CRYPTANALYTIC STUDY OF PROPERTY-PRESERVING ENCRYPTION

BY FATMA BETÜL DURAK

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

David Cash

and approved by

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE DISSERTATION

Cryptanalytic Study of Property-Preserving Encryption

by Fatma Betül Durak

Dissertation Director: David Cash

Property-preserving encryption (PPE) provides accessible methods to encrypt the databases in an efficient way by preserving specific fuctionalities of databases. PPE as an impactful research area accommodates deterministic encryption, order-revealing encryption (ORE) and format-preserving encryption (FPE). Although the simplicity and compatibility of PPE makes it demanding in real world systems, the security of these primitives has been unclear since their invention. In this work, we study and explore the range of threats in ORE and FPE established with a Feistel network as the elementary unit. We take system- and theory-centric approach to address the issues in ORE, FPE, and Feistel networks which form building blocks of some particular FPE schemes (and potentially many other constructions) of our interest.

We start with ORE where the ordering relation of messages is preserved (and revealed) on corresponding after encryption. In our work, we demonstrate two issues in ORE: firstly, we show that when multiple columns of correlated data are encrypted with ORE, attacks can use the encrypted columns together to reveal more information than prior attacks could extract from the columns individually. Secondly, we apply known attacks, and develop new attacks, to show that the *leakage* of concrete ORE schemes on non-uniform data leads to more accurate plaintext recovery than is suggested by the security theorems which only deal with uniform inputs.

We then consider the recently published FPE standards: FF1 and FF3, by The National Institute of Standards and Technology (NIST) . Particularly, FF1 and FF3 are both tweakable block cipher based on Feistel network (FN). Feistel network in FPE is an iterative cipher that are composed of multiple rounds to encrypt the plaintext by breaking it into left and right branches. Each iteration, called round, consists of a secret round function (i.e. a keyed pseudo-random function) and modular addition. In each round, one half of the input is evaluated with a round function and then applied by modular addition with the other half (which is preserved for the next round). The outputs are permuted and then input to the next round. The security of FN hinges on the security of its round functions and the number of iterations (more iteratitions assure stronger security at the cost of efficiency). In the NIST standards, FF1 and FF3 are AES-based modes of operation composed with 10-round and 8-round Feistel network respectively. FPE is specifically designed for small domain sizes when $\mathsf{N}^2 \geqslant 10$ where N is the branch size of Feistel network.

In this work, we investigate the security of Feistel network thoroughly since its security has not yet been understood well with already existing theoretical analyses to integrate with small domains. We develop a set of new generic attacks for Feistel network that have direct effect on FF1 and FF3. First, we give a generic known-plaintext attack to a 4-round Feistel network that reconstructs the entire tables for all round functions. It requires a data complexity of $\mathsf{N}^{\frac{3}{2}}\left(\frac{\mathsf{N}}{2}\right)^{\frac{1}{6}}$ known plaintexts and time complexity of $\mathsf{O}(\mathsf{N}^3)$. Our 4-round attack can be easily extended to five or more rounds with complexity $\mathsf{N}^{(r-5)\mathsf{N}+o(\mathsf{N})}$. We continue further to explore more generic attacks on arbitrary number of rounds that covers 8-round and 10-round Feistel network. In that direction, our ideas are developed around exhaustive search on the round functions by using an early abort that eliminates as many candidates as possible during the reconstruction phase of round function. We conclude that neither FF1 when $\mathsf{N} \leqslant 11$ nor FF3 when $\mathsf{N} \leqslant 14$ offers 128-bit security.

Finally, we give a new total break attack to the FF3 scheme by exploiting the bad domain separation in its design. It is practical when the domain size is small. Our attack is not generic and enables slide attack due to its weak design. Luckily, we can provide an easy and intuitive fix to prevent the FF3 scheme from our attack. A follow-up work after the slide attack to the FF3, we develop a new generic attacks on Feistel networks (without restricting ourselves to any of its system design). Unfortunately, our research indicates that the security assurance of FF3 is still not viable even with our suggested patch when the domain size is tiny.

Acknowledgements

Starting a Ph.D. in the USA was a turning point in my life. Succeeding it was the most stimulating, mentally reinforcing and satisfying experience overall. It was not easy. I had to pass through many challenging paths. Nevertheless, I tried to produce great work, collaborated with many people, have met many smart researchers, and made a lot of friends. Sometimes, I was alone in this journey, sometimes not; there were times I was fully satisfied, while at the same time questioning where I am and what I am doing. At the end, given the fact that I am writing these lines, I feel that I have fulfilled one of the big dreams of my life. So far!

I would like to thank my adviser, Prof. David Cash, for being very helpful, kind, and present for me to succeed my thesis. I would like to heartily thank Prof. Serge Vaudenay to have opened his opportunities and his great projects to me. Without our coffee times and lengthy discussions, I wouldn't have been able to keep my excitement in my work, I wouldn't have been able to pursue my dream, and I wouldn't have been able to feel confident enough in my job. I also would like to thank my committee members Rebecca Wright, Shubhangi Saraf, Thomas Ristenpart and all my collaborators with whom I have worked. It is my pleasure to thank Berra Beyoglu and Hülya Bicer with whom I have shared a significant amount of my time during last few years together.

Finally, I thank my family for encouraging me in every decision of my life with full support. Biggest thank you to my mother, Emine Durak, for her genuine and loving care for me; my father, Muharrem Durak, for his support to keep my dreams; my brother, Haci Veli Durak, for his belief in me and making my life more fun. More thanks to my beloved cousins Hamide Arslan, Adnan Arslan, and Mustafa Güven; who helped me to become who I am, who did more than they should have for me to lead me where I wanted to be.

Dedication

I proudly dedicate my dissertation to my grandmother, Hafize Cangüven, for being the greatest person in my entire life, for her unconditional love and genuine care, and for being a unique person in every respect. Grandmother, you are my lifetime idol with all your humanitarian values that you instilled in me. You will be sorely missed. Rest in peace.

Table of Contents

Chapter 1

Introduction

An increasing amount of sensitive and private data are being collected and stored in databases by enterprises and government agencies. These databases are either kept local or outsourced to third parties in order to reduce the burden of data management and maintenance cost. Despite its prevalence and merits of such systems, unprotected databases result in severe security and privacy issues. Indeed, data breaches such as hacking of Cellebrite, the world famous iPhone and device cracker, caused the break-in of 900 GB of sensitive corporate data to be compromised. This includes the list of companies which bought phone cracking technology, various types of technical data and databases. This occurrence and extensive compromises of systems made security a primary concern of the design processes of every commodity database management system (DBMS). The possible hedge against compromises with the DBMS has been suggested with *encrypted databases* so that the data is protected from the attacker.

Standard encryption with a key held by the client could only offer protection from the damages caused by the compromised server to some extent. However, encrypting the data to secure it causes some major drawbacks to the existing services. In a single client-server settings, for instance, configuring and deploying standard encryption prevents the client from performing basic operations such as keyword search or range queries over the encrypted data (the client must either provide the decryption key or dump the entire database). This mitigates the most significant advantage of the systems and makes the access cumbersome for the data owner. Moreover, encrypting legacy systems in order to secure it is not possible without making considerable costly modifications to the existing databases. A database created decades ago could be processing a 9-digit of social security number (SSN). This particular system can only accommodate 9-digits of SSN and simply cannot handle larger data which would be generated with standard encryption, i.e. 128-bit AES algorithm. Moreover, the applications built on top of these databases expect the data in the form of the messages (i.e. 9-digits of decimal strings from the set of valid SSNs), whereas the encrypted data could be any binary string with special characters or letters.

A specific and efficient class of encryption methods has been developed to deal with this riddle. These methods fall into the category of so-called *property-preserving encryption* (PPE) [PRZB11, GHH$^+$14] (such as *deterministic encryption, order-revealing encryption, format-preserving encryption*) and have been increasingly marketed and deployed in many systems. Due to its compatibility with the real world database systems, PPE-based encrypted databases look like a seeming solution to the problems created by standard encryption on databases. The main intuition behind PPE is to allow a server to execute its part of functionality on encrypted data and preserve the certain properties of plain data so that the data can be encrypted preserving those properties. PPE allows for the least amount of changes to the legacy systems, as it allows applications to operate on encrypted data in the same way that they would on plain data. It lets the systems take advantage of all the optimization techniques of standard database management systems that makes the query processing quite fast. However, the design of encryption methods in PPE exposes some information called *leakage* which poses a lot of risks. In fact, those risks could be devastating even for the designs that are admitting the amount of leakage (the leakage supported with strong claims by means of “provable security”).

In the present work, we study the security of PPE-based methods and their building blocks.

1.1 Terminology

In this section, we overview the basic concepts that we will need throughout this work. We then concretely explain our contributions.

Relational databases. Relational database is a set of data items organized in *tables* where each row corresponds to a unique instance of an individual (e.g. an employee or a customer of a corporation), and each column corresponds to a category associated with a row (name, date of birth, address, etc). In general, we would be interested in encrypting a given column of data using PPE techniques so that certain properties of the columns are preserved.

Deterministic encryption. It was first formally defined by Bellare, Boldyreva, and O'Neill in [BBO07] along with a few instantiations of the primitive and its security definitions/analysis. Briefly, a symmetric deterministic encryption (DTE) scheme is a 3-tuple $(\mathsf{Gen}(\cdot), \mathsf{Enc}(\cdot,\cdot), \mathsf{Dec}(\cdot,\cdot))$ where $\mathsf{Enc}(\cdot,\cdot)$ takes two inputs as a secret key K generated by $\mathsf{Gen}(\cdot)$ algorithm and a plaintext to map the plaintext into a *unique* ciphertext under the secret key K. Dec is the inverse of Enc function. DTE is not a randomized encryption scheme, hence the equality of two messages encrypted under the same key will be preserved and also revealed to the adversary.

Searchable encryption. It allows a data owner to encrypt his/her *documents* along with a keyword index to outsource. The data owner can later interact with an untrusted server to securely search on the encrypted documents. Searchable encryption (SE) has been a popular field of research initiated with Song, Perrig, and Wagner [SWP00a]. Briefly, a static SE scheme consists of a tuple $(\mathsf{SetUp}(\cdot), \mathsf{Token}(\cdot,\cdot), \mathsf{Search}(\cdot,\cdot,\cdot), \mathsf{Token}(\cdot), \mathsf{Dec}(\cdot,\cdot))$ which defines an interactive protocol between a client and a server for keyword search operation. In this tuple, $\mathsf{SetUp}(\cdot)$ takes a document collection as input and outputs a private state for the client and an encrypted index. $\mathsf{Token}(\cdot,\cdot)$ is an algorithm run by the client. It inputs a secret key and a keyword, outputs an encoded keyword for Search algorithm. $\mathsf{Search}(\cdot,\cdot,\cdot)$ is an interactive protocol which inputs an encrypted index, an encoded keyword token, a private client state and outputs a set of identifiers that correspond to the documents containing the queried keyword (returned by the server). $\mathsf{Dec}(\cdot,\cdot)$ returns the plaintext of document identifiers and updates the client state. SE can be defined in dynamic setting by allowing Update protocol that enables addition or deletion of non-existing documents.

The security of SE primitive is defined with the concept of information *leakage* to the service provider that holds the encrypted documents in order to perform search (under the honest-but-curious adversarial model). For instance, all the SE constructions leak the hashes of keywords that reveals the keyword search repetition (called search pattern) and identifiers of the documents returned by each search (called access pattern). There exist many construction of SE in both static and dynamic setting that target different security levels [NPG14, KPR12, CJJ$^+$14, SPS14]. There have been many studies of the information leakage through access and search patterns [CGPR15, PW16, ZKP16]. Statistical inference attack even reveal information about encrypted queries and files accessed by the query [IKK12].

Oblivious random access memory (ORAM) introduced by Goldreich and Ostrovsky [GO96, SSS11, SvDS$^+$13] can hide access pattern, therefore can prevent most of the information leakage in SE. However, all known ORAM constructions in [SCSL11] are more complicated and less efficient than SE. In particular, they require many round trips between the client and the server with significantly larger memory complexity spent on the server. Therefore, ORAM techniques for SE suffer from inefficiency and are in practice not preferred.

Order-preserving encryption. Order-preserving encryption (OPE) enables an untrusted database service provider to process range queries on a column in relational databases without completely revealing the numerical values in that column. Briefly, a symmetric OPE scheme has four algorithms $(\mathsf{Gen}(\cdot), \mathsf{Enc}(\cdot,\cdot), \mathsf{Compare}(\cdot,\cdot), \mathsf{Dec}(\cdot,\cdot))$ with the following property: if $m_1 > m_2$, then $\mathsf{Compare}(\mathsf{Enc}(K, m_1), \mathsf{Enc}(K, m_2)) = 1$ under the same secret key K indicating that the first message is greater than the second message (similarly, if $m_1 < m_2$, then $\mathsf{Compare}(\mathsf{Enc}(K, m_1), \mathsf{Enc}(K, m_2)) = -1$ under the same secret key K). Simply put, the comparison relation on plaintexts is preserved across the corresponding ciphertexts.

OPE in practice is very intuitive to apply in relational databases. In a single client-server model with a proxy which holds the client's secret key, the client (with its proxy) encrypts the individual columns of its interest to query a range (under different keys for each column) and outsources it to the server. When the client processes a range query with an interval $[x, y]$, the proxy encrypts x and y with a key corresponding to the column and sends encrpyted end points to the server. Then, the server looks for the values in between $\mathsf{Enc}(\mathsf{K}, x)$ and $\mathsf{Enc}(\mathsf{K}, y)$ to return it to the proxy. The proxy then decrypts them and sends to the client as a result. Hence, small modifications to the existing services would be enough to upgrade the security of the databases (the server requires inconsequential changes). Among the practical OPE schemes, some are blockcipher based whereas some are interactive protocols [PLZ13, KS14, Ker15], maintaining the security definitions.

Order-revealing encryption which is a generalized form of OPE with some distinctions. In particular, the $\mathsf{Compare}$ algorithm in OPE is a standard comparison operation on numerical values, whereas $\mathsf{Compare}$ algorithm in ORE works with any form of ciphertexts along with numerical values. The distinctions are not important for this work, hence we refer to only ORE which subsumes OPE for the rest of this work.

Several practical ORE constructions are available [BCLO09, CLWW16]. By the definition of the ORE primitive, it has a weaker security than standard encryption even when it is used in a randomized manner. In particular, an adversary that has been given chosen-plaintext power breaks any ORE construction with enough number of queries (due to the binary search). ORE schemes have been proven secure under some cryptographic assumptions. The security of ORE was first studied by Naveed et al. as in [NKW15]. In their work, Naveed et al. showed various practical attacks on *ideal* ORE construction where the information leaked by the ORE scheme was "only" the ordering relation of messages (there are certain practical OPE and ORE schemes which leaks additional information such as plaintext bits or statistical information). Their attacks work under the assumption that the attacker obtains the encryption of almost the entire message domain or the adversary has given some auxiliary information such as statistical information to enable frequency analysis. Their attack has been applied on medical dataset collected from hospitals to show its accuracy and impact. An independent and concurrent work by Grubbs et al. in [GSB+17] defines improved attacks against both ideal and non-ideal ORE instantiations. Their work introduces a new bipartite graph structure, and uses dynamic programming algorithms in their framework to make attacks effective beyond the dense domains attacked by Naveed et al. Along with our work, these constitute the first empirical analysis of non-ideal ORE.

Format-preserving encryption. It provides a way to encrypt a message of a specific "format" into the same "format", meaning that if a message is 16-digit credit card number (CCN) with a valid checksum, the ciphertext is also a valid 16-digit of CCN. Similarly, if messages are formed as a 6-digit of passcodes, the encryption algorithm maps them to the ciphertexts of the same domain. Brightwell and Smith [BS97] introduced the first known format-preserving encryption schemes.

An FPE is a deterministic symmetric key encryption with a tuple defined as ($\mathsf{Gen}(\cdot)$, $\mathsf{Enc}(\cdot,\cdot,\cdot), \mathsf{Dec}(\cdot,\cdot)$). $\mathsf{Enc}(\cdot,\cdot,\cdot)$ function takes a key K, a format F and a message X from a domain D to return a ciphertext Y which is an element of D. There is a corresponding decryption function $\mathsf{Dec}(\cdot,\cdot)$ such that Dec is an inverse of Enc, i.e. they are permutations over domain D.

FPE is specifically designed for short messages (i.e. smaller than 128-bit message length) which are not necessarily in binary form. The National Institute of Standards (NIST) published a specification for practical FPE constructions accepting three important schemes [NIS16]. NIST motivated its standard with the following words "FPE has emerged as a useful cryptographic tool, whose applications include financial-information security, data sanitization[1], and the transparent encryption of fields in legacy databases.".

The FPE schemes in NIST standards assert security claims which are supported by crytanalytic techniques, and there has been no proofs of the security of the constructions. In this work, we focus on two specific FPE standards called FF1 and FF3 and analyze their security with new and sophisticated techniques which are practical when the domain size is sufficiently small. There already existed a recent attack by Bellare et al. in [BHT16] which studied the FF1 and FF3 by exploiting the weakness of underlying building block which is a Feistel network. Another weakness explored by Patarin and enchanced by Bellare et al. is a bias that is introduced due to the non-invertable round functions used in the Feistel network. The query complexity of the attack exceeds the domain size with many tweaks. It is interesting when the adversary

[1]The sanitization of personally identifiable information in a database -whether by FPE or other methods- does not necessarily provide strong assurance that individuals cannot be re-identified

can make known-plaintext queries under multiple tweaks.

Given all these PPE-based primitives, one fundamental research direction is understanding the security of these primitives in practice by cryptanalysis of these methods. Even though all of the OPE and SE constructions are provably secure under widely believed cryptographic assumptions, there is a gap between what theory suggests and how they achieve security in practice. This gap is mostly due to the unrealistic assumptions, such as uniform data distribution rather than any cryptographic assumption itself, and due to the fact there is not enough theory studied for the underlying building blocks (specifically in the NIST standards). Throughout this work, we will study the primitives used in practice, their security claims, and the meaning of security in practice under given various novel cryptanalysis techniques.

In our work, we will explicitly focus on Feistel networks which have been used in the FF1 and FF3 FPE standards. Feistel Network is an iterative cipher that generates a secure pseudo-random permutation defined on larger domain from secure pseudo-random function or pseudo-random permutation over a smaller domain. Each iteration called a round consists of an uninvertable round function which makes the Feistel network interesting. We are convinced that there has been no thorough analysis of this primitive and its legitimate security in practice has been overlooked.

1.2 Our Contribution

This work categorically focuses on order-revealing encryption, format-preserving encryption and Feistel networks. In Chapter 2, we present a collection of observations, experiments, and attacks dealing with the use of ORE on data that will be encountered in practice. We point out some overlooked properties of leakage profiles, perform experiments to measure the security of ORE against known attacks but on non-uniform data, and give new attacks showing that ORE is less secure than theory indicates.

We first identify new properties of datasets that can cause information to be leaked, even when an *ideal* ORE is used. Then, we turn to analyzing the leakage of concrete (non-ideal) instantiations of ORE.

In a database table with multiple columns, PPE-based systems use ORE to encrypt each column independently under different keys. Using a different key for each column prevents direct comparison of ciphertexts across columns, thus leaking less information. The first part of the work investigates the security of ideal ORE on multiple columns. We observe that columns of data in a table are usually *correlated* because a row of a table usually corresponds to an individual record. We study the effect of correlation using simple multi-column versions of attacks by [NKW15] on ideal ORE. We analyze our attacks using visualizations and measurements of the attack accuracy on geographic datasets (described with more details in Chapter 2) where latitude and longitude were correlated.

The attacks in the first part of our work apply to any ORE that reveals order, but most ORE schemes reveal more. In the second part of our work, we try to understand the leakage allowed by these concrete ORE instantiations [BCLO09,BCO11,CLWW16].

In practice, the data in a column are generally not uniformly random and independent, therefore the one-wayness proofs by [BCO11, CLWW16] do not apply. That means the information leaked by these constructions may be greater than the theorems suggest. We show the effect of the leakage caused by non-uniformity of data (technically violating the assumptions of the theorems). We start by measuring the amount of information that can be directly inferred from leakage profiles of existing ORE constructions on different types of data, including synthetic and real location datasets and timestamps for mobile phone usage. This measurement consists of running simple attacks and in some cases composing them, to produce guesses of plaintexts which are then evaluated for accuracy depending on the context.

We found that the security gap guaranteed by one-wayness theorems were not always applicable on real data. Concretely, we consider the construction of Chenette et al. [CLWW16], which was proved to be one-way in a quantitatively stronger sense than the prior work of Boldyreva et al. [BCLO09, BCO11]. But by simulating the Chenette et al. and Boldyreva et al. leakage profiles on a real dataset of 2000 latitude-longitude pairs, we found that essentially the same number of plaintext bits (about 50%) were explicitly leaked by both schemes. On larger datasets Chenette et al. may leak even more.

More precisely, we took a closer look at the Boldyreva et al. scheme [BCO11]. Theoretical results [BCLO09] suggested that this construction “leaks the most-significant half of the plaintext bits,” but our experiments showed that this conclusion was too generous: simple attacks recover more.

Our results above show that the Chenette et al. construction can leak the same number of bits as Boldyreva et al., despite the one-wayness gap. We go further in attacking the Chenette et al. construction, showing how to infer additional plaintext bits beyond those explicitly leaked. Our attack against the Chenette et al. construction was able to predict almost every point in a location dataset of California road intersections to within 5km (and most to within 0.5km), while the leakage profile did not explicitly leak the location of any plaintext to within less than 400 km. We note that our quantitative claim depends on the attacker determining the two most significant bits of the longitudes by hand, since the ORE construction did not leak these explicitly. (This amounts to putting a California-shaped blob of points in one of four possible places on the globe.)

In Chapter 3, we introduced the format-preserving encryption and Feistel network (FN) primitives before investigating the security in the following chapters. We did not provide any contribution in this chapter. In Chapter 4, we developed a set of attacks for various number of rounds of Feistel networks with specific design (two branches and modular addition) that was used to construct FPE schemes. We showed a new generic known-plaintext attack on 4-round Feistel networks (we insert this attack in our slide attack to break the FF3 construction). Our techniques to develop a 4-round attack is novel and different from any previously known attack on Feistel networks. In our attack, we fully recover the round functions with $\mathsf{N}^{\frac{3}{2}}\left(\frac{\mathsf{N}}{2}\right)^{\frac{1}{2\mathsf{L}}}$ known plaintext and with a time complexity $\mathsf{O}(\mathsf{N}^{2+\frac{3}{\mathsf{L}}})$ for four rounds. Furthermore, we utilized our 4-round FN attack to extend the round function recovery to more rounds. Due to the generic and known plaintext nature of our 4-round FN attack, we easily adapted it to a chosen-plaintext attack to apply it on five and more rounds Feistel structures.

We continued our studies to strengthen the attack to five or more rounds. We explored Feistel networks with two branches, random functions, and any arbitrary number r of rounds further with known generic exhaustive search attack. We started with a brief survey of different types of attacks on FN and presented their complexities and then proposed improved attacks. For small number of rounds, the best attack is an improvement of Meet-In-The-Middle (MITM) attack. However, we found a better attack for larger number of rounds based on partial exhaustive search. This attack for Feistel network with arbitrary number of rounds over small domains show that both FF1 and FF3 do not provide the intended security over tiny domains. Our attack shows that neither FF1 with $\mathsf{N} = 11$ nor FF3 with $\mathsf{N} \leqslant 14$ (even with our fix to the FF3 attack) offer 128-bit security.

In Chapter 5, we gave a total practical break to 8-round Feistel network based FF3 FPE standard over a small domain. In this part, rather than generic attacks on 8-rounds FF3, our attack exploited the "bad domain separation" in FF3. Namely, the specific design choice of FF3 allows us to permute the round functions by changing the tweak and it leads us to develop a slide attack (using only two tweaks). The attack works with chosen plaintexts and tweaks when the message domain is small. It requires $\mathsf{O}(\mathsf{N}^{\frac{7}{4}+\frac{1}{4\mathsf{L}}})$ chosen plaintexts and two tweaks, with time complexity $\mathsf{O}(\mathsf{N}^5)$, where N^2 is input domain size to the Feistel network and L is a parameter in our attack which is typically set to 3 in experimental results. Luckily, the fix to prevent FF3 against our attack is quick and easy to incorporate without changing the main structure of the scheme.

