UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Format and Order Revealing Encryption

(Versão Final Após Defesa)

Rui Paiva

Dissertação para obtenção do Grau de Mestre em Engenharia Informática

Mestrado em Engenharia Informática

(2º ciclo de estudos)

Orientador: Prof. Doutor Paul Andrew Crocker
Co-orientador: Prof. Doutor Simão Melo de Sousa

Covilhã, Junho de 2018

Acknowledgements

I would like to thank Professor Dr. Paul Andrew Crocker and Professor Dr. Simão Melo de Sousa, who both played a very important role in the completion/execution of this project as my supervisor and Co-supervisor, for all the guidance and inspiration throughout not only this last year but all my time as a Release member.

I would also like to thank my parents, especially my mother, who has been supportive and encouraged me to continue my studies and provided everything I needed to succeed.

To my sisters, Alexandrina and Susana, a special thanks for all the support and advices that always kept me on track.

I would also like to thank my close friends that helped me throughout the years. To my girlfriend, Joana Duarte, that never stopped supporting me and always had the right words to motivate me.

I would like to thank *Release* and all its students and professors, who, in one way or another, have supported me throughout the years at UBI.

Finally, I would like to thank João Reis and Luís Horta for all the moments, struggles and laughs we've shared together. I'll never miss you guys and best of luck for the both of you.

Resumo

Com a explosão de serviços baseados na nuvem que ocorre nos dias de hoje, torna-se imperativo que os dados que são consumidos por este tipo de serviços sejam de alguma forma protegidos contra ataques ou roubos[Cen18]. O principal problema com este tipo de serviços é que, normalmente, estes serviços precisam de acesso aos dados para conseguirem fazer pesquisas e correlacionar dados de forma a que seja possível fornecer diversos serviços. Esta dissertação tem como objetivo estudar o mundo da criptografia de forma a perceber que tipo de garantias são oferecidas pelos esquemas criptográficos existentes nos dias de hoje para serviços baseados na nuvem.

Este trabalho é motivado por um problema real de delegação de dados para a nuvem. Este problema envolve a proteção de dados sensíveis que precisam de ser analisados por entidades externas. Embora não haja uma abordagem simples para resolver este tipo de problemas, nesta dissertação iremos discutir três abordagens que, potencialmente, poderão resolver este problema. Uma abordagem tenta definir o que poderia ser a estrutura geral de um novo esquema criptográfico que pudesse lidar com o problema específico em análise. Numa outra abordagem iremos utilizar ferramentas existentes para tentar resolver o problema em questão. Iremos também tentar unir dois esquemas criptográficos existentes, de forma a tentar combater este problema em específico.

Foi também realizado um estudo a vários esquemas criptográficos de forma a perceber quais as soluções que existem hoje em dia para problemas relacionados com a delegação de dados para entidades externas, como também, tentar perceber que esquemas criptográficos que ainda são resultados meramente teóricos mas que possam vir, no futuro, a ser úteis para combater esta problemática.

Os resultados desta dissertação mostram que resolver um problema relacionado com criptografia nem sempre é fácil, uma vez que, a má utilização destes esquemas poderá levar a uma falha grave de segurança. Por fim, concluímos que, resolver um problema desta natureza através de ferramentas existentes é bastante mais fácil do que tentar desenvolver esquemas criptográficos novos, mas que irá perder o poder de poder ser aplicado a outros problemas semelhantes.

Palavras-chave

Nuvem, Autenticação, Privacidade, Criptografia, Pesquisa sobre dados Cifrados, Format Preserving Encryption, Order Revealing Encryption

Resumo alargado

Com o aumento de fornecimento e procura de serviços baseados na nuvem, aumenta também o nível de risco de fuga ou roubo de informação usada por estes serviços. Esta informação pode ser utilizada para outros fins aos quais ela era inicialmente destinada, como também, esta pode acabar por cair nas mãos de alguém que não deveria ter acesso. Estes fatores de risco, quando associados à necessidade de que estes prestadores de serviços necessitam de aceder aos dados sem estes estarem cifrados de forma a poderem fornecer os seus serviços, aumenta significativamente o risco de fugas de informação.

Os riscos aumentam ainda mais quando esta informação se trata de informação sensível, como no caso de cartões bancários, que são usados para diversos tipos de compras. Compras essas que podem ser feitas **online** ou em quase todas as superfícies comerciais. Este tipo de informação é extremamente sensível, uma vez que, cada transação bancária tem a ela associada vários tipos de informações pessoais do utilizador que podem comprometer a sua segurança e privacidade.

Sendo os cartões bancários uma das principais formas de pagamento nos dias de hoje, o risco de roubo de informação continua a subir. Com um número elevado de transações é necessário também que estas transações sejam validadas de forma automática. Validações essas que, em grande parte das vezes, é feita por entidades externas aos bancos. Havendo a necessidade de enviar informação sensível para entidades externas, o risco de fuga ou roubo de informação é ainda mais elevado, uma vez que estas entidades externas podem introduzir novos fatores de risco.

Para combater o risco de fuga ou roubo de informação nos pagamentos com cartões bancários, existe um conjunto de standards que são obrigatórios para qualquer empresa que fornece o pagamento com cartões bancários. Esses standards variam consoante o volume de transações que cada empresa tem durante um ano. Estes standards não são por vezes suficientes para proteger completamente todas as etapas e entidades por onde a informação terá de ser processada.

Quando o problema se trata de proteger informação sensível, como é o caso de informação bancária, a resposta vem normalmente do mundo da criptografia onde é possível encontrar uma solução para o problema de confidencialidade e privacidade. No entanto, para este problema onde múltiplas entidades estão envolvidas e, por forma a poder fazer validações nas próprias compras por cartão bancário, é necessário que muitos dos dados sensíveis estejam disponíveis para que várias entidades os possam analisar. Normalmente, quando queremos proteger informação de ser vista por terceiros, é possivel aplicar diversos tipos de esquemas criptográficos que irão cifrar completamente esses dados e apenas quem possuir a chave certa a vai conseguir decifrar. Isto introduz um problema para o caso dos pagamentos com cartões bancários, uma vez que, as entidades externas que analisam estas transações necessitam de acesso aos dados originais.

Nesta dissertação, fizemos um estudo do mundo atual da criptografia de forma a perceber que tipo de soluções podem ser aplicadas num problema de delegação de dados para entidades externas. Foram estudados vários esquemas criptográficos, alguns que podem potencialmente

resolver todo o tipo de problemas relacionadas com delegação de dados, enquanto que outros apenas podem ser usados para casos específicos. Estudamos também alguns esquemas que apenas são resultados teóricos, mas que, caso um dia venham a ser possíveis de ser implementados, têm o potencial de aumentar exponencialmente a nossa segurança no mundo *online*.

Para esta dissertação foram também desenvolvidos dois protótipos de esquemas que foram usados durante esta dissertação, mais propriamente, Order Revealing Encryption e Format Preserving Encryption. O desempenho destes protótipos foram comparados com outras implementações disponíveis publicamente.

Por fim, foram propostas várias possíveis soluções para este problema. Uma proposta passa por usar esquemas de criptografia já existentes e usá-los de forma a resolver o problema em questão. Numa outra proposta iremos usar ferramentas existentes do mundo da criptografia que podem ser usados em simultâneo com esquemas criptográficos para resolver este problema. Como proposta de solução temos também a abordagem criptográfica onde definimos os requisitos que um esquema de criptografia terá de ter para que possa manter a confidencialidade dos dados, mas ao mesmo tempo, fornecer informações suficientes sobre a informação original para que seja possível fazer uma análise destes dados.

Abstract

As more and more cloud services emerge so does the need for new methods for securing the data these services consume, especially since data leaks have become the norm rather than the exception. Since most cloud services require some kind of access to our private data in order to perform searches and provide services, new ways of securing our data in the cloud is needed. This dissertation examines the current state of the cryptographic world in order to try to and understand and resume what solutions currently exist for this particular type of problem.

This work is motivated by a particular problem of data delegation to a cloud infrastructure. This problem involves the protection of sensitive data whilst it's analysed by a third party. While there is no simple approach to solve this particular problem, this dissertation discusses three main approaches to tackle this problem. One approach attempts to define a new cryptographic scheme with a leakage profile that would allow a third party to only have access to some information of the plaintext but, at the same time, keep the plaintext safe from attackers. Another approach attempts to use already existing cryptographic schemes, such as, Format Preserving Encryption and Order Revealing Encryption to solve this particular problem. A final approach tries to solve this problem by utilising cryptographic tools, such as hash-functions and hash-based message authentication codes.

An extended study was also conducted in many cryptographic schemes, both current and old cryptographic schemes. This study allowed for a better view of the cryptographic world and how these schemes could help us achieve a solution. For this dissertation, a prototype was also implemented of some recent cryptographic schemes. These prototype implementations allowed for a deeper understanding of how these schemes work and also allowed us to conduct some experiments while trying to combine two cryptographic schemes.

The results of this dissertation show that that trying to solve a problem via creating a new cryptographic scheme is not an easy feat especially when one wants to define correctly the strict security requirements and also the work needed to understand the mathematical workings of similar schemes. Lastly we conclude that solving the problem with the help of already existing tools may be the easiest solution, but, it may also only work for a specific scenario and hence is of no use in other similar situations. A solution to the particular problem studied in this thesis is also presented at the end of this dissertation, although, it only applies to this specific problem and does not solve the more general problem of privacy of data delegation to the cloud.

Keywords

Cloud, Authentication, Privacy, Cryptography, Security, Search over encrypted data, Format

Preserving Encryption, Order Revealing Encryption, Payment Card Industry, Credit Card Fraud

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ABE	Attribute Based Encryption
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CBC	Cipher Block Chaining
CPA	Chosen Plaintext Attack
CCN	Credit Card Number
FE	Functional Encryption
FFX	Format Preserving Feistel-based Encryption
FPE	Format Preserving Encryption
FPEQ	Format Preserving Encryption with Plain-text Equality
FHE	Fully Homomorphic Encryption
HE	Homomorphic Encryption
HMAC	Keyed Hash Message Authentication Code
IBE	Identity Based Encryption
IND-OCPA	Indistinguishability under Ordered Chosen Plaintext Attack
IO	Indistinguishability Obfuscation
IP	Internet Protocol
IV	Initialisation Vector
MIFE	Multi-Input Functional Encryption
NIST	National Institute of Standards and Technology
PCI	Payment Card Industry
PCI-DSS	Payment Card Industry Data Security Standard
PRF	Pseudo-Random Function
PRP	Pseudo-Random Permutation
SE	Structured Encryption
SSN	Social Security Number
OPE	Order Preserving Encryption
ORE	Order Revealing Encryption
OTP	One-Time Pad
UBI	University of Beira Interior
XOR	Exclusive OR

Chapter 1

Introduction

As the worlds data moves into the cloud, the privacy and security of our data moves into the ground. It has become common to hear news about yet another data leak, and still, these leaks keep happening[Cen18]. Why do these leaks keep happening? How can the security of these systems be improved? How can we achieve better data security and privacy while still allowing for instance third party cloud services have access to private information without the possibility of leaks?

The problem of keeping data secure while still allowing cloud services to run over the data, performing searches and other tasks, is not a simple problem to solve. These cloud services need access to client information in order to perform computations in a way that both benefits the company that is running the service, as well as the end user who can benefit from the result of that computation.

One can think of many ways to approach this problem. From correctly implementing security measures to ensure that data, even if leaked, can resist to offline attacks.

Although many approaches exist to try and solve this `cloud problem`, none of them actually solves the problem in its entirety and as more and more cloud services emerge so does the desire for a solution to this problem.

In this dissertation we take a look at one particular problem that many companies and individuals face on a daily basis. Delegating sensitive data to be stored in the `cloud` is not always an easy choice to make. Ever had the feeling of ”Am I giving away too much personal information?” when filling online forms? That is because, deeply inside, we know that our information can end up in the wrong hands, but at the same time, we know that our information is needed in order for us to acquire services.

The problem that we are analysing in this dissertation is a similar case of data delegation to a third party company that needs to process this data for credit card frauds. This data is, obviously, confidential and could cause problems for the information owner, in this case a bank,

to lose this information to an untrusted party. How do we allow a third party to have access and process our data (confidential data) but still keep it secure enough to make sure that the third party in question cannot use the data to its own advantage, or, how do we make sure that even if the data gets stolen in transit or while in the hands of the third party that it stays secure?

These are some questions that we will analyse in this dissertation with the help of recent cryptographic advancements.

