still working correctly.

Once again, care had to be taken with the experiment to not encrypt the usernames of system users.

4.3.3 Inbox Section

The Inbox section was where all current work items are shown and a user could claim an item for processing. This screen shows the Customer Detail field, which was successfully de-identified. The “Manage Accounts / Disbursements” screen correctly showed the result of connecting to standard screens in the Inbox section of the lending system to verify that all PII has been successfully de-identified.

The Inbox and claim process sections worked successfully with the de-identified dataset. The screen showed a connection to the disbursements screen and that all data in the “client full name” and “client ID no.” fields were de-identified.

Figure 18 on page 66, the “Disbursement / View Details” screen, correctly showed a connection to the disbursements / Client details screen and that all PII data were de-identified.

All the user actions related to viewing Client disbursement worked correctly with the de-identified dataset.

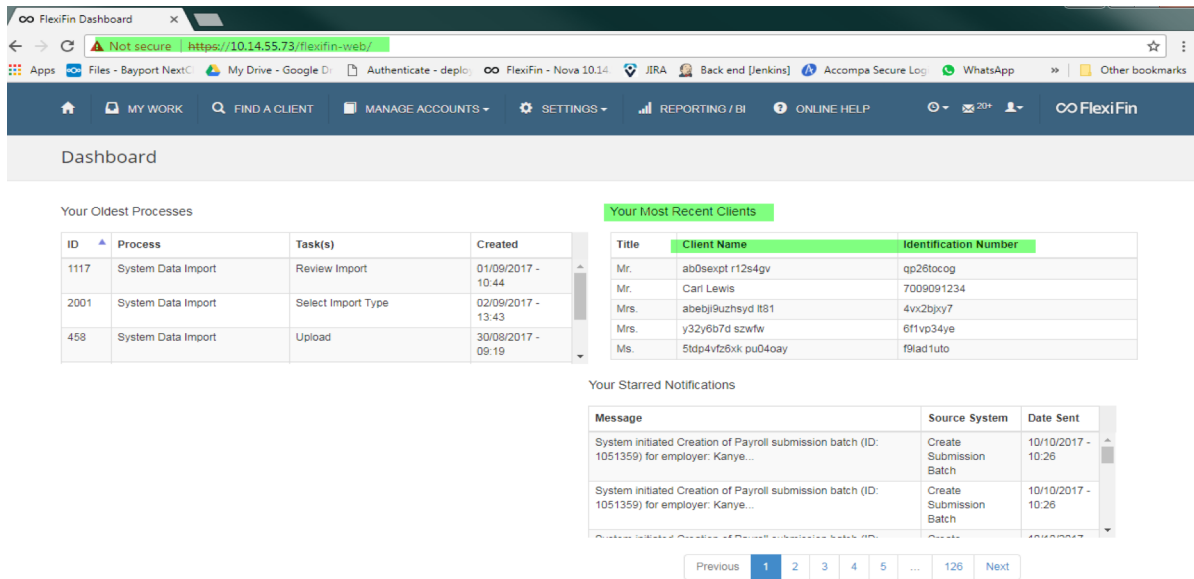


Figure 22: Experiment-2: Dashboard - Your Most Recent Clients

4.3.4 Dashboard

The Dashboard shows the activities available to the user, including: “Your Oldest Processes”, “Your Most Recent Clients” and “Your Starred Notifications”. Figure 22 shows the result of connecting to the Dashboard section of the lending system to verify that all PII had been successfully de-identified.

4.3.5 Launch Process

The Launch Process sections shows the “Loan Agreement Outcome” and “Loan Quote” sections. the “ Loan Agreement Outcome” correctly shows the “First name”, “Last name”, “Identification Number” and “Date of Birth” fields correctly de-identified. The “Loan Quote” screen is shown in Figure 23 with the PII correctly de-identified. This screen also shows all de-identified data being set to the same size as the input data.

your future now

BAYPORT
FINANCIAL SERVICES

Loan Quote

Personal Details	
Title	Mrs.
First Name(s)	y32y6b7d
Last Name	szwfw
Gender	Female
Date of Birth	1965-03-08
ID Number	6f1vp34ye
Marital Status	Married

Employer Details	
Organisation	Government of Botswana-PD
Employer	Ministry of Education and Skills Development-PPM-PD
Employee No	429420902
Employment Start	1997-01-01
Employment Type	Permanent

Contact Details	
Mobile	81960030

Contact Details	
Work	2755069

Banking Details	
Bank Name	Bank ABC
Branch Name	Fairground Branch
Branch Code	550267
Account Type	CURRENT
Account Name	3b54g6go6pds4
Account Number	7657029116554

Address Details	
Postal	4in9m12v, uwpatbjweik4, Francistown, Francistown, North-East District, Botswana

Address Details	
Residential	jh002tdc2iv1375gn3a45g7, uhvdge0e6, Francistown, Francistown, North-East District, Botswana

Primary Loan Details	
Client Id	1023703
Interest Rate	2.36%
Principal Loan Value	92,750.00 BWP
No of Installments	72

Loan Components	
Disbursement Amount	92,750.00 BWP
Loan Amount	92,750.00 BWP
Monthly Installment	3,092.56 BWP
Total Collectable	222,664.32 BWP

Outlet & Agent Details	
Outlet Name	Francistown
Field Manager	uc4owgoqr ixnilydqfdm
Agent Name	n71trnx9llf aiccmqog

Bayport Management Ltd (reg. No. 54787 C1/GBL) ("BML") is a Public Limited Company incorporated in the Republic of Mauritius under section 299 of the Mauritius Companies Act. BML's Registered Office is situated at C/o DTOS Ltd, 10th Floor, Raffles Tower, 19 Cybercity, Ebene, Mauritius. BML is licensed and regulated by the Financial Services Commission of Mauritius.