The author's series of publications are listed below.

- Circular Security Reconsidered by F. Betül Durak, Serge Vaudenay (2016). Published in Innovative Security Solutions for Information Technology and Communications, SECITC 2016, Lecture Notes in Computer Science, vol 10006, Springer [BDV16]. (Not covered in this dissertation.)

- What Else is Revealed by Order-Revealing Encryption? by F. Betül Durak, Thomas M. DuBuisson, and David Cash (2016). Published in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016 [DDC16]. (Covered in Chapter 2)

- *Patent Application.* Method for Multi-Server Searchable Symmetric Encryption with Fully Oblivious Search Pattern by Guajardo Merchan Jorge, Jain Shalabh, Hoang Thang, F. Betül Durak. Application date: 14/11/2016. Application Number: 62/421579. (Not covered in this dissertation.)

- Breaking the FF3 Format-Preserving Encryption Standard over Small Domains by F. Betül Durak, Serge Vaudenay (2017). Published in Advances in Cryptology, CRYPTO 2017, International Association for Cryptologic Research, Lecture Notes in Computer Science, vol 10402, Springer [DVb]. (Covered in Chapter 4, 5)

- Breaking the FF3 Format-Preserving Encryption Standard over Small Domains by F. Betül Durak, Serge Vaudenay (2017). Published in Proceedings of Early Symmetric Crypto, ESC 2017 [DVa]. (Covered in Chapter 4, 5)

- Practical Oblivious Dynamic Searchable Encryption via Distributed PIR and ORAM by Thang Hoang, Jorge Guajardo, Attila Altay Yavuz, F. Betül Durak. Submitted to Annual Computer Security Applications Conference (ACSAC) 2017. (Not covered in this dissertation.)

- Cryptanalysis of Generic Feistel Network by F. Betül Durak, Serge Vaudenay. (Covered in Chapter 4)

Chapter 2

What Else is Revealed By Order-Revealing Encryption?

Property-preserving encryption (PPE) encrypts data in a way that certain functionalities that we have with plaintexts can be computable on ciphertexts by someone without the key. PPE flavors include *searchable encryption* [SWP00b] for keyword searching in text, *deterministic encryption*, where the equality of two plaintexts are preserved, and *order-revealing encryption (ORE)* [AKSX04, BCLO09], where the order of two plaintexts can be detected from corresponding ciphertexts without decrypting. PPE in general achieves weaker security than traditional encryption because, by the definition of the primitives, they inherently leak information about plaintexts. On the other hand, PPE enables efficient encrypted database applications [PRZB11, GHH+14, enc15, ABE+13].

This chapter investigates the security of ORE in real world applications. An ORE scheme is an encryption algorithm $\mathcal{E}$ that takes numbers from some domain as input such that, given two ciphertexts $\mathcal{E}_K(x)$, $\mathcal{E}_K(y)$, anyone can tell if $x < y$ without the secret key K. ORE is a very useful primitive for encrypted databases. Namely, columns in a database table that are encrypted with ORE allow range queries. In a single client/server model, in order to issue a query for the range (a, b) the client encrypts a and b and sends the ciphertexts to the server that holds the ORE encrypted columns in its tables. The server can then find all rows with values between the encrypted endpoints without decrypting them. Hence, ORE provides a way to upgrade the security of a database while keeping the functionality.

In this chapter, we present a collection of observations, experiments, and attacks dealing with the use of ORE on data that will be encountered in practice. We point out overlooked fundamental properties of ORE security, perform experiments to measure the security of ORE against known attacks but on non-uniform data, and give new attacks showing that ORE is less secure than theory indicated.

This work is a joint work with David Cash from Rutgers University and Thomas DuBiusson from Galois Inc. published in the proceedings of 2016 Conference on Computer and Communications Security (CCS 2016) [DDC16].

2.1 Order-Revealing Encryption

An ORE scheme consists of three algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ for key generation, encryption, and comparison respectively. The key generation algorithm outputs a key K. The encryption algorithm may be randomized, takes an input message x from an associated ordered plaintext space, and emits a ciphertext y. The comparison algorithm takes two ciphertexts $\mathsf{y}_0, \mathsf{y}_1$ as input and outputs a bit b indicating that the message in y_b is larger (or $\perp$ if the messages are equal). When the algorithm $\mathcal{C}$ is a canonical numerical comparison operator [1], the scheme is called an *order-preserving encryption (OPE)* scheme, although this distinction is not important for our attacks. Therefore, we will analyse the security of the the OPE construction given by Boldyreva et al. [BCO11] under the general security results of ORE.

2.2 Security of Order-Revealing Encryption

Starting with Boldyreva et al. [BCLO09], ORE schemes have been proven secure with respect to mostly incomparable definitions. We follow Chenette et al. [CLWW16] in defining ORE security with a parameter called a *leakage profile* $\mathcal{L}$. Formally, $\mathcal{L}$ can be any function on vectors of messages. Intuitively, an ORE scheme must leak the order of the plaintexts, but it may also be allowed to leak more. The formal definition requires that an adversary cannot win a game requiring it to compute more information than is output by $\mathcal{L}$. Theoretical properties, like one-wayness, were identified for some

[1]or equivalently, if $\mathcal{C}$ defines a total order in the output space of $\mathcal{E}$ (in ORE, it only defines a total order on the subset of values which have a preimage).

definitions and used to compare leakage profiles. The proofs of one-wayness for ORE assume that data are uniformly distributed over the message space, but this does not appear to be the case in any application we could think of. Thus, it is not clear what the theoretical security results of ORE mean in practice. We proceed informally as our attacks do not depend on the details of the definition, but rather only on the leakage profile.

Five leakage profiles have been considered in the literature. We term them **Ideal**, **ROPF**, **MSDB**, **RtM**, **MtR**. The first two were introduced by Boldyreva et al. [BCLO09], who constructed an **ROPF**-secure ORE. Later, Boldyreva et al. [BCO11] proved that **ROPF**-security requires an ORE to roughly reveal half the plaintext bits under the assumption that data to encrypt is picked uniformly at random. Later, Chenette et al [CLWW16] defined profiles **MSDB**, **RtM**, **MtR** and built ORE schemes achieving them. They proved a result showing that these profiles leak fewer bits on uniform data. We review them in more detail now.

Ideal. The *ideal* leakage profile only reveals the ordering relation of the ciphertexts. This profile, for instance, hides any statistical information about the messages. The profile is achievable using (currently impractical) theoretical tools [BLR+15] or by *interactive* variants of ORE where encryption is a protocol between two parties [PLZ13, KS14, Ker15]. We consider these protocols in-scope for this work, and will treat them as ORE. We note that [Ker15] actually achieves stronger-than-ideal security by hiding the frequency of plaintexts in the column. Some of our attacks will still apply to this construction.

ROPF. The *random order-preserving function* profile [BCLO09] is defined with respect to a plaintext space and range. Given a column of data $\mathbf{d}$ with n rows, the leakage profile chooses a *random order-preserving function* f from the plaintext space to the range, and outputs $f(\mathbf{d}[1]), \dots, f(\mathbf{d}[n])$, i.e. the function applied component-wise to the dataset.

The **ROPF** profile was later shown to reveal approximately the most-significant half of the plaintext bits [BCO11] of a random message, and also to hide roughly the other half. The selection of the range set is a parameter to be set when configuring the instantiation.

MSDB. The *most-significant-differing bit* profile [CLWW16], on a column $\mathbf{d}$ with n entries, will output the order of the plaintexts in $\mathbf{d}$ along with, for all $1 \leqslant i < j \leqslant n$, a number $\mathrm{diff}_{i,j}$ that indicates the index of the most significant bit where the plaintexts $\mathbf{d}[i]$ and $\mathbf{d}[j]$ differ, along with the values of the bits at that position. Equivalently, $\mathrm{diff}_{i,j}$ is the length of the longest common prefix of $\mathbf{d}[i]$ and $\mathbf{d}[j]$, plus one.

For example, if the plaintexts are $0000, 0001, 1000$, the **MSDB** profile would allow one to infer the first most significant bit of all three plaintexts, and the last most significant bit of the first two plaintexts, along with equality of appropriate prefixes. For this example, an adversary would learn that the plaintexts must be of the form $0uw0, 0uw1, 1xyz$, where u, w, x, y, z are variables for bits that are not explicitly leaked.

RtM and MtR. We consider two more profiles that are induced by composing multiple ORE schemes as suggested by Chenette et al. [CLWW16]. **RtM** stands for "**ROPF** then **MSDB**" and **MtR** stands for "**MSDB** then ropf". The former, **RtM**, is induced by first applying the leakage profile **ROPF** to get a vector $(f(\mathbf{d}[1]), \ldots, f(\mathbf{d}(n)))$, and then applying **MSDB** to this vector (treating it as if it was a plaintext). That is, the profile leaks the index of the most significant differing bit of $f(\mathbf{d}[i])$ and $f(\mathbf{d}[j])$ for each $i < j$.

The profile **MtR** will be scheme-dependent. The scheme is defined by composing some **MSDB**-secure OPE scheme with an **ROPF**-secure ORE scheme. The leakage is defined by the output of the composed scheme. Note that we must assume that the **MSDB** scheme here is OPE, not just ORE, in order to define the compare algorithm for this version. Indeed, in [CLWW16], Chenette et al. showed how to convert their ORE scheme into an OPE scheme.

The two profiles **MtR**, **RtM** were originally introduced without distinction, but we observe that they provide practically different security.

2.2.1 Existing Attacks

2.2.1.1 Theoretical Result on Leakage Profiles

One-wayness of ROPF. Boldyreva et al. [BCO11] introduced the notion of *window-one-wayness (WOW)* for ORE. An ORE is L-WOW if given $\mathcal{E}_K(x_i)$ for several uniformly random strings $x_i \in \{0,1\}^m$, it is infeasible to determine an interval of size L containing some x_i. To make our experiments easier to compare, we consider an alternative version that we call ℓ-bit-WOW, which says that it should be infeasible to compute the ℓ-bit prefix of some x_i.

Boldyreva et al. [BCO11] also proved that any **ROPF**-secure OPE with plaintext space $\{0,1\}^m$ is L-WOW secure for L approximately $2^{m/2}$, meaning that it is hard to compute the lower $m/2$ bits of a uniformly random plaintext.

One-wayness of MSDB. Chenette et al. [CLWW16] proved that **MSDB**-secure OREs have stronger WOW security. That work proved ℓ-bit-WOW security for **MSDB** schemes, for a much smaller ℓ of about $\ell \approx 1/\log \epsilon$, where ϵ is the desired bound on the success probability of the adversary. (A smaller ℓ means the result is stronger, and it is hard to guess even a smaller prefix of the plaintext.) A closely matching attack against ℓ-bit-WOW of **MSDB** is almost immediate, as one can show that the **MSDB** leakage profile will provide roughly the same number of bits of a plaintext prefix.

Security of RtM and MtR. Composed ORE schemes were suggested in [CLWW16] to combine the security of **ROPF** and **MSDB**. It was proven that the composition of individual OREs will result in a construction that inherits both security notions. We remark that this is only a lower bound on the quality of security achieved, and that the composition may be strictly more secure than **ROPF** or **MSDB**.

2.2.1.2 Attacks on Leakage Profiles

Scaling attack against ROPF. Boldyreva et al. [BCO11] introduced what we term the *scaling attack* and denote $\mathsf{ScalingAtk}$ against the WOW security of an **ROPF**. It works as follows: to attack an ORE with plaintext space $\{0,\dots,M\}$ and range $\{0,\dots,N\}$, one maps a ciphertext y for an unknown plaintext x to $x' = \lceil yM/(N+1) \rceil$. It was proved that this estimate will satisfy $|x - x'| < 8\sqrt{M}$ with high probability.

We observe that when $M = 2^m$ and $N = 2^n$ are powers of two, this can be approximated by a simple bitshift: $x' \approx y \gg (n-m)$ i.e. the right-shift of y until it is the same bit-length as a plaintext. The estimate for these parameters becomes $|x-x'| < 2^{m/2+3}$. This now implies that for most x, approximately the upper half of the bits of x' will match those of x. We omit a formal analysis of the variation due to high-order bit rollovers.

Sort attack against Ideal. Recent work by Naveed et al. gave attacks on **Ideal**-secure ORE schemes (and thus on all of the other profiles except that of [Ker15]). The first was the sort attack, denoted SortAtk below, which we recall in detail for use later. The attack assumes knowledge of a plaintext space that we denote as $\{1, 2, \ldots, M\}$, and attempts to guess the plaintexts used to generate a vector of ciphertexts $\mathbf{c}$ that it takes as input. The attack sorts the vector $\mathbf{c}$ (using the ORE comparison algorithm), and then guesses that the smallest unique ciphertext corresponds to 1, that the second smallest ciphertext corresponds to 2, and so on. Formally, it is defined as follows. Let $c_1, c_2, \ldots$ be the ciphertexts in $\mathbf{c}$ in sorted order. $\mathsf{SortAtk}(\mathbf{c})$ outputs a mapping α from ciphertexts to $\{1, \ldots, M\}$, where

$$\alpha(c) = \begin{cases} i & \text{if } c \in \mathbf{c},\ c = c_i \\ \perp & \text{otherwise} \end{cases} .$$

The sort attack was shown to correctly invert a large fraction of ciphertexts in a simulated attack. However, it required that the plaintexts were "dense" in the message space, meaning that almost all possible plaintexts in $\{1, \ldots, M\}$ are encrypted in $\mathbf{c}$. This was true for several columns in certain hospital databases, like age (years), length-of-stay (0-365), and others.

Cumulative attack against Ideal. Naveed et al. gave a second attack, called the cumulative attack and denoted CumulativeAtk that works when plaintexts are not dense in the message space. This attack also takes as input a vector of ciphertexts $\mathbf{c}$, but additionally requires a training vector $\mathbf{z}$ of data which should be drawn from the same distribution as the plaintexts in $\mathbf{c}$.

The cumulative attack outputs a map α from ciphertexts in $\mathbf{c}$ to plaintexts in $\mathbf{z}$. The map α is computed via a linear program that minimizes the error in frequencies and in the cumulative distribution (i.e., the fraction of plaintexts less than a given plaintext in $\mathbf{z}$ versus $\mathbf{c}$). We omit further details, but we note that the training input $\mathbf{z}$ is essential for two reasons: First, the map α will only output plaintexts in $\mathbf{z}$, and thus if a plaintext has not been seen, then $\mathsf{CumulativeAtk}(\mathbf{c}, \mathbf{z})$ will never guess it. Second, the guesses are entirely dependent on using $\mathbf{z}$ as a "typical" distribution with frequencies that correspond to the target data.

Other attacks. It was observed by Boldreva et al. [BCLO09] that *chosen-plaintext attacks* allowed the easy extraction of plaintexts. This, and attacks that observe queries, are not considered in this work but are likely to further diminish security in practice.

2.2.2 Datasets and Implementations

We use two real geographic datasets `Cal`, `SpitzLoc`, one synthetic geographic distribution `Globe`, and one real time-stamp dataset `SpitzTime`.

The dataset `Cal` represents the latitude and longitude of about 21,000 intersections in the California road network (also used by Mavroforakis et al. [MCO+15] in their ORE work). [2] Latitudes are numbers between −90 and 90 and longitude are numbers between −180 and 180, both given to six decimal digits. The latitudes are all between 32.541302 and 42.017231, and longitudes were between −124.389343 and −114.294258. We encode a given latitude x as $10^6(x+90)/180$ in binary, and a longitude y as $10^6(x+180)/360$ in binary. Encoded latitude is represented in 27 bits, and similarly longitude is represented in 28 bits.

The dataset `SpitzLoc` consists of latitude and longitude coordinates tracking the movement of German Green party politician Malte Spitz over six months. The dataset records the location of towers used for voice calls and text messages, as well as times (which we used for the `SpitzTime` dataset below). The data were provided by Deutsche

[2]Dataset obtained from `http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm`

Telecom and posted publicly as an example of highly sensitive information recorded by service providers. [3] The dataset consists of about 36,000 rows, with many locations missing. We extracted from this 1,477 non-repeating (latitude, longitude) pairs in Germany that we call `SpitzLoc`. The points were available up to 8 decimal digits of accuracy, but we encoded them as 28- and 29-bit strings as in `Cal`.

The `SpitzTime` dataset consists of 30,492 time-stamps represented as seconds from an epoch of January 1, 2000. The actual range is between 2009-08-31 to 2010-02-27 and mostly daylight hours. Concretely, the timestamps were integer values between 305,020,620 and 320,629,560.

Finally, we select a distribution `Globe` to represent the latitude and longitude of a uniformly random points on Earth. We encode the points as bitstrings of length 32, using the same method as before. We chose this distribution because it does not result in uniform samples of pairs from $\{0,1\}^{32}$, but it may model the distribution of some datasets.

All experiments were performed on recent Mac laptops. Our experiments were written in Python and used the **ROPF** implementation from CryptDB [PRZB11]. The other leakage profiles could be simulated exactly without full implementations.

2.3 Inter-Column Correlation

The sort- and cumulative-attacks [NKW15] showed that the ciphertexts (produced with **Ideal**-secure ORE that only leaks order and frequency) in an individual encrypted column could sometimes be inverted, but required at least one of the following conditions to be met:

- The plaintext data present in the column is *dense* in the plaintext space, meaning that most or all of the possible plaintext values encrypted at least once.
- The plaintext data has *low entropy*, meaning that most of the possible plaintexts appear frequently, and particularly that a training set will have many plaintexts in common with the column under attack.

[3] The data is downloadable at `http://www.zeit.de/datenschutz/malte-spitz-data-retention`

We add to this another condition: when two or more encrypted columns hold data that are *correlated*. We show that information may be leaked even when the data in the column is the encryption of small subset of huge domain, when all encrypted values are unique (and without any training data), and an **Ideal**-secure ORE is used. We call this *inter-column correlation*, and hereafter we experiment with applying the sorting attack on multiple columns at once.

2-D Sort attack. We consider applying the sort attack to two columns at once. Recall that, given a ciphertext vector $\mathbf{c}$, $\mathsf{SortAtk}(\mathbf{c})$ outputs a mapping α from ciphertexts to the set $\{1, \ldots, M\}$, where M is the number of unique ciphertexts in $\mathbf{c}$.

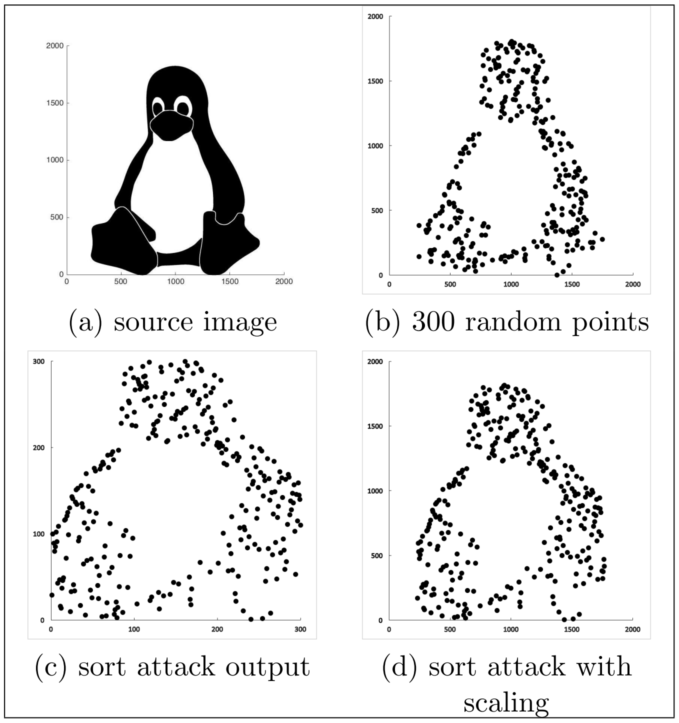
We simply apply this attack to two columns independently. That is, we define the attack $\mathsf{2DimSortAtk}(\mathbf{c}_1, \mathbf{c}_2)$ to output (α_1, α_2) given by $\alpha_i \leftarrow \mathsf{SortAtk}(\mathbf{c}_i)$ for $i = 1, 2$. In other words, this attack independently sorts each ciphertext vector, and emits guesses about the plaintext vectors using the same technique as $\mathsf{SortAtk}$. Below, we argue that this attack should be interpreted differently from the original single-column attack.

Visual example. We start with an example. In Figure 2.1 (a), we plot an image, which is formally a set of approximately 10^6 points in $\{1, \ldots, 2000\} \times \{1, \ldots, 2000\}$ (the black points correspond to points in the set). In 2.1 (b), we select a random subset of only 300 points from the set, conditioned on none of the coordinates repeating (that is, none of the chosen points have the same x or the same y value).

We model the chosen subset as a dataset $\mathbf{d} = (\mathbf{d}_x, \mathbf{d}_y)$ consisting of a pair of columns $\mathbf{d}_x$ and $\mathbf{d}_y$ each with 300 cells, where each row represents the location of a black point in the image. We selected the data in this way to ensure that prior attacks against the two individual columns would not be effective. As all points in each column are unique, the frequency information is trivial. Moreover, each column only consists of 300 out of 2000 possible plaintext values, so the columns are not dense in the plaintext space.

Despite the prior attacks failing to recover the plaintexts in $\mathbf{d}$, we show that information can be recovered by considering the columns together via the 2-D sort attack. The results of our attack are plotted in 2.1 (c), where the rough features of $\mathbf{d}$ are still visible.

Our multi-dimensional sorting attack does not guess any point correctly – The



(a) source image (b) 300 random points

(c) sort attack output (d) sort attack with scaling

Figure 2.1: Visualization of the 2-D sort attack on an image dataset encrypted with **Ideal** ORE.

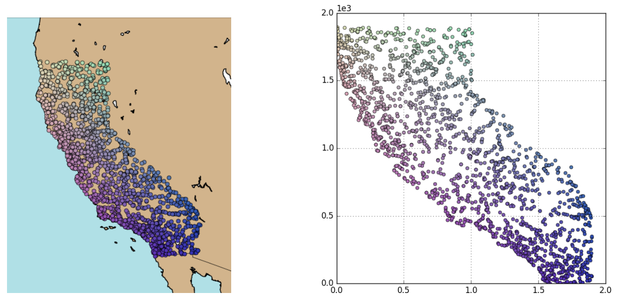
original points are in $\{1, \ldots, 2000\}$ but the sort attack only emits guesses in $\{1, \ldots, 300\}$. Thus, by the metrics of [NKW15], the attack does not work. But it is clear that a significant portion of the structure of the image is recovered, including relatively fine details like the arrangement of points from the penguin's foot in the lower right. (By scaling the plot in 2.1 (d) it resembles the plaintext points more strongly.)

This experiment suggests that we expand our consideration of the leakage of ORE in two senses. First, even when individual encrypted columns are useless for an adversary, the leakage from *correlated* encrypted columns may combine to reveal a harmful level of information, even to an adversary with no training data to help its analysis. Second, even when an attack does not recover plaintexts correctly, it may still be plausibly considered successful since it recovers partial information.

2.3.1 Sort Attack on Location Datasets

We experimented with the 2-dimensional sort attack on two location datasets: `Cal` and `SpitzLoc` (see Section 2.2.2 for details).

`Cal` Dataset visualization. For the `Cal` dataset, we selected a subset of 2,000 random points and ran 2DimSortAtk on the ideal leakage for that subset. In Figure 2.2, we plot the plaintext points and the output of 2DimSortAtk (the colors are used to track plaintexts between the two plots). We observed that the shape of California was still clearly visible, and some other features like western protrusion one third of the way up were also visible (several other runs on random points were similar). Below we return to this attack and try to evaluate it quantitatively.



(a) 2,000 plaintext points (b) 2DimSortAtk output

Figure 2.2: Visualization of 2DimSortAtk on a subset of the `Cal` dataset.