1.1 Objectives and Contributions

The main objective of this dissertation is to develop a solution for a very specific problem of data delegation to an external third-party. This kind of problem can be found in many situations where data needs to be stored or processed by a cloud service provider. As previously mentioned, there are many ways to approach this particular problem, one of them being, a cryptography approach, where a specially designed cryptographic scheme could solve this exact problem.

To achieve the main objective of this dissertation we first need to understand what current cryptographic schemes exist that try to solve this or similar issues. To study their development throughout the years and the current state of such schemes. If such schemes exist then we need to understand how they work and where and if they can be applied to our specific problem. If no such scheme exists, then, we will try to understand if similar problems have been solved before and what kind of work has been done so far in this area that might apply to our problem.

Another possible approach is to try to solve this particular problem using current existing tools that might allow us to solve this particular problem, the only downside being that usually when using existing tools, that solution can only be applied to a single instance of the problem and not as a general solution.

The contributions of this thesis include a study of two distinct cryptographic primitives, namely FPE and ORE, including their implementation and subsequent performance using the Python language. A detailed description a particular problem related to the security and privacy of data delegation was given as well as a discussion of possible solutions and the difficulties and challenges encountered relating to the proposed solutions.

1.2 Thesis Outline

This dissertation is divided into six chapters.

- First Chapter - Introduction. This chapter has a brief description of the motivation behind this dissertation and presents the objectives it aims to achieve.

- Second Chapter - Problem Definition. This chapter contains the definition of the problem being studied in this dissertation. We give a detailed explanation of the problem at hand, namely from the Payment Card Industry and how it can affect the security of customer data in credit card transactions in several identified scenarios.

- Third Chapter - State of the Art. This chapter gives an introduction to the relevant background information that is needed in order to fully understand the ideas behind our proposed solutions. This section is composed of both, a theoretical explanation of concepts and cryptographic schemes, as well as, their purpose and definitions.

- Fourth Chapter - Implementation. This chapter contains the high level description of prototype implementations of several schemes that were made during this dissertation. These prototypes have as a basis the concepts presented in the second chapter. Although these prototypes are not relevant for the proposed solutions, they give a different perspective on how the involved schemes actually work.

- Fifth Chapter - Proposed Solutions. In this chapter several approaches to tackle the problem of delegating encrypted data to a third party are proposed. These solution are analysed in light of the preceding chapters, in particular their characteristics and whether they really solve this particular problem.

- Sixth Chapter - Conclusion and Future Work. In this chapter we take a look at what was accomplished during this dissertation, as well as, what was left to be done and what might be a possible path forward if this work is to be continued. We then conclude this dissertation by giving a view of the current state of the current cloud service providers and the security they offer.

Chapter 2

Problem Definition

2.1 Introduction

Due to business logic, marketing and regulatory frameworks, data will often be outsourced in order to be used as part of some analysis that can help a company improve its business strategies, conform to current regulations or some other business necessity. With this in mind the main goal of this dissertation was to study and propose a solution to a real world problem related to sensitive data being delegated to a cloud infrastructure where it is then subjected to a third-party analysis.

The main problem when delegating data to a third party, even when that third party is simply some other department of the same organisation, is that this data, which may be sensitive, is exposed to a whole new world of possible attacks and leaks, privacy issues and industry compliance requirements.

One can think of many solutions to solve the problem of data being delegated to third parties. For instance using strong access controls and thus only allowing certain personnel to have access to sensitive data, or even encrypting the data so that only those who have the secret key can access the underlying data. These solutions, although improving the sense of security of the people involved, actually create a whole new world of insecurities and organisational problems.

When looking for solutions to keep data safe from attacks, we have to think of all possible scenarios and leaks. We need to think of online and offline security, what primitives to use, hash functions, access control, physical access control, how data is encrypted at rest or when it's being used or transferred. If we need to process data, how will it be decrypted or how much information of the original data can be leaked to allow for services to still use our data but at the same time be secure enough against attackers.

Data delegation is a fairly common problem in today's cloud based world where more and more services require the upload of sensitive information. Encryption schemes are usually designed

to prevent attackers from recovering the original data from secured ciphertexts and not to allow some information to be leaked and allow for services to fully operate under still secure data. Due to this current limitation of encryption schemes, each cloud platform has its own security policy and protocols that usually ensure the safekeeping of sensitive data for both storage and processing. Also the various industry standards further restrict the solutions that may be proposed.

The problem we will define comes form the PCI. The main set of standards relating to the transmission of payment type card data is known as simply *PCI*. Payment cards may be debit, credit or prepaid cards that are branded with one of the major associations of players in this area, namely Visa, MasterCard, American Express, Discover and JCB. PCI-DSS [Rou] is a set of security standards drawn up to offer guidelines for companies that deal in data related to payment cards, in fact all business that store, process or transmit payment cardholder data must be PCI Compliant in order to participate in the global network of card transactions. The level of security, i.e the set of guidelines that must be adhered to, is determined by a number of factors, these factors include the number of transactions that the company processes each year, whether or not card data is being stored or outsourced to a third party and more ominously by the number of so called “data breaches” or other type of security incident that the company has been exposed to, unfortunately as the public are all to aware attackers and malicious insiders continue to access company networks and steal payment data using compromised credentials. Due to the difficulties of adhering to the more stricter PCI standards compliance levels many companies outsource components of the card transaction process to third parties, such as payment acceptance, fraud analysis, settlement, in fact often the whole e-commerce checkout/payment process is outsourced to specialised third parties. Having multiple parties involved credit card purchases further increases the risk of data being stolen, as well as, increasing the difficulty of standard adaptation by all the parties involved.

This chapter contains a detailed description of the particular problem being studied, as well as, all the solutions that were found during the research process. A detailed explanation of all the proposed solutions and their pros and cons are also included with each of the proposed solution descriptions.

2.2 Problem Definition

As previously mentioned, the main inspiration for this dissertation is to study and propose several solutions for a real world problem regarding data delegation to third parties or cloud infrastructures.

The problem itself is inspired from a real company that assists other companies in maintaining high levels of security throughout their technological infrastructure.The company has several successful projects in the security area and works directly with banks to help them keep their data secure.

One of the biggest problems that banks face is fraud that results in theft. Stealing money or assets may occur in several different ways credit card frauds being just one of them. Credit card purchases or transactions are one of the preferred methods for online (card not present) purchases and (card present) physical store purchases. Currently there is such a high amount of transactions frauds that may occur can simply blend in amongst all other transactions, thus a high quality automated system for detecting frauds is mandatory. When the problem was first introduced for this dissertation, it came with the revelation that credit card transactions are often not processed or analysed by the bank itself, but rather these transactions are sent to a third party company that specialises in fraud detection and transaction analysis. While this revelation actually makes sense, because it is better to have a company that is dedicated to detect credit card frauds analyse a bank's transaction, it also raises concerns about the security and confidentiality of the millions of transactions that are legitimate.

For a better understanding of how each credit card transaction occurs, 2.1 shows the overall structure of the credit card network from when the customer uses his credit card to make a payment at the terminal station, along with all the middle processing stations,from collecting payment information at the merchant, through payment gateways, sending that information to a dedicated acquiring service/bank, credit card network etc. until it reaches the customers issuing bank.

Most of the credit card transactions go through a very similar network architecture. This architecture, as presented in 2.1 already presents a problem for data security as both the transaction information, the credit card information and also the customers personal information needs to travel through several processing stations before reaching the actual bank. This presents an opportunity for an attacker to steal sensitive information.

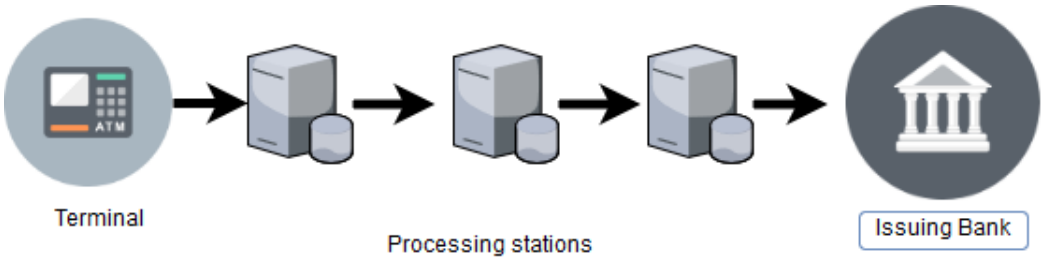

Figure 2.1: Credit Card Transaction

Most payment systems, however, taken steps to protect customers data by adding FPE[1] into the current network architecture or some system of Tokenizing (tokenizing is the act of replacing sensitive data with some of the value called a token). Adding FPE is not only good for data security, but also good for maintaining the same processing stations algorithms without having to adapt the algorithms and associated IT infrastructure to a new cryptographic scheme because, since FPE preserves the format of the data, for the processing stations the data will look like normal credit cards going through the system. This is especially important for systems that use legacy software which is designed to deal with real credit card numbers and not fully tokenized (encrypted) data.

Figure 2.2 shows how a FPE can be used in a credit card transaction.

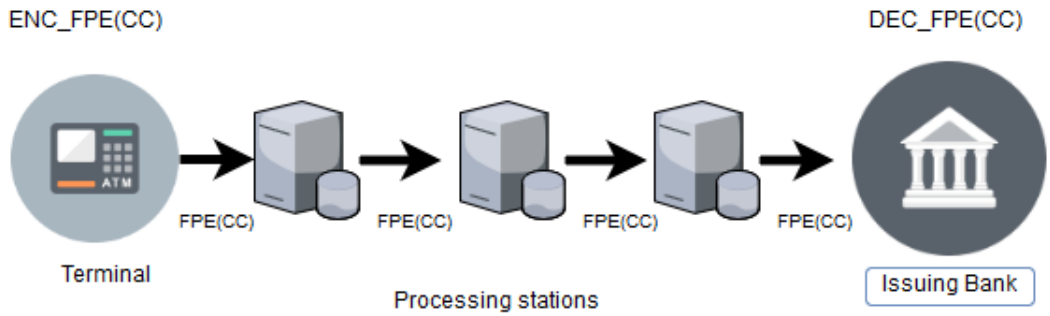

Figure 2.2: Credit Card Transactions with FPE.

An important problem with credit card transactions is how to detect possible credit card frauds. This can be done in multiple ways and at multiple points in the processing pipeline. For instance the third-party can be called on when each transaction occurs to look for a possible misbehaving customer or the third-party can be invoked at a later stage, at the end of the day for example, and go through several transactions for a given bank and correlate data from multiple transactions for multiple customers.

To illustrate both of these two possibilities, 2.3 and 2.4 show how a third party can come into

[1]FPE is an encryption scheme that allows data protection where the cipher texts have the same format as the plain texts. This scheme will be fully explained in the following section.

play when a credit card transaction is made.

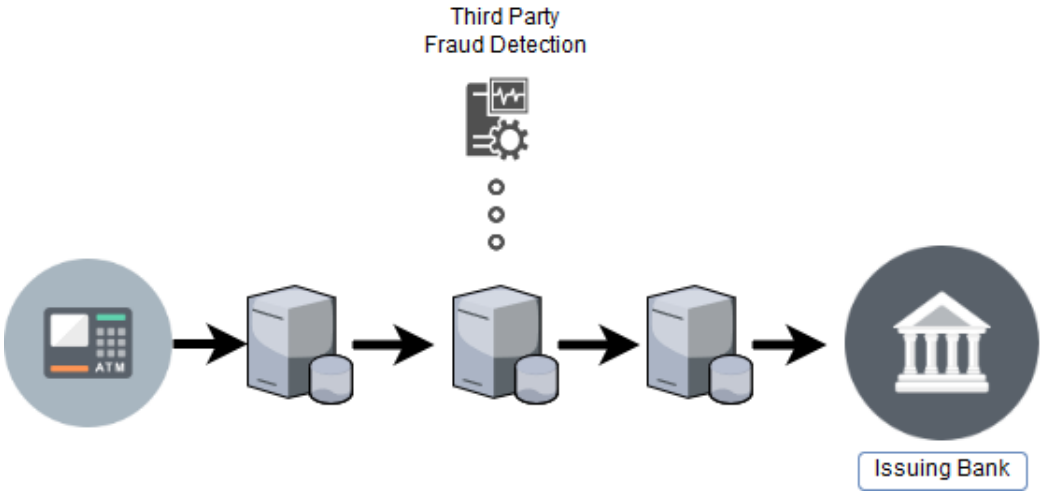

Figure 2.3: Third Party processing each transaction

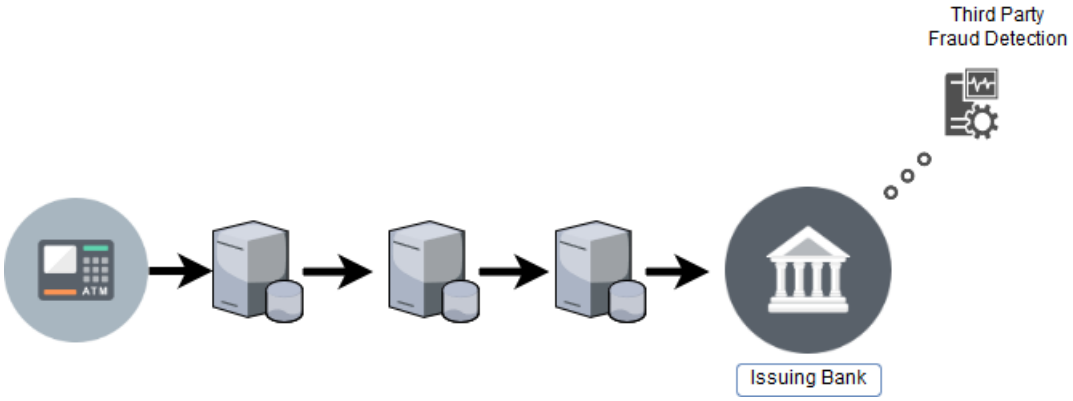

Figure 2.4: Third party processing after several transactions.