Figure 23: Experiment-2 - Launch Process - Loan Quote

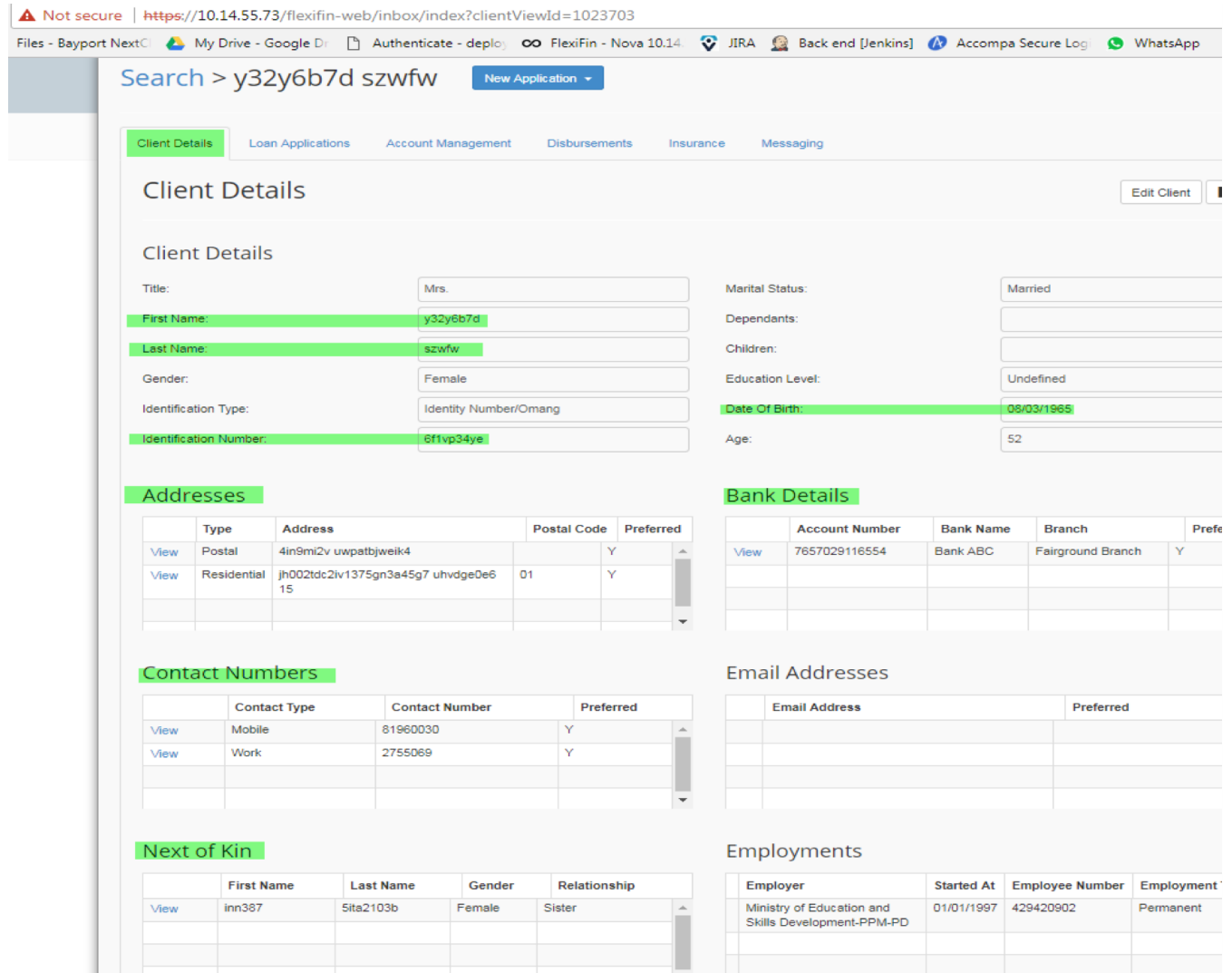


Figure 24: Experiment-2: Find a Client -All Details de-identified

4.3.6 Find a Client

The Find a Client section shows the screens to find and manage a client in the Lending software. QA staff would generally search for a customer by name, but since this was de-identified, the client was searched for by their Client Number, which was an internal, unique identifier and was not PII. Figure 24 shows all client PII de-identified in the Find a Client section.

4.3.7 Search First Name

The Search First Name section shows the screens to find and manage a client in the Lending software. Once the dataset was de-identified, staff could no longer search by a client's real first name. When they searched by first name they would be searching by the de-identified name.