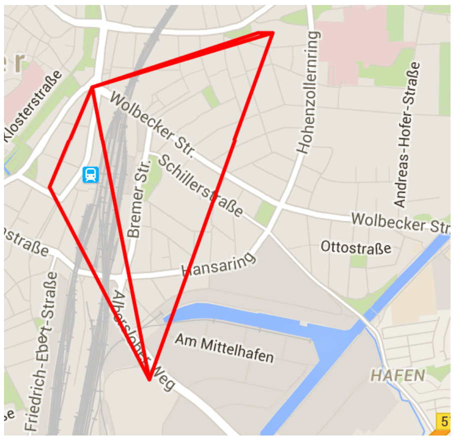
`SpitzLoc` Dataset visualization. The database we worked with includes the complete road network data for California with 21K intersection points. However, we did not always have a complete road network of a state or country, but what we had was a subset of geographical data in a state/country. For an example, personal location data which map providers, such as Google maps or Apple, keep track of, from their customers, in order to provide better service is stored in the cloud in a way that shows the trips of an individual with time-stamps. For a person who lives in a specific city or travels around this particular city during a period of time, a map provider stores the locations of the person with a fixed frequency during these trips. When an application wants to secure this type of data so that range queries are enabled on the dataset, **Ideal** security is likely to be preferable due to its ideal leakage. However, as in California data leakage with **Ideal** schemes, it is not clear what an adversary learns from a set of points that forms a trip from a start point to an end point. This trip can be visualized as interpolating the consecutive points in the database for both the plaintexts in a map and their orderings in ciphertexts in a 2D grid.

To explore further in this direction, we use a dataset of German Green party politician Malte Spits. We take his trips in the month of October in 2009 and considered investigating his trips in a single day, in a single week, and in a complete month. We order the ciphertexts based on only his provided locations stored in the database (in Germany) instead of the complete geographical location of the country. Then, we plot both the locations as plaintexts on Google maps for his daily, weekly and monthly trips and the ordering leakage for ciphertexts in 2D grid. The results are shown in 2.3 for his one day trip in October first, for his complete trip in the first week of October, and for his complete trip in the entire month October in 2009. The results noticeably indicate the nature of the trips such as if it is taken at walking distance in a small area, or if it coordinates are from south to north.

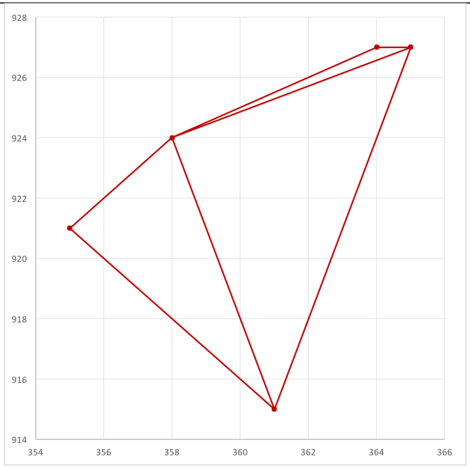
Discussions. The authors initially investigated the security of ORE for two projects. The first required encrypting a database that included locations of naval vessels, and second considered encrypting personal mobile phone GPS histories for location-based queries. Our experiments above showed that even the best possible leakage (**Ideal**) would still result in a concerning loss in secrecy.

We conjecture that the security in practice may be much worse. This is because we generated the plots above *without any training data or side information.* An attack which knows that `Cal` points are taken from California, or that `SpitzLoc` points are taken from the movements of a German citizen, can use side information like the shape of the movement zone and the distribution of cities and other points of interest when generating guesses at the plaintext data.

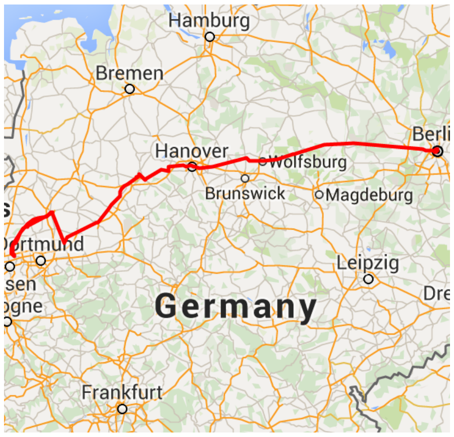
Moreover, our attacks are on relatively small amounts of data. For instance, a column with 2,000 rows could be downloaded and searched locally in most scenarios,



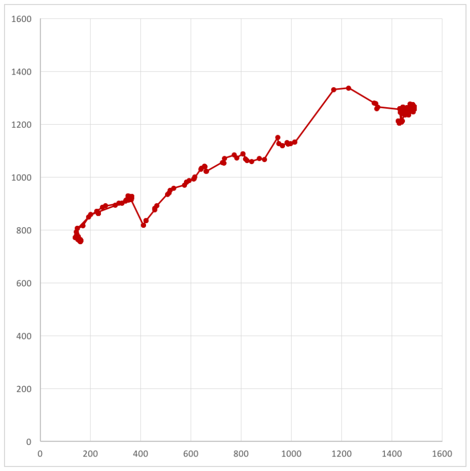
(a.1) October 1



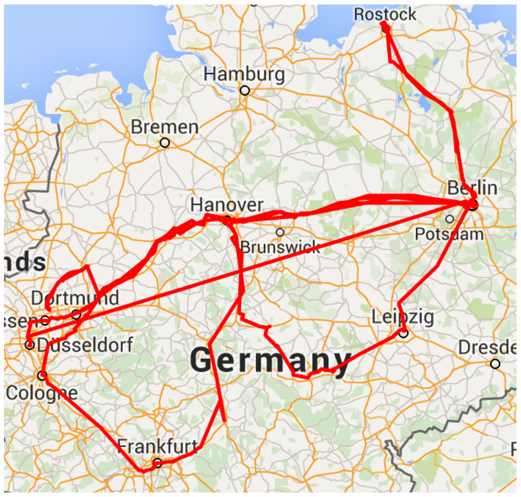
(b.1) Attack output on (a.1)



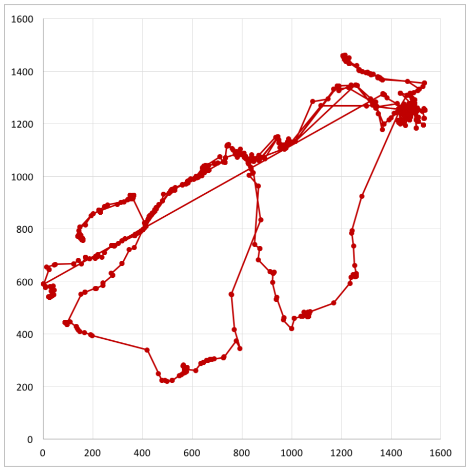
(a.2) October 1–7



(b.2) Attack output on (a.2)



(a.3) October 1–31



(b.3) Attack output on (a.3)

Figure 2.3: Visualization of subsets of the `2DimSortAtk` output on the `SpitzLoc` dataset.

meaning that ORE may be unnecessary there. On larger datasets, such as years of location movements, the guesses might be even more accurate.

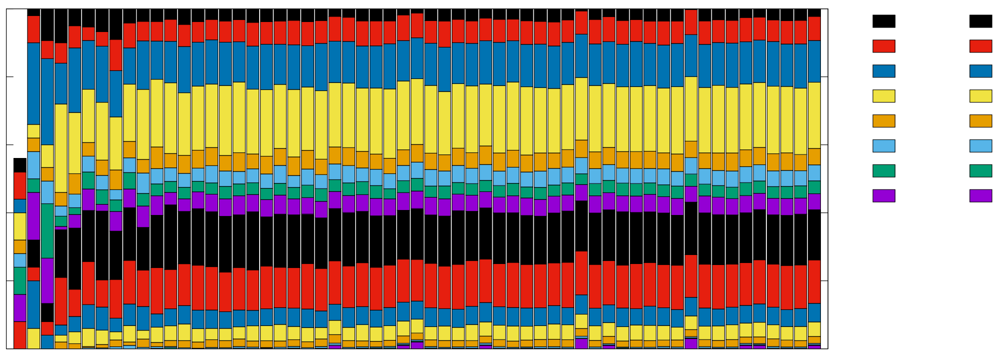


Figure 2.4: Histogram of 2DimSortAtk output on subsets of 25 up to 20,000 points (non-uniform step sizes). Distances in km from the correct plaintext point.

2.3.2 Sort Attack Accuracy with Bounds

We now consider extending the 2-D sort attack to use the additional hint of *bounds* on the possible plaintexts and generate guesses on the plaintexts that can be quantitatively evaluated. Concretely, we consider the following variant of 2DimSortAtk, denoted Bnd2DimSortAtk. In addition to the encrypted columns $(\mathbf{c}_1, \mathbf{c}_2)$, the attack also takes as input pairs of numbers (a_1, b_1) and (a_2, b_2). First, it runs the original 2DimSortAtk twice to generate mappings (α_1, α_2) from the ciphertexts to $\{1, \ldots, M\}$. We then compose these mappings with functions f_1, f_2 that evenly space the guesses within the given bounds, resulting in the following attack:

```
Bnd2DimSortAtk($\mathbf{c}_1, \mathbf{c}_2, a_1, b_1, a_2, b_2$)
01 Compute $(\alpha_1, \alpha_2) \leftarrow$ 2DimSortAtk($\mathbf{c}_1, \mathbf{c}_2$)
02 Define the function $f_1$ by $f_1(i) := (i-1) \cdot \frac{b_1 - a_1}{|\mathbf{c}_1|} + a_1$
03 Define the function $f_2$ by $f_2(i) := (i-1) \cdot \frac{b_2 - a_2}{|\mathbf{c}_2|} + a_2$
04 Output $(f_1 \circ \alpha_1, f_2 \circ \alpha_2)$.
```

In the algorithm, $|\mathbf{c}|$ denotes the number of ciphertexts in the encrypted column $|\mathbf{c}|$.

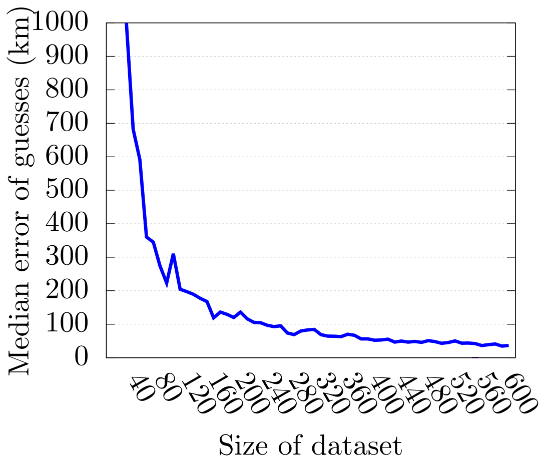
`Cal` Dataset attack with bounds. We used random subsets of $25, 50, ..., 2000$ points from the `Cal` dataset to evaluate Bnd2DimSortAtk. The results are plotted in Figure 2.4. In each case we gave the attack the same bounds, which were set to the greatest and smallest latitudes/longitudes in California and in particular, they were not the maxes and mins over the actual subset under attack. We plotted the quality of guesses as stacked histograms in Figure 2.4 (each bar reports on a different run, showing the proportion of points that were guessed to within different accuracies on that run).

The maximum error in any of the experiments was around 140 km while the minimum was about 2 km. We note that our plot reveals that the quality of guesses was not improving with the number of points, which is interesting because we expected a dense set of points to reveal more ordering information. The explanation (which we found via inspection and plotting the guesses) is that bad guesses tend to stay bad, even with many points, and good guesses tend to stay good. We stress however that an attack, with, for example, a training set of points in California could likely do much better than this simple attack.

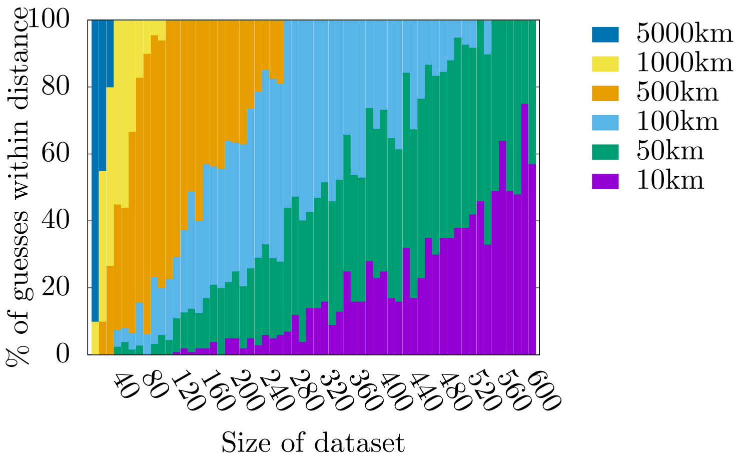
Discussion. Strictly speaking, this attack no longer exploits inter-column correlation because Bnd2DimSortAtk could be adapted to run on individual columns, and it would generate the same guesses for the latitude and longitude columns independently. However, we suggest it as a technique for evaluating the confidentiality of **Ideal** ORE on geographic data.

2.4 Leakage-Enabled Attacks

This section contains evaluations of known attacks on ORE but with non-uniform data (sections 2.4.1, 2.4.2, 2.4.3), showing that in some cases leakage is much worse than on uniform data. Then new attacks are presented (sections 2.4.4, 2.4.5, 2.4.6), and we also present our observations regrading modular order-revealing encryption (MORE) in Section 2.4.7.



(a) median line



(b) histogram

Figure 2.5: Accuracy for **MSDB** leakage on random globe points.

2.4.1 MSDB on Random Globe Points

We evaluated **MSDB** leakage on the `Globe` dataset. Recall that **MSDB** allows one to infer some bits explicitly but keeps others hidden. In order to quantitatively compare the leakage to the plaintext values, we filled in all unknown bits arbitrarily.

The accuracy of our guesses is evaluated in Figure 2.5. The **MSDB** leakage profile tends to leak more information on larger datasets (it is *monotonic* in the sense that it will never leak less when a subset of data is included in a larger set). We note that after 600 points, we found that over half the points could be guessed within 10km, and the other half could be guessed within 50km. Thus, even on a tiny dataset of random values, the geometric meaning of this leakage (which is limited mostly to high-order bits) reveals significant information about the dataset.

It is a fair observation that the `Globe` dataset is geographically uniform and numerically non-uniform, which undoubtedly impacts the quality of the attack. The same attack on numerically uniform data, which has a geographic distribution biased towards the poles, results in a slight improvement in the mean attack accuracy without taking advantage of the known distribution. We did not analyze datasets with encodings that maintain numeric and geographic uniformity.

2.4.2 ROPF on Real Locations

Boldyreva et al. [BCO11] proved that any **ROPF**-secure ORE with plaintext space $\{0,1\}^m$ is L-WOW (see Section 2.2.1.1) for $\mathsf{L} \approx 2^{m/2}$. This roughly implies that it is hard to guess more than $m/2$ of the most significant bits of a random plaintext. Their result was involved, and generalizing it to other plaintext distributions does not seem easy. Moreover, real data may not obey the distribution assumed in a proof anyway.

Thus, we instead evaluated how the ScalingAtk (see Section 2.2.1.2) performed on our datasets. We encrypted subsets of `Cal` using an **ROPF**-secure ORE of [BCLO09], and recorded the length of the plaintext prefix that appears in the corresponding ciphertext. The average number of bits preserved between the plaintexts and corresponding ciphertexts over the entire longitude column of `Cal` is 15 bits out of 27. This result matches the theory for uniform data. Moreover, Boldyreva et al. [BCO11] also showed that increasing the ciphertext length by more than few bits beyond the plaintext length does not have any effect on the security for uniform data. We also experimented with varying ciphertext lengths, and the average number of preserved bits remained around 15 bits for larger output sizes.

We have no theoretical framework to explain if and how this experiment will generalize to other data. It seems prudent (and easy) for practitioners to simulate this attack on test data before deployment.

2.4.3 ROPF on Small and Large Messages

Boldreva et al. proved roughly that an **ROPF**-secure ORE will leak half of the bits of a random input message, but not much more, even when the output space is only one bit longer than the input space. Here, we experimentally show that this result does not apply when encrypting messages that are close to the minimum and maximum elements of the message space. That is, for small and large messages, the Boldyreva et al. construction leaks far more.

We performed the following experiment. We fixed the message space of the Boldyreva et al. ORE to $\{0,1\}^{64}$, and output space to either $\{0,1\}^{65}$ or $\{0,1\}^{128}$ in two independent runs. We encrypted the plaintexts $x = 2^0, 2^1, \dots, 2^{63}$ to generate ciphertexts $c_0, c_1, \dots, c_{63}$, and computed $x'_i \leftarrow \mathsf{ScalingAtk}(c_i)$ for each i. We computed the error $e_i \leftarrow |x_i - x'_i|$ for each i, and averaged the e_i over 10 independent runs (i.e. we selected a new key each time).

Note that the largest message in our experiment is $x = 2^{63}$, the midpoint of the message space. By symmetry, similar results would be obtained for the large messages near 2^{64}, so we did plot these results.

In Figure 2.6, we plot the logarithms (base 2) of the errors e_i compared to i, the logarithm (base 2) of the message. We find that the scaling attack performs much better on small messages than on random messages, which can be guessed to within a distance of about 2^{31}. The ciphertext for $x = 1$ was predicted exactly in every run. Ciphertexts up to 2^4 were recovered to with distance 2 on every run. The rest of the guesses were much more accurate than 2^{31}, until we reach larger messages. There was no significant variation when we changed the output length of the cipher.

Let n be the input length. The trend is that an input $x \approx 2^i$ or $x \approx 2^n - 2^i$ can be predicted to within about $2^{i/2}$ accuracy. This can be stated alternatively as a new rule of thumb: *An **ROPF**-secure ORE leaks* all *of the leading zeros (or ones) of a message, and* additionally *the most-significant half of the remaining bits.*

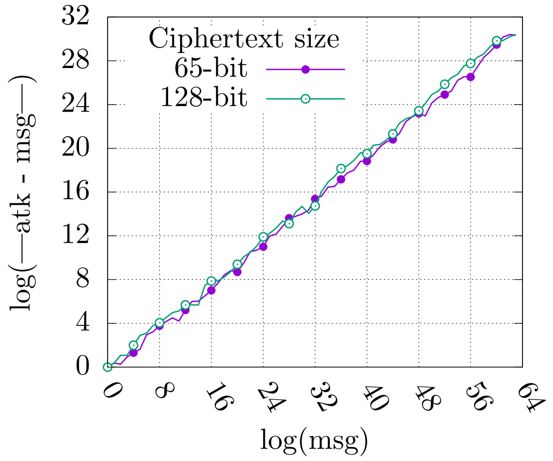


Figure 2.6: Accuracy of ScalingAtk against **ROPF** on small messages.

2.4.4 MSDB on Real Locations: The Distance Minimization Attack

A stronger one-wayness result was proved for the **MSDB** profile. When random data are encrypted, the proof showed that only the k most significant bits will be leaked, except with probability about $|\mathbf{d}|/2^k$. Again, a detailed result on general distributions seems difficult to derive and may not be useful in practice anyway. Intuitively, the result follows because two random plaintexts will have the same k-bit prefix with probability $1/2^k$, and if this does not happen then no bits beyond the k-th will be leaked.

We evaluated this profile on the `Cal` dataset. AS points are not uniformly random, the location of differing bits between pairs will depend strongly on the distribution. Moreover, by exploiting properties of the distribution, even more bits may be inferred, as we show below.

Intuition. We first visualize **MSDB** leakage on the `Cal` dataset. Recall that this leakage profile explicitly leaks some bits of the plaintexts and keeps other bits hidden (see Section 2.2). In order to visualize the geometry of the leakage on `Cal`, we pretend that the unknown bits are the average of their possible values – that is, instead of one or zero, they are actually "0.5". After filling in these values, we plot the result in Fig. 2.7. The large groups of points are separated from the main group when a relatively

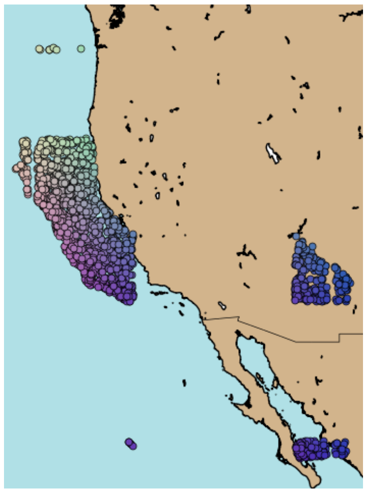


Figure 2.7: Visualization of **MSDB** leakage on `Cal` dataset.

high-order bit is hidden. There are also many other low-order bits hidden, but their effect is harder to see. Our intuition is that the large groups are trivial to move to the correct location by hand, and thus some of the hidden bits are easy to guess.

We give an attack to automate this, called the *distance minimization attack*, which is described and evaluated below. This attack will consider an individual encrypted column, and create guesses for every plaintext bit by moving the points to possible locations and seeing which is "closer" to the aggregate group, with the intuition being that correlated data will not often exhibit behavior like the irregular groups in Fig. 2.7.

The attack. The attack is given in pseudocode here, and a description follows.

```
DistMinAtk($\mathbf{c}$)
01 Initialize an empty guess vector $\mathbf{g}$
02 Foreach $\mathbf{c}[\mathrm{i}] \in \mathbf{c}$
```

```
03      Set $\mathbf{g}[i] \in \{0, 1, \perp\}^m$ using **MSDB** leakage
04      Reset $\mathbf{g}[i] \in \{0, 1, 0.5\}^m$ by replacing $\perp$ with 0.5
05  For $j = m$ down to 1
06      Foreach $\mathbf{g}[i]$ with $j$-th bit unresolved (i.e. set to 0.5)
07          Try assigning $j$-th bit of $\mathbf{g}[i]$ to 0 and 1
08          For each setting, compute $S = \sum_k |\mathbf{g}[i] - \mathbf{g}[k]|$.
09          Set $j$-th bit of $\mathbf{g}[i]$ to minimize $S$.
```

The distance minimization attack is given a column $\mathbf{c}$ of ciphertexts encrypted with an **MSDB**-secure ORE. It first initializes a guess vector $\mathbf{g}$ using **MSDB** leakage naively (line 03). That is, it starts with all bits unknown. Then, for every pair of ciphertexts $\mathbf{g}[i], \mathbf{g}[j]$, it performs the comparison to learn their differing bit, and then fills in that bit in the entries of $\mathbf{g}[i]$ and $\mathbf{g}[j]$.

After this initial stage, the guesses are strings in $\{0, 1, \perp\}^m$, where $\perp$ represents that a bit was not leaked. The rest of the algorithm will assign every $\perp$ entry to either zero or one. The algorithm will need to interpret the guesses $\mathbf{g}[i]$ as numbers, even when it has unknown bits. We will temporarily set $\mathbf{g}[i]$ to the "average" of all the possible values that it could be. Concretely, we convert each $\mathbf{g}[i]$ into a vector over $\{0, 1, 0.5\}$ by replacing $\perp$ with 0.5. Now when we need to consider $\mathbf{g}[i]$ as a number, we can compute its value using the binary expansion formula, but with values $0, 1, 0.5$ instead of just $0, 1$.

Starting on line 05, the attack begins to resolve the 0.5 entries in the guesses to either 0 or 1. It begins with the most significant bits, and considers the guesses with most significant bit unknown individually. When considering the guess $\mathbf{g}[i]$, the attack tries setting the unknown bit of $\mathbf{g}[i]$ to 0 and 1. For each setting it measures the sum of absolute differences between the resulting point and all of the points in the guess set (it is at this point that we are considering the guesses as *numbers*, to compute differences). The attack selects the bit setting that results in a smaller sum, and moves on to the next guess.

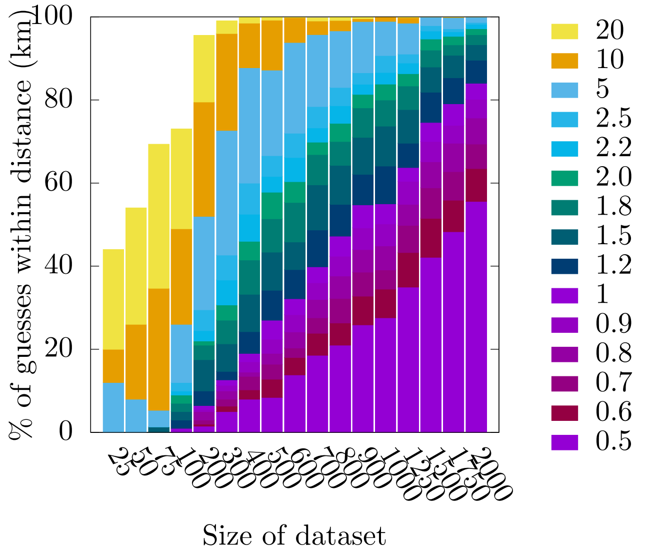


Figure 2.8: Accuracy of DistMinAtk against **MSDB** on `Cal`.