Let us now briefly discuss each situation. For the first scenario, represented by 2.3, the third party needs to evaluate each individual transaction and provide feedback to the network regarding the validity of the transaction. If a transaction is considered as fraudulent, the issuing bank may reject the transaction.

Although these is a good approach to decrease the likelihood of a fraudulent transaction to go through the system unnoticed, this also presents a problem when there is cryptographic protection of the credit card information that goes through the network, as shown in 2.2. By having the third party evaluating each individual transaction, the third-party also needs access to the decryption key of the FPE scheme in place. This alone, increase the risk of unauthorised access to confidential data.

For the second scenario, the third party only has access to the transactions after the bank has validated the transactions. This can happen in periodic reports, at the end of the day for

example. This approach allows the bank to potentially take additional measures to protect customer data by using other cryptographic approaches before sending transaction data to the third party for analyses.

These two scenarios and others will be analysed in detail in the following sections. However our model will only assume that the third party receives credit card information where in a real world scenario much more information would have to be delivered to the third party, such as, names, postal codes and so on. We also assume that the third party had to correlate data from multiple users at the same time and not only of just one customer, in which case a FPE scheme would suffice.

Assuming that the credit card information is protected by Format Preserving Encryption let us consider the following examples of a third-party company conducting an audit of a customer transaction information.

2.2.1 Scenario 1 - Single Transaction Processing

This case considers the case of a third party processing every transaction as they happen and as shown in Figure 2.3. If we consider that every transaction is protected by a FPE scheme, then we must determine how to allow the third party to access the original data, so that it can determine the validity and legitimacy of the transaction.

In this scenario, the third party needs to correlate credit card information in order to assess their validity and if that data is encrypted with a FPE scheme, then we must have a way to let the third party have access to that protected data. Since the current cryptographic schemes do not allow for partial decryption of data we must give the third party access to whole content of the transaction.

2.2.2 Scenario 2 - Multiple Transaction Processing

For this study, we also considered the case where a third party analysis is conducted after several transactions have already been made, as shown in Figure 2.4. In this scenario, the third party might need to check individual transactions, as described in the previous scenario, but it may also need to correlate data from several transactions and several credit cards.

Since this analysis only occurs after the transactions are complete, we can take additional steps to secure this data at the issuing bank before actually sending the data to the third party. For example we can remove data that is irrelevant for the third party to analyse (e.g., remove customer names).

2.2.3 Scenario 3 - Anonymous Processing

Anonymous processing is the ideal scenario where the third party would be able to correlate data from multiple transactions in order to spot fraudulent transactions, but at the same time only has access to the encrypted versions of the data and not the data itself. For the third party to be able to spot fraudulent transactions it must be able to tell if two credit card numbers, even if protected by a FPE scheme, are the same or not. For a better understanding of what anonymous processing is, Figure 2.5

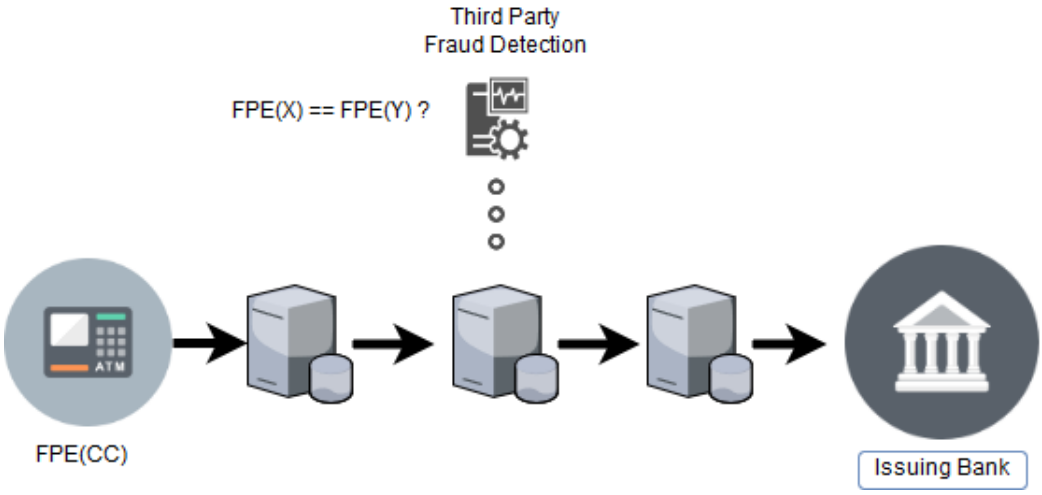

Figure 2.5: Third party correlating transaction data.

Let us consider the following example. Let X be a valid credit card number that is (FPE) encrypted to Y and transmitted from A to B via a series of third parties. In particular B uses a third party C in order to detect credit possible card fraud and theft, for instance by analysing the recent payment behaviour of this card X in terms of geographical location, purchase history, credit limit etc.

In order to ensure the privacy of the credit card users we don't wish to give C the encryption key. However in order to detect fraud C needs to determine which transactions are from card X.

One possible solution is to simply use for each particular customer a constant "tweak", however

this would mean that all transactions from X are encrypted to the same Y, which completely destroys the security of the FPE scheme (That is, the scheme can't be chosen-plaintext attack (CPA) secure).

In order to solve this problem we therefore require that the associated encryption process is associated with an additional function, FEQ, and key, KC, which when given to C, gives C the ability to determine which credit cards are the same whilst simultaneously permitting privacy over the transmission process, i.e any attacker is unable to distinguish equivalent credit cards. In other words let

$$\begin{aligned} Y_1 &= FPE(K, T_1, X) \\ Y_2 &= FPE(K, T_2, X) \end{aligned} \tag{2.1}$$

C is then able to apply the equality revealing function $FEQ(..)$ to determine if Y_1 and Y_2 are valid encryptions for the same X, or equivalently we have

$$FEQ(Y_1, T_1, Y_2, T_2, KC) = \text{TRUE or FALSE}$$

This type pf scheme with a Function FEQ is inspired by the order revealing scheme with "leakage function"[CLWW16][LW16] presented in section 3.3.3 where the order of two ciphertexts can be determined. This function has the same purpose, but instead of revealing the ordering, it would only reveal the equality of the underlying credit card information which is a much weaker requirement.
It then remains to analyse in whats sense this scheme can be considered a distinct type of cryptographic primitive or as a special case of more general cryptographic schemes.

The main objective of this dissertation is to try to understand if this FPE scheme for credit card frauds can be augmented with other cryptographic primitives that leak some information by design.

In this case, one proposed scheme is ORE. ORE leaks just enough information to allow for comparisons to be made between ciphertexts and determine the underlying plaintext ordering. By combining both these schemes, a third party company conducting an investigation for credit card fraud would have the power to, even if protected by FPE, determine if several different credit card numbers belong to the same customer.

In order to determine if a junction between these two schemes is possible we need to first

understand how much leakage is needed for a scheme to have in order to be of use for the company running the analysis and if this leakage allows a scheme to remain secure.

Even if such a scheme is possible to construct and still remain secure alternative solutions may also exist that don't involve constructing a new scheme but instead make use of other existing tools. Therefore to allow for a full investigation to occur we also need to consider these other possibilities in case they in fact offer more logical or efficient solution.

During this dissertation there was a great effort to understand, even the small details, of both schemes, FPE and ORE. What they do, how they do it, where and when to use them, how much and if they leak and why.

Chapter 3

State of the Art

3.1 Introduction

In this section we will start with a review of the basic definitions of security due to their importance when discussing cryptographic primitives in general. We will also take a look at some cryptographic primitives that were studied during this dissertation and that are in some way related to the *cloud problem*. Finally, a review of cryptographic primitives that are, currently, only theoretical results, or in other words, schemes that have the potential to solve some of the problems discussed in the previous section but are not yet considered practical, in the sense that a practical implementation can not yet be built.

3.2 Defining Security

In this section we introduce the most basic definitions of security that are related to the cryptographic primitives that were studied during this dissertation. The most important aspect of understanding any cryptographic scheme is being able to understand its security definitions. In order to understand the various cryptographic schemes presented in the following sections, it is important to first understand the fundamentals of security and where they originated.

3.2.1 Perfect Secrecy and Semantic Security

When studying or working with cryptographic schemes, it is of paramount importance to understand how the security definitions define the scheme itself. For any good cipher, there are always security definitions attached, and without these definitions, the scheme holds no ground. In a way, the scheme is as good as its security definition, or, in another words, a cryptographic scheme is nothing without a good security definition.

These security definitions may depend on the functionality of the underlying scheme, and some definitions may be more strict than others.

Considering the case of Perfect Secrecy, formalised by Shannon[Sha49], it defines how perfect security can be achieved in a cryptographic scheme at the cost of functionality. Having perfect secrecy means that a given ciphertext does not have any information that can be related with the original plaintext. One scheme that has perfect secrecy is called OTP. This scheme achieves perfect secrecy because the key length is always equal to the size of the plaintext. This allows for the maximum level of security but it also lacks the functionality any scheme needs. In the case of OTP, functionality suffers from the security definitions of the scheme. Having perfect secrecy means that the key material is at least as big as the plaintext itself, hence, when exchanging information over a secure channel, one may as well exchange the plaintext directly instead of the encryption key, the only exception being, in the case where one would exchange an encryption key via a secure channel in order to, in a later date, exchange an encrypted message over an insecure channel.

This limitation on functionality makes perfect secrecy not practical in most real world scenarios.

In order to cope with real world scenarios, cryptographic schemes are usually built with another weaker security definition in mind called, Semantic Security. This notion of security affects most security schemes used in today's world. Its goal is to provide *enough* security against real world attackers that are not infinitely powerful and that actually have limited power to decipher encrypted data. This notion is based on the amount of computation power any potential attacker might hold when trying to attack a particular cryptographic scheme, thus, limiting the amount of *security* needed for a scheme to hold it's confidential information. Another factor in favour of semantic security is that, any scheme that is semantically secure can exponentially increase its security factor by only increasing its key size by a few bits, while an attacker can only increase its power linearly (by using more computers to brute force).

One famous quote from the original paper of semantic security by Goldwasser and Micalli [GM84]:

> Whatever is efficiently computable about the clear text given the cyphertext, is also efficiently computable without the cyphertext.

This means that, if our scheme does not have any security flaw, besides its key space (which is reduced based on semantic security), then, as computation power of our adversary grows,

we can simply increase the key size of our scheme, which means an exponential increase in computation for our adversary.

The security definitions of many schemes are usually built around the idea of the computation power of an adversary, it can't be unlimited (except for perfect security). This computation power and the ability of these schemes to resist brute-force attacks are often enough to reason about the security properties of a particular scheme.

This notion of semantic security and indistinguishability, is useful in the security definitions of many primitives that are useful in cryptography. As an example that is useful when we later discuss Feistel networks we give the definition of a strong pseudo-random permutation from [KL07].