4.3.8 New Loan

The Create New Loan section shows the screens to create a new client and add a new loan. This functionality worked correctly with the de-identified dataset. New client data was not de-identified on-the-fly. It was only de-identified when the de-identification ETL process was run to create a QA database instance. QA staff entered fake data when they added test clients and test loans. It was possible to upload customer documents, review the Loan Quote, review the Application Outcome, Create Disbursements, View Account Summary, and Loan Statement with the de-identified dataset, which were key functions of the Lending software.

4.3.9 Edit Client

The Edit Client section shows the screens to edit a client. It was possible to edit all client details correctly when using the de-identified dataset. No additional encryption is being performed once the QA database is delivered as the QA testers use fictitious data and the QA database does not need to be re-identified. FPE is only performed when copying from production to QA.

4.3.10 Document Upload

The Document upload section shows the screens to upload documents for a client. It was possible to upload all client documents correctly when using the de-identified dataset and issue a manual receipt.

4.4 Summary

This research has demonstrated that the QA staff could perform all their duties successfully as shown in Table 3 on page 73 and Table 4 on page 77 using the de-identified databases created using the LibFFX library and a Python de-identification script.

As such, FPE was an adequate solution for encrypting any SQL database, any alphanumeric data and for any user audience.

De-identified data needs to maintain a similar structure to the original data to be user-friendly.

If the data needs to re-identified then the LibFFX library would need to be updated to support special characters.

Minor changes in the QA process were required, such as adjusting the search criteria to use client numbers instead of client names.

The de-identification script used in Experiment-2 has since been taken into production use.

5 Conclusion

5.1 Introduction

The research aimed to show that data could be de-identified without damaging the database schema and be used by the QA / test teams to perform their tasks as normal. It further aimed to verify that the QA / test teams' full set of tests can be run successfully on a de-identified data set.

Two experiments were performed implementing the FF1 algorithm from NIST 800-38G [21] using the LibFFX library [23], downloaded from Github. The experiments were performed using two different databases from a South African Financial Services Provider (FSP). The databases were sourced from the FSP's subsidiaries in Colombia and Botswana. Since the FSP is based in South Africa, the Protection of Personal Information (POPI) Act will apply to the use of the databases, as they contained customer Personally Identifiable Information (PII), once the POPI effective date is reached. The PII was de-identified using an extract, transform and load process discussed in section 2.3.5 on page 17 which loads the data from a database, then encrypts it and loads back into a new database for testing. The de-identified database was reviewed by Quality Assurance (QA) testers using the Lending software user interface to ensure that the Lending software was still usable with the de-identified data.

The FF1 algorithm was an excellent candidate for the de-identification of data which was stored in a structured database because of its ability to encrypt data from any alphabet without changing the length of the output. The FF1 feature was used effectively in Experiment-1 as shown in section 4.2 on page 73 and Experiment-2 as shown in section 4.3 on page 77.

The tests shown in Table 3 on page 73 were performed in Experiment-1 and the tests shown in Table 4 on page 77 were performed in Experiment-2. The tests entailed logging into the Lending system and performing all the usual actions that a system user would require to verify that the systems still worked and that the PII was correctly de-identified.

The normal activities are: Login, Check User Inbox, Find a Client, Check Dashboard, View Client Details, View Account Statement, Print Disbursement, Capture Documentation, Edit Client and Verify Employer Contacts.

5.2 Chapter Summary

Chapter 2 is a summary of legislative requirements in the jurisdictions used in the experiments and a summary of cryptographic research and literature leading up to the testing and ratifying of the FF1 and FF3 algorithms used for Format-preserving encryption (FPE) published by NIST in NIST “Special Publication 800-38G | Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption” [21].

Chapter 3 is a summary of the open-source and commercial software tools reviewed before performing the experiment and the software library chosen to implement FPE. A Python script was written to perform the experiment and the initial testing results are shown.

Chapter 4 showed the analysis and discussion of the results and output from the tests performed by the QA team.

5.3 Research Goals

1. This thesis aimed to show that data could be de-identified without damaging the database schema and be used by the QA / test teams to perform their tasks as normal.
2. It was verified that the QA / test teams’ full set of tests could be run successfully on a de-identified data set.

Research goal 1 was tested by de-identifying the database and loading it into the QA instance of the Lending software for the QA team to test. The QA team was able to perform all of the standard tests shown in Table 3 on page 73 and Table 4 on page 77 with the de-identified database. As they were no longer able to view customer names they had to search for customers using their internal customer number as shown in Section 4.2.4 on page 74 and Figure 16 on page 65. This was expected behaviour of the de-identified database and was discussed with the QA team prior to the test phase.

Research goal 2 was resolved early on in the test phase as the QA team were completely happy to use de-identified data in place of customer names. The backup strategy, if the QA

team found the de-identified names difficult to work with, was to make use of a tokenisation approach as per section 2.3.2 on page 16 to make encrypted customer names a more human-friendly output. This would have been feasible based on a risk assessment of the PII data and the QA user group as well as the risk of re-identification as per the Anonymisation Decision-Making Framework [25].

The experiments demonstrated that it is completely possible to do a full set of QA tests on new software versions using a de-identified dataset. Since the de-identified dataset is using a NIST compliant algorithm to perform the encryption, ensuring that the requirement to comply with the eight conditions of POPI was not necessary. The eight conditions are “Accountability, Processing limitation, Purpose specification, Further processing limitation, Information quality, Openness, Security safeguards, and Data subject participation” [83]. More detail can be found in Section 2.2.1 on page 10.