Evaluation. We evaluated this attack on random subsets of the `Cal` dataset of varying size. In `Cal`, the two most significant bits of longitude never change, so the attack cannot automatically infer these bits. This is exactly the sort of information that **MSDB** is designed to protect, but we assert that these can sometimes be guessed. In our setting, and attacker can run the attack without guessing these bits, notice that the shape of California appearing, and then fill in the missing significant bits by selecting one of four possible positions on the globe.

We evaluated the accuracy of our algorithm assuming the two most significant bits of longitude were given. In Fig. 2.8, we measure accuracy on subsets of sizes 25 to 2000. The figure reflects the improvement on the accuracy of guesses as the size of the dataset grows. On a dataset of 2,000 points, the algorithm guesses 95% of the plaintexts to within 2 km, and has average error 0.6 km. We note that no location was leaked to within less than 400km explicitly, and the improvement comes from the attack inferring hidden bits.

Discussion. For this particular example, leakage and correlation allowed accurate estimation of essentially every plaintext. More generally, use of any ORE scheme on any dataset that contains correlations should be viewed with a healthy dose of caution.

Limiting databases to small datasets might not offer meaningful protection and anyway negates the benefits of off-loading computation, which is a primary motivation for ORE. Practitioners desire simple rules for determining if property-preserving encryption and leaks inherent to the selected mechanism are acceptable, but no such rules have been proposed while examples of dangerous uses continue to mount.

2.4.5 Combined Attacks on MtR and RtM.

In this section, we consider how prior attacks can be combined to extract information from the composed leakage profiles **MtR** and **RtM** defined in Section 2.2. We start by describing how each can be attacked, and then evaluate the attacks.

In this section, let $\mathcal{E}^{\mathrm{r}}$ be the **ROPF**-secure ORE from [BCLO09] and $\mathcal{E}^{\mathrm{m}}$ be the **MSDB**-secure ORE from [CLWW16]. We will specify the appropriate domains and ranges for $\mathcal{E}^{\mathrm{r}}, \mathcal{E}^{\mathrm{m}}$ as needed below.

MtR. This profile describes the security of the following ORE scheme: It uses two random keys $\mathsf{K}^{\mathrm{m}}, \mathsf{K}^{\mathrm{r}}$. To encrypt a plaintext x, it computes

$$c_{\mathrm{in}} \leftarrow \mathcal{E}^{\mathrm{m}}_{\mathsf{K}^{\mathrm{m}}}(x); \quad c_{\mathrm{out}} \leftarrow \mathcal{E}^{\mathrm{r}}_{\mathsf{K}^{\mathrm{r}}}(c_{\mathrm{in}})$$

and outputs c_{out}. Note that we have assumed that the range of $\mathcal{E}^{\mathrm{m}}$ is contained in the domain of $\mathcal{E}^{\mathrm{r}}$. In order for the scheme to be correct (i.e. order-preserving), we need that $\mathcal{E}^{\mathrm{m}}$ is an *OPE* scheme, not just an *ORE* scheme, as the composition prevents running a general comparison algorithm on c_{in}.

Now, suppose we are given a column $\mathbf{c}$ of ciphertexts encrypted with the above construction. We apply the scaling attack to each ciphertext $\mathbf{c}[i]$ by running $\mathbf{y}[i] \leftarrow \mathsf{ScalingAtk}(\mathbf{c}[i])$ (see Section 2.2.1.2). According to the WOW-security results on **ROPF**, we expect approximately the most significant half of $\mathbf{y}[i]$ to match the bits of the corresponding **MSDB**-secure ciphertext produced as an intermediate ciphertext.

Next, we simply treat the column $\mathbf{y}$ as ciphertexts emitted by the [CLWW16] **MSDB**-secure construction. We carry out all of the pair-wise comparisons, recording when the differing bits are revealed in a vector of guesses $\mathbf{g}$. We can then interpret the vector **g** as we did when attacking **MSDB** alone.

RtM. This attack simply reverses the steps, so we sketch the important differences. The outer encryption is now **MSDB**-secure, so one can compute the differing bits on those ciphertexts to get a vector of guesses **g** (that will have known and unknown bits recorded). Then, we can apply the scale attack to the entries of **g** (we leave the unknown bits in place, and bit-shift the string as before). The result will issue guesses for some known bits which we can use as the attack output. We can also replace the unknown bits with 0.5 as in the DistMinAtk to quantitatively approximate the plaintexts.

Selecting a combined scheme. We remark that the constructions achieving **MtR** and **RtM** might not provide equivalent security. (The proof only shows that they achieve *at least* **MSDB** and **ROPF** security separately, but the combined modes might be strictly stronger.) It was unclear which will be better in practice.

There is, however, an efficiency difference between the existing instantiations of **MtR** and **RtM**. In [CLWW16], **MSDB**-secure ORE is constructed with relatively short ciphertexts (about 1.58 times the size of a plaintext), but the OPE version of their scheme has much longer ciphertexts (it expands the plaintext by a factor λ which corresponds to the “security parameter” and may be set to 80 or much larger). Thus **RtM**, which does not require the **MSDB**-secure part to be OPE, results in a much more efficient construction, and it is the one we evaluate below.

Evaluation. In Fig. 2.9, we plot the performance of the combined attack on **RtM** encryption. The results are similar to Fig. 2.8 so we conclude that, on this type of data, **MSDB** and **RtM** provide similar security.

2.4.6 RtM on timestamp data

We now turn to less dispersed data encrypted under **RtM**. We randomly draw time-stamps from the `SpitzTime` dataset, which are distributed over a several-month period. Our attack first encrypts this data using **ROPF** then models the **MSDB** leakage. From this model we then generate our guesses and compare these values to the plaintext data.

The analysis of time-stamp data can not be performed with distances between the

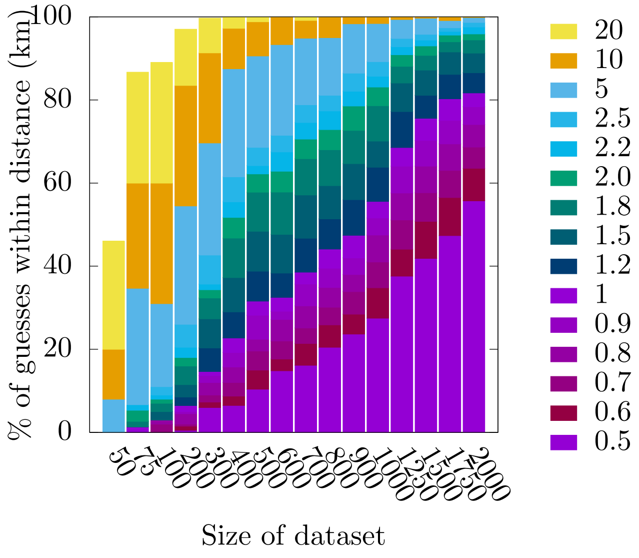
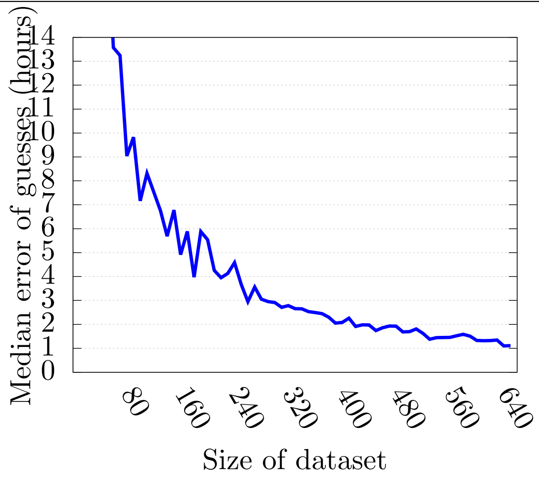


Figure 2.9: Accuracy of DistMinAtk against **RtM** on `Cal`.

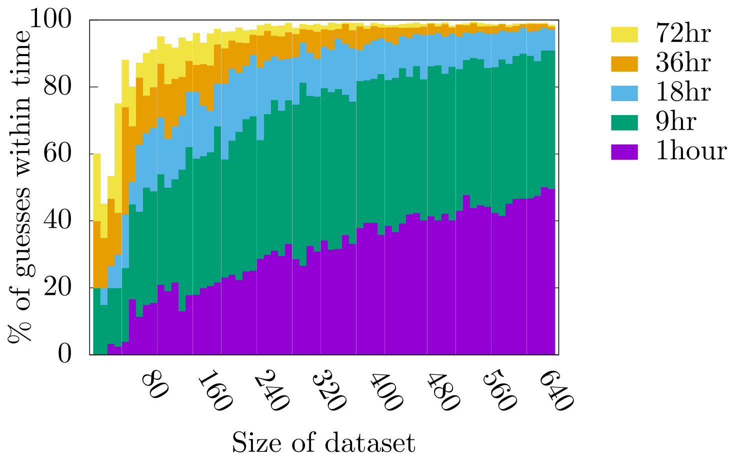
guessed and actual values, as done for the random locations. This is because the plaintexts are concentrated in a narrow portion of the domain, resulting in all ciphertexts sharing a significant matching prefix. Instead of a direct difference, our analysis and attack are a form of distance windowed one-wayness adversary [BCO11]. Informally, we compare the guessed distance between each pair of ciphertext (in order) with the distance between the plaintext data. More formally, for the sorted vector of guessed values, g, and matching plaintexts, p, the metric of interest is $|(g_i - g_{i-1}) - (p_i - p_{i-1})|$.

Now, we describe the setup, attack, and results. First, `SpitzTime` data beginning times are parsed into a 32 bit number of seconds starting at an epoch of January 1, 2000. All data is OPE encrypted and shifted as with the scaling attack. Then the **MSDB** leaks are modeled and inferences are made. Finally, we generate the guessed data set for comparison.

The results of the attack are shown in Fig. 2.10. Even for extremely small databases, many time-stamp differences are accurately guessed to within one-hour. The vast majority of guesses were correct to within two days.



(a) median line



(b) histogram

Figure 2.10: Accuracy of **RtM** leakage on `SpitzTime`.

2.4.7 Modular ORE on Real Locations

Modular ORE (MORE) was suggested by Boldyreva et al. [BCO11] to address the leakage of **ROPF**-secure ORE. When an adversary sees a column of data encrypted with **ROPF**, it can extract about half of the plaintext via their scaling attack. Thus, they suggest modifying an **ROPF**-secure ORE $\mathcal{E}^{r}$ as follows: In addition to the usual key K, store another string j, chosen at random from the plaintext space $\{0,1\}^{\mathsf{m}}$. Then define $\mathcal{E}^{\mathrm{mod}}_{(\mathsf{K},\mathsf{j})}(\mathsf{x}) := \mathcal{E}^{r}_{\mathsf{K}}(\mathsf{x}+\mathsf{j})$, where the addition $\mathsf{x}+\mathsf{j}$ is computed modulo 2^{m}. The construction $\mathcal{E}^{\mathrm{mod}}$ is no longer strictly speaking an ORE scheme, since the addition wraps sometimes. It was shown that efficient range queries are still possible (see [BCO11]). The scaling attack fails completely against $\mathcal{E}^{\mathrm{mod}}$ because it recovers the higher-order bits of $\mathsf{x}+\mathsf{j} \mod 2^{\mathsf{m}}$, which are independent of x. Recent work [MCO+15] developed query protocols to hide the shift value but left the actual encryption algorithm as defined above.

We took the `Cal` dataset, which Mavroforakis et al. used in testing their MORE algorithms, and applied their MORE construction to the latitude and longitude columns independently (with different keys and different "shifts" for each columns). Then, we applied the scaling attack to the ciphertexts and plotted the results in Fig. 2.11.

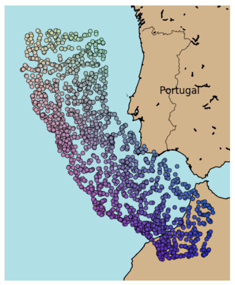


Figure 2.11: Visualization of MORE on the `Cal` dataset.

We observe that the fine details of the data are preserved (as expected, because **ROPF**-OPE was used), but it is also obvious how to correct both shifts by hand. We did not explore a quantitative or automated way to remove the shift. The general issue appears to be that for some distributions, shifting by a random value still preserves enough structure so that the shift is easily corrected.

2.5 Conclusions and Recommendations

This work shows that the conclusions of one-wayness theorems about ORE security should not be assumed to hold on real data. But, in practice, we expect attacks to recover even more information than our experiments did. Our datasets are relatively small, and leakage gets worse as the dataset grows. In this section, we highlight the performance of attacks on specific datasets, but these numbers should not be taken as "typical" numbers to inform deployment decisions. This is especially true because the relationship between the semantics of data and its encoding will affect the attacks. Instead, we recommend that one runs our attacks, which are all simple to implement and ran in a few minutes on our datasets, on test data (similar to production data for the intended use-case) before using ORE.

Some of the work in this sections grew out of exactly this approach, where the authors were considering ORE to store personal location histories. While we could not quantify the attacks theoretically, the plots in Figure 2.3 convinced us that ORE provide inadequate security, especially against adversaries with side information on the individual.

Chapter 3

Format-Preserving Encryption and Feistel Network

This chapter introduces two important concepts that will be analyzed cryptographically in the following chapters. The chapter is not intended to be equipped with any contribution, therefore it only provides a set of cryptographic tools and their security results. We briefly define what *format-preserving encryption* (FPE) and Feistel network is. A Feistel network is basically the main building block of some specific instantiations of FPE and it turns out that the meaning of its security is not well understood yet. In the following chapters, we will cryptanalyse this primitive along with Feistel-based FPE.

3.1 Format-Preserving Encryption

Format-preserving encryption (FPE) provides a method to encrypt data in a specific *format* into a ciphertext of the same *format*. A *format* in FPE schemes refers to a finite set of characters such as the decimal (or binary) numerals or alpha-numerals along with the length of the sequence of the characters that form the plaintexts. FPE has been staging in applied cryptography community due to the desirable functionality. It secures data while keeping the database scheme or communication protocols intact. For instance, given a legacy database system, upgrading the database security requires a way for encrypting credit card numbers (CCN) or social security numbers (SSN) in a way transparent to its applications.

Brightwell and Smith [BS97] introduced a first known format-preserving encryption which was termed as *data-type preserving encryption* in 1997. They wanted to encrypt an existing database to let all the applications access encrypted data just as they accessed non-encrypted data. Their solution for this was reduced to preserve the particular datatype of entries in the databases. The term *format-preserving encryption* is due to Terence Spies from Voltage Security [Spi08]. Though FPE dates back to late 90's, the demand to make FPE-based databases has created an active area of research during last few years. There have been many techniques proposed to build FPE schemes such as prefix cipher, cycle walking, Feistel network, Feistel modes [AB96, BRRS09, BRS, BR02, Luc96, Spi08, SK96]. The complete list of FPE schemes for small domain size along with their description and their security level can be found in a synopsis by Rogaway [Rog, p. 6,7]. In his list, Rogaway considers the schemes that are built with pseudorandom functions (that itself might be constructed from block ciphers).

FPE is specifically designed to work on small domain sizes such as four-digit PIN codes (with domain size 10^4) or nine-digit decimals of SSNs (with domain size $10^9 \approx 2^{30}$) or CCN (with domain size $10^{16} \approx 2^{54}$). In these cases, conventional block ciphers such as AES are somewhat rigid to use because of the small domain size which is typically smaller than the set of 128-bit blocks. Additionally, FPE with small domain sizes are vulnerable to dictionary attacks as encrypting the same message twice will generate the same ciphertext under the same secret key. Tweakable encryption has been used to avoid dictionary attacks in FPE.

3.2 Tweakable Encryption

A tweakable block cipher (TBC) is a tuple $(\mathsf{Gen}(\mathcal{K}), \mathcal{E}_{\mathsf{K}}(\cdot,\cdot), \mathcal{D}_{\mathsf{K}}(\cdot,\cdot))$ formed of three algorithms for key generation, encryption, and decryption with a key K; all efficiently computable algorithms. A TBC is defined over a key space $\mathcal{K}$, a message space $\mathcal{X}$, and tweak space $\mathcal{T}$. Namely, for every key $\mathsf{K} \in \mathcal{K}$, and every tweak $\mathsf{T} \in \mathcal{T}$, the algorithms $\mathcal{E}_{\mathsf{K}}(\mathsf{T},\cdot), \mathcal{D}_{\mathsf{K}}(\mathsf{T},\cdot)$ are inverse permutations over the domain $\mathcal{X}$.

In the standard model, the tweakable block ciphers [LRW11, BRRS09] are used to construct tweakable format-preserving encryption schemes since tweakable encryptions provide better resistance to dictionary attacks.

A Tweakable Format-Preserving Encryption (TFPE) scheme is a block cipher that preserves the format of the domain in the output. A TFPE function $\mathsf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \mapsto \mathcal{X}$ is defined from a key space $\mathcal{K}$, a tweak space $\mathcal{T}$, and a domain $\mathcal{X}$ to the same domain $\mathcal{X}$ where the domain is defined with *format* $\mathcal{F}$ (it is specified with set of characters and the length of messages formed with these characters). A decryption function is the inverse of the TFPE function E under the same key K and tweak T.

The security of TFPE varies under the situations when the adversary is given chosen-plaintext (CPA), chosen-ciphertext (CCA), or chosen-plaintext and ciphertext (CPCA) power. The aim of the adversary is typically to be able to decrypt a challenge message without querying it to the decryption oracle in the case of CCA and CPCA.

One way to construct TFPE is to use a Feistel network with tweakable round functions. In the next section, we will describe Feistel networks along with two Feistel-based TFPE schemes that are NIST and ANSI approved standards.

3.3 Feistel Network

A common way to construct a family of cryptographically secure functions consists of iterating over a relatively simple function r times. Each iteration is called a round and the resulting system is called an r-round cipher. Most widely used iterating systems are Substitution-Permutation-Networks (SPN) and Feistel Networks (FN). The focus of this chapter is Feistel networks that have been used in constructing many block ciphers such as DES [DES], FEAL [SM88], Blowfish [Sch94], BEAR and LION [AB96], RC5 [Riv95], Camellia [AIK+00], and Twofish [Fer99].

The Feistel network was named after Dr. Horst Feistel in IBM labs in 1973. The classical FN gives a way to construct a permutation over $\{0,1\}^{2n}$ with round functions over $\{0,1\}^{n}$. We call it a balanced Feistel network. The internal round functions do not have to be invertible, yet the decryption of Feistel network can still be defined. Fig. 3.1 (a) represents a 4-round FN with modular addition. Other well known types of Feistel networks are unbalanced FN, alternating between contracting and expanding round functions. An unbalanced Feistel network uses a random round function from

$\mathtt{n}_l$ bits to $\mathtt{n}_r$ bits and from $\mathtt{n}_r$ bits to $\mathtt{n}_l$ bits in order to obtain a permutation from $\mathtt{n}_l + \mathtt{n}_r$ bits to $\mathtt{n}_l + \mathtt{n}_r$ bits. One of the famous unbalanced FN construction was given in BEAR and LION [AB96] by Anderson and Biham.

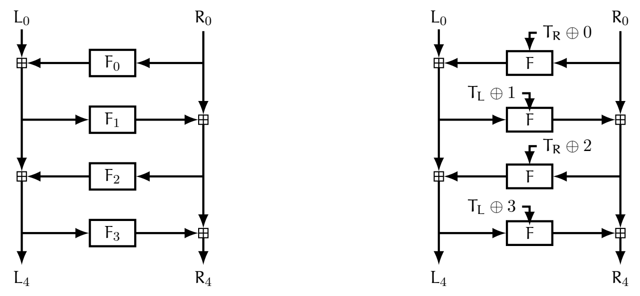
As we will not necessarily assume messages in binary, we use the notation $\mathsf{N}_l, \mathsf{N}_r$ as the domain size of the round functions. We are specifically interested in the Feistel network with the following properties: two branches with domain size N_l and N_r, with modular addition modulo N_l and N_r, secret random round functions which are balanced ($\mathsf{N} = \mathsf{N}_l = \mathsf{N}_r$) or unbalanced but with $\mathsf{N}_l \approx \mathsf{N}_r$. Moreover, we are interested in small domain size. The FN we consider in this work are typically (assumed to be) indistinguishable from a truly random function. More precisely, we investigate the FN when the round function is entirely unknown instead of a publicly known round function that mixes the input with a secret key (i.e. round function is $\mathsf{F}_\mathtt{i} = \mathtt{f}_\mathtt{i}(\mathtt{k}_\mathtt{i}, .)$, where $\mathtt{k}_\mathtt{i}$ is the round key in $\mathtt{i}^{\mathtt{th}}$ round). We do not assume that round functions are bijective. The Feistel networks with these properties are used to build various tweakable format-preserving encryption schemes [BRRS09, BRS, BPS]. Among these constructions, FF1 [BRS] by Bellare et al. and FF3 [BPS] by Brier et al. were standardized by NIST published in March, 2016 [NIS16]. Furthermore, the AEZ authenticated encryption [HKR17] includes modes with very small domains based on FN with these properties, as well.

3.3.1 Feistel-based Tweakable Format-Preserving Encryption

Probably, it is natural to build FPE schemes based on a Feistel network since it can be used with already existing conventional block ciphers, such as AES [AES01].

We are particularly interested in the TFPE schemes named FF1 and FF3 in NIST standards. Their designs are based on the Feistel network depicted in Fig. 3.1 (a).

We use the following notations for the rest of the work. The domain $\mathcal{X}$ consists of strings of characters; $\mathtt{s}$ represents the cardinality of the set $\mathtt{S}$ of characters and $\mathtt{b}$ represents the length of the messages in the domain $\mathcal{X}$ whose size is $\mathtt{s}^\mathtt{b}$. For example, the credit card numbers (CCNs) consists of 16 digits of decimal numerals with $\mathtt{S} = \{0, 1, \ldots, 9\}$, $\mathtt{s} = 10$ and $\mathtt{b} = 16$ where we have $10^{16} \cong 2^{54}$ possible distinct numeral



(a) Feistel Network (b) FF3 Encryption

Figure 3.1: 4-round Feistel Network and FF3 Encryption

strings. We set the minimum length of the message block $\mathtt{minlen} = 2$ and the maximum length of the message block to $\mathtt{maxlen} < 2^{32}$ in FF1 and $\mathtt{maxlen} = \lfloor \log_s(2^{f-32}) \rfloor$ in FF3, where f is the input/output size of the round function used in Feistel scheme (typically, $f = 128$). Both in FF1 and FF3, we set the lower bound $s^b \geqslant 100$. We represent the number of rounds in the scheme with w.

Unlike standard Feistel schemes which use the exclusive or (XOR) (denoted by $\oplus$), FF1 and FF3 use the modular addition that is denoted by $\boxplus$.

We define the following notations for three functions:

$\mathbf{STR}_s^b$ **:** a function that maps an integer x where $0 \leqslant x < s^b$ to a string of length b in base s with most significant character first, e.g. $\mathsf{STR}_{12}^4(554) = 03A2$.

$\mathbf{NUM}_s$ **:** a function that maps a string X to an integer x such that $\mathsf{STR}_s^b(x) = X$. For instance, $\mathsf{NUM}_2(00011010) = 26$.

$\mathbf{REV(X)}$ **:** a function that reverses the order of the characters of string X.

$\mathbf{PRF_K(X)}$ **:** Cipher Block Chaining Encryption mode on the input string X and returns the final block of the ciphertext.

The length of string X is denoted by $|X|$. The concatenation of strings is denoted by $\|$. The first (left-most) character of string X is $X[0]$. The i^{th} one is $X[i-1]$. We denote $X[i \cdots j]$ the substring of X formed with $X[i]X[i+1]\cdots X[j]$.

3.3.2 The FF3 Standard

The FF3 uses a tweakable block cipher as a round function, $\mathsf{F_K}(\mathsf{T},\mathsf{X}) = \mathsf{Y}$ with $\mathsf{X},\mathsf{Y} \in \{0,1,\ldots,2^f-1\}$ and $\mathsf{T} \in \{0,1\}^{32}$, where K is a key and T is one half of the FF3 tweak with an offset. The FF3 encryption algorithm is given in Algorithm 1.