Définition 1. *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient polynomial time computable keyed permutation. In this case the calculation of the permutation, and also its inverse, given the key and some value are both known and efficiently computable.*
Then F is called a strong pseudo-random permutation if for all probabilistic polynomial time distinguishers, D*, there exists a negligible function* $negl$ *such that:*
$|Pr[D^{F_k(.),F_k^{-1}(.)}(1^n) = 1] - Pr[D^{f_k(.),f_k^{-1}(.)}(1^n) = 1]| \leq negl(n)$

where $k \leftarrow \{0,1\}^*$ *is chosen uniformly at random and* f_n *is chosen uniformly at random from the set of all permutations on n-bit strings.*

This definition basically says that distributions of strong pseudo-random permutation are indistinguishable from a uniform distribution to an efficient observer. A distinguisher is any algorithm, or statistical experiment, chosen by some adversary who wants to refute (or prove) a given hypothesis.

These types of definitions give us guarantees when it comes to on how many schemes actually protect our data by having the same properties as, in the case of a Feistel network, truly random functions. Since we can not have truly random functions these definitions need to be adjusted to represent the amount of power an opponent, or attacker, might have to break a scheme.

3.2.2 Security Tradeoffs

These security definitions can, however, be too strict for some scenarios. Consider the problem that this dissertation is addressing. How can we share sensitive information with a third party whilst keeping our data protected by a semantically secure encryption scheme? This, of course, without sharing the secret keys that protect the data.

Some schemes, and in order to provide a greater amount of functionality, do allow for some leakage to occur.

Just as the case of schemes that have semantic security, these schemes also rely on modelling an attacker that has limited power at its disposal. Whilst for semantic security, we assume that an attacker doesn't have an infinite amount of power, for schemes that have some kind of leakage, the assumption is made around the fact that the attacker is further limited in its capabilities. Plus, the inherent behaviour of these schemes goes against the definition of semantic security.

As an example on why these schemes need to, sometimes, use security definitions that are not as strict as they can be, consider the case of OPE, which we will properly introduce later in this chapter. OPE, is a scheme that allows ciphertexts to be compared directly and output the ordering between the two. A scheme like OPE cannot have a security model that is, for instance, CPA secure, because an attacker can easily break this notion just by comparing the two ciphertexts. So, in order to have a definition for this scheme, the behaviour of the attacker has to be limited by a great amount, in this specific case, only to allow for ordered messages to be allowed. This notion is also called, IND-OCPA[BCLO09].

Just as OPE, ORE also has a leakage profile and has an adapted set of security definitions that take into account the fact that the inherent properties of the scheme will leak some information to attackers.

3.3 Cryptographic Primitives and Notation

A cryptographic primitive consists of a set of algorithms, Π, which often contains the *Setup*, *Encryption* and *Decryption* algorithms. Although, some cryptographic (cipher) primitives may have additional algorithms, these are the three basic ones that almost every scheme has. A

cryptographic scheme is also defined over a message space, $\mathcal{M}$, which defines what is accepted as an input for the scheme, in other words, the domain on which the scheme can operate. i.e., If a message space is [0..9], then, the scheme can only operate on digits ranging from 0 through 9. Additionally, a scheme is also defined over a key space, which basically defines the amount of distinct keys accepted by the scheme. The key space is usually, but not always, defined as *K*.

3.3.1 Format Preserving Encryption

Format Preserving Encryption FPE is a cryptographic scheme that is somewhat different from other classical cryptographic schemes as here the main goal is to allow for the format of the original message to be preserved throughout the encryption process [BRS10a]. For example, when using FPE to encrypt, say, a credit card number, the result of the encryption algorithm will also be a valid credit card number while still allowing whoever has the encryption key the ability to obtain the original credit card number. FPE has the ability to not only accept credit card numbers, but any format that can be defined by the user, such as, emails, home addresses, IP-addresses and so on.

There are many ways of constructing a format preserving scheme but in this dissertation, we will only discuss the standardised FFX mode of operation. This mode of operation relies on a Feistel network to protect the original plaintext that is given as input [BRS10b]. The X in the FFX name stands for the different parameters that this mode of operation allows. Allowing for a greater grain of control by the user.

A Feistel network is a general method of constructing an invertible function (in fact a permutation) using a non-invertible function (or functions) in several keyed rounds, thus forming a sort of network. It was designed by Horst Feistel whilst working at IBM.

Figure 3.1 shows the internal configuration of the Feistel network for the FPE.

This Feistel network accepts several variables alongside the plain text of length n (denoted by A||B), namely a key (k) and a variable which is incremented in each round of the network, in Figure 3.1 we note this variable as i. Each round of the Feistel network also works with the length of the `Tweak` given as input, t. The length of the input itself is also used in each round in order to modify the other half of the input by modular addition. The round function is usually a block cipher.

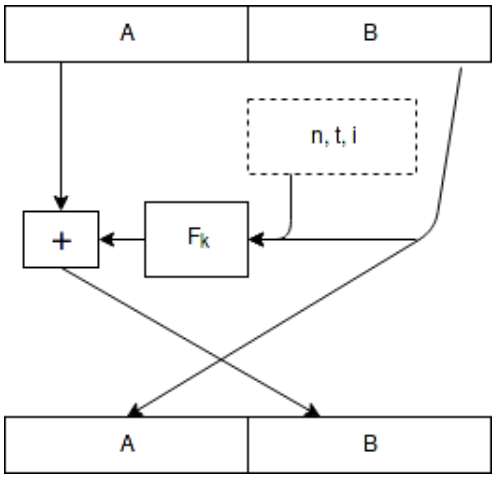

Figure 3.1: FPE Balanced Feistel Network

The following algorithm represents the encryption process of FPE. This algorithm takes as input the string X to be encryption and the tweak T which will also be used during the encryption process. The result of this algorithm a string Y such that the length of Y is equal to the length of X. The input string must be over a pre-defined alphabet of radix/base characters. Both the string contents as well as their lengths will be used during the encryption process, for both the input string and the tweak. For this encryption algorithm, there is also an PRF which is a predefined 128-bit block cipher used in every round of the Feistel network. This PRF is used to combine pre-defined values, such as, the radix used, the length of the input, the length of the tweak and so on. The output of the PRF will then be used as the seed for the final value that will become the input of the next round of the Feistel network.

Algorithm 1 FPE encryption algorithm.

Data: Input: String X of length n. Tweak T of length t
Result: String Y, such that length(Y) = n
initialization
Let $u = floor(n/2); v = n - u$
Let $A = X[1..u]; B = X[u+1..n]$
Let $b = ceil(ceil(v*$ log(radix))/8)
Let $d = 4$ceil(b/4) + 4
Let P = $[1]^1$ || $[2]^1$ || $[1]^1$ || $[radix]^3$ || $[10]^1$ || $[u \ mod \ 256]^1$ || $[n]^4$ || t^4
for *i = 0; i < 9; i++:* **do**
  Let Q = T || $[0]^{(-t-b-1) mod 16}$ || $[i]^1$ || $[B]^b$
  Let R = PRF(P || Q)
  Let S be the first d bytes of the following string of ceil(d/16) blocks:
  R || $CIPH_k(R \oplus [1]^{16})||CIPH_k(R \oplus [2]^{16})...CIPH_k(R \oplus [ceil(d/16)-1]^{16})$
  if i is even, let m = u; else, let m = v.
  Let c = $(A + S)$ mod $radix^m$
  Let C = (c)
  Let A = B
  Let B = C
end
Return A || B

Format and Order Revealing Encryption

For the encryption process to occur, additional data is needed before actually encrypting for example a credit card, or any other input, the most important of which is the so called Tweak. This Tweak value acts almost like an IV for a Block Cipher in CBC mode, allowing an encryption of the same credit card with the same number to have different results, i.e randomised encryption. (Assuming the Tweak value is different).

The decryption process of the FPE scheme is identical to the encryption process. For the decryption process to occur, we only need to change the round number of the Feistel network, thus, we now start from round 8 and go all the way to 0. Also, instead of performing a modular addition between the left side of the input with the result of round function, we need to compute a modular subtraction. And finally, the last change is the how the input sides, A and B in the encryption function, are swapped. These small differences between the encryption and decryption process make FPE an easy scheme to implement and means that the scheme has identical performance in both the encryption and decryption algorithms

Finally it should be stated that the basis of this scheme, the Feistel network, has been extensively studied, starting with the work by Luby and Rackoff[LR88] who proved that 3 rounds of the network are sufficient to make the resulting block cipher a pseudo-random permutation, or if we increase the number of rounds to 4, the result will be a strong pseudo-random permutation of the input. This assumption is based on definitions such as the one presented in the previous section 3.2.1.

3.3.2 Order Preserving Encryption

OPE[BCLO09] is a cryptographic scheme that, as the name suggests, preserves the ordering of the ciphertexts in relation to the plaintexts.OPE has generated a lot of interest due to its use when performing range queries over a database of encrypted data. The first formal treatment of this primitive if by Agrawal [AKSX04] in 2004. The paper in 2011 by Boldyreva et. al. in [BCO11] is however more interesting as it gives a treatment of OPE in terms of a scheme that leaks some information, this opened the way for many new schemes that have some kind of leakage profile, or in other words, where something about the plaintext is revealed in order to increase functionality at the cost of some level of security. The OPE by Boldyreva was the first in a line of many cryptographic schemes that aimed to both improve the overall security of data that is stored by third-parties, as well as, give these third parties the ability to search and index data that is encrypted.

OPE allows for direct comparisons to be made in between ciphertexts, which not only allows for comparisons to be made without any cryptographic key, but also allows for database indexing and range queries to be made, just as they would be made in a scenario without OPE.

The direct comparison of ciphertexts can be defined simply as:

$$x > y \iff Enc(s_k, x) > Enc(s_k, y)$$

which means that, if the value of x is greater than the value of y then, the encryption of x will have a greater value than the encryption of value y under the same encryption key, s_k.

Having this property does, however, bring a reduced level of security. This reduced level of security comes with the fact that, by preserving the ordering of the plaintext, we are actually giving away information of the plaintext itself, just by giving the ciphertext. This goes against the definitions of semantic security itself defined in the previous chapter.

This forces this scheme to have a lower level of security when compared to other schemes that achieve semantic security. For some scenarios however, having a lower definition of security could allow for a better functionality of the system.

In the case of OPE, the amount of information leaked by the scheme was proven to be too much to allow for a still safe to use scheme and with enough functionality to be used.

Although the OPE scheme has some leakage that is desirable, equality and ordering, it also has leaks that were not intentional. OPE leaks the relative distance of the ciphertexts, which means that, by looking at the ciphertexts alone, we can roughly determine the relative distance of the original plaintexts. This additional leakages can be translated to a scheme leaking half of the bits of a given ciphertext. Having half of the bits leaked, allows for attackers to recover almost all of the original plaintexts. This paper[GSB+17] by Paul Grubbs et. al. conducted attacks against OPE and ORE schemes and showed that around 90% of the values of a database protected by the aforementioned schemes were decrypted without the use of the original key.

Despite having more leakage than it was designed for, OPE could still be used in a scenario with high-entropy values where the ordering of this data is not a relevant factor.

3.3.3 Order Revealing Encryption

ORE is a cryptographic primitive that aims to provide a much wanted functionality over ciphertexts, namely, the ability to extract some information from two ciphertexts about the order relation between the original plaintexts. Of course the primitive should do this without compromising the security of the plaintext. In the ORE case, a public comparison function is provided that can be used without a private key in order to obtain the original order relation. The definition and analysis of the schemes security is achieved by defining a leakage profile of the scheme, the leakage profile defines what information is leaked about the plaintext.

ORE was first defined in a seminal work by Dan Boneh et. al. [BLR$^+$15] with a scheme that is not practical due to the fact that it relies on multilinear maps. Besides not being computationally practical, the security of multilinear maps is not well understood, thus schemes that rely on multilinear maps are not generally used in real world applications.

The first ORE scheme that did not rely on multilinear maps was introduced by Chenette et. al. [CLWW16]. This scheme aims to allow comparisons to be made between two or more ciphertext to reveal their ordering, without revealing anything else. This leakage profile, if secure, brings great functionality for its users, allowing for somewhat searchable databases to be built. For a better understanding of this scheme, lets first define its leakage profile.

$$\mathcal{L}_f(m_1, \ldots, m_t) = \{(ind_{diff}(m_i, m_j),\ \mathbf{1}(m_i < m_j)) : 1 \leq i < j \leq t\}, \quad \textbf{(3.1)}$$

At a first glance, this function can be confusing. This function simply means that, when comparing two ciphertexts encrypted by an ORE scheme, comparisons can be made by directly comparing the indices of each ciphertext, where the first bit that differs will determine the actual ordering of the ciphertexts.

In order to understand how this scheme encryption and comparison works, we need to look at the scheme construction. This ORE[CLWW16] scheme is defined as follows.