We have also successfully de-identified the Lending system database and used the de-identified database without making any changes to the software or the database schema.

The goals of the research were met and areas for further study were identified.

5.4 Significance

The significance of this work is that it showed that it was possible to de-identify PII in a SQL database without making changes to the database structure, which was required by older methods of encryption.

The two experiments showed that it was possible to encrypt two separate databases and use the Lending System correctly in both cases.

The research from Bellare et al. [5] also showed that the encryption of FPE is just as strong as the underlying block-cipher algorithm that is used, which in this case was AES, even for small domains.

The research further showed that the utility of the data is not lost, as the QA team could

perform their full set of standard tests on the de-identified database.

Additionally, the research showed that it was possible to use de-identified names using FPE without having to revert to a tokenisation approach.

5.5 Future Work

The LibFFX open-source encryption library should be enhanced to encrypt all alphanumeric, special characters and double-byte characters used in other languages. The new code can easily be merged with the LibFFX library on Github. Currently LibFFX does not encrypt special characters, so the only simple way to handle these was to strip them out of the input string. Reinserting the special characters into the outputted string to maintain consistency could potentially increase the likelihood that the data could be re-identified. Building these into the LibFFX alphabet would allow for the easy encryption of special characters without having to strip them out of the input text. It would also allow for the re-identification of a dataset if this is required.

A manual should be written for LibFFX which makes it easier for other users to utilise. It was possible to follow and emulate the user example, but it would be simpler to follow some more detailed instructions.

LibFFX should be extended to easily support email addresses as discussed in section 4.1.2 on page 67, so that the encrypted output retains the same format as the input. A specific datatype could be created for email addresses to allow for the easy encryption as discussed in section 4.1.2 on page 67 as dealing with the “@” and “.” symbols were difficult, and subsequently, made de-identified email addresses difficult for users to work with. Retaining the format of the inputted email address makes these simpler to use, which helps to ensure that the change process of moving to an acceptable de-identification standard is easier for users to adopt.

A “name” datatype should be created in the LibFFX library that, when used, ensures that the encrypted output is capitalised and does not contain any numeric characters. Again, this would ensure that the de-identified output is easier to use.

An open-source encryption library should be built that will encrypt PII data that is embedded

in a graphics format such as JPEG or PDF, without damaging the non-PII portions of the file, This would be useful for customer documentation, such as pay slips, that are scanned into the Lending system. This was not attempted in this research. Since it would be very difficult to identify the PII in an image file, an FPE algorithm or fuzzing algorithm could be used to de-identify the whole image file.

5.6 Limitations of the Research

There are limitations to how one is able to encrypt dates, since the software requires that the lessor was at least 18 years old when the loan is originated and that the lessor not be within five years of retirement. If the year was isolated for encryption the potential outputs were 0000 to 9999. Valid year dates needed to be between 1953 to 2000 to pass age validation criterion. A solution to this could be to leave the year unchanged and only de-identify the month and day, as this would adequately de-identify the date without breaking the Lending system's internal data integrity rules. There would be a slightly increased likelihood of identification if an attacker knew the birth year was not encrypted, since there are only 12 valid months to choose from and 31 valid days to choose from, however the set of customers was big enough to render this almost completely moot.

The security of very long or very short strings creates a potential risk for re-identification by an attacker. Since the length of the string is retained, an attacker who has knowledge of the source data or population may be able to re-identify an individual based on length. LibFFX has an upper limit of 128 characters, so if a persons' name is 40 characters for instance, an attacker may have prior knowledge of who that person is. A solution to this would be to pre-calculate the average length of each field, then choose an acceptable maximum and minimum length then clip the de-identified data to match these parameters.

There is a limitation regarding special characters, as these were not supported by the LibFFX library. These were handled by removing the special characters as discussed in section 4.1.3 on page 68. As discussed in section 5.5, the LibFFX library should be updated to encrypt special characters.

Glossary

BPS	Brier Peyrin Stern Encryption
CCN	Credit Card Number
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
COTS	Commercial off-the shelf
COTS	Common off the shelf software
Crypto-C	RSA BSAFE Crypto-C
CTR	Counter
DB	Database
DFA	Deterministic Finite Automaton
DPA	Data Protection Act
DPD	European Union Data Privacy Directive
DPO	Data Protection Officer
DTP	Datatype-Preserving Encryption
E.U.	European Union
ERD	Entity Relationship Diagram
ETL	Extract, Transact, Load
FF1	Feistel Format 1
FF3	Feistel Format 3
FFX	Feistel-based encryption
FIPS	Federal Information Processing Standard (FIPS)
FPE	Format-Preserving Encryption

FTE	Format Transforming Encryption
GDPR	European Union General Data Protection Regulation
HPE	Hewlett Packard Enterprise
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
PAN	Credit Card Primary Account Number
PCI	Payment Card Industry
PCIDSS	Payment Card Industry Data Security Standard
PHI	Protected Health Information
PII	Personally Identifiable Information
POPI	Protection Of Personal Information
PRF	Pseudo-Random Function
PRP	Pseudo-Random Permutation
S.A.	South Africa
SSN	Social Security Number
SST	Secure Stateless Tokenisation
U.K.	United Kingdom
U.S.A.	United States of America
VIL	Variable Input Length Enciphering