Algorithm 1: FF3 Encryption

```
   Input  : string $X$ in base $s$ of length $b$ such that $b \in [minlen \cdots maxlen]$, a
            tweak bit string $T$ such that $|T| = 64$.
   Output: string $Y$ such that $|Y| = b$
 1 Let $\ell = \lceil \frac{b}{2} \rceil$; $r = b - \ell$.
 2 Let $L_0 = NUM_s(REV[X[1 \cdots \ell]])$ and $R_0 = NUM_s(REV[X[\ell+1 \cdots b]])$
 3 Let $T_L = T[0 \cdots 31]$ and $T_R = T[32 \cdots 63]$
 4 foreach $i = 0 \cdots w-1$ do
 5     if $i$ is even then
 6         $L_{i+1} = L_i \boxplus F_K(T_R \oplus STR_2^{32}(i), R_i) \pmod{s^\ell}$
 7         $R_{i+1} = R_i$
 8     end
 9     else
10         $R_{i+1} = R_i \boxplus F_K(T_L \oplus STR_2^{32}(i), L_i) \pmod{s^r}$
11         $L_{i+1} = L_i$
12     end
13 end
14 return $REV[STR_s^\ell(L_w)] \| REV[STR_s^r(R_w)]$
```

In lines 1-2, the encryption algorithm splits the input X into two substrings L_0 and R_0. In lines 5-8 (respectively in lines 10-12), the algorithm first takes the tweak $\mathsf{T_R}$ (respectively $\mathsf{T_L}$) XORed with the encoded round index i and $\mathsf{R_i}$ (respectively $\mathsf{L_i}$) to input tweakable PRF $\mathsf{F_K}$. Second, it applies modular addition of the output of $\mathsf{F_K}$ to $\mathsf{L_i}$ (respectively $\mathsf{R_i}$).

For simplicity and by abuse of notations, we say that FF3 encrypts the plaintext $(\mathsf{L}_0, \mathsf{R}_0)$ into the ciphertext $(\mathsf{L}_w, \mathsf{R}_w)$ with tweak $(\mathsf{T_L}, \mathsf{T_R})$, so that we only concentrate on lines 4-14. We illustrate the 4-round FF3 scheme in Fig. 3.1 (b).

In the concrete proposal, $w = 8$, $f = 128$ and

$$\mathsf{F_K}(\mathsf{T},\mathsf{X}) = \mathsf{NUM}_2(\mathsf{AES_K}(\mathsf{T} \parallel \mathsf{STR}_2^{f-32}(\mathsf{X})))$$

where AES maps an f-bit bitstring to an f-bit bitstring [NIS16].

The FF3 construction is an 8-round FN that uses a tweak XORed with a round counter as an input to the block cipher. The XOR operation guarantees that round functions are pairwise different. This is usually called "domain separation". The security of FF3 asserts that it achieves several cryptographic goals including chosen-plaintext security or even pseudo-random-permutation (PRP) security against an adaptive chosen-plaintext and ciphertext attack under the assumption that the underlying round function is a good pseudorandom function (PRF). In Chapter 5, we show that its security goal has not been met even when the round functions are replaced by secure PRFs and we give a round-function-recovery attack on FF3.

3.3.3 The FF1 Standard

We present the FF1 tweakable FPE standard in Algorithm 2 with the same notation as in FF3. In this standard, $w = 10$.

Algorithm 2: FF1 Encryption

```
Input  : string $X$ in base $s$ of length $b$ such that $b \in [2 \cdots 2^{32} - 1]$, a tweak bit
         string $T$ such that $|T| = t$.
Output: string $Y$ such that $|Y| = b$
1  Let $\ell = \lceil \frac{b}{2} \rceil$; $r = b - \ell$.
2  Let $L_0 = \mathsf{NUM}_s([X[1 \cdots \ell])$ and $R_0 = \mathsf{NUM}_s(X[\ell + 1 \cdots b])$.
3  Let format of domain be parameterized as $\mathsf{frmt} = s, b, t$.
4  foreach $i = 0 \cdots w - 1$ do
5      if $i$ is even then
6          $L_{i+1} = L_i \boxplus \mathsf{NUM}_2(\mathsf{PRF}_K(\mathsf{frmt}, i, T, R_i)) \pmod{s^\ell}$
7          $R_{i+1} = R_i$
8      end
9      else
10         $R_{i+1} = R_i \boxplus \mathsf{NUM}_2(\mathsf{PRF}_K(\mathsf{frmt}, i, T, L_i)) \pmod{s^r}$
11         $L_{i+1} = L_i$
12     end
13 end
14 return $\mathsf{STR}_s^\ell(L_w) || \mathsf{STR}_s^r(R_w)$.
```

Chapter 4

Generic Attacks On Feistel Network

In this chapter, we investigate the security of a generic Feistel Network. We consider FN with truly random functions, balanced, with two branches, and a group operation which is not necessarily the XOR. We start with already existing results in Section 4.1 and give our new generic attacks in Section 4.2 (on four rounds and more) and Section 4.2.4 based on optimized bruteforce. Each section includes attacks: 1) for FN with small number of rounds with "small" complexity and some of them have straightforward extensions for larger number of rounds with much bigger complexity 2) for FN with arbitrary number of rounds when the complexity is exponential with respect to the number of rounds. Given all these attacks on FN, our results conclude that the FF1 and FF3 standards with very low parameters (i.e. the domain size) do not offer the expected security.

All the work presented in this chapter has been done with Prof. Serge Vaudenay from Ecole Polytechnique Fédérale de Lausanne. Part of the studies described here were published in the proceedings ESC'17 [DVa], and the proceedings of Crypto'17 [DVb].

4.1 Existing Security Results of Feistel Network

Since its invention, Feistel networks and their security analysis have been studied widely. Many cryptanalytic studies have been done to outline key-recovery, message-recovery, round-function-recovery, and differential attacks on different types of Feistel networks [BLP16, DDKS15, PNB06, IS13, HR10, NVP13].

The most famous security results which date back to late 80's, are from Luby-Rackoff [LR88]. Luby and Rackoff first show that a three round Feistel construction is a pseudorandom permutation from $2n$ bits to $2n$ bits. In their seminal paper, Luby and Rackoff showed that for more than three rounds FN, all generic CPA attacks on Feistel schemes require $q = \Omega(2^{\frac{n}{2}})$ queries where n is the input/output size to the round function. Information theoretically, the number q of queries provides $2qn$ bits of information. For r-round FN, we need $rn2^n$ bits of information to recover the round functions (each round function can be represented with a string of size $n2^n$). Therefore, $q = \frac{r}{2}2^n$ may be enough to reconstruct the round function, in theory. Patarin [Pat10] further showed that for $q \ll 2^n$, four rounds are secure against known-plaintext attacks (the advantage would be bounded by $\frac{4q}{2^n} + \frac{q^2}{2 \cdot 2^n}$ for $q \leqslant \frac{2^n}{67n}$), five rounds are secure against chosen-plaintext attacks (the advantage would be bounded by $\frac{5q}{2^n} + \frac{q^2}{2 \cdot 2^n}$ for $q \leqslant \frac{2^n}{67n}$) and six rounds are secure against chosen-plaintext and ciphertext attacks (the advantage would be bounded by $\frac{8q}{2^n} + \frac{q^2}{2 \cdot 2^n}$ for $q \leqslant \frac{2^n}{128n}$). This suggests that there exists no distinguisher with given bounds, hence there exists no stronger attack with given bounds.

First of all, we observe that the round functions do not uniquely define the codebook. That means we can find a set of equivalent round functions that gives the same codebook as the"true" round functions. Namely, if the tables of a tuple $(F_0, \dots, F_{r-1})$ maps entire plaintext space to the corresponding ciphertext space, then we can construct tables of a set of many other tuples that are equivalent to the tables of "true" round functions. Indeed, for any set of values $\alpha_0, \dots, \alpha_{r-1}$ such that $\alpha_1+\alpha_3+\alpha_5+\cdots = \alpha_0+\alpha_2+\alpha_4+\cdots = 0$, we can define

$$F'_j(u) = F_j(u - \alpha_{j-1} - \alpha_{j-3} - \alpha_{j-5} - \cdots) + \alpha_j$$

for all j and u to obtain an equivalent tuple of round functions. As the "true" round functions will always be defined up to $\alpha_0, \dots, \alpha_{r-1}$, we can fix one point arbitrarily in $F_0, \dots, F_{r-3}$ when looking for recovery of tables for each round function. Therefore, we define the round-function-recovery as recovering the tables of one of the equivalent tuples of round functions in a Feistel network. It is enough to uniquely reconstruct the entire codebook of the FN. We will exploit this in order to give round-function-recovery

attack method	**attack type/goal**	**requirement**	**time complexity T**	**data complexity q**	**ref**
yo-yo	known pt, $r = 3$		$O(N \ln N)$	$N \ln N$	[DVb]
cycle finding	known pt, $r = 4$		$O(N^3)$	$N^{\frac{3}{2}}$	[DVb]
guess and determine	chosen pt, $r = 4$		$O\left(N^{\frac{3}{2}}\right)$	$N^{\frac{3}{2}}$	[BLP16]
cycle finding	chosen pt, $r = 5$		$O\left(N^{\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[DVb]
integral attack	chosen pt, $r = 5$	F_1 or F_3 permutation	$O(N^{2.81})$	N^2	[BLP16]
yo-yo	full codebook, $r = 5$	$\oplus$-Feistel	$O(N^2)$	N^2	[BLP16]
guess and determine	full codebook, $r = 5$		$O\left(N^{N^{\frac{3}{4}}}\right)$	N^2	[BLP16]
SAT solver	full codebook, $r \leqslant 5$		not specified	N^2	[BP15]
yo-yo	full codebook, $r = 6$	$\oplus$-Feistel	$O\left(N^{\frac{1}{2}N}\right)$	N^2	[BLP16]
yo-yo	full codebook, $r = 7$	$\oplus$-Feistel	$O(N^N)$	N^2	[BLP16]
cycle finding	chosen pt		$O\left(N^{(r-5)N+\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[DVb]
MITM	known pt		$O\left(N^{\lceil\frac{r}{2}\rceil N}\right)$	$r\frac{N}{2}$	Eq. (4.1), Sec. 4.1.2
impr MITM	chosen pt		$N^{\frac{r-4}{2}N(1+o(1))}$	$r\frac{N}{2}$	Eq. (4.2), Sec. 4.1.3
random partial exhst search	known pt		$N^{\frac{r(r-2)}{r-1}N\left(\frac{N}{q}\right)^{\frac{1}{r-1}}(\beta+o(1))}$	$q < N^2$	Eq. (4.6), Sec. 4.2.4.1
random partial exhst search	chosen pt		$N^{(r-1)N^{1-\frac{1}{r-2}}(\beta+o(1))}$	$\beta N^{2-\frac{1}{r-2}}$	Eq. (4.9), Sec. 4.2.4.1
random partial exhst search	chosen pt		$N^{\frac{r(r-2)}{r-1}N\left(\frac{N}{q}\right)^{\frac{1}{r-2}}(\beta+o(1))}$	$q < N^{2-\frac{1}{r-2}}$	Eq. (4.7), Sec. 4.2.4.1
iterated partial exhst search	known pt		$N^{(r-2)N\left(\frac{N\ln N}{q}\right)^{\frac{1}{r-2}}(\beta+o(1))}$	$N \ln N \ll q \leqslant N^2$	Eq. (4.12), Sec. 4.2.4.7
iterated partial exhst search	chosen pt		$N^{(r-2)N^{1-\frac{1}{r-2}}(\ln N)^{\frac{1}{r-2}}(\beta+o(1))}$	$\beta N^{2-\frac{1}{r-2}}(\ln N)^{\frac{1}{r-2}}$	Eq. (4.15), Sec. 4.2.4.7
iterated partial exhst search	chosen pt		$N^{\frac{q}{N}-1+(r-3)N\left(\frac{N\ln N}{q}\right)^{\frac{1}{r-3}}(\beta+o(1))}$	$N \ln N \ll q \leqslant N^2$	Eq. (4.14), Sec. 4.2.4.7

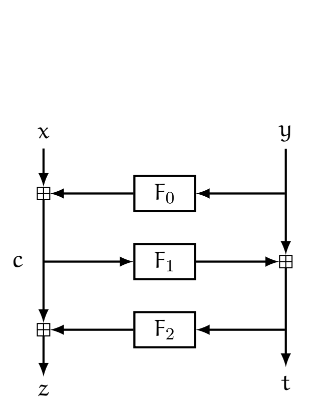
Table 4.1: Function Recovery attacks against generic balanced 2-branch r-round FN with N branch domain size. (All β are different constants such that $\beta < 1$.)

attack against generic Feistel network. Moreover, we aim to do it for query complexity less than the domain size since it is trivial to reconstruct the codebook with the query complexity equal to domain size.

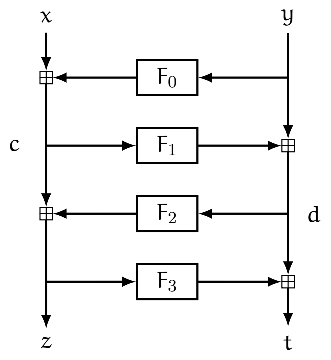
We summarize the best function recovery attacks in Table 4.1.[1] The complexities are given in terms of number of encryption.

For the rest of this work, we define the Feistel network over a group of order N^2. Typically, this group is $\mathbb{Z}_N \times \mathbb{Z}_N$. It is due to fact that Feistel-based tweakable FPE can be defined over domains of integers. Without loss of generality, FPE schemes with integral domains are relatively easier to construct and generalize to any domains due

[1]Table 4.1 only reports function recovery attacks. It does not include attacks applying with round functions in a small space of N (instead of N^N). It does not include distinguishers such as the ones from Patarin [Pat08] either.



(a) 3-round Feistel network



(b) 4-round Feistel network

Figure 4.1: 3-round and 4-round Feistel Schemes

to the ranking.

4.1.1 Generic Round-Function-Recovery Attack with Guess and Determine Method [BLP16]

In [BLP16], chosen-plaintext and ciphertext attacks are given for 4 and 5-round FN with modular addition. Their attack is based on a distinguisher for a 3-round FN introduced by Luby and Rackoff in [LR88]. For this distinguisher, refer to the Fig. 4.1 (a). Let the adversary have access to both encryption and decryption oracle. The adversary selects a δ, it queries the encryption oracle with arbitrary $(x\|y)$ and $(x+\delta\|y)$, and obtains $(z\|t)$ and $(z'\|t')$ respectively. Then, the adversary queries $(z+\delta\|t)$ to the decryption oracle and obtains $(x''\|y'')$. The distinguisher checks if $t-y''=t'-y$ to distinguish 3-round Feistel Network from a random permutation.

In 4-round attack given by Biryukov et al. (refer to the Fig. 4.1 (b)), consider a type of plaintext/ciphertext of the form $(xyzt)$, $(x''y''(z+\delta)t'')$, and $((x+\delta)yz't')$ with corresponding d, d, d' values in plaintexts/ciphertexts respectively. More precisely, the attacker starts with one arbitrarily fixed value of F_3 and one guessed value for F_3 (meaning that it iterates N times what follows). He sets z and $z+\delta$ such that their image by F_3 is known. For each d, it sets t and t''. With the queries to the decryption oracle for $(z\|t)$ and $(z+\delta\|t'')$, the attacker obtains $(x\|y)$ and $(x''\|y'')$. With a query to the encryption oracle on $(x+\delta\|y)$, it gets $(z'\|t')$. With all the obtained values, the attacker can use the 3-round property of the above defined distinguisher and can find an $F_3(z') = t' - d + y'' - y$ output of F_3 for a new value. The adversary iterates on d to get new outputs of F_3 until it finds no conflict. On average, the conflicts occur after $O(\sqrt{N})$ trials for d. The time complexity of this attack is $O\left(N^{\frac{3}{2}}\right)$ with $N^{\frac{3}{2}}$ data complexity.

The 5-round attack in Biryukov et al. is extended form of 3-round distinguisher and 4-round attack with more guesses. Its time complexity is $O\left(N^{N^{\frac{3}{4}}}\right)$ with N^2 data complexity.

4.1.2 Meet-In-The-Middle (MITM) Attack

The MITM attack was introduced by Diffie and Hellman [DH77]. MITM is a generic known-plaintext attack. More specifically, if the system includes multiple encryption schemes with independent keys, we can decompose the system into two smaller parts where we can attack separately and combine the results from each part. The idea is to evaluate smaller systems that require less complexity, reducing the complexity of the overall system dramatically. For instance, the MITM attack can recover the key for iterating systems such as 3-DES.

Briefly, consider an r round encryption $E_1, E_2, \ldots, E_r$ and corresponding $D_1, D_2, \ldots,$ D_r decryption algorithms with keys $K_1, K_2, \ldots, K_r$ of length k. Let $P_1, P_2, \ldots, P_q$ be the plaintexts of length n that we apply r-round encryption that produces $C_1, C_2, \ldots, C_q$. Let the intermediate values after i^{th} round be $P_1^i, P_2^i, \ldots, P_q^i$ for $1 \leqslant i < r$. The adversary enumerates each possible combination of the keys $K_1, K_2, \ldots, K_u$ for the first $u = \lfloor \frac{r}{2} \rfloor$ rounds and it computes the intermediate values for each plaintexts as $P_1^u, P_2^u, \ldots, P_q^u$ until u^{th} round. Then, these values along with their possible keys are stored in a table (The memory complexity is 2^{uk}). Then, the adversary partially decrypts the ciphertext $C_1, C_2, \ldots, C_q$ for each value of the keys $K_r, K_{r-1}, \ldots, K_{u+1}$ backward. Finally, the adversary looks for a match between the partially decrypted values and the rows of the stored table. Each match (there is only one) suggests keys for $K_1, K_2, \ldots K_r$ and the adversary recovers all the keys. The time complexity of the MITM attack is $2^{(r-u)k}$ and memory complexity is 2^{uk}.

In order to improve the memory complexity of MITM attack, a new technique called dissection attack has been introduced by Dinur et. al in [DDKS12]. The main idea behind this new technique is to guess the appropriate middle values in order to work from the middle towards the end points recursively (in MITM attack, we start with the end points by partially encrypting and decrypting them). Unfortunately, the time complexity goes worse than MITM attack when applied to Feistel networks since in FN, we can simply ignore the middle round in MITM attack while in dissection we still have to guess them. Later in 2015, Dinur et al. [DDKS15] showed a new technique that combines MITM attack with dissection attack for FN.

We can apply the MITM attack to the Feistel networks with r rounds and q known plaintext/ciphertext pairs. Since our domain size is much smaller than key size in FN, we are interested in recovery of round functions instead of key recovery for round functions. Therefore, the standard MITM attack is equivalent to setting $k = N\log_2(N)$ and the time complexity is $N^{(r-u)N}$ with same memory complexity. We label the time complexity as follows:

$$T^{MITM} = O\left(N^{\lceil \frac{r}{2} \rceil N}\right) \tag{4.1}$$

with $q = \frac{rN}{2}$ **known plaintexts**.

4.1.3 Improved MITM

In this section, we elaborate and extend the attack mentioned briefly in [DDKS12, DDKS15] on r-round FN. We take $u = \lceil \frac{r}{2} \rceil - 1$ and $v = \lfloor \frac{r}{2} \rfloor - 1$ so that $r = u + v + 2$ and $u \approx v$. Consider the FN in Fig. 4.2 for r even (When r is odd, we can set $u = \lfloor \frac{r}{2} \rfloor - 1$ so that $r - u - 2 = \lceil \frac{r}{2} \rceil - 1$). We can split the $(2u + 2)$- round FN in 4 parts: starting with a single round F_0; a u-round Feistel Network called G, whose output is an input to $(u + 2)^{th}$ round function F_{u+1}, and finally another v-round Feistel Network called H.

An intuitive attack works as follows. Fix a value $M_R^{(0)} = a$ for a packet of N plaintexts so that we need $\frac{q}{N}$ values for a. We set the output of F_0 for one value of a

arbitrarily. For all the values of M_L^0, we query $(M_L^0 \| a)$ and obtain N $(C_L \| C_R)$ values. We enumerate all the functions of H, and compute $(M_L^{(u+2)} \| M_R^{(u+2)})$ from $(C_L \| C_R)$ by decrypting. We set $Z = M_L^{(u+2)} = M_L^{(u+1)}$ if u is even and set $Z = M_R^{(u+2)} = M_R^{(u+1)}$ if u is odd. We store each Z in a hash table. We then enumerate all the functions of G, and compute $(M_L^{(u+1)} \| M_R^{(u+1)})$ from $(M_L^{(1)} \| M_R^{(1)})$. For each of the computed values of $M_L^{(u+1)}$ (for u even) or $M_R^{(u+1)}$ (for u odd), we look for a match in the hash table for stored Z values (since they have to be equal). The complexity of this approach is to enumerate $N^{\nu N}$ ν-round functions with memory complexity $\nu N \log_2(N)$ to store the hash table.

We can make the attack better by only guessing the 1-bit of intermediate value Z. The meet-in-the-middle is done only on one bit for each plaintext-ciphertext pair and all others are recovered by a yo-yo attack on 3 rounds [DVb] with complexity $O(N)$. More precisely, the attack works with $q = \frac{rN}{2}$ known plaintext/ciphertext pairs. We enumerate all the possible values for $M_L^{(0)}$ and pick $\frac{q}{N} = \frac{r}{2}$ arbitrary $M_R^{(0)} = a$. Among the $\frac{r}{2}$ possible $M_R^{(0)}$, we can fix one of them arbitrarily, therefore, we can guess only $\frac{r}{2} - 1$ outputs of F_0. The complete attack is given in Algorithm 3.

In this attack, we have to guess $N^{\frac{q}{N}-1}$ values for F_0, $N^{(u-1)(N-1)}$ values for enumerating $F_1, F_2, \ldots, F_{u-1}$ and 2^{N-1} values for F_u for a single bit (we guess $N^{(\frac{q}{N}-1)+(u-1)(N-1)}2^{(N-1)}$ values in total). And, we guess $N^{(\nu-1)(N-1)}$ values for enumerating $F_{u+3}, F_{u+4}, \ldots, F_{r-1}$, 2^N values for F_{u+2} (we guess $N^{(\nu-1)(N-1)}2^N$ in total). Therefore, the complexity is $O\left(N^{(\frac{q}{N}-1)+(\frac{r}{2}-2)(N-1)}2^{N-1}\right)$ for r is even and $O\left(N^{(\frac{r-1}{2}-1)(N-1)}2^{N-1}\right)$ for r is odd. We label the time complexity for the attack as:

$$\begin{aligned} T^{MITM^*} &= O\left(N^{(\frac{r}{2}-1)+(\frac{r}{2}-2)(N-1)}2^{N-1}\right), \quad \text{for } r \text{ even} \qquad (4.2)\\ T^{MITM^*} &= O\left(N^{(\frac{r-1}{2}-1)(N-1)}2^{N-1}\right), \quad \text{for } r \text{ odd} \end{aligned}$$

with $q = \frac{rN}{2}$ **chosen plaintexts**.