We begin by choosing a security parameter $\lambda \in \mathbb{N}$, an integer $\mathcal{M} \geq 3$, and a secure PRF $\mathcal{F}$ where $\mathcal{K}$ x $([n]$ x $\{0,1\}^{n-1}) \rightarrow \mathbb{Z}_{\mathbb{M}}$

ORE is then defined by three algorithms, *Setup*, *Encrypt* and *Compare*.

- ORE.Setup(1^λ) $\rightarrow$ A uniformly random key k for the PRF $\mathcal{F}$ is chosen. k is the master key sk

- ORE.Encrypt(sk, m) $\rightarrow$ Considering $b_1 \dots b_n$ as the binary representation of the message m, the encryption algorithm is as follows for each $i \in [n]$:

$$u_i = F(sk, (i, b_1 b_2 \dots b_{i-1} || 0^{n-i})) \ + \ b_i \ (mod \ M)$$

- ORE.Compare(ct_1, ct_2). Given two ciphertext for comparison, the comparison algorithm is defined as follows. Both ciphertexts are constructed as:

$$ct_1 = (u_1, u_2, \dots, u_n)$$

$$ct_2 = (u'_1, u'_2, \dots, u'_n)$$

  First we consider i to be smallest index where $u_i \neq u'_i$. If no such index exists, we output 0. If an index exists where $u_i \neq u'_i$, then, we output 1 if $u'_1 = u_i + 1 \ (mod \ M)$ and 0 otherwise.

The aforementioned construction of this ORE scheme is fairly straightforward and simple to implement, which greatly reduces the risks of insecure implementations. As noticeable, this scheme does not include a decryption algorithm. Since the main objective of this scheme to allow others to perform comparisons on encrypted data, there's no need to define a decryption algorithm. Although, one could easily be constructed using binary search by the owner of the original data. By starting with an encrypted random number, we can then use the ORE compare function to compare the encryption of that random number and compare it to the ciphertext that we want to decrypt. Depending on the result of the comparison function, we know that the ciphertext is either greater or less than our encrypted random number, which means that we can then generate another number based on the comparison result, moving closer and closer to the original number.

Unfortunately, the leakage profile of this version of ORE ended up leaking too much information, allowing attacks to fully reconstruct databases protected by ORE[DDC16]. Allowing attackers to know the exact position in the ciphertext where differences occur simply means that too much information is being revealed.

Format and Order Revealing Encryption

In another attempt to make a secure ORE scheme, Lewi et. al. in [LW16], were able to reduce the amount of information that leaks from their scheme, allowing only an attacker to know which block of ciphertext (8 bit blocks) differs. The fact that the attacker no longer has the exact position where the difference between ciphertext occurs greatly increases the overall security of the scheme. In this work, Lewi et. al. also have an interesting way of dealing with offline attacks against databases that might store sensitive information, by actually splitting a ciphertext in two parts (left and right). The left part of the ciphertext is to be stored on the user side, while the right part of ciphertext is meant to stay in a third party database. Both parts of the ciphertext have different security requirements and functionalities. While the left part of the ciphertext has a lower security requirement, for obvious reasons, it will be used as a key for the right part of the ciphertext, allowing for comparisons to be made in between left and right parts. Lewi et. al. also achieved semantic security on the right part of the ciphertext, allowing it to withstand against offline attacks in the case of information theft from the third party. Although this scheme is not the main focus of this dissertation, a brief overview of its construction, as defined in [LW16], ORE is composed by four main algorithms, Setup, Encrypt(left), Encrypt(right) and Compare.

- $ORE.Setup(1^\lambda)$:
  - PRF Key: $k \xleftarrow{R} \{0,1\}^\lambda$
  - Random permutation: $\pi : [N] \to [N]$
  - Output secret key pair: (k, π)
- $ORE.Encrypt_L(sk, x)$:
  - Computes and returns: $ct_l = (\mathcal{F}(k, \pi(x)), \pi(x))$
- $ORE.Encrypt_R(sk, y)$:
  - Random nonce: $r \xleftarrow{R} \{0,1\}^\lambda$
  - For each $i \in [N]$:
    $v_i = CMP(\pi^{-1}(i), y) + \mathcal{H}(\mathcal{F}(k, i), r) \ (mod\ 3)$
  - Output: $ct_r = (r, v_1, v_2, \ldots, v_N)$

- $ORE.Compare(ct_l, ct_r)$:

  - where $ct_l = (k', h)$ and $ct_r = (r, v_1, v_2, \ldots, v_N)$

  - Computes and returns: $v_h - H(k', r) \ (mod\ 3)$

This version of ORE still allows for comparisons to be made between ciphertexts and actually increases the overall security of the scheme by only revealing the block that is different between the two ciphertexts.

3.3.4 Functional Encryption

The most common encryption mechanisms that we use on a day to day basis, only allows to either fully encrypt or fully decrypt data, or the so called *all or nothing approach*[BSW11]. Whenever we try to allow data to leak from the ciphertext for the sake of functionality, the resulting security level is always inferior. FE is a public key crypto system that was first defined by Boneh. et al. in [BSW11]. FE allows for a specific piece of the ciphertext to be revealed to the key holder. The specific piece of the ciphertext that is revealed for a given key is usually called a function which is always bound to a specific function key. This function will allow the corresponding key holder to decrypt a specific function of the ciphertext and nothing else. As an example, if we consider a ciphertext that contains credit card information, then, we could have a function that would allow its key holders to extract only the first few digits of the credit card number to check whether the card is a VISA or a Mastercard. The ability to have this level of access control to the encrypted data gives data owners the ability to control who and how their data can be accessed. This approach where we can decrypt portions of the ciphertext would be the ultimate solution to the current cloud problem, described in chapter 1, that many cloud service providers have. Unfortunately, there doesn't seem exists way that FE could be implemented. The first real definition of a FE is given in the [BSW11] in a where the concept of FE is introduced, as well as, how other schemes, such as, IBE[BF01] and ABE[GPSW06], can be viewed as special cases of FE. A FE scheme could be constructed as follows[BSW11]:

- Setup$(1^\lambda) \rightarrow (pp, mk)$

  The setup algorithm generates a master and a public key based on the security parameter λ.

Format and Order Revealing Encryption

- Keygen$(mk, k) \rightarrow (sk)$

  The Keygen algorithm generates a new secret key from the master key for the input k.

- Encryption$(pp, x) \rightarrow c$

  The encryption algorithm encrypts x based on the public key given as input.

- Functional Decryption$(sk, c) \rightarrow y = F(k, x)$

  The decryption algorithm decrypts c based on the secret key given as input and outputs a function of the plain text x.

The main problem of FE is how such a scheme could be constructed in a efficient manner. There a few proposals for how a FE scheme could be constructed but nothing seems to be efficient enough to be of practical use. One way of constructing a FE scheme is through IO. Obfuscation is the technique that tries to make computer programs unintelligible to human eyes while preserving its functionality. If obfuscation was strong enough to hide every possible program function, then, we could use programs to hide sensitive information, such as, private keys. Unfortunately, as Barak et. al showed in their paper [BGI$^+$01], achieving obfuscation is impossible for some program functions. In the same paper, Barak et. al. also give the definition of IO. IO focus on the indistinguishably of the circuits themselves and not what is obfuscated. IO simply states that, given two equivalent circuits of the same size and functionality should be computationally indistinguishable. If IO could be constructed in a efficient manner, it would allow for most cryptographic primitives to be instantiated. This would allow schemes, such as, FHE and obviously FE. There is, however, only a few candidates to build IO. The first candidate for IO is from Gentry et. al. [GGH$^+$16] with a multilinear maps construction, which is far from being efficient. Recent works have however reduced the level of multilinearity required to a 5 degree multilinear map[AS17]. In conclusion, being able to implement a FE scheme would mean a tremendous advance to the cryptographic community as a whole new world of possibilities would emerge from the ability to say who and how much of the ciphertext could be decrypted. This would allow for many cloud services to still use our data in a user controlled way.

3.3.5 Multi-Input Functional Encryption

FE could be a true problem solver for the data delegation. Having the ability to specify which users have access to specific parts of encrypted data, and even then, not being able to learn anything else about the original plaintext is remarkable. FE still suffers from a flaw that may

be hinder cloud providers to adopt such a scheme (if it ever comes to be practical). FE does not allow for a function to be computer over multiple ciphertexts encrypted with different keys. This cause a service provider to be unable to conduct searches over multiple ciphertexts. MIFE however gives the ability to compute functions over multiple ciphertexts encrypted with different encryption keys, derived from a master key. Using the Master key the scheme permits the derivation of a special key which can be used to compute a function over the cipher texts for instance providing the ability to perform range queries over multiple ciphertexts.
By allowing range queries, MIFE could be a true problem solver for the cloud world and allow for a whole variety of services to securely perform computations over encrypted data.

A MIFE scheme is composed of four algorithms (Setup, Keygen, Encryption and Decryption) [GGG+14]

- $Setup(1^k, n) \rightarrow EK_n$
  Based on the security parameter k, the setup algorithm,outputs n encryption keys, $EK_1, EK_2, \ldots, EK_n$ and the schemes Master Key MSK

- $Keygen(MSK, f) \rightarrow SK_f$
  The key generation algorithm, takes as input the master key MSK and a n-ary function f, and outputs the secret key SK for the corresponding function f.

- $Encryption(EK, x) \rightarrow CT$
  The encryption algorithm, takes as input an encryption key EK and a plaintext x and outputs the ciphertext CT.

- $Decryption(SK_f, CT_1, \ldots, CT_n) \rightarrow y$
  The decryption algorithm takes as input a secret key for a specific function as well as any ciphertexts that are needed for the function f to be computed. The output of this algorithm is the result of applying the function f to the ciphertexts provided as input.

Although MIFE completely overtakes FE in its functionality, it also comes at a higher cost. From the same paper[GGG+14], their results show that MIFE implies IO. The same is not true for FE. This means that, any further improvements to the complexity of assumptions of MIFE only comes with the correspondent improvements for IO constructions. This implication means that, an implementable MIFE scheme might still be a far from happening. Although other proposals for building MIFE schemes exists, such as, the work by Boneh et.al[BLR+15] which does not rely on obfuscation constructions, but instead is inspired by obfuscation techniques, is based

on multilinear maps, which although being more pratical than IO, their security is not well understood, which may hinder the development of a MIFE scheme in a multilinear map setting.

Chapter 4

Implementation

This chapter discusses the implementation of several cryptographic primitives that were made during this dissertation. Implementing cryptographic primitives can give the developer a better understanding on how practical these schemes actually are and generally give a different perspective on how things work and how everything comes together to form a cryptographic scheme.

This chapter will discuss mainly two different cryptographic primitives. Order Revealing Encryption and Format Preserving Encryption.

For each of these schemes we will present some implementation choices that were made during development. For Order Revealing Encryption, two implementations were developed. These implementations reflect both scheme versions that were introduced by Chenette[CLWW16] and Lewi[LW16] respectively.

Although these implementations are not necessarily needed for this dissertation, implementing them certainly helped in understanding the core concepts behind each cryptographic scheme. We also include in this chapter timing comparisons between our prototypes and well defined implementations available online.

4.0.1 Order Revealing Encryption

The first implementation was of ORE[1] as stated in [CLWW16]. Implementation of this scheme was done by following the main construction of this scheme as it is presented in chapter 3 (Main Construction) of the aforementioned paper.

The main goal of this implementation was to follow the exact construction given in the paper, hence, the implementation tries to reflect, as close as possible, the construction presented in the paper and as defined in section 3.3.3.

[1] Implementation of ORE available at: https://github.com/rtpaiva/ORE

This prototype was implemented in Python [2] due to its flexibility regarding the type system, which allows for a quick prototype development.

This implementation is divided into two main blocks. The first block implements the ORE algorithms for encryption and comparison of data. The second block implements basic tests to assert application correctness.

The first block consists of two functions, the encryption function and the comparison function. The encryption function receives as parameters the key for the PRF and the plaintext to be encrypted. This function makes use of the BitArray[3] package for data type conversions. This function first converts the plaintext given as parameter to a binary representation. This representation will then be used in the main for loop. For each iteration of the for loop, the function gets the prefix of the message, which is, all binary values from the start of the message up to the index *i*. As an example, for the third iteration of the loop, where the *i* variable is set to 2, the prefix will contain all the binary characters from the start of the message up up to the *ith* value of the message.