References

- [1] Luhn algorithm - Wikipedia, Accessed: 2018-03-15.
- [2] BAKIBINGA-GASWAGA, E. Data Protection in the Commonwealth: Key Instruments and Current Practices. unctad.org/meetings/en/Presentation/dtl_eweek2016_EBakibinga-Gaswaga_en.pdf, Accessed: 2017-02-01.
- [3] BANK OF ENGLAND. Prudential Regulation Authority (Bank of England). <http://www.bankofengland.co.uk/pru/Pages/default.aspx#>, Accessed: 2017-12-02.
- [4] BAYPORT FINANCIAL SERVICES. Bayport Finance. <http://www.bayportfinance.com/>, Accessed: 2017-09-01.
- [5] BELLARE, M., RISTENPART, T., ROGAWAY, P., AND STEGERS, T. Format-Preserving Encryption. In *Lecture Notes in Computer Science*, vol. 5867 LNCS. 2009, pp. 295–312.
- [6] BELLARE, M., ROGAWAY, P., AND SPIES, T. Addendum to The FFX Mode of Operation for Format Preserving Encryption. <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec2.pdf>, Accessed: 2017-01-05.
- [7] BELLARE, M., ROGAWAY, P., AND SPIES, T. The FFX mode of operation for format-preserving encryption. <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf>, Accessed: 2016-08-11.
- [8] BLACK, J., AND ROGAWAY, P. Ciphers with Arbitrary Finite Domains. *CT-RSA 2002 - The Cryptographers' Track at the RSA Conference 2002* (2002), 114–130.
- [9] BLACKMER, W. GDPR: Getting Ready for the New EU General Data Protection Regulation. <http://www.bna.com/final-european-union-n57982067329/>, Accessed: 2017-02-25.
- [10] BLOOMBERG. The Final European Union General Data Protection Regulation | Bloomberg BNA. <https://www.bna.com/final-european-union-n57982067329/#!>, Accessed: 2017-03-18.

- [11] BREACHWATCH. ICO DPA Fines | Breach Watch. <http://breachwatch.com/ico-fines/>, Accessed: 2017-09-24.
- [12] BRIER, E., PEYRIN, T., AND STERN, J. BPS: a Format-Preserving Encryption Proposal. <http://ai2-s2-pdfs.s3.amazonaws.com/0be5/d4c77e333d78ddab5c4bf55d15649a660771.pdf>, Accessed: 2017-04-17.
- [13] BRIER, E., PEYRIN, T., AND STERN, J. BPS Authors Patent Declaration. <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-ip.pdf>, Accessed: 2016-08-12.
- [14] BUCKLEY, J. B. FFX Codec. <https://github.com/brandt/ffxcodec>, Accessed: 2017-08-04.
- [15] CRUNCHBASE. Semtek Innovative Solutions | crunchbase. <https://www.crunchbase.com/organization/semtek-innovative-technologies-corporation#/entity>, Accessed: 2017-06-26.
- [16] DATA PROTECTION WORKING PARTY. Guidelines on Data Protection Officers. http://ec.europa.eu/information_society/newsroom/image/document/2016-51/wp243_en_40855.pdf, Accessed: 2017-02-25.
- [17] DATAWAREHOUSE4U.INFO. ETL process. <http://datawarehouse4u.info/ETL-process.html>, Accessed: 2017-03-12.
- [18] DEFUSE SECURITY. Free Password Hash Cracker. <https://crackstation.net/>, Accessed: 2016-08-01.
- [19] DURAK, F. B., AND VAUDENAY, S. Breaking the FF3 Format-Preserving Encryption Standard over Small Domains. In *Lecture Notes in Computer Science*, vol. 10402 LNCS. 2017, pp. 679–707.
- [20] DWORKIN, M. Recommendation for Block Cipher Modes of Operation Methods and Techniques. <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>, Accessed: 2016-03-05.

- [21] DWORKIN, M. NIST Special Publication 800-38G | Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. <http://dx.doi.org/10.6028/NIST.SP.800-38G>, Accessed: 2016-08-12.
- [22] DWORKIN, M., AND PERLNER, R. Analysis of VAES3 (FF2). <https://eprint.iacr.org/2015/306>, Accessed: 2017-07-25.
- [23] DYER, K. P. LibFFX : The FFX Mode of Operation for Format-Preserving Encryption. <https://github.com/kpdyer/libffx>, Accessed: 2017-04-29.
- [24] ELLIOT, M., MACKEY, E., O'HARA, K., AND TUDOR, C. U.K. Anonymisation Network. <http://ukanon.net/>, Accessed: 2016-07-17.
- [25] ELLIOT, M., MACKEY, E., O'HARA, K., AND TUDOR, C. *The Anonymisation Decision-Making Framework*, 1 ed., vol. 1. UKAN Publications, University Of Manchester, Manchester, 2016.
- [26] FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATIONS (FIPS). Advanced Encryption Standard Process. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard_process, Accessed: 2017-02-14.
- [27] FINANCIAL CONDUCT AUTHORITY. Financial Conduct Authority. <https://www.fca.org.uk/>, Accessed: 2017-09-24.
- [28] GEMALTO. SafeNet - World-Leading Identity and Data Protection Solutions from Gemalto. <https://safenet.gemalto.com/>, Accessed: 2017-05-09.
- [29] GEMALTO. P2P Encryption with SafeNet. <https://safenet.gemalto.com/partners/verifone-inc/>, Accessed: 2017-05-09.
- [30] GEMALTO. SafeNet ProtectDB : Column-Level Database Encryption and Protection Encrypt sensitive column-level data. <https://safenet.gemalto.com/data-encryption/data-center-security/protect-db-database-encryption/>, Accessed: 2017-05-09.
- [31] GEMALTO. SQL SECURITY SECURITY. <https://safenet.gemalto.com/data-encryption/database-encryption-solutions/>, Accessed: 2017-05-09.