Algorithm 3: Improved MITM round-function-recovery attack

```
1  Pick $\frac{q}{N}$ arbitrary inputs $a$ of $F_0$ and consider the vector of $q$ plaintexts of the form $(x\|a)$.
2  Encrypt the plaintexts and get the vector of ciphertexts.
3  For the first plaintext-ciphertext, fix arbitrarily the output of each $F_i$ round function so that they match and remove this plaintext and ciphertext from the vectors.
4  For the round function $F_{u+3}, \dots, F_{r-1}$ of $H$, guess the outputs of remaining $N-1$ inputs. (Loop of $N^{(v-1)(N-1)}$.)
5  For $F_{u+2}$ in $H$ (that we did not enumerate), we guess only the first output bit (out of $\log_2 N$) of the output. (Loop of $2^{N-1}$.)
6  For each guess, compute the vector $Z$ of $q-1$ bits by partial decryption of $v$ rounds and store it in a hash table.
7  Guess the outputs of $F_0$ for the remaining $\frac{q}{N}-1$ values $a$. (Loop of $N^{\frac{q}{N}-1}$.)
8  For the round functions $F_1, F_2, \dots, F_{u-1}$ of $G$, guess the outputs of $N-1$ unset inputs of these functions. (Loop of $N^{(u-1)(N-1)}$.)
9  For $F_u$ in $G$ (that we did not enumerate), we guess only the first output bit (out of $\log_2 N$) of the output. (Loop of $2^{N-1}$.)
10 For each guess, compute the vector $Z$ of $q-1$ bits by partial encryption of $u+1$ rounds and check if it is in the hash table.
11 For each match, try to complete the tables of $F_u$, $F_{u+1}$, and $F_{u+2}$ by a yo-yo attack on 3 rounds.
```

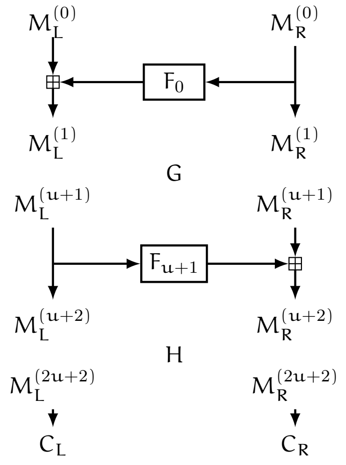


Figure 4.2: (2u+2)-round Feistel Network (with u even on the picture)

4.1.4 Message Recovery Attack [BHT16]

In their recent work by Bellare et al. in [BHT16] on FFX FPE schemes, they consider an FN scheme with round functions built as tweakable block ciphers. They gave a message recovery attack with data complexity larger than the domain size by using small number of messages per tweak. Basically, their attack is a differential attack that exploits the bias introduced on the left/right part of the input in Feistel networks. The idea of the bias they exploit was discovered by Patarin [Pat92]. Namely, consider two messages $M^{(0)} = (M_L^{(0)}, M_R^{(0)})$ and $M'^{(0)} = (M_L'^{(0)}, M_R^{(0)})$ as an input with same $M_R^{(0)}$ to the FN with modular addition under the same tweak. Let $M_L^{(i)}$ (resp. $M_L'^{(i)}$) be the output of left part of FN in i^{th} round. Then, we can show that $M_L^{(i)} - M_L'^{(i)}$ is most likely to be $M_L^{(0)} - M_L'^{(0)}$.

In [BHT16], more specifically, the authors consider two messages M and M' encrypted under FN, where they share the same right that is known to the adversary. In the attack, the adversary obtains the encryption of M and M' under q tweaks, the entire message M' and the shared half of the messages. At the end, the adversary outputs the unknown half of the message M with probability close to 1 by using a bias. The bias simply works as the following: Consider q pairs (M, C_i) and (M', C_i') for each tweak. Apply modular subtraction to the left part of the C_i and C_i' and modular addition to known left part under each tweak. The attack observes a value more likely than the others, and outputs this value as the unknown half of the message. The data complexity of the attack for an r round FN is $q = 24(\log(N) + 4)N^{(r-3)}$, where N is the input size of the right branch in FN. The time complexity is linear in q.

4.2 Our Generic Attacks on Feistel Network

The rest of the section is organized as follows: in Section 4.2.1, we give a heuristic attack for 3-round FN and analyze its time complexity. We report the ratio of success recovery in Fig. 4 with the parameters that the attack takes. In Section 4.2.2, we present an attack for the 4-round FN that leverages our 3-round attack. The correctness and further analysis is presented with formally stated lemmas. In Section 4.2.3, we expand our attack for five rounds or more and derive the time complexities. Finally, in Section 4.2.4, we define an exhaustive search algorithm dealing with partial functions in order to recover round functions for an arbitrary round number.

4.2.1 Round-Function-Recovery on 3-Round Feistel Scheme

Consider a 3-round Feistel Scheme with three round functions $\mathsf{F}_0, \mathsf{F}_1, \mathsf{F}_2$ and modular addition. Given x and y in $\mathcal{X}$, we define:

$$\begin{aligned} c &= x + \mathsf{F}_0(y), \\ t &= y + \mathsf{F}_1(c), \\ z &= c + \mathsf{F}_2(t). \end{aligned} \tag{4.3}$$

Due to the symmetry of the set of solutions $(\mathsf{F}_0, \mathsf{F}_1, \mathsf{F}_2)$ (as already observed), we can fix F_0 on one point arbitrarily. The idea of our attack is to concentrate on data for which we know how to evaluate F_0 so that we can deduce the output for the round function F_2. Then, we concentrate on data for which we know how to evaluate F_2 and we deduce more points in F_0. We continue by alternating the deduction between F_0 and F_2 until we recover them all. When we continue iterating as described, we can fully recover the tables for all three round functions $(\mathsf{F}_0, \mathsf{F}_1, \mathsf{F}_2)$. Our attack is presented in Algorithm 4 in more detail.

We model our set S as a bipartite graph with two parties of N vertices (one for the y's and the other for the t's) and edges for each (y, t) pair represented by tuples from S. What our algorithm just does is to look for a connected component of a random starting point y with complexity $O(\theta N)$. Following the theory of random graphs [Sal95], we need θN random edges so that the graph is likely to be fully connected when $\theta \approx \ln(N)$. For a constant $\theta \geqslant 1$, it is likely to have a giant connected component. This component corresponds to a constant fraction of the tables of F_0 and F_2. Therefore, after $\log_\theta N$ iterations, we can reconstruct F_0 and F_2 which allow us to reconstruct F_1. For any random y, we can see that it does not appear in S with probability $\left(1 - \frac{1}{N}\right)^{\theta N} \approx 1 - e^{-\theta}$.

Algorithm 4: (F_0, F_1, F_2) Recovery Attack

```
1 Collect a set $S$ of tuples $(xyzt)$ of size $\theta N$.
2 Take a subset $S_1 \subseteq S$ of size $\theta$ such that $y$ is constant in $S_1$.
3 Fix $F_0(y) = 0$ arbitrarily and deduce $\theta$ tuples $(cyzt)$ in $S_1$ by $c = x + F_0(y)$. We
  collect $\theta$ equations of the form $F_2(t) = z - c$.
4 Take the subset $S_2 \subseteq S$ of all $(xyzt) \in S$ such that $\exists(x'y'z't') \in S_1$ with $t = t'$.
  The expected size of $S_2$ is $\theta^2$.
5 Using the $\theta$ points of $F_2$, we deduce $\theta^2$ tuples $(xyct)$ by $c = z - F_2(t)$. From
  these tuples, we obtain $\theta^2$ equations of the form $F_0(y) = c - x$.
6 Take the subset $S_3 \subseteq S$ of all $(xyzt) \in S$ such that $\exists(x'y'z't') \in S_2$ with $y = y'$.
  The expected size of $S_3$ is $\theta^3$.
7 Using the $\theta^2$ points of $F_0$, we deduce from $\theta^3$ tuples $(cyzt)$...
8 We iterate through $S_1 \subseteq S_3 \subseteq S_5 \subseteq \cdots \subseteq S$ and $S_2 \subseteq S_4 \subseteq \cdots \subseteq S$ to complete
  the tables of $F_0$ and $F_2$.
```

Thus, we can only hope to recover a fraction $1 - e^{-\theta}$ of the table of F_0. The same holds for F_1 and F_2. **Therefore, with data and time complexity N, we recover a good fraction of all tables. With data and time complexity $N \ln N$, we recover the full tables with good probability.**

We implemented our attack. On Fig. 4.3, we plot the average fraction of recovered F_0 values depending on θ for several values of N. For this, we computed an average over 10,000 independent runs. For $\theta = 1$, the fraction is about 40%. We also plot the fraction of the trials which fully recovered all functions. These two values can be taken as an approximation of the expected fraction of recovered table for F_0 and the probability to fully recover all functions, respectively. As we can see, the first value does not depend so much on N (we have a giant connected component for θ around 1), but the second one jumps for θ proportional to $\ln N$ (the graph becomes fully connected). For $\theta = \ln N$, the probability is roughly $\frac{1}{3}$.

4.2.2 Round-Function-Recovery on 4-Round Feistel Scheme

In this section, we present an attack to fully recover the round functions of a 4-round Feistel scheme.

Consider a 4-round Feistel scheme with round functions F_0, F_1, F_2, F_3. Given x and y in $\mathcal{X}$, we define the following equations (see Fig. 4.4 (a)):

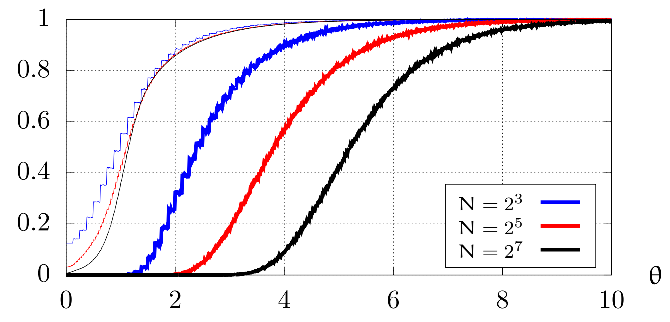


Figure 4.3: Fraction of recovered F_0 depending on θ in the 3-round attack (in thin) and fraction of experiments which fully recovered all functions (in bold) over 10,000 trials.
.

$$\begin{aligned}
c &= x + F_0(y),\\
d &= y + F_1(c),\\
z &= c + F_2(d),\\
t &= d + F_3(z).
\end{aligned}$$

Assume that we collected M random pairwise different plaintext messages (xy). We collect the pairs:

$$V = \{(xy, x'y') \mid z' = z, t' - y' = t - y, xy \neq x'y'\}$$

and,

$$V_{\mathsf{good}} = \{(xy, x'y') \mid z' = z, c' = c, xy \neq x'y'\}$$

where c, d, z, t (respectively c', d', z', t') are defined from (xy) (respectively form $(x'y')$) as above. We define $\mathsf{Label}(xy, x'y') = x - x'$.

We form a directed graph $G = (V, E)$ with the vertex set V as defined above. We take $(x_1y_1x_1'y_1', x_2y_2x_2'y_2') \in E$ if $y_1' = y_2$ (i.e. a pair of tuples $x_1y_1x_1'y_1'$ is connected to a pair $x_2y_2x_2'y_2'$ if the y_2 in the second message in former tuple is same as in the first message in latter tuple). Furthermore, we let $E_{\mathsf{good}} = (V_{\mathsf{good}} \times V_{\mathsf{good}}) \cap E$ and define

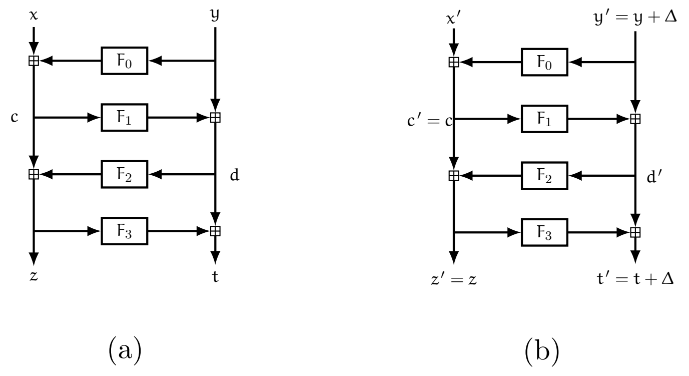


Figure 4.4: 4-round Feistel Scheme Attack

the sub-graph $G_{good} = (V_{good}, E_{good})$.

Then, we have the following Lemma with four properties:

Lemma 1. *Given a graph* G *with a vertex set* V *defined as above:*

1. $V_{good} \subseteq V$.

2. *If* $(xy, x'y') \in V$, *then* $y \neq y'$.

3. *If* $(xy, x'y') \in V_{good}$, *then* $F_0(y') - F_0(y) = \mathsf{Label}(xy, x'y')$.

4. *For all cycles* $v_1 v_2 \cdots v_L v_1$ *of* G_{good}, $\sum_{i=1}^{L} \mathsf{Label}(v_i) = 0$.[2]

Proof. The proofs are straightforward:

1. Clearly, $z' = z$ and $c' = c$ imply that $t' - y' = t - y$, hence $V_{good} \subseteq V$.

2. If $t' - y' = t - y$ and $y' = y$, then $t' = t$. If we further have $z' = z$, then we deduce $c' = c$. If $c' = c$, then $x' = x$, thus $xy = x'y'$. Hence, we cannot have $(xy, x'y') \in V$.

3. If $c' = c$ then $F_0(y') - F_0(y) = x - x' = \mathsf{Label}(xy, x'y')$.

4. Let $v_i = (x_i y_i, x'_i y'_i)$. If $v_i \in V_{good}$ then $F_0(y'_i) - F_0(y_i) = \mathsf{Label}(v_i)$. If we have a cycle then $y'_i = y_{i+1}$ with $y_{L+1} = y_1$. Hence, $\sum_i \mathsf{Label}(v_i) = 0$.

[2]Note that the cycle length notation L should not be confused with the subscript L indicating the left part of a plaintext or a ciphertext.

□

The principle of our attack is as follows: if we get vertices in V_{good}, the property 3 from Lemma 1 gives equations to characterize F_0. One problem is that we can identify vertices in V, but we cannot tell apart good and non-good (bad) ones. One way to recognize good vertices is to use property 4 in Lemma 1: to find cycles with zero sum of labels. For this, we will prove in Lemma 4 that this is a characteristic property of good cycles, meaning that all the vertices in these cycles are good vertices with high probability. First, we estimate the number of vertices and edges with the following two Lemma.

Lemma 2. *For* x, y, x', y' *random and* F_0, F_1, F_2, F_3 *random,*

$$\Pr[(xy, x'y') \in V_{good} \mid (xy, x'y') \in V] = \tfrac{1}{2-\frac{1}{N}} \approx \tfrac{1}{2}.$$

Proof. We compute the following probabilities:

$$\begin{aligned}
\Pr[xy, x'y' \in V_{good}] &= \Pr[z' = z, c' = c, x'y' \neq xy] \\
&= \Pr[z' = z, c' = c, y' \neq y] \\
&= \Pr[y' \neq y] \Pr[c' = c \mid y' \neq y] \Pr[z' = z \mid c' = c, y' \neq y] \\
&= \left(1 - \frac{1}{N}\right) \frac{1}{N^2}. \qquad (4.4)
\end{aligned}$$

$$\begin{aligned}
\Pr[xy, x'y' \in V \setminus V_{good}] &= \Pr[z' = z, t' - y' = t - y, c' \neq c, xy \neq x'y'] \\
&= \Pr[z' = z, d' - y' = d - y, c' \neq c, y' \neq y] \\
&= \Pr[y' \neq y] \Pr[c' \neq c \mid y' \neq y] \\
&\quad \Pr[d' - y' = d - y \mid y' \neq y, c' \neq c] \\
&\quad \Pr[z' = z \mid d' - y' = d - y, y' \neq y, c' \neq c] \\
&= \left(1 - \frac{1}{N}\right)\left(1 - \frac{1}{N}\right)\left(\frac{1}{N}\right)\left(\frac{1}{N}\right).
\end{aligned}$$

Hence,

$$\begin{aligned}\Pr[xy, x'y' \in V_{\mathsf{good}} \mid xyx'y' \in V] &= \frac{\Pr[xy, x'y' \in V_{\mathsf{good}}]}{\Pr[xy, x'y' \in V]} \\ &= \frac{1}{1 + \frac{\Pr[xy,x'y' \in V \setminus V_{\mathsf{good}}]}{\Pr[xy,x'y' \in V_{\mathsf{good}}]}} \\ &= \frac{1}{2 - \frac{1}{N}} \approx \frac{1}{2}.\end{aligned}$$

□

Lemma 3. *The expected number of elements in V_{good} is $\frac{M(M-1)\left(1-\frac{1}{N}\right)}{N^2} \approx \frac{M^2}{N^2}$.*

Proof. We have $M(M-1)$ possible pair of tuples $xy, x'y'$ with $xy \neq x'y'$ to construct V_{good}. From Eq. (4.4), the probability of each vertex in V_{good} is $\frac{1}{N^2}\left(1-\frac{1}{N}\right)$. Thus, we expect to have $\frac{M(M-1)\left(1-\frac{1}{N}\right)}{N^2} \approx \frac{M^2}{N^2}$ elements in V_{good}. □

We have the property that for each cycle $v_1v_2\cdots v_Lv_1 \in G$, if $v_1, \ldots, v_L$ are all in V_{good}, then the sum of $\mathsf{Label}(v_i)$ is zero due to Lemma 1, property 4. If one vertex is not good, the sum may be random. This suggests that a way to find good vertices in V is to look for long cycles in G with a zero sum of labels.

Lemma 4. *($L = 2$ case) If $v_1 = (x_1y_1, x_1'y_1')$ we say that v_1 and v_2 are permuting if $v_2 = (x_1'y_1', x_1y_1)$. If $v_1v_2v_1$ is a cycle in G with zero sum of labels, and v_1, v_2 are not permuting, then v_1 and v_2 are likely to be good. More precisely, for $v_1 = (x_1y_1x_1'y_1')$ and $v_2 = (x_2y_2x_2'y_2')$ random, we have $\Pr[v_1, v_2 \in V_{\mathsf{good}} \mid v_1v_2v_1$ is a cycle$, v_1, v_2$ not permuting, $\sum_{i=1}^{2} \mathsf{Label}(v_i) = 0] \geqslant \frac{1}{1+\frac{10}{N-5}}$.*

Proof. Before we start computations, we state the following definitions:

$[\mathsf{good}]$: the event that v_1 and v_2 are both in V_{good}.

$[\mathsf{bad}]$: the event that v_1 and v_2 are both in V but not both in V_{good}.

$[\mathsf{cyc}]$: the event that $y_1' = y_2$ and $y_2' = y_1$.

$[\mathsf{perm}]$: the event that $x_1y_1 = x_2'y_2'$ and $x_1'y_1' = x_2y_2$.

$[\Sigma = 0]$: the event that $\mathsf{Label}(v_1) + \mathsf{Label}(v_2) = 0$.

$[\#\{d\} = 4]$: the event that d_1, d_1', d_2, d_2' are pairwise different.

$[\#\{d\} = j]$: the event that there are exactly j pairwise different values among d_1, d_1', d_2, d_2'.

Let $p_{good} = \Pr[good, cyc, \neg perm, \Sigma = 0]$.

Let $p_{bad} = \Pr[bad, cyc, \neg perm, \Sigma = 0]$.

We are interested in $\Pr[good \mid cyc, \neg perm, \Sigma = 0] = \frac{1}{1+\frac{p_{bad}}{p_{good}}}$.

We want to upper bound $\frac{p_{bad}}{p_{good}}$. And, we start with the probability p_{good}.

Note that if $[good]$, we have $[\Sigma = 0]$ and it is equivalent to $[c_1 = c_1', c_2 = c_2', d_1 \neq d_1', d_2 \neq d_2', z_1 = z_1', z_2 = z_2']$. When $[c_1 = c_1', c_2 = c_2', cyc]$ holds, $[perm]$ is equivalent to $[c_1' = c_2]$. When $[c_1 = c_1', c_2 = c_2', y_1' = y_2]$ holds, $(d_1 - y_1) - (d_2' - y_2') = F_1(c_1) - F_1(c_2') = F_1(c_1') - F_1(c_2) = (d_1' - y_1') - (d_2 - y_2) = d_1' - d_2$. So, $y_1 = y_2'$ is equivalent to $d_1 - d_2' = d_1' - d_2$.

We let A be the event $[c_1 = c_1' \neq c_2 = c_2', \#\{d\} = 4, d_1 + d_2 = d_1' + d_2']$ which consists of only c and d. Picking the xy is equivalent to picking cd. So, A only depends on the c, d. We have $\Pr[A] \geqslant \frac{1}{N^3}\left(1 - \frac{1}{N}\right)^2\left(1 - \frac{3}{N}\right) \geqslant \frac{1}{N3}\left(1 - \frac{5}{N}\right)$ (We first pick c_1 and d_1, then $c_2 \neq c_1$, $d_1' \neq d_1$, and $d_2 \notin \{d_1, d_1', 2d_1' - d_1\}$). When A holds, $[y_1' = y_2]$ only depends on F_1 and occurs with probability $\frac{1}{N}$. When A holds, $[z_1 = z_1', z_2 = z_2']$ only depends on F_2 and occurs with probability $\frac{1}{N^2}$. Therefore,

$$
\begin{aligned}
p_{good} &= \Pr[good, cyc, \neg perm, \Sigma = 0] \\
&= \Pr[c_1{=}c_1'{\neq}c_2{=}c_2', d_1{\neq}d_1', d_2{\neq}d_2', d_1{+}d_2{=}d_1'{+}d_2', y_1'{=}y_2, z_1{=}z_1', z_2{=}z_2'] \\
&\geqslant \Pr[c_1{=}c_1'{\neq}c_2{=}c_2', \#\{d\}{=}4, d_1{+}d_2{=}d_1'{+}d_2', y_1'{=}y_2, z_1{=}z_1', z_2{=}z_2'] \\
&= \Pr_{c,d}[A] \Pr_{F_1}[y_1' = y_2 | A] \Pr_{F_2}[z_1 = z_1', z_2 = z_2' | A] \\
&\geqslant \frac{1}{N^6}\left(1 - \frac{5}{N}\right)
\end{aligned}
$$

Now, we compute the probability p_{bad}.

We know that $[bad]$ is equivalent to $[c_1 \neq c_1'$ or $c_2 \neq c_2', F_1(c_1) = F_1(c_1'), F_1(c_2) = F_1(c_2'), d_1 \neq d_1', d_2 \neq d_2', z_1 = z_1', z_2 = z_2']$. When $[cyc]$ occurs, $[\neg perm]$ is equivalent to $[c_1' \neq c_2$ or $c_1 \neq c_2']$. When $[F_1(c_1) = F_1(c_1'), F_1(c_2) = F_1(c_2')]$ holds, $[cyc]$ is equivalent to $[d_1 + d_2 = d_1' + d_2', y_1' = y_2]$. When $[cyc]$ holds, $[\Sigma = 0]$ is equivalent to $[c_1 + c_2 = c_1' +$

$c_2']$. So, when $[\mathsf{cyc}, \Sigma = 0]$ occurs, $[c_1 \neq c_1' \text{ or } c_2 \neq c_2']$ is equivalent to $[c_1 \neq c_1', c_2 \neq c_2']$.