To protect the contents of the prefix the ORE construction uses a PRF, in our case an HMAC. Once the prefix is protected, we now need a way to determine the order relations between ciphertexts. The order relations are maintained in the last step of the for loop by adding the current index of the message to the output of the PRF. The result of the addition is then masked by the modulo operation, which must be any value >= 3.

The result of this operation is added as the new block of the final ciphertext. When the for loop ends, the function returns the resulting combination of ciphertext blocks.

Code snippet 4.1 shows the implemented encryption function.

```
1 def encrypt(key: bytearray, message):
2     ciphertext = []
3     message_bitArray = BitArray(bin=bin(int(message)))
4 
5     # For each bit in the message
6     for i in range(0, len(message_bitArray.bin)):
7         # get the prefix
8         prefix = message_bitArray.bin[0:i]
```

[2] Python Software Foundation. Version 3.6.1. Available at http://www.python.org
[3] BitArray Package, version 0.8.1, available at https://pypi.python.org/pypi/bitarray/

```
9
10          # Get the PRF output with the key and prefix
11          prf_output = crypto.prf(key, BitArray(bin=prefix))
12          # Add the current bit from the message to the prf_output
13          block = (prf_output.uint + int(message_bitArray.bin[i])) % 3
14
15          # add block to the final ciphertext
16          ciphertext = ciphertext + [block]
17
18      return ciphertext
```

Listing 4.1: ORE Encryption Function

The second implemented function is the comparison function. This function simply takes two ciphertexts as parameters (ctx1 and ctx2) and outputs -1 if ctx1 < ctx2, 0 if ctx1 == ctx2 or 1 if ctx1 > ctx2. This function compares both ciphertext, byte by byte, and applies the comparison formula given in section 3.3.3. Code snippet 4.2 shows the implemented comparison function.

```
1  # compare ciphertext 1 and ciphertext 2
2  # if : ctx1 < ctx2 return -1
3  # if : ctx1 > ctx2 return 1
4  # if : ctx1 = ctx2 return 0
5  def compare(ctx1, ctx2):
6      if len(ctx1) > len(ctx2):
7          return 1
8      if len(ctx2) > len(ctx1):
9          return -1
10     else:
11         for i in range(0, len(ctx1)):
12             if ctx1[i] != ctx2[i]:
13                 if int(ctx2[i]) == (int(ctx1[i]) + 1) % 3:
14                     return -1
15                 elif int(ctx2[i]) == (int(ctx1[i]) -1) % 3:
16                     return 1
17         return 0
```

Listing 4.2: ORE Comparison Function

In order to determine the correctness of the implementation, the following test was designed to ensure that the encryption and comparison function were working as intended. This test generates random numbers in the interval $1000 <= \S < 5000$. For each pair of random numbers, the test algorithm calls the ORE encryption function and stores the result. The next step is then to compare the result of the ORE comparison function on these ciphertexts and compare the result with the actual ordering of the randomly generated numbers. If the results from the comparison function match the original comparison (without encryption), then, the algorithm counts the test as being positive. Any other result and it is considered a failed test. In a default configuration, this process is repeated 10000 times. The number of successful and failed tests is then printed in the standard output. From multiple test runs performed, all ORE comparisons were successful.

An implementation of this scheme is also given by the same authors of the paper [CLWW16].

Although this implementation is a prototype, several steps have been made by the same authors to increase the overall performance of the prototype in order to measure the encryption and comparison functions of the scheme. This prototype was developed in the C programming language and also uses the AES-NI instruction set. This allows the prototype to achieve a much higher performance than our simple python prototype. Just to see how much of a difference our python implementation has against the C implementation, we measured the average time it takes for both the encryption and comparison function to perform their corresponding tasks.

For comparison, the following table presents the time comparison between our implemented ORE scheme and the aforementioned authors, as well as, a publicly available ORE implementation by the github user *kpatsakis*[4].

Table 4.1: Encryption and compare time comparison

	key size	**Iterations**	**Encryption Average (μs)**	**Compare Average (μs)**
ORE (python)	128 bits	100000	1332.06	2.749
ORE (C + AES-NI)	128 bits	100000	0.00129	0.33
ORE - kpatsakis (python)	128 bits	100000	158.04	1.4168

For a small scheme such as ORE which as a relatively easy implementation difficulty, we can clearly see that our implementation of ORE is 10 times slower than a *random* implementation found on a Google search. This is mainly due to the choice of cryptographic library that is used. In our implementation, both the random values and the hash functions that are needed for this

[4]ORE implementation by the user "kpatsakis" on github. Available at: https://github.com/kpatsakis/OrderRevealingEncryption

scheme are computed using a importable cryptographic library. This together with bigger hash sizes used in our implementation may increase the overall encryption and comparison time.

4.0.2 Order Revealing Encryption New Construction

As previously mentioned, the Order Revealing Encryption scheme as presented in [CLWW16] is currently considered insecure due to flaws in the scheme design that allow an attacker to recover 99% of the plaintext values in a database[DDC16].

Kevin Lewi and David Wu[LW16] then proposed a new Order Revealing scheme, which is able to achieve semantic security on the data that is stored in a remote database.

This scheme, although achieving a higher level of security when compared with previous schemes, it also lowers the usability for any party involved, requiring more steps per user then other schemes. The final goal of this scheme is to withstand against offline attacks that have been more and more common in online service providers. Offline attacks happen when an attacker is able to compromise an online service provider and make a copy of the contents of a database, thus, being able to then attack the database as it own will.

To protect against offline attacks, this scheme splits the ciphertext into two parts, called the left and right ciphertext. The left ciphertext contains within itself the vital information that is needed to perform comparisons. This information includes the private key and the index value to compare to when making comparisons. The right ciphertext, which achieves semantic security, only contains the result of several comparisons made between the number that the user is trying to hide and a random nonce. All the comparisons results inside the right ciphertext are then protected by a PRF. The right ciphertext is then sent to the remote server where it is to be stored.

This scheme was also implemented in python, although only for small domains.[5]

[5]Implementation available at: https://github.com/rtpaiva/small_domain_ore

4.0.3 Format Preserving Encryption

The FPE implementation[6] followed the NIST[7] standard for the format preserving encryption mode FFX.

The first challenge of the FPE implementation was to actually choose which of the available formats would be implemented. Since the goal of the implementation was to only produce a prototype version, the FFX mode of operation was chosen because it relies on Feistel networks and gives a wider range of plaintext and tweak space than other formats.

The implementation for FPE is more complex than the implementation of ORE, thus, we will only discuss what we think are important aspects or functionalities of the implementation.

As previously mentioned, the implemented mode of operation was the FFX mode as provided in the NIST standard. Since the program to be implemented is merely a proof of concept, the programming language of choice was, again, Python. In the standard, several function definitions are presented in order to fully implement an FPE scheme. However, some of these functions were not implemented because they only provide data type conversions, which, the python language, as some other languages, already support it natively.

The main focus of this section will be on the two main functions that implemented the FPE scheme, namely, the Encryption and Decryption function.

The main component of these functions is the Feistel network structure. These functions are almost identical, with the exception of the Feistel network order, where the decryption function computes the Feistel network in a reverse order.

Before the Feistel network is processed on the given input, a setup phase is necessary. The setup process is fairly straightforward, it consists of extracting vital information from the input, such as the input length in order to divide it into two parts, which will be fed into the Feistel network and the left and right part of the input. The setup phase is also used to create the P value which is, essentially, the initial block to be fed into the Feistel network. The P block depends on the length of the input, as well as, the tweak length.

Code snippet 4.3 shows the setup process of the FPE encryption function.

[6]FPE implementation available at: https://github.com/rtpaiva/FPE
[7]NIST Special Publication (SP) 800-38G, available at:http://dx.doi.org/10.6028/NIST.SP.800-38G

```
def encrypt(K: bytes, X, T):
    assert check_config_file()

    X = str(X)
    T = str(T)
    assert len(X) > 0, error.ErrorHandler.ERROR_EMPTY_MESSAGE
    n = len(X)
    t = len(T)
    u = math.floor(n / 2)
    v = n - u
    A = X[0:u]
    B = X[u:n]
    assert len(A) + len(B) == n, error.ErrorHandler.
    ERROR_FEISTEL_MESSAGE_LENGTH
    b = math.ceil(math.ceil(v * math.log(config.RADIX, 2)) / 8)
    d = 4 * int((b+3)/4)
    P = generate_p(u % 256, n, t)
    bmagnitude = 10 ** int((n+1)/2)
```

Listing 4.3: FPE Encryption Function

Once all the required information has been constructed in the setup phase, the Feistel network is then launched. This Feistel network is launched with 10 rounds, as stated in the standard.

For each round of the Feistel network there are three main computations that need to be performed. The first one is to generate another block of information that will then be encrypted using an instance of AES in CBC mode.

This block that is generated in each round, called Q, depends on several pieces of information in order to assure that, for each round, the outputs of the AES cipher are different. The Q is constructed based on the following values:

- T - The Tweak given as input;
- t - The Tweak length;
- i - The Feistel round number;

- B - The right half of the message to be encrypted;
- b - The number of bytes needed to represent the right half of the message.

The Q value is extremely important for the Feistel network because it has different values for each round of the Feistel network, thus, increasing the security of the final computed value.

This Q value will then be concatenated with the P value, which was constructed during the setup phase. The concatenation of these two values will then be used as an input for the PRF function.

Once the PRF has been computed, it will then be added to the left part of the input message to form the next right side of the Feistel network for the next round. During this modular addition, the format and length of the message is also kept due to the RADIX value.

The exact same process is then repeated another 9 times. Once completed, the return value consists of the concatenation of the left and right side of the Feistel network computed in the last round.

```
    for i in range(0, ROUNDS):
        Q = generate_q(T, i, B, b)

        to_encrypt = P + Q

        AES = crypto.AESCipher(K)
        R = AES.encrypt(to_encrypt)[-16:]

        (...)

        c = (num_radix(A) + y) % (config.RADIX ** m)
        C = str(c)
        A = B
        B = C

    return str(int(A) * bmagnitude + int(B))
```

Listing 4.4: FPE Encryption Function - Feistel Network

Similarly, the decryption function is almost identical to the encryption function, with the difference of the round numbers are reversed. The A and B values that correspond to the input sides are swapped and the modular addition at the end of each round is replaced by a modular subtraction.

In addition to the encryption and decryption function, some tests were included in order to ensure that the computations made by these functions were indeed correct.

These tests were based on publicly available test vectors[8] where several aspects of the FFX mode of operation are tested.

Additional test were also generated in order to assure the implementations correctness. These tests range from data conversion tests to configuration file fault detection tests.

Just as we did for ORE the table below shows the encryption and decryption averages for a data set of 100000 iterations. In this table we compare the computation time of our FPE implementation against a python package that implements FPE. This package, called *pyffx*[9] is, currently a package available through the python foundation website and is the reference for FPE implemented in python. This package is also the top search result for python implementations of FPE, hence, a suitable candidate for comparison. The second comparison is made against a FPE[10] scheme made in Java by the github user, Robshep. This second comparison had to come from another programming language due to the lack of python implementations available at the time.

Table 4.2: FPE time comparison

	Key Size (bits)	**Iterations**	**Encryption Average (μ s)**	**Decryption Average (μ s)**
FPE	128	100000	108.615	108.67
FPE (pyffx)	128	100000	107.18	100.1
FPE (robshep)	128	100000	52.03	40.46

Both encryption and decryption operations are fairly similar in both python implementations. This is due to both implementations being fairly similar in the sense of following the specification for FPE implementation without much concern for optimisation. The pyffx implementation does, however, have a better usability for other users because it was designed to be a package to be implemented by other users, by offering an easy-to-use implementation.

[8] FFX AES Test Vector Data generated by Voltage Security, Available at: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/aes-ffx-vectors.txt

[9] pyffx package, available at: https://pypi.python.org/pypi/pyffx/0.2.0

[10] Robshep Java implementation, available at: https://github.com/robshep/JavaFPE

The Robshep implementation written in Java is a fairly simple implementation. It does not use any external libraries for it's computations, and although the average time take for encryption and decryption is half when compared to the python implementations, it is also made in Java, which may completely change the computation time.

Chapter 5

Proposed Solutions

Multiple approaches were taken in order to try to solve the aforementioned issue. The best possible solution for us involved an out of the box cryptographic scheme that would solve this or similar issues where FPE and ORE properties were necessary. Other approaches were also taken into account in order to find multiple solutions for this problem. Other techniques, such as, Tokenization were also studied and considered.

Since studying a more cryptographic solution is very different from studying the possible of using existing crypto tools, the following two sections contain detailed information about each method.