- [32] GOMES-PINZON ZULETA. Colombia's New Data Protection Law. https://iapp.org/media/pdf/knowledge_center/Colombian_DPA_Summary_2011_11.pdf, Accessed: 2016-10-01.
- [33] GOVERNMENT OF THE UNITED KINGDOM. Data Protection Act of the United Kingdom 1998.
- [34] GREEN, M. P. A Few Thoughts on Cryptographic Engineering Some random thoughts about crypto. <https://blog.cryptographyengineering.com/2011/11/10/format-preserving-encryption-or-how-to/>, Accessed: 2017-02-26.
- [35] HERON, S. Advanced Encryption Standard (AES). *Network Security 2009*, 12 (dec 2009), 8–12.
- [36] HEWLETT PACKARD ENTERPRISE. Format-Preserving Encryption (FPE) | HPE Security - Data Security. <https://www.voltage.com/technology/data-encryption/hpe-format-preserving-encryption/>, Accessed: 2016-08-11.
- [37] HEWLETT PACKARD ENTERPRISE. HPE SecureData. <https://www.voltage.com/products/data-security/hpe-securedata-enterprise/>, Accessed: 2016-11-06.
- [38] HEWLETT PACKARD ENTERPRISE. HPE SecureData for Test / Dev Test / Dev Environments Present Security Risks. <https://www.voltage.com/products/data-security/hpe-securedata-testdev/>, Accessed: 2016-08-10.
- [39] HEWLETT PACKARD ENTERPRISE. HPE Security - Data Security Patents. <https://www.voltage.com/technology/standards/patents-and-research/>, Accessed: 2017-02-19.
- [40] HINTZE, M. Viewing the GDPR Through a De-Identification Lens: A Tool for Clarification and Compliance. <https://fpf.org/wp-content/uploads/2016/11/M-Hintze-GDPR-Through-the-De-Identification-Lens-31-Oct-2016-002.pdf>, Accessed: 2017-03-15.
- [41] HYAMS, O. The Right to Be Forgotten. <http://lessexcourt.wordpress.com/2014/05/15/the-right-to-be-forgotten/>, Accessed: 2017-02-25.

- [42] INFORMATION COMMISSIONER'S OFFICE. Anonymisation: managing data protection risk code of practice. http://ico.org.uk/for_organisations/data_protection/topic_guides/~media/documents/library/Data_Protection/Practical_application/anonymisation-codev2.pdf, Accessed: 2016-07-17.
- [43] INFORMATION COMMISSIONER'S OFFICE. Big Data and Data Protection. http://ico.org.uk/news/latest_news/2014/~media/documents/library/Data_Protection/Practical_application/big-data-and-data-protection.pdf, Accessed: 2016-10-01.
- [44] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC 18033-3:2010 - Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers. <https://www.iso.org/standard/54531.html>, Accessed: 2017-11-08.
- [45] JOHNSON, K. Format-Preserving Encryption Library for Java. <https://sourceforge.net/projects/format-preserving-encryption/>, Accessed: 2017-02-26.
- [46] KEETSHABE, A. A. M. DEVELOPMENT OF ICT LEGAL AND REGULATORY FRAMEWORK IN BOTSWANA. www.gov.bw/globalassets/.../e—legislation—ict-pitso-2012-legal-framework.ppt, Accessed: 2016-08-31.
- [47] KRISHNAMURTHY, S. V. S. Revised letter of assurance for essential patent claims. FFX Mode of Operation for Format-Preserving Encryption. <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-voltage-ip.pdf>, Accessed: 2016-08-12.
- [48] LACHAUP, D., AND DYER, K. P. LibFTE : A toolkit for constructing practical, format-abiding encryption schemes. <https://libfte.org/>, Accessed: 2017-06-02.
- [49] LEVAN, C. P. Protegrity Blog: WHAT IS THE DIFFERENCE BETWEEN TOKENIZATION AND ENCRYPTION? <http://www.protegrity.com/difference-tokenization-encryption/>, Accessed: 2017-05-01.