From symmetry, the $[c_1' \neq c_2 \text{ or } c_1 \neq c_2']$ case is at most twice the $[c_1' \neq c_2]$ case. Let B be the event $[c_1 \neq c_1' \neq c_2 \neq c_2', c_1 + c_2 = c_1' + c_2', d_1 + d_2 = d_1' + d_2', d_1 \neq d_1', d_2 \neq d_2']$ which consists of only c and d. When B holds, $[F_1(c_1) = F_1(c_1'), F_1(c_2) = F_1(c_2'), y_1' = y_2]$ only depends on F_1. Therefore,

$$\begin{aligned}
p_{\mathsf{bad}} &= \Pr[\mathsf{bad}, \mathsf{cyc}, \neg\mathsf{perm}, \Sigma = 0] \\
&= \Pr[c_1{\neq}c_1', c_2{\neq}c_2', c_1'{\neq}c_2 \text{ or } c_1{\neq}c_2', c_1{+}c_2{=}c_1'{+}c_2', F_1(c_1){=}F_1(c_1'), F_1(c_2){=}F_1(c_2'), \\
&\qquad d_1{+}d_2{=}d_1'{+}d_2', d_1{\neq}d_1', d_2{\neq}d_2', y_1'{=}y_2, z_1{=}z_1', z_2{=}z_2'] \\
&\leqslant 2\Pr[c_1{\neq}c_1'{\neq}c_2{\neq}c_2', c_1{+}c_2{=}c_1'{+}c_2', F_1(c_1){=}F_1(c_1'), F_1(c_2){=}F_1(c_2'), d_1{+}d_2{=}d_1'{+}d_2', \\
&\qquad d_1{\neq}d_1', d_2{\neq}d_2', y_1'{=}y_2, z_1{=}z_1', z_2{=}z_2'] \\
&= 2\Pr_{c,d,F_2}[B, z_1{=}z_1', z_2{=}z_2'] \Pr_{F_1}[F_1(c_1){=}F_1(c_1'), F_1(c_2){=}F_1(c_2'), y_1'{=}y_2 | B, z_1{=}z_1', z_2{=}z_2'] \\
&= 2\Pr_{c,d,F_2}[B, z_1 = z_1', z_2 = z_2'] \times \frac{1}{N^3}
\end{aligned}$$

We split B following the $[\#\{d\} = j]$ cases for $j = 2, 3, 4$. Each case is denoted B_j. When we have $[d_1 \neq d', d_2 \neq d_2', \#\{d\} = 2, d_1 + d_2 = d_1' + d_2']$, we have either $[d_1 = d_2', d_1' = d_2]$ or $[d_1 = d_2, d_1' = d_2', d_1' = d_1 + \frac{N}{2}]$. When we have $[d_1 \neq d_1', d_2 \neq d_2', \#\{d\} = 3]$, we have $[d_1 = d_2 \text{ or } d_1' = d_2']$ (If we have $[d_1 = d_2' \text{ or } d_1' = d_2]$, then $d_1 + d_2 = d_1' + d_2'$ and $\#\{d\} = 2$ conflicts). When we have $[d_1 \neq d_1', d_2 \neq d_2', \#\{d\} = 4]$, we have no equality of d's. For B_4,

$$\begin{aligned}
&\Pr_{c,d,F_2}[B_4, z_1 = z_1', z_2 = z_2'] \\
= \ &\Pr_{c,d}[B_4] \Pr_{F_2}[z_1 = z_1', z_2 = z_2' | B_4] \\
= \ &\Pr_{c,d}[c_1{\neq}c_1'{\neq}c_2{\neq}c_2', c_1{+}c_2{=}c_1'{+}c_2', d_1{+}d_2{=}d_1'{+}d_2', \#\{d\}{=}4] \Pr_{F_2}[z_1 = z_1', z_2 = z_2' | B_4] \\
\leqslant \ &\Pr_{c,d}[c_1 + c_2 = c_1' + c_2', d_1 + d_2 = d_1' + d_2'] \Pr_{F_2}[z_1 = z_1', z_2 = z_2' | B_4] \\
= \ &\frac{1}{N^4}
\end{aligned}$$

For each of the two cases of B_3, either $z_1 = z_1'$ or $z_2 = z_2'$ occurs with probability $\frac{1}{N}$.

So,

$$\begin{aligned}
&\Pr_{c,d,F_2}[B_3, z_1 = z_1', z_2 = z_2'] \\
\leqslant\ & 2\Pr_{c,d}[c_1 + c_2 = c_1' + c_2', d_1 + d_2 = d_1' + d_2', d_1 = d_2] \Pr_{F_2}[z_1 = z_1' | d_1 \neq d_1'] \\
=\ & \frac{2}{N^4}
\end{aligned}$$

For B_2,

$$\begin{aligned}
&\Pr_{c,d,F_2}[B_2, z_1 = z_1', z_2 = z_2'] \\
\leqslant\ & \Pr_{c,d}[B_2] \\
=\ & \Pr_{c,d}[c_1 \neq c_1' \neq c_2 \neq c_2', c_1 + c_2 = c_1' + c_2', d_1 + d_2 = d_1' + d_1', d_1 \neq d_1', d_2 \neq d_2', \#\{d\} = 2] \\
=\ & \Pr_{c,d}[c_1 + c_2 = c_1' + c_2', d_1 + d_2 = d_1' + d_2', d_1 = d_2', d_1' = d_2'] + \\
& \Pr_{c,d}[c_1 + c_2 = c_1' + c_2', d_1 = d_2, d_1' + d_2', d_1' = d_1 + \tfrac{N}{2}] \\
=\ & \frac{2}{N^4}
\end{aligned}$$

Therefore, $\Pr_{c,d,F_2}[B, z_1 = z_1', z_2 = z_2'] \leqslant \frac{5}{N^4}$ and $p_{\mathsf{bad}} \leqslant \frac{10}{N^7}$.

Finally, $\frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}} \leqslant \frac{10}{N-5}$. We deduce

$$\Pr[\mathsf{good} \mid \mathsf{cyc}, \neg\mathsf{perm}, \Sigma = 0] \geqslant \frac{1}{1 + \frac{10}{N-5}}$$

$\square$

We give an extended version of Lemma 4 as follows:

Lemma 5. *If* $v_1 v_2 \cdots v_i \cdots v_L v_1$ *is a cycle of length* L *in* G *with zero sum of labels and the vertices use no* d_i *or* c_i *in common, then all* v_i *are likely to be good. More precisely, for* $v_i = (x_i y_i x_i' y_i')$ *random, we have*

$\Pr\left[\forall i, v_i \in V_{\mathsf{good}} \middle| v_1 \cdots v_i \cdots v_L v_1 \text{ is a cycle}, \left(\#\{c\} = \#\{c'\} = L, \forall i \neq j\ \ c_i \neq c_j'\right), \left(\#\{d\} = L, \forall i,j\ \ d_i \neq d_j'\right),\right.$
$\left.\sum_{i=1}^{L} \mathsf{Label}(v_i) = 0\right] \geqslant \frac{1}{1 + \frac{2^L - 1}{N}}.$

Proof. We compute $p = \Pr[\mathsf{good} \mid \mathsf{good} \vee \mathsf{bad}, \mathsf{cyc}, \neg\mathsf{repeat}_c, \neg\mathsf{repeat}_d, \Sigma = 0]$, where we use the same notation as in Lemma 4 with new $[\neg\mathsf{repeat}_c]$ and $[\neg\mathsf{repeat}_d]$ notations. We define them as follows:

We note that when all v_i are vertices (good or bad), since $F_1(c'_i) = F_1(c_i)$, $y'_{i+1} = y_i$ is equivalent to $d'_i - d_{i+1} = F_1(c_i) - F_1(c_{i+1})$. We further note that when this holds, then $\sum d_i = \sum d'$. To be able to compute the probability of $[\mathsf{cyc}]$, we introduce a condition on the non-repetition of the c and c', except for the possible equalities $c_i = c'_i$ in good vertices. Namely, we define

$$[\neg\mathsf{repeat}_c] : \left(\#\{c\} = \#\{c'\} = L \quad , \quad \forall i \neq j \quad c_i \neq c'_j\right)$$

When $[\neg\mathsf{repeat}_c, \sum d = \sum d']$ holds and all v_i are vertices, $[\mathsf{cyc}]$ occurs with probability $\frac{1}{N^{L-1}}$. Therefore, $\Pr[\mathsf{cyc} \mid \mathsf{good} \vee \mathsf{bad}, \neg\mathsf{repeat}_c, \Sigma d = \Sigma d'] = \frac{1}{N^{L-1}}$

The event $[\forall i \quad z_i = z'_i]$ is equivalent to $c_i + F_2(d_i) = c'_i + F_2(d'_i)$. To be able to compute its probability, we introduce a condition on the non-repetition of the d and d'. Namely, we define

$$[\neg\mathsf{repeat}_d] : \left(\#\{d\} = L \quad , \quad \forall i, j \quad d_i \neq d'_j\right)$$

Hence, when $[\neg\mathsf{repeat}_d]$ occurs, $[\forall i \quad z_i = z'_i]$ occurs with probability $\frac{1}{N^L}$: $\Pr[z' = z \mid \neg\mathsf{repeat}_d] = \frac{1}{N^L}$. Finally, when $[\mathsf{cyc}]$ holds, $[\Sigma = 0]$ is equivalent to $\Sigma(c - c') = 0$, and $[\mathsf{good} \vee \mathsf{bad}]$ is equivalent to $[F_1(c) = F_1(c'), z' = z]$.

We define

$$\begin{aligned} p_{\mathsf{good}} &= \Pr[c{=}c', \neg\mathsf{repeat}_c, \mathsf{cyc}, \neg\mathsf{repeat}_d, z'{=}z] \\ p_{\mathsf{bad}} &= \Pr\left[\neg(c{=}c'), F_1(c){=}F_1(c'), \textstyle\sum(c-c'){=}0, \neg\mathsf{repeat}_c, \mathsf{cyc}, \neg\mathsf{repeat}_d, z'{=}z\right] \end{aligned}$$

with obvious shorthands $[c = c']$, $[z' = z]$, $[F_1(c) = F_1(c')]$, $[\sum(c - c') = 0]$.

We upper bound $\frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}}$ to compute p.

We have

$$\begin{aligned}
p_{\mathsf{good}} &= \Pr[c = c', \neg\mathsf{repeat}_c, \mathsf{cyc}, \neg\mathsf{repeat}_d, z' = z] \\
&= \Pr\left[c = c', \neg\mathsf{repeat}_c \sum d = \sum d', \mathsf{cyc}, \neg\mathsf{repeat}_d, z' = z\right] \\
&= \frac{1}{N^{2L-1}} \Pr[c = c', \neg\mathsf{repeat}_c] \Pr\left[\sum d = \sum d', \neg\mathsf{repeat}_d\right] \\
&= \frac{1}{N^{3L-1}} \frac{N(N-1)\cdots(N-L+1)}{N^L} \Pr\left[\sum d = \sum d', \neg\mathsf{repeat}_d\right]
\end{aligned}$$

$$\begin{aligned}
p_{\mathsf{bad}} &= \Pr\left[\neg(c{=}c'), F_1(c){=}F_1(c'), \textstyle\sum(c-c'){=}0, \neg\mathsf{repeat}_c, \mathsf{cyc}, \neg\mathsf{repeat}_d, z'{=}z\right] \\
&= \Pr\left[\neg(c{=}c'), F_1(c){=}F_1(c'), \textstyle\sum(c-c'){=}0, \neg\mathsf{repeat}_c \sum d{=}\sum d', \mathsf{cyc}, \neg\mathsf{repeat}_d, z'{=}z\right] \\
&= \frac{1}{N^{2L-1}} \Pr\left[\neg(c{=}c'), F_1(c){=}F_1(c'), \textstyle\sum(c-c'){=}0, \neg\mathsf{repeat}_c\right] \Pr\left[\textstyle\sum d{=}\sum d', \neg\mathsf{repeat}_d\right]
\end{aligned}$$

So,

$$\begin{aligned}
\frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}} &= \frac{N^{2L}}{N(N-1)\cdots(N-L+1)} \Pr\left[\neg(c{=}c'), F_1(c){=}F_1(c'), \textstyle\sum(c-c'){=}0, \neg\mathsf{repeat}_c\right] \\
&= \frac{N^{2L}}{N(N-1)\cdots(N-L+1)} \sum_{I \neq \emptyset} \Pr\left[\begin{array}{l} \neg\mathsf{repeat}_c \\ \forall i \notin I \quad c_i{=}c'_i \\ \forall i \in I \quad c_i{\neq}c'_i, F_1(c_i){=}F_1(c'_i) \\ \sum_{i\in I}(c_i - c'_i){=}0 \end{array}\right] \\
&\leqslant \frac{N^{2L}}{N(N-1)\cdots(N-L+1)} \sum_{I \neq \emptyset} \Pr\left[\begin{array}{l} \neg\mathsf{repeat}_c \text{ except } c'_{\max I} \\ \forall i \notin I \quad c_i{=}c'_i \\ \forall i \in I \setminus \{\max I\} \quad c_i{\neq}c'_i, F_1(c_i){=}F_1(c'_i) \\ \sum_{i\in I}(c_i - c'_i){=}0 \\ F_1(c_{\max I}){=}F_1(c'_{\max I}) \end{array}\right] \\
&= \frac{N^{2L}}{N(N-1)\cdots(N-L+1)} \sum_{I \neq \emptyset} \frac{N(N-1)\cdots(N-L-\#I)}{N^{2L+\#I}} \\
&= \sum_{I \neq \emptyset} \frac{(N-L)(N-L-1)\cdots(N-L-\#I)}{N^{\#I}}
\end{aligned}$$

$$\leqslant \sum_{I \neq \emptyset} \frac{1}{N} = \frac{2^L - 1}{N}$$

where $\left[\neg \mathsf{repeat}_c \text{ except } c'_{\max I}\right]$ means

$$\begin{cases} \#\{c\} = L \\ \#\{c'_1, \dots, c'_{\max I - 1}, c'_{\max I + 1}, \dots, c'_L\} = L - 1 \\ \forall i \forall j \neq \max I \ i \neq j \implies c_i \neq c'_j \end{cases}$$

By relaxing the constraints on $c'_{\max I}$, we can compute the probability of $\Sigma(c - c') = 0$ conditioned to other events about c and c'. This probability is $\frac{1}{N}$.

Therefore,

$$\frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}} \leqslant \frac{2^L - 1}{N}$$

and we have

$$\frac{1}{1 + \frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}}} \geqslant \frac{1}{1 + \frac{2^L - 1}{N}}$$

□

We believe that Lemma 4 remains true for valid cycles of small length except in trivial cases. In Lemma 5, we extend to $L > 2$ for cycles satisfying some special non-repeating condition $[\neg \mathsf{repeat}]$ on the c and d values to rule out many trivial cases. However, this condition $[\neg \mathsf{repeat}]$ cannot be checked by the adversary. Instead, we could just avoid repetitions of any message throughout the cycle (as repeating messages induce repeating c's or d's). We use the following conjecture (which is supported by experiment for $L = 3$).

Conjecture 1. *If $v_1 v_2 \cdots v_L v_1$ is a cycle of length L in G with zero sum of labels and the vertices use no messages in common, then $v_1 \cdots v_L$ are all good with probability close to 1.*

For M known plaintexts, the expected number of valid cycles in G_{good} of a given length L is $\frac{M^{2L}}{N^{3L}}$.

The aim of our attack is to collect as many F_0 outputs as possible to reconstruct a table of this function. Thus, we are interested in vertices whose labels are defined as $\mathsf{Label}(v_i) = F_0(y) - F_0(y'), \forall i \in \{0, 1, \ldots, |V|\}$ and we generate another graph to represent the collection of many independent equations for F_0.

We have a valid cycle $v_1 v_2 \cdots v_L v_1$ of length L in G when $v_i \in V$,

$$\sum_{i=1}^{L} \mathsf{Label}(v_i) = 0$$

and vertices use no messages in common. Now, let us define an undirected graph $G' = (V', E')$, where $V' = \{0, 1, \ldots, N-1\}$ and E' is defined as follows: for each vertex $v_i = (xy, x'y')$ in a valid cycle $v_1 v_2 \cdots v_L v_1$ of length L, add $\{y_i, y_i'\}$ as an edge in E' with label set to $\mathsf{Label}(v_i)$. The purpose of such a graph G' is to put y values which are dependent on each other in a single connected component, and put apart with independent y values in separate connected components.

When we model G' as a random graph, we can adjust M so that we can have a large connected component in G'. Given the vertex set size $|V'| = N$ and the edge size $|E'| = m$, $m = \frac{N(N-1)}{2}p$, where p is the probability that G' has an edge between two vertices. From Erdős-Rényi model [ER59] on random graphs, we want $Np \geqslant 1$. We know that $Np \sim 2\frac{m}{N}$. So, we want $m \geqslant \frac{N}{2}$. We have $\frac{M^{2L}}{L \cdot N^{3L}}$ expected good cycles (counted without repetition of their L circular rotations) of length L, thus $m \sim \frac{M^{2L}}{N^{3L}}$. Therefore, we need to set $M = \lambda N^{\frac{3}{2}} \left(\frac{N}{2}\right)^{\frac{1}{2L}}$ for a constant $\lambda \geqslant 1$ to have a large connected component in G'. Our attack works with $M = N^{\frac{3}{2}+\epsilon}$ for $\epsilon > 0$ small, with complexity $O(2^L N^{(1+2\epsilon)L})$ and a constant probability of success. If our attack recovers at least $\sqrt{N}$ points in F_0 correctly (which is the case when we have a large connected component in G'), we obtain $M \times \frac{\sqrt{N}}{N} \gg N$ samples to apply the attack on 3-rounds so that it recovers a good fraction of F_1, F_2, F_3. It is enough to bootstrap a yoyo attack (Steps 9–18 of Algorithm 5), and our attack succeeds.

Now, we give the full algorithm of our attack to 4-round Feistel scheme.

Experimentally, we noticed that $\lambda = 0.8$ is too small to obtain a large enough connected component for $L = 3$. Conversely, for $\lambda = 2$, G' is more connected but the

Algorithm 5: (F_0, F_1, F_2, F_3) Recovery Attack (Strategy S_2)

```
1  Pick $M$ known plaintexts and retrieve their ciphertext.
2  Create $G = (V, E)$.
3  Find valid cycles of length $2, 3, \ldots, L$ and collect the vertices in these cycles.
4  Create $G'$ from $\{y, y'\}$ from the collected vertices.
5  Find the largest connected component in $G'$.
6  Assign one $F_0(y)$ value arbitrarily and deduce $F_0$ on the connected component.
7  For all known plaintexts using $y$ in the connected component, evaluate and
   deduce a tuple for the 3-round Feistel scheme based on $(F_1, F_2, F_3)$.
8  Apply the attack on 3-round Feistel scheme from Section 4.2.1 to recover a
   constant fraction of $(F_1, F_2, F_3)$.
9  while nothing more revealed do
10     foreach of the M plaintext/ciphertext pairs do
11         if $F_0$ and $F_1$ are known for this plaintext then
12             deduce one point for $F_2$ and $F_3$
13         end
14         if $F_2$ and $F_3$ are known for this ciphertext then
15             deduce one point for $F_0$ and $F_1$
16         end
17     end
18 end
```

giant component contains many bad edges that we want to avoid.

Let E_j be the event that the sizes of the j largest connected components sum to greater than $\sqrt{N}$ with no bad edges in G'. Let $E_{\leqslant j}$ be the event that either of $E_1, E_2, \ldots, E_j$ occurs. We simulated the attack for various values of N and $\lambda = 1, 2, 3$ and report the numbers for $E_{\leqslant 1}, E_{\leqslant 2}, E_{\leqslant 3}$ on Table 4.2. When we read the table, by taking $\lambda = 1$ and $j = 3$, our attack recovers $\sqrt{N}$ points of F_0 with probability at least 23 %. In our attack, if we look at j connected components, we need to multiply the complexity by N^{j-1} (We can fix F_0 on one point for free, then all values in its connected components are inferred, but for each additional connected component, we must guess one value of F_0). It is likely that we can mitigate this N^{j-1} factor by early abort during the attack on 3-rounds.

In our experiments, we observe better success probability of our attack with $\lambda = 1$. With λ larger, the attack hardly ever succeeds. It may look paradoxical to say that if λ is too large, then the attack fails, but this is due to higher chances to collect bad edges. However, when G' is heavily connected, we could propose algorithms to

N	$M(\lambda)$	#trials	$\Pr[E_{\leqslant 1}]$	$\Pr[E_{\leqslant 2}]$	$\Pr[E_{\leqslant 3}]$
2	2(0.71)	5022	0.00 %	0.00 %	0.00 %
4	5(0.56)	7098	1.51 %	1.51 %	1.51 %
8	15(0.53)	7010	0.36 %	4.07 %	4.07 %
16	46(0.51)	6665	0.05 %	1.23 %	1.23 %
32	144(0.50)	6103	0.02 %	0.03 %	0.16 %
64	457(0.50)	7986	0.00 %	0.00 %	0.01 %
128	1449(0.50)	7460	0.00 %	0.00 %	0.00 %
256	4598(0.50)	6879	0.00 %	0.00 %	0.00 %
512	14597(0.50)	6094	0.00 %	0.00 %	0.00 %
2	3(1.06)	4316	0.00 %	0.00 %	0.00 %
4	8(0.89)	4153	15.19 %	15.19 %	15.19 %
8	23(0.81)	6703	5.83 %	18.54 %	18.54 %
16	73(0.81)	6886	4.57 %	13.87 %	13.87 %
32	230(0.80)	6952	2.52 %	7.12 %	10.98 %
64	730(0.80)	6568	1.40 %	5.65 %	9.18 %
128	2318(0.80)	6189	0.29 %	1.13 %	2.83 %
256	7357(0.80)	8054	0.02 %	0.30 %	0.86 %
512	23355(0.80)	626	0.00 %	0.00 %	0.00 %
2	3(1.06)	4352	0.00 %	0.00 %	0.00 %
4	9(1.00)	3864	23.08 %	23.08 %	23.08 %
8	29(1.02)	5791	15.59 %	35.02 %	35.02 %
16	91(1.01)	6585	16.20 %	29.90 %	29.90 %
32	288(1.00)	6814	14.66 %	27.09 %	31.67 %
64	913(1.00)	6981	18.16 %	34.69 %	40.87 %
128	2897(1.00)	6609	16.31 %	33.53 %	40.73 %
256	9196(1.00)	6176	16.27 %	36.90 %	46.50 %
512	29193(1.00)	486	11.73 %	32.10 %	44.86 %
8	58(2.03)	988	22.77 %	23.99 %	23.99 %
16	182(2.01)	2504	6.71 %	6.79 %	6.79 %
32	575(2.00)	3425	0.53 %	0.55 %	0.55 %
64	1825(2.00)	5727	0.02 %	0.02 %	0.02 %
128	5793(2.00)	1948	0.00 %	0.00 %	0.00 %
256	18391(2.00)	125	0.00 %	0.00 %	0.00 %
512	58386(2.00)	7	0.00 %	0.00 %	0.00 %
32	863(3.00)	1389	0.00 %	0.00 %	0.00 %
64	2737(3.00)	2666	0.00 %	0.00 %	0.00 %
128	8689(3.00)	167	0.00 %	0.00 %	0.00 %
256	27586(3.00)	9	0.00 %	0.00 %	0.00 %

Table 4.2: Experimental $\Pr[E_{\leqslant j}]$ over several trials for various N, λ, and j;the number of trials correspond to the successful runs of the whole attack on FF3 in the first step out of 10 000 using $L = 3$.

eliminate inconsistencies in labels and get rid of bad edges. It means that we would have a successful attack for any $\lambda \geqslant 2$. We let it as future work.

Therefore, we have a double phase transition. The first phase transition occurs when we have enough data to be able to make the graph and find cycles. Our attack quickly succeeds after this phase transition. The second phase transition occurs when we start having bad edges in the collected cycles. Then, our attack must be enriched to be able to work any longer. We did not do it on purpose as we noticed there is a sufficient window in between these two phase transitions to break the scheme with good probability of success without caring about possible bad edges.

In Table 4.3, we show the experimental results of success probability of the entire attack for various strategies. Let S_j be an event with strategy j. In S_1, we accumulate the three largest connected components and abort, unless the accumulated size is at least $\sqrt{N}$ and they have no bad edges, i.e., S_1 is exactly $E_{\leqslant 3}$. In S_2, we just look at the largest connected component and fail unless it has no bad edges in G' (we remove the condition on size of the connected component that is greater than $\sqrt{N}$). In S_3 (and S_4 resp.), we look at the two largest (three largest resp.) connected components that have no bad edges. What we report in Table 4.3 includes the success probability $\Pr_{\mathsf{succ}}$ of S_i and we recover the entire tables for each round function. These various strategies are considered for experimental purpose even though we have theoretical results that suggests the condition on the size of the connected component.