5.1 Cryptographic Approach

A cryptographic oriented approach requires a deep knowledge of cryptographic schemes, how they involved through time and what mistakes to avoid when constructing new schemes. Security definitions for a new scheme are as important as the scheme itself, hence, building a new cryptographic scheme is not an easy task nor a pre-determined set of steps. To build this new cryptographic scheme a thorough study of many other cryptographic schemes, their evolution and their security definitions was conducted. The first thing that seemed obvious for this new scheme was to establish it's leakage profile, or in other words, how much leakage was allowed in order for the scheme to have functionality and security. This leakage profile allows us to pre-define the scheme behaviour and it's security definitions.

For a cryptographic approach we were able to delimit a possible work plan for a future scheme that would allow equality information to be leaked from a FPE scheme which we called FPEQ.

Lets now loosely define what we require in general terms for a symmetric format preserving encryption scheme with plain text equality revealing function, or as we call it, FPEQ.

5.2 FPEQ

A format preserving with equality revealing encryption (FPEQ) scheme is a tuple of algorithms Π = (FPEQ.Setup; FPEQ.Encrypt; FPEQ.Decrypt; FPEQ.Compare) defined over some well-ordered domain D with the following properties:

- FPEQ.Setup($(1^\lambda) \rightarrow sk, qk$. On input security parameter , the setup algorithm outputs two secret keys, an encryption key sk and an equality revealing key qk.
- FPEQ.Encrypt($sk, t, m) \rightarrow ct$. On input the secret key sk and a message m $\in D$ and tweak t, the encryption algorithm outputs a ciphertext ct $\in D$
- FPEQ.Decrypt
- FPEQ.Compare($ct_1, ct_2, qk) \rightarrow b$ On input two ciphertexts ct1, ct2 and the compare key the compare algorithm FPEQ:Compare outputs a bit b $\in 0, 1$, TRUE or FALSE.

This scheme would have a leakage function which is different to the case of ORE which also reveals the ordering (the first bit or byte) of the plain text. This new leakage Function would not have to reveal any bit or block of bits that differ only the equality or inequality of the plain texts The best possible Leakage function would then simply be

$$\mathcal{L}_f((m_1, t_{1_1}) \dots (m_1, t_{1_{n1}}) \dots (m_k, t_{k_{nk}})) = \{m_i = m_j \forall ((m_i, t_i), (m_j, t_j))\}, \quad \textbf{(5.1)}$$

The alternative proposed here is to use the above FPEQ scheme in the transaction processing pipeline, hence no decryption and tokenizing would be necessary. The third party has access to a public or keyed function that allows him to determine if different credit card numbers originate from the same customer. The third party would only need to perform the equality revealing function on the credit card numbers received and group them.

The goal of a FPE scheme with a keyed plain text equality revealing function may be ambitious. Therefore as a first phase we can imagine an encryption scheme that is semantically secure and that is also decorated with a keyed plain text equality revealing function. The underlying encryption scheme could be symmetric or public/private key.

If we consider our proposal for the FPEQ scheme, one can obtain the same scheme using a more

general class cryptographic schemes, such as, FE or MIFE.

These schemes, although not currently practical, provide the tools needed for our proposed scheme.

With a FE scheme, we can perform computations over the ciphertexts, allowing an audit to occur over encrypted data, while maintaining the underlying data protected.

While feasible, functional encryption requires the usage of multiple keys (one for each ciphertext, or for each function). This might become a problem when comparing multiple ciphertexts, as in the case of credit card audits.

This inconvenience, lead us to take a look at multi-input functional encryption. MIFE allows for range queries to be computed over a set of ciphertexts. This is extremely useful for our proposed scheme, because, the number of keys needed to conduct an audit is greatly reduced. MIFE also allows the owner of the master key to derive functional keys that allow a third-party to perform range queries over the specific ciphertexts.

To better understand how our FPEQ scheme could be incorporated in a MIFE scheme, consider the definition of a Multi-Input Functional Encryption scheme as defined in the first chapter of this dissertation.

For our FPEQ scheme the function that is introduced during the Keygen algorithm would compute the equality of the underlying plaintext data. This would allow anyone with the proper keys to determine if two or more ciphertext have the same underlying plaintext.

This is, of course, assuming that MIFE is implementable, which is not the case, currently. Hence, for our FPEQ scheme, a weaker scheme is needed, where one might reduce the security definitions of the scheme in order to achieve its goal, which is, to allow a property of the underlying plaintexts to be publicly computable, even if protected by a format preserving encryption scheme.

If such a scheme could be implemented in an efficient manner, than, this scheme could be used to solve multiple problems that are currently present in today's cloud platforms, ultimately allowing cloud service providers to process encrypted data, thus, exponentially increasing the security of the data that they need to process.

5.3 FPE

Another possible approach to solve this problem is to work with the existing cryptographic approaches and possibly modify them in order to solve the current problem. When working with existing cryptographic schemes, the main concern always involves the security of the already defined scheme. The security definitions of one scheme are usually defined in terms of a particular model and for a particular algorithm and protocol. The security definitions only assure that the scheme is secure for the way that it was designed to. If any alteration is made to the scheme structure, then, the security definitions need to be completely re-evaluated.

For this dissertation we decided to study the possibility of changing the FPE scheme in order to meet our needs. We decided to experiment with how FPE works internally and alter the Feistel network that is used to perform an FPE encryption without concerning ourselves about the security definitions, at least for now.

The main idea behind changing the behaviour of the Feistel network is that, if we can get a property to be maintained, in other words an invariant, throughout a sequence of Feistel network rounds, then that property can also be used as well as the final cipher text having the desired (format preserving) format. Of course the property in question is *Comparison* or even just *Equality*. By allowing some property of the plaintext to be maintained throughout the Feistel network we might also be compromising the overall security of the scheme by introducing additional leakage to the scheme. There were several approaches taken. All of these are separated into the following sections.

5.3.1 Approach 1

The first approach to change the Feistel network structure involved adding an ORE encryption to the output of the round function is combined with the first half of the input. Figure 5.1 shows the Feistel network structure. This approach tries to apply ORE throughout the rounds of the Feistel network so that, at the end, the ORE compare function would be usable (with some modifications to account for the repeated usage of ORE). If we apply directly the ORE algorithm then this approach, however, has the flaw of making FPE losing its ability to preserve the format of the input. Since the first half of the input has go through ORE encryption, it would lose its original format. Losing the format would mean that, by the second round of the Feistel

network, we had to deal with two different formats. By the end of the second round, all of the original input format would be lost.

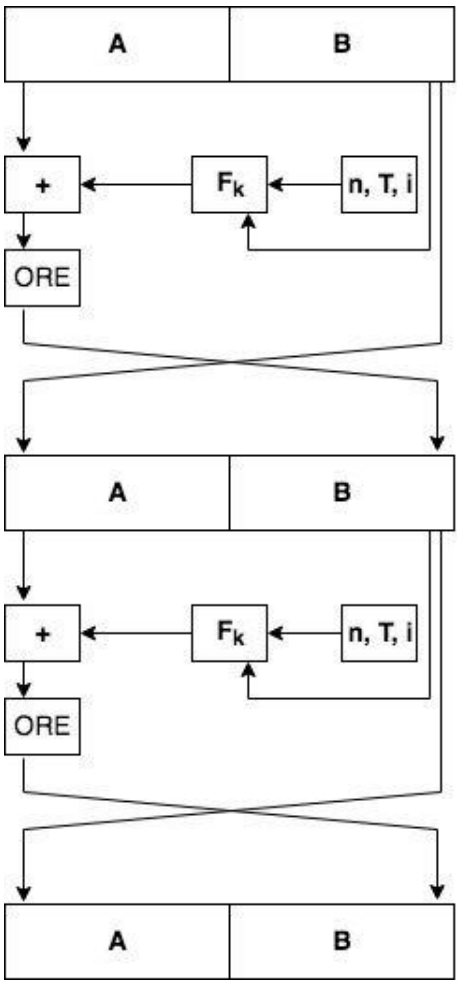

Figure 5.1: ORE Encryption after Sum operation

In an attempt to solve the format issue, one can think of trying to change the ORE encryption position in the Feistel round. Figure 5.2 shows an ORE encryption right before the sum operation that joins the result of the round function and the first half of the input. This encryption suffers from the same problem of the last approach. By encrypting the result of the round function, we are essentially providing the wrong format to the sum operation, thus, losing format anyway.

5.3.2 Approach 2

Other approaches were also considered where we could replace the round function of the Feistel network by an ORE encryption as shown in Figure 5.3. This approach brings several problems. First, we still need to consider decryption. ORE by itself does not support decryption. Decryption can be achieved by a `brute-force` manner, but, it has no default algorithm to decrypt an ORE

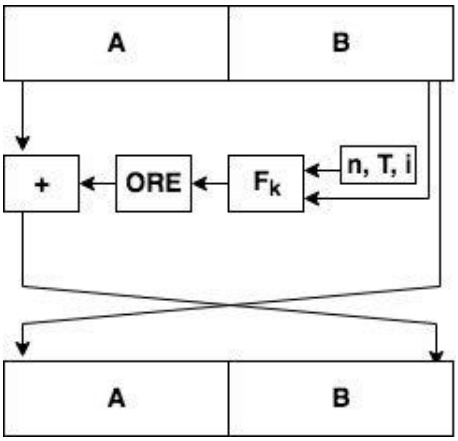

Figure 5.2: ORE Encryption before Sum operation

ciphertext. This means that, if we use ORE, the Feistel network is not longer invertible, thus, we also lose the possibility to have FPE decryption. The main problem of this approach is, however, the ability to perform a ORE comparison. Since FPE, depending on the operation mode, requires at least 8 rounds of Feistel network to achieve the desired security, we still need to perform comparisons after 8 rounds of Feistel network.

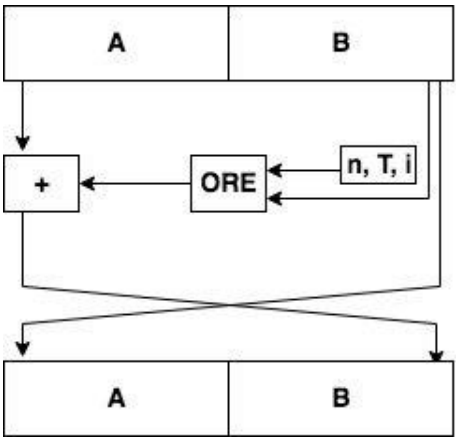

Figure 5.3: FPE Balanced Feistel Network

Other approaches were also considered, as seen in Figure 5.4 and in Figure 5.5. Both these approaches have the same problem of losing the ability to preserve the format of the FPE scheme, thus, losing the functionality that is important for our specific problem.

All of these approaches were merely an attempt to try to combine both schemes together. The main problem of having a Feistel network to allow for a comparison function is that the ability to perform comparisons is lost whenever we have a new Feistel round.

In the event that we could combine the two schemes (FPE and ORE), we would be able to solve this particular scenario of credit card transactions. This would mean, however, that other cloud

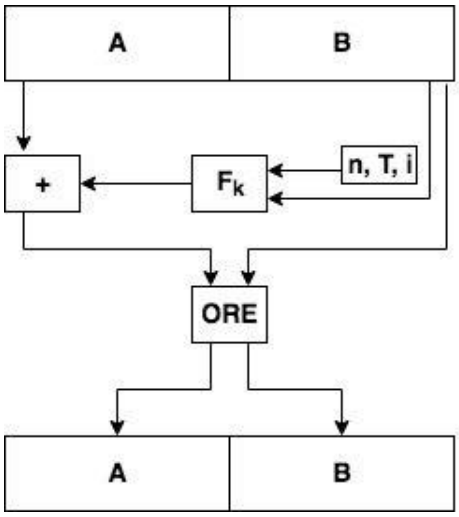

Figure 5.4: FPE single ORE instance.

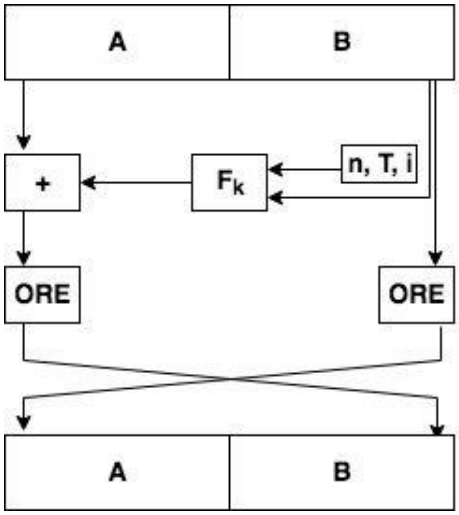

Figure 5.5: FPE with two ORE instances.

related security problems might not be solvable with this approach, only the ones that needed to preserve the format of the original data, which may not be suitable for many scenarios.