- [50] LEYDEN, J. T. R. Last year's ICO fines would be 79 times higher under GDPR. https://www.theregister.co.uk/2017/04/28/ico_fines_post_gdpr_analysis/, Accessed: 2017-09-24.
- [51] LINKEDIN. Semtek: Overview | LinkedIn. <https://www.linkedin.com/company/121359/>, Accessed: 2017-08-03.
- [52] LLOYD, J. Botan Format Preserving Encryption. <https://botan.randombit.net/manual/fpe.html>, Accessed: 2017-02-25.
- [53] LORÜNSER, T., SLAMANIG, D., LÄNGER, T., AND PÖHLS, H. C. PrismacloudTools: A Cryptographic Toolbox for Increasing Security in Cloud Services. *Proc. of the Workshop on Security, Privacy, and Identity Management in the Cloud to be held at the 11th International Conference on Availability, Reliability and Security (ARES SECPID 2016)* (aug 2016).
- [54] LUCHAUP, D., DYER, K. P., JHA, S., RISTENPART, T., AND SHRIMPTON, T. LibFTE: A Toolkit for Constructing Practical, Format-Abiding Encryption Schemes. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, 2014), USENIX Association, pp. 877–891.
- [55] LUHN, H. Computer for verifying numbers.
- [56] MATTSSON, U. T. *Format-Controlling Encryption Using Datatype-Preserving Encryption*, vol. 1.
- [57] MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. Chapter 01: Overview of Cryptography. In *Handbook of Applied Cryptography*. 1996, pp. 1–48.
- [58] MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. Chapter 07: Block Ciphers. In *Handbook of Applied Cryptography*. 1996, pp. 223–282.
- [59] MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. *Handbook of Applied Cryptography*, 5 ed., vol. 106. CRC Press, 1996.
- [60] MEYSTRE, S. M. S., FRIEDLIN, F. J., SOUTH, B. R., SHEN, S., AND SAMORE, M. H. Automatic de-identification of textual documents in the electronic health record: a review of recent research. *BMC Medical Research Methodology* 10, 1 (aug 2010), 70.

- [61] MICHALSONS. POPI Commencement Date or POPI Effective Date. <https://www.michalsons.com/blog/popi-commencement-date-popi-effective-date/13109>, Accessed: 2017-02-06.
- [62] MICHALSONS. POPI Act - Protection of Personal Information Workshops. <https://www.michalsons.com/focus-areas/privacy-and-data-protection/popi-act-protection-of-personal-information>, Accessed: 2017-02-06.
- [63] MICHALSONS. POPIA Opinion : Can you transfer personal information outside of South Africa and store it in the cloud ? Question Key points. Tech. rep., 2017.
- [64] MIRACL. MIRACL: Redefining Trust for the Modern internet. <https://www.miracl.com/>, Accessed: 2017-05-09.
- [65] MOONSTONE. Possible POPI Commencement Dates | Moonstone. <https://www.moonstone.co.za/possible-popi-commencement-dates/>, Accessed: 2017-12-09.
- [66] MORRIS, B., ROGAWAY, P., AND STEGERS, T. How to Encipher Messages on a Small Domain. In *Advances in Cryptology*. 2009, pp. 286–302.
- [67] MZEKANDABA, S. I. Exclusive interview: Pansy Tlakula readies Information Regulator | ITWeb. <https://www.itweb.co.za/content/gGb3BwMWO6jM2k6V>, Accessed: 2017-02-14.
- [68] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Computer Security Division - Computer Security Resource Center Example Algorithms. <http://csrc.nist.gov/groups/ST/toolkit/examples.html>, Accessed: 2016-08-01.
- [69] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. FIPS 46-3, Data Encryption Standard (DES) (withdrawn May 19, 2005). <https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf>, Accessed: 2017-12-06.
- [70] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Recent Cryptanalysis of FF3. <https://csrc.nist.gov/News/2017/Recent-Cryptanalysis-of-FF3>, Accessed: 2017-08-11.

- [71] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. Sample AES-FF1. <https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/examples/ff1samples.pdf>, Accessed: 2016-10-01.
- [72] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Data Encryption Standard. *Computers and Security* (1981), 250–253.
- [73] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. The Advanced Encryption Standard (AES). <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>, Accessed: 2017-12-06.
- [74] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. NIST Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation : Methods for Format-Preserving Encryption: Draft. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html, Accessed: 2016-08-01.
- [75] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. NIST Computer Security Publications - Archived FIPS Publications. <http://csrc.nist.gov/publications/PubsFIPSArch.html>, Accessed: 2016-11-01.
- [76] OFFICE OF THE PRESIDENT. Proceedings of the e-Government Strategy Stakeholders Conference of Botswana. http://www.gov.bw/globalassets/portal-team/egovt-stakeholders-conference-proceedings_final-report-without-annexes.pdf, Accessed: 2017-03-29.
- [77] PCI SECURITY STANDARDS COUNCIL. Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards. https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss, Accessed: 2017-12-01.
- [78] PRIMEFACTORS. A Better Way to Protect Sensitive Data. <https://www.primefactors.com/secure-data-encryptright>, Accessed: 2017-05-03.
- [79] PROTEGRITY. Database Protector. http://info.protegrity.com/l/49532/2016-11-10/3wr2y1/49532/79092/Protegrity_Database_Protector.pdf, Accessed: 2016-05-03.