5.4 Existing tools

As previously mentioned, another possible approach to solve this problem is to use already existing cryptographic tools that can, when used together, augment the utility of already in use cryptography schemes, e.g., FPE.

With this approach we need to be certain that, when used together, the overall data security is not put at risk and that an attacker cannot obtain further relevant information to possibly

recover the original data.

If we consider the original problem of delegating confidential data to a third party entity, where this data needs to be processed, we can try to use existing tools to allow secure leakage of the plaintext.

The first considered approach is allowing the third party to have access to the decryption key and revert the credit card numbers to the originals whenever needed. While this solution is simple it would also defeats the purpose of using FPE altogether and compromise the confidentiality of the data.

Since we want to protect the credit card numbers, we need to be able to give the third party some information that can be linked with the original credit card number, but not the credit card number itself.

The second approach that was considered is an approach that uses `tokenization`. The process of tokenization is straightforward. A string of random bits is generated for every new credit card number. This random string is then associated with each credit card and stored in a safe location. Whenever a trusted party needs access to this token or it needs to check if a token is connected to a credit card number, the trusted party can query this database in order assert the link between the two. By using a token to identify credit card numbers, we allow ourselves to still use FPE, because the tokens are linked with the original credit card number and not the one generated by the FPE encryption algorithm. This means that, even if we encrypt the same credit card number multiple times with different tweaks and different keys, as long as we provide the same token that is linked with the original credit card a link can be established with the original credit card by anyone with access to the secure token database. Although achieving its goal, tokenization can bring extra work load to the credit card fraud analysis. Having to link every single credit card with its token while conducting a fraud analysis can put extra logic and work load on the analysis itself.

5.5 ORE as an existing tool

As a last already existing tool that we can consider for the aforementioned problem, is the use of ORE. Instead of combining FPE and ORE to make a single encryption scheme, we can use them

both, but at different stages to achieve the needed protection for our data.

Since all the transactions to be analysed are held by the bank, we can also use different cryptographic techniques to further reduce the amount of sensitive information that is sent to the third party. For instance, since we hold all of the original transaction information, we can now apply an ORE scheme to the credit card numbers. This way, the third party only has information about which credit cards are the same but not to the original numbers.

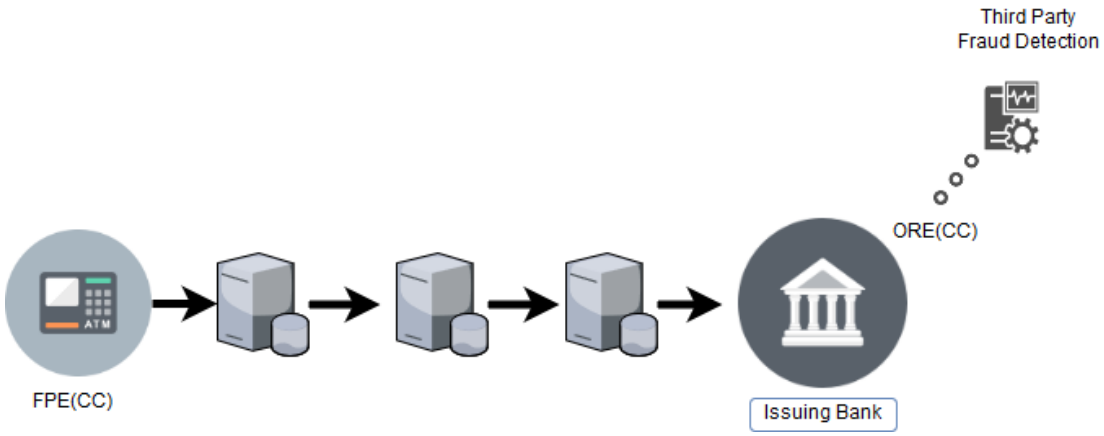

Figure 5.6: ORE before third party analysis.

Figure 5.6 shows where in the transaction process, the ORE scheme would be applied.

Using this approach enables us to solve the original issue for this particular case only. Combining these two schemes will not be possible for many other scenarios of data delegation.

5.6 Conclusion

This section presented the main approaches when trying to solve the main issue of this dissertation. All approaches propose some ideas of what can be done to solve the problem at hand. The cryptographic approach gives an idea of what a possible scheme might look like which has the desired properties. The FPE approach attempts to combine two existing schemes that have the ideal properties that would be able to solve our problem and combine these two schemes into one. The last approach attempts to solve the problem by using existing cryptographic tools. There are also other approaches that were not included in this chapter. Schemes such as SE[CK10], which allows data to be stored in search trees to later allow searches over encrypted data, may be a strong candidate in solving our current issue.

Chapter 6

Conclusion and Future Work

In this dissertation we analysed how a problem of data delegation to a cloud infrastructure could be resolved. We presented a brief review of the principle cryptographic schemes that were used in this dissertation and discussed and compared details of their implementations and performances. By analysing the specific problem related to credit card transactions we identified three possible approaches to attempt to solve this problem. One approach tries to define a new model for a cryptographic scheme to handle this specific situation. The other approach attempts to solve the existing problem by modifying an existing cryptographic scheme that is being used in this situation, namely, FPE. This approach tries to combine FPE and ORE in order to try to utilize the comparison function of ORE together with the format preserving properties of FPE. The third and last approach attempts to solve the problem through existing cryptographic tools. This approach, although simpler, may only be useful in this specific situation and not on other similar cases where data delegation to the cloud might become a problem.

6.1 Future Work

This dissertation leaves a lot of possible future work. Each possible approach described in the previous section leaves room for improvement. The first approach where we try to define a new model for a new cryptographic scheme could also use a strong security definition study to determine whether this new scheme meets the security requirements for such a scheme. This study would have to take the amount of leakage, that is inherent from this schemes, into account and determine that, given its leakage profile, if such a scheme even if built, would be of any practical use. This security study would require a thorough analysis of the security definitions of many other schemes and apply that knowledge to this scheme. That security analysis was never taken into account for our scheme. For the second approach that tries to combine both FPE and ORE, an extra study would also be needed on how to properly join the two schemes in order to make the ORE properties to be maintained throughout the Feistel network. If such a requirement is possible in this scenario, then, a security analysis of the resulting scheme would

also be needed since FPE security definitions could be completely broken with the introduction of additional leakage that is inherent from the ORE scheme.

Lastly, the final approach were we combine both FPE and ORE at different stages of the credit card network, would also need testing further testing and an security analysis of the resulting infrastructure. For this approach, we would need to make sure that at every stage of the payment process, data would be safe from attackers and eavesdroppers. Such an analysis was not conducted.

Overall, all aspects of this dissertation could use an extra layer of study and attention to solidify its results, including the implementations that were mentioned in section 4, which would need proper testing and security validation.

6.2 Conclusion

Currently, cloud service providers face a major challenge when it comes to protecting users private data. This challenges will not become any easier even if new schemes designed for the cloud world become practical. It usually takes years of adaptation and security analysis for a new cryptographic scheme to be accepted by companies and users.

The research in this area has, however, grown exponentially in the last few years and there are signs that things might be turning into the right direction by the most recent conferences on cryptography. Many schemes are taking into account the fact that searches might be needed over the encrypted data.[CAGR17][Rus18] This ability that these new schemes have to allow for searches to occur, might help cloud service providers to increase the security level of customers data. Some of these new schemes, namely SE, do require data to be stored and organised in specific ways to allow for range queries. This requirement might make adaptation to be difficult by organisations that have already established their data structures.

Although there is not a current one-shot solution for every cloud problem, there are a few alternatives that, depending on the situation, might completely solve some of the problems. These solutions though, need to be analysed for each individual scenario. This situation makes adoption of current tools to be a problematic for many companies and service providers.

The future of cloud service security seems to be improving, as more and more end-users become

aware of the risks involved by delegating personal information to third parties and providers.

Bibliography

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 563–574, New York, NY, USA, 2004. ACM. Available from: http://doi.acm.org/10.1145/1007568.1007632. 21

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 152–181. Springer, 2017. 27

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer, 2009. 18, 21

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*, pages 578–595. Springer, 2011. 21

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001. 26

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001. 27

[BGJS15] Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input functional encryption for unbounded arity functions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 27–51. Springer, 2015.

[BHT16] Mihir Bellare, Viet Tung Hoang, and Stefano Tessaro. Message-recovery attacks on feistel-based format preserving encryption. In *Proceedings of the 2016 ACM SIGSAC*

Conference on Computer and Communications Security, pages 444–455. ACM, 2016.

[BLR+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 563–594. Springer, 2015. 23, 28

[BPS10] Eric Brier, Thomas Peyrin, and Jacques Stern. Bps: a format-preserving encryption proposal. *Submission to NIST, available from their website*, 2010.

[BRRS09] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In *International Workshop on Selected Areas in Cryptography*, pages 295–312. Springer, 2009.

[BRS10a] M Bellare, P Rogaway, and T Spies. Addendum to "the ffx mode of operation for format-preserving encryption": A parameter collection for enciphering strings of arbitrary radix and length, draft 1.0, natl. inst. stand. technol.[web page], 2010. 19

[BRS10b] Mihir Bellare, Phillip Rogaway, and Terence Spies. The ffx mode of operation for format-preserving encryption. *NIST submission*, 20, 2010. 19

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011. 26

[CAGR17] Shujie Cui, Muhammad Rizwan Asghar, Steven D Galbraith, and Giovanni Russello. Secure and practical searchable encryption: A position paper. In *Australasian Conference on Information Security and Privacy*, pages 266–281. Springer, 2017. 52

[CD14] Sanjit Chatterjee and M Prem Laxman Das. Property preserving symmetric encryption revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 658–682. Springer, 2014.

[Cen18] InfoWatch Analytical Center. Global data leakage report, 2017, 2018. v, 1

[CK10] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer, 2010. 49

[CLOZ16] David Cash, Feng-Hao Liu, Adam O'Neill, and Cong Zhang. Reducing the leakage in practical order-revealing encryption. Technical report, Cryptology ePrint Archive, Report 2016/661, 2016.

[CLWW16] Nathan Chenette, Kevin Lewi, Stephen A Weis, and David J Wu. Practical order-revealing encryption with limited leakage. In *International Conference on Fast Software Encryption*, pages 474–493. Springer, 2016. 12, 23, 31, 34, 35

[DDC16] F Betül Durak, Thomas M DuBuisson, and David Cash. What else is revealed by order-revealing encryption? In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1155–1166. ACM, 2016. 24, 35

[DF14] Sashank Dara and Scott Fluhrer. Fnr: Arbitrary length small domain block cipher proposal. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 146–154. Springer, 2014.

[GGG+14] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 578–602. Springer, 2014. 28

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. 27

[GKP+13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM, 2013.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984. 16

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194. ACM, 1987.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006. 26

[GSB+17] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 655–672. IEEE, 2017. 22

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, volume 20, page 12, 2012.

[JP16a] Marc Joye and Alain Passelègue. Function-revealing encryption. 2016.

[JP16b] Marc Joye and Alain Passelègue. Practical trade-offs for multi-input functional encryption. Technical report, Cryptology ePrint Archive, Report 2016/622, 2016.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. 17

[Lin16] Yehuda Lindell. How to simulate it-a tutorial on the simulation proof technique. *IACR Cryptology ePrint Archive*, 2016:46, 2016.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. 21

[LW16] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1167–1178. ACM, 2016. 12, 25, 31, 35

[M+09] Ulf T Mattsson et al. Format controlling encryption using datatype preserving encryption. *IACR Cryptology ePrint Archive*, 2009:257, 2009.

[NKW15] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC*

Conference on Computer and Communications Security, pages 644–655. ACM, 2015.

[PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–391. Springer, 2012.

[Rog10] Phillip Rogaway. A synopsis of format-preserving encryption. In *UNPUBLISHED MANUSCRIPT*. Citeseer, 2010.

[Rou] Margaret Rouse. Pci dss compliance (payment card industry data security standard compliance). Available from: `http://searchcompliance.techtarget.com/definition/PCI-compliance` [cited 07.01.2018]. 6

[Rus18] Giovanni Russello. Obliviousdb: Practical and efficient searchable encryption with controllable leakage. In *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers*, volume 10723, page 189. Springer, 2018. 52

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell Labs Technical Journal*, 28(4):656–715, 1949. 16