- [80] PROTEGRITY. Why Memory Tokenization will replace Encryption - Protegrity. <http://www.protegrity.com/why-memory-tokenization-will-replace-encryption/>, Accessed: 2017-05-03.
- [81] PYTHON.ORG. pyffx. <https://pypi.python.org/pypi/pyffx>, Accessed: 2017-07-16.
- [82] RAUTENBACH, C. A family home, five sisters and the rule of ultimogeniture: Comparing notes on judicial approaches to customary law in South Africa and Botswana. *African Human Rights Law Journal* 16, 1 (2016), 145–174.
- [83] REPUBLIC OF SOUTH AFRICA. Protection of Personal Information, Act 4 of 2013.
- [84] ROGAWAY, P. A Synopsis of Format-Preserving Encryption. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.299.769>, Accessed: 2017-04-25.
- [85] ROZENBERG, B. Prismacloud - Analysis of the state of the art of FPE, OPE and tokenization schemes. <https://prismacloud.eu/?p=1688&preview=true>, Accessed: 2017-03-11.
- [86] SCHNEIER, B., WAGNER, D., KELSEY, J., HALL, C., AND FERGUSON, N. The Twofish Team’s Final Comments on AES Selection. *Security* (2000), 1–13.
- [87] SPIES, T. Format preserving encryption. <https://www.voltage.com/>, Accessed: 2016-08-01.
- [88] SRINIVASA, S. FPE in Ruby. <https://gist.github.com/sandys/3755219>, Accessed: 2017-04-01.
- [89] STALLINGS, W. *Cryptology and Network Security*, 5 ed., vol. 4301 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [90] THE COSORT COMPANY. IRI CoSort : Big Data Transformation Management. <https://www.iri.com/products/cosort/overview>, Accessed: 2017-03-12.
- [91] THE COSORT COMPANY. IRI FieldShield: Data Masking Software Overview. <https://www.iri.com/products/fieldshield/overview>, Accessed: 2017-02-25.
- [92] THE EUROPEAN PARLIAMENT. EUR-Lex - 31995L0046 - EN. *Official Journal L 281* , 23/11/1995 P. 0031 - 0050 (1995).

- [93] THE EUROPEAN UNION. General Data Protection Regulation. <http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1489641783187&uri=CELEX:32016R0679>, Accessed: 2017-01-16.
- [94] THE INFORMATION COMMISSIONER'S OFFICE. The Information Commissioner's Office. <https://ico.org.uk/>, Accessed: 2016-11-09.
- [95] THE INFORMATION COMMISSIONER'S OFFICE. Taking action - data protection. <https://ico.org.uk/about-the-ico/what-we-do/taking-action-data-protection/>, Accessed: 2017-12-02.
- [96] THE INFORMATION REGULATOR. Protection of Personal Information Act draft regulations. <http://www.justice.gov.za/inforeg/docs/InfoRegSA-RegulationsDraft-Aug2017.pdf>, Accessed: 2017-09-11.
- [97] THE THALES GROUP. Thales completes acquisition of Vormetric. <https://www.thalesgroup.com/en/worldwide/press-release/thales-completes-acquisition-vormetric>, Accessed: 2017-08-03.
- [98] TOBIN, P., CATO, J., AND TAYLOR, D. The Protection of Personal Information Act is law: where to from here? <http://www.insurancegateway.co.za/Botswana/PressRoom/ViewPress/-URL=The+Protection+of+Personal+Information+Act+is+law+where+to+from+here#>, Accessed: 2017-03-28.
- [99] UNITED STATES CONGRESS. Health Insurance Portability and Accountability Act OF 1996. <https://www.gpo.gov/fdsys/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf>, Accessed: 2017-11-09.
- [100] VANCE, J. VAES3 scheme for FFX An addendum to The FFX Mode of Operation for Format-Preserving Encryption. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.693.8704&rep=rep1&type=pdf>, Accessed: 2017-06-01.
- [101] VOLTAGE. HP Acquire Voltage to Expand Data Encryption. <https://www.voltage.com/releases/hp-acquire-voltage-security-expand-data-encryption-security-solutions-cloud-big-data/>, Accessed: 2017-08-03.

- [102] VORMETRIC. Vormetric Transparent Encryption | Tokenization | Application Encryption | Cloud Security | Data-at-Rest Encryption. <https://www.vormetric.com/>, Accessed: 2017-05-04.
- [103] VORMETRIC. Operational efficiencies from Vormetric Orchestrator. <https://www.vormetric.com/sites/default/files/sb-vormetric-orchestrator.pdf>, Accessed: 2017-05-02.
- [104] WEISS, M., ROZENBERG, B., AND BARHAM, M. Practical Solutions For Format-Preserving Encryption. *In Proceedings of the 9th Workshop on Web 2.0 Security and Privacy (W2SP) 2015 abs/1506.0* (2015), 1–10.
- [105] WES, M. Looking to comply with GDPR? Here’s a primer on anonymization and pseudonymization. <https://iapp.org/news/a/looking-to-comply-with-gdpr-heres-a-primer-on-anonymization-and-pseudonymization/>, Accessed: 2017-12-09.
- [106] WIKIPEDIA. Advanced Encryption Standard. https://de.wikipedia.org/wiki/Advanced_Encryption_Standard, Accessed: 2017-01-31.
- [107] WIKIPEDIA. Format-preserving encryption. https://en.wikipedia.org/wiki/Format-preserving_encryption, Accessed: 2016-08-11.