The data complexity of our attack in Algorithm 5 is $M = O(N^{\frac{3}{2}+\frac{1}{2L}})$. We compute the time complexity for the algorithm based on the step 2, 3, 4, and 5, as the other steps are much shorter. In step 2, creating our graph G is defined as forming the vertices in G. This can be done in $M\log(M)$ time with collision detection for M known plaintext/ciphertext pairs. In step 3, we look for the cycles of length L. The cycles of length L in our graph can be found with multiplication in an adjacency matrix (which is sparse). Matrix multiplication can be done in $O(|V|^2 d)$ where $d = \frac{|E|}{|V|}$ is the average degree of a vertex. Therefore, the complexity is $O(|V||E|)$. With the Floyd-Warshall algorithm, we need $(L-1)$ multiplications by the adjacency matrix in the max-plus algebra that leads us to a complexity $O(L|V||E|)$. With $|E| \sim \frac{|V|^2}{N}$, where

N	$M(\lambda)$	#trials	$\Pr[\mathsf{succ}, S_1]$–$(\Pr[S_1])$	$\Pr[\mathsf{succ}, S_2]$–$(\Pr[S_2])$	$\Pr[\mathsf{succ}, S_3]$–$(\Pr[S_3])$	$\Pr[\mathsf{succ}, S_4]$–$(\Pr[S_4])$
2	2(0.71)	5022	0.00 %–(0.00 %)	0.00 %–(100.00 %)	0.00 %–(49.70 %)	0.00 %–(49.70 %)
4	5(0.56)	7098	0.00 %–(1.51 %)	0.00 %–(99.42 %)	0.00 %–(36.97 %)	0.00 %–(36.97 %)
8	15(0.53)	7010	0.00 %–(4.07 %)	0.00 %–(98.49 %)	0.00 %–(36.01 %)	0.00 %–(36.01 %)
16	46(0.51)	6665	0.00 %–(1.23 %)	0.00 %–(97.99 %)	0.00 %–(38.86 %)	0.00 %–(38.84 %)
32	144(0.50)	6103	0.05 %–(0.16 %)	0.77 %–(98.33 %)	2.24 %–(45.55 %)	2.24 %–(45.53 %)
64	457(0.50)	7986	0.01 %–(0.01 %)	2.02 %–(98.32 %)	6.36 %–(53.72 %)	6.41 %–(53.72 %)
128	1449(0.50)	7460	0.00 %–(0.00 %)	2.01 %–(98.75 %)	7.02 %–(67.63 %)	7.67 %–(67.57 %)
256	4598(0.50)	6879	0.00 %–(0.00 %)	0.74 %–(98.92 %)	5.16 %–(80.23 %)	6.67 %–(80.20 %)
512	14597(0.50)	6094	0.00 %–(0.00 %)	0.31 %–(99.38 %)	3.02 %–(92.25 %)	5.07 %–(92.16 %)
2	3(1.06)	4316	0.00 %–(0.00 %)	0.00 %–(100.00 %)	0.00 %–(76.90 %)	0.00 %–(76.90 %)
4	8(0.89)	4153	0.07 %–(15.19 %)	0.07 %–(93.74 %)	1.13 %–(59.64 %)	1.13 %–(59.64 %)
8	23(0.81)	6703	3.88 %–(18.54 %)	2.27 %–(90.23 %)	4.83 %–(57.72 %)	4.85 %–(57.69 %)
16	73(0.81)	6886	10.30 %–(13.87 %)	21.71 %–(87.71 %)	29.65 %–(67.25 %)	29.67 %–(67.14 %)
32	230(0.80)	6952	10.34 %–(10.98 %)	43.18 %–(88.62 %)	57.44 %–(79.67 %)	57.44 %–(78.88 %)
64	730(0.80)	6568	8.82 %–(9.18 %)	59.10 %–(91.21 %)	75.29 %–(88.78 %)	75.21 %–(87.62 %)
128	2318(0.80)	6189	2.70 %–(2.83 %)	65.89 %–(93.89 %)	84.15 %–(93.75 %)	84.15 %–(92.39 %)
256	7357(0.80)	8054	0.84 %–(0.86 %)	67.21 %–(96.52 %)	87.79 %–(96.52 %)	88.33 %–(95.49 %)
512	23355(0.80)	626	0.00 %–(0.00 %)	67.09 %–(98.24 %)	90.58 %–(98.24 %)	91.53 %–(97.76 %)
2	3(1.06)	4352	0.00 %–(0.00 %)	0.00 %–(100.00 %)	0.00 %–(75.30 %)	0.00 %–(75.30 %)
4	9(1.00)	3864	3.03 %–(23.08 %)	3.60 %–(88.69 %)	7.27 %–(64.65 %)	7.27 %–(64.65 %)
8	29(1.02)	5791	27.65 %–(35.02 %)	29.11 %–(78.62 %)	34.31 %–(65.88 %)	34.31 %–(65.76 %)
16	91(1.01)	6585	28.44 %–(29.90 %)	49.83 %–(73.27 %)	54.08 %–(68.37 %)	54.08 %–(67.84 %)
32	288(1.00)	6814	30.69 %–(31.67 %)	62.91 %–(71.79 %)	65.17 %–(70.75 %)	65.10 %–(68.80 %)
64	913(1.00)	6981	39.52 %–(40.87 %)	73.80 %–(77.14 %)	73.24 %–(77.14 %)	72.87 %–(74.03 %)
128	2897(1.00)	6609	39.17 %–(40.73 %)	83.10 %–(83.83 %)	79.77 %–(83.83 %)	79.03 %–(79.89 %)
256	9196(1.00)	6176	45.16 %–(46.50 %)	88.52 %–(88.76 %)	85.80 %–(88.76 %)	85.01 %–(85.82 %)
512	29193(1.00)	486	44.03 %–(44.86 %)	93.21 %–(93.21 %)	90.95 %–(93.21 %)	89.92 %–(90.95 %)
8	58(2.03)	988	23.99 %–(23.99 %)	25.40 %–(25.40 %)	25.40 %–(25.40 %)	25.40 %–(25.40 %)
16	182(2.01)	2504	6.79 %–(6.79 %)	6.79 %–(6.79 %)	6.79 %–(6.79 %)	6.79 %–(6.79 %)
32	575(2.00)	3425	0.55 %–(0.55 %)	0.55 %–(0.55 %)	0.55 %–(0.55 %)	0.55 %–(0.55 %)
64	1825(2.00)	5727	0.02 %–(0.02 %)	0.02 %–(0.02 %)	0.02 %–(0.02 %)	0.02 %–(0.02 %)
128	5793(2.00)	1948	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
256	18391(2.00)	125	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
512	58386(2.00)	7	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
32	863(3.00)	1389	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
64	2737(3.00)	2666	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
128	8689(3.00)	167	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)
256	27586(3.00)	9	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)	0.00 %–(0.00 %)

Table 4.3: Experimental $\Pr[S_j]$ and success probability over many trials for various N and j using $L = 3$.

$|V| = 2\frac{M^2}{N^2} = 2^{3-\frac{1}{L}}N^{1+\frac{1}{L}}$ and L constant, we have $O(\frac{|V|^3}{N})$ which is equal to $O(N^{2+\frac{3}{L}})$. Another method to find cycles is to enumerate all L-tuples of vertices in $O(|V|^L)$ which is $O(N^{L+1})$. Therefore, we compute the minimum between the two methods which is $O(N^3)$ for any L and it is the complexity of step 3. (It can even be lower for $L > 3$.) Step 4 takes N time and finally step 5 takes $\frac{M^{2L}}{N^{3L}} = \frac{N}{2}$. As the complexity is weighted by step 3, **we have time complexity of our algorithm as $O(N^3)$ for $L = 3$ and a smaller $O(N^{2+\frac{3}{L}})$ for $L > 3$.** Instead of $L-1$ multiplications with a sparse matrix in the max-plus algebra, we could also use $O(\log L)$ general purpose matrix multiplications over the integer with the Coppersmith-Winograd algorithm [CW90]. We would reach a complexity of $O(|V|^{2.38}\log L)$ which is not better.

4.2.3 Round-Function-Recovery on 5-Round Feistel Scheme and More

Given the 4-round full recovery attack from Section 4.2.2, we can extend it to attack 5-round Feistel network. The attack for 5-round Feistel network is straightforward; it uses chosen plaintexts and guess strategies. First of all, consider our 4-round attack and the known plaintexts from this attack. We choose plaintexts for the 5-round so that the right half of the messages have as little different values as possible, then guess the corresponding images through F_0. It means that for the right halves of the messages, we generate all the possible partial tables of the first round function for these right values. Then, we guess which table is consistent after running the attack on the next 4-round. The data complexity of our 4-round attack is $\lambda N^{\frac{3}{2}+\epsilon}$, **hence our time complexity for 5-round recovery with chosen plaintexts is $O(N^{\lambda N^{\frac{1}{2}+\epsilon+3}})$.** The data complexity is unchanged.

We can attack $r-$rounds similarly with complexity $O(N^{(r-5)N+\sqrt{N}+3})$ by guessing the round functions on the last $(r-5)$ rounds. The data complexity is unchanged. We can apply this to FF1 ($r = 10$) and FF3 ($r = 8$). We obtain a complexity lower than 2^{128} for FF1 with $N = 7$ and for FF3 with $7 \leqslant N \leqslant 10$. (For lower N, exhaustive search on either the codebook or the round functions reaches the same conclusion.) Hence, **these instances of FF1 and FF3 do not offer 128-bit security**.

4.2.4 Round-Function-Recovery by Partial Exhaustive Search

We consider exhaustive search algorithms dealing with partial functions. Normally, a function F_i is defined by its set of all possible $(z, F_i(z))$ pairs. We call *partial table* any subset of its table. The *density* of a partial table is the ratio θ of its cardinality by N. For example, $\theta = \frac{1}{N}$ corresponds to a partial table defined on a single point z and $\theta = 1$ corresponds to the full table. A partial table is an *extension* of another partial table if the former is a superset of the latter. We deal with partial tables for each round function. We define r-tuples T of partial tables in which T_i denotes the partial table of F_i in T. We say T is homogeneous with density θ if for all i, T_i has density θ. Similarly, a tuple T' is an extension of T if for each i, T'_i is an extension of T_i. An *elementary tuple* is a homogeneous tuple of density $\frac{1}{N}$. This means that each of its partial functions are defined on a single point.

We say that a tuple T *encrypts* a plaintext M into a ciphertext C (or *decrypts* C into M) if we can evaluate the FN on M with the partial information we have about the round functions and if it gives C. We say that a pair (M, C) is *computable except for* r' *rounds* for a tuple T if the partial functions are enough to encrypt M for up to i rounds and to decrypt C for up to $r - i - r'$ rounds.

We say a tuple T of partial tables is *compatible* with (M, C) if one of the following conditions is satisfied: (i) T encrypts M into C; (ii) (M, C) is computable, except for two rounds or more; (iii) (M, C) is computable, except for one round and the half of the encryption of M that can be computed after $i + 1$ round matches the respective half of the decryption of y after $r - i - 1$ rounds. Clearly, if T is not compatible with (M, C), then no extension of T can encrypt M into C so we can prune an exhaustive search.

4.2.4.1 Random Partial Exhaustive Search

Assume that q plaintext/ciphertext pairs are known to the adversary. Consider an homogeneous r-tuple of partial functions $F_0, F_1, \ldots, F_{r-1}$ of density θ_i (i.e. for each round function, we know the fraction of θ_i of the table). Let $\mathtt{Pool}_i$ be a set of such tuples which are compatible with all pairs. Our aim is to start from $\theta_0 = \frac{1}{N}$ and to reconstruct the tables with $\theta = 1$ iteratively.

In Pool_i, for each tuple we have two types of plaintext/ciphertext pairs. For a tuple, it can either encrypt the pair with the partial tables it has, or it does not have complete enough partial tables to compute a pair (starting from a known plaintext, some partial tables do not contribute to verify the ciphertext). What we like to do is to extend these partial tables on arbitrary new points, in a way that the compatibility with all q pairs is preserved. We need to use some of q known plaintext/ciphertext pairs in order to check the validity of a tuple T in Pool_i. For each known plaintext/ciphertext pair out of q pairs, we can compute the fraction θ_i^r pairs with Pool_i, thus, we expect to compute $\theta_i^r q$ pairs given a tuple in Pool_i. Each fully computable pair is compatible with probability $\frac{1}{N^2}$. This decimates the pool of possible tuples by a factor $N^{2\theta_i^r q}$. We can also compute, for all but one round, a fraction $r\theta_i^{r-1}(1-\theta_i)$ of pairs, which are compatible with probability $\frac{1}{N}$. This decimates the pool of tuples by a factor $N^{r\theta_i^{r-1}(1-\theta_i)q}$. Partial functions are defined on $\theta_i N$ arbitrary points (i.e. not on any possible set of $\theta_i N$ points). That means we have $N^{r\theta_i N}$ possible tuples. Hence, among all $N^{r\theta_i N}$ possible r-tuples, we expect to have

$$|\mathsf{Pool}_i| \approx N^{r\theta_i N - r\theta_i^{r-1}(1-\theta_i)q - 2\theta_i^r q} = N^{r\theta_i N - r\theta_i^{r-1}q + (r-2)\theta_i^r q}.$$

We can construct Pool_{i+1} from Pool_i as shown in Algorithm 6, where $\theta_0, \theta_1, \ldots, \theta_w$ are the parameters of the algorithm. Our aim is to reach $\theta_w = 1$. For $\theta_0 = 0$, we have $|\mathsf{Pool}_i| = 1$ (the tuple of empty tables), but we rather start with $\theta_1 = \frac{1}{N}$. Indeed, we can set Pool_0 to an arbitrary tuple encrypting M_1 into C_1. This still keeps a solution. If we do so, we have $\mathsf{Pool}_0 = 1$ which reduces the size of the other pools. We continue decimating the consecutive pools with other $q-1$ known plaintext/ciphertext pairs. We obtain the formula

$$cns \times N^{r\theta_i N - r\theta_i^{r-1}(q-1) + (r-2)\theta_i^r (q-1)}$$

with a constant cns such that $|\mathsf{Pool}_0| = 1$ hence $cns \approx N^{-r}$.

Algorithm 6: Random partial exhaustive search round-function-recovery attack

```
1  Collect $q$ plaintext-ciphertext pairs $(M_i, C_i)$, $i = 1, \ldots, q$.
2  Get an arbitrary elementary tuple $T$ which encrypts $M_1$ to $C_1$.
3  Initialize $\mathsf{Pool}_0 = \{T\}$.
4  foreach $i = 1, \ldots, w$ do
5      foreach $T \in \mathsf{Pool}_{i-1}$ do
6          foreach homogeneous extension $T'$ of $T$ of density $\theta_i$ do
7              if all $(M_1, C_1), \ldots, (M_q, C_q)$ are compatible with $T'$ then
8                  Add $T'$ in $\mathsf{Pool}_i$.
9              end
10         end
11     end
12 end
13 output: $\mathsf{Pool}_w$
```

The complexity of the algorithm is the number of tuples in Pool_{i-1} (Step 5) multiplied by the number of possible extensions (Step 6) and number of plaintext/ciphertext pairs q (Step 7) summed over $i \in \{1, \ldots, w\}$ (Step 4). We have

$$T^{\mathsf{Pool}} = \sum_{i=1}^{w} q|\mathsf{Pool}_{i-1}|N^{r(\theta_i N - \theta_{i-1} N)} = \sum_{i=1}^{w} qN^{r\theta_i N - r\theta_{i-1}^{r-1}(q-1) + (r-2)\theta_{i-1}^{r}(q-1) - r} \tag{4.5}$$

One strategy consists of taking $\theta_i = \theta_{i-1} + \frac{1}{N}$ and $w = N$. In that case, we can compute T^{Pool} with Eq. (4.5). The T^{Pool} sum is dominated by a maximal term corresponding to a critical value θ_c of θ.

4.2.4.2 Approximation.

To simplify the analysis, we neglect θ_{i-1}^{r} against θ_{i-1}^{r-1}. If we substitute $\theta_i = \theta_{i-1} + \frac{1}{N}$, Eq. (4.5) is a sum of terms in θ_{i-1} only and the maximum of this function is reached by $\theta_c = \left(\frac{N}{(r-1)(q-1)}\right)^{\frac{1}{r-2}}$. We obtain the complexity of Algorithm 6 is

$$T^{\mathsf{Pool}} \approx qN^{\frac{r(r-2)}{r-1}N\left(\frac{N}{(r-1)(q-1)}\right)^{\frac{1}{r-2}}} \tag{4.6}$$

with q **known plaintexts**.

We checked experimentally that the values of Eq. (4.5) and Eq. (4.6) have a logarithm within the same order of magnitude (up to 20% difference), so we could use Eq. (4.6) as an indicative estimate and Eq. (4.5) for a more precise value.

4.2.4.3 Chosen plaintext extension.

As an improvement, we can save one more round with a chosen plaintext attack by guessing only $\frac{q}{N} - 1$ points in F_0 and applying the Pool attack on the remaining $r-1$ rounds. We obtain

$$T^{\mathsf{Pool}^*} = N^{\frac{q}{N}-1}T^{\mathsf{Pool}}_{r-1} \approx qN^{\frac{q}{N}-1+\frac{(r-1)(r-3)}{r-2}N\left(\frac{N}{(r-2)(q-1)}\right)^{\frac{1}{r-3}}} \tag{4.7}$$

with q **chosen plaintexts**.

4.2.4.4 Optimization with larger q.

For $q = N^{1+\alpha}$ and $N \to +\infty$, we have $T^{\mathsf{Pool}^*} = N^{O(N^{1-\frac{\alpha}{r-3}})}$ for $0 < \alpha \leqslant 1 - \frac{1}{r-2}$ and $T^{\mathsf{Pool}^*} = N^{O(N^{\alpha})}$ for $1 - \frac{1}{r-2} < \alpha \leqslant 1$. Precisely, q makes the exponent of T^{Pool^*} minimal for

$$q = \frac{r-1}{r-2}(r-1)^{-\frac{1}{r-2}}N^{2-\frac{1}{r-2}}$$

chosen plaintexts for which we obtain

$$T^{\mathsf{Pool}^*} \approx \frac{r-1}{r-2}(r-1)^{-\frac{1}{r-2}}N^{1-\frac{1}{r-2}+(r-1)^{1-\frac{1}{r-2}}N^{1-\frac{1}{r-2}}}. \tag{4.8}$$

To simplify, this is

$$T^{\mathsf{Pool}^*} \approx N^{(r-1)N^{1-\frac{1}{r-2}}(\beta+o(1))} \tag{4.9}$$

for $\beta = (r-1)^{-\frac{1}{r-2}} < 1$ when $N \to +\infty$, with $q = \frac{r-1}{r-2}\beta N^{2-\frac{1}{r-2}}$ **chosen plaintexts**.

4.2.4.5 Discussion.

Eq. (4.9) makes this attack asymptotically better than an exhaustive search on a single round function, i.e. $N^{N+o(N)}$ as N is large enough, hence a better complexity than the MITM strategy which requires full exhaustive searches in each round. In fact, $N^{(r-1)\beta N^{1-\frac{1}{r-2}}} < N^N$ is equivalent to $N > (\beta(r-1))^{r-2}$, meaning that $N > (r-1)^{r-3}$ when $\beta = (r-1)^{-\frac{1}{r-2}} < 1$ is plugged. **Hence, for $N > (r-1)^{r-3}$, our Pool^* attack is faster than exhaustive search on a single round function.** For instance, with $r = 8$ and $N = 2^{16}$, Eq. (4.8) gives $T^{\mathsf{Pool}^*} = 2^{835\,939}$ while $N^N \approx 2^{1\,048\,576}$.

It is interesting to see for which q the complexities T^{Pool^*} and T^{MITM^*} (with $q = r\frac{N}{2}$) become equal. For an arbitrary r, $T^{\mathsf{Pool}^*} \approx T^{\mathsf{MITM}^*}$ (meaning equality of the higher terms in the exponents) for $q = \frac{N}{r-2}\left(2\frac{(r-1)(r-3)}{(r-2)(r-4)}\right)^{r-3} \sim N\frac{2^{r-3}}{r-2}e^2$.

Using optimal complexity $q = r\frac{N}{2}$, the best round-function-recovery attack is MITM^* given in Sec. 4.1.3. **With $N^2 > q > N\frac{2^{r-3}}{r-2}e^2$, our Pool^* algorithm becomes better than MITM^*.**

4.2.4.6 Improvement.

We can speed up the algorithm by adding more points in the tuples when we can compute them. Concretely, if one plaintext/ciphertext pair can be "computed" except in one or two rounds, we can deduce the values in the missing rounds and define them in the tuple. Adding x points reduce the number of iterations to define the next pool by N^x.

4.2.4.7 Iterative Partial Exhaustive Search

In this section, we improve the Pool strategy. Instead of considering random extensions of partial tables, we only consider extensions that are able to encrypt one more pair. We proceed as defined by Algorithm 7. More precisely, we start with an arbitrary elementary tuple encrypting the first pair. At each iteration i, we extend each tuple from Pool_{i-1} by each possible extension encrypting the ith pair while being compatible with all others.

Algorithm 7: Iterative partial exhaustive search round-function-recovery attack

```
1  Collect $q$ plaintext-ciphertext pairs $(M_i, C_i)$, $i = 1, \ldots, q$.
2  Get an arbitrary elementary tuple $T^1$ which encrypts $M_1$ to $C_1$.
3  Initialize $\mathsf{Pool}_1 = \{T^1\}$.
4  foreach $i = 2, \ldots, q$ do
5      Initialize $\mathsf{Pool}_i$ to empty.
6      foreach $T \in \mathsf{Pool}_{i-1}$ do
7          foreach elementary tuple $T^i$ which encrypts $M_i$ to $C_i$ do
8              Set $T'$ with $T'_j = T_j \cup T^i_j$, $j = 0, \ldots, r-1$. (Extend $T$ with $T^i$.)
9              if $T'$ is a valid extension of $T$ then
10                 if all $(M_{i+1}, C_{i+1}), \ldots, (M_q, C_q)$ are compatible with $T'$ then
11                     Add $T'$ in $\mathsf{Pool}_i$.
12                 end
13             end
14         end
15     end
16 end
17 Output $\mathsf{Pool}_q$.
```

In practice, instead of enumerating all elementary tuples T^i then checking compatibility with T, we can limit ourselves to the enumeration of all compatible elementary tuples. With an appropriate data structure, we can also avoid to retry to encrypt M_j or decrypt C_j and directly go to the next computable round in every pair if any. This saves the inner loop. With these algorithmic tricks, the complexity is expected to be close to the size of the pools.

We can estimate $|\mathsf{Pool}_i|$ with the same analysis as for the Pool algorithm. What changes is that the first i pairs always decimate the pool and that random decimation is based on the remaining $q - i$ pairs. Essentially, we approximate the size of the pool with N^{X-Y} where X is the number of entries in the partial functions (i.e. the number of defined points throughout all rounds) and Y is the number of equations over N-values which a tuple must satisfy to be compatible. To treat the fact that we start with an arbitrary elementary tuple, we subtract r to X (that is, we do not enumerate all tuples in the first iteration) and we subtract 2 to Y (i.e., we consider that the taken tuple is already decimated by the first pair). We obtain

$$|\mathsf{Pool}_i| \approx \mathsf{cns} \times N^{r\theta_i N - r - 2(i-1) - r\theta_i^{r-1}(1-\theta_i)(q-i) - 2\theta_i^r(q-i)} \tag{4.10}$$

with cns such that $|\mathsf{Pool}_1| = 1$, so $\mathsf{cns} \approx 1$. To estimate θ_i, we can observe that at each iteration we need to map one random input in each round function and it may be either new or not. At each round i, with probability θ_{i-1} the table T_j does not increase (the input is not new), and with probability $1-\theta_{i-1}$, it increases by one element (the input is new). Therefore, on average we have

$$\theta_i = {\theta_{i-1}}^2 + (1-\theta_{i-1})\left(\theta_{i-1} + \frac{1}{N}\right) = \theta_{i-1}\left(1 - \frac{1}{N}\right) + \frac{1}{N}.$$

We deduce $\theta_i = 1 - \left(1 - \frac{1}{N}\right)^i$. Then, the complexity is

$$T^{\mathsf{Iter}} = \sum_{i=1}^{N} N^{r\theta_i N - 2i - r\theta_i^{r-1}(1-\theta_i)(q-i) - 2\theta_i^r(q-i) - r + 2} \tag{4.11}$$

4.2.4.8 On the validity of Eq. (4.10).

In order to come up with Eq. (4.10), we modeled the behavior of Iter to construct Pool_i. For that, we realize that picking an elementary tuple encrypting one plaintext (no matter what the ciphertext is) corresponds to picking a one random input for each round function. We call this a trial. An input to one round function corresponds to a ball with a number from 0 to $N-1$. A round function is a bag of N balls. So, we have r bags of balls and a trial consists of picking one ball in each bag. Balls are replaced in their respective bags after picking them. We do q trials in total but the first i and the last $q-i$ play different roles. The balls which were picked during the first i trials are called good balls. We estimate the random variable X as the total number of good balls, to which we subtract r (because we enumerate all possibilities in every trials but the first one). Then, we look at the number of good balls in each trial (except the first one). The random variable Y is set to the number of trials with $r-1$ good balls plus twice the number of trials with r good balls (the first one does not count but the next $i-1$ ones always count, by construction). Clearly, conditioned to a density of good balls of θ_j in round j, we have $E(X) = \sum_{j=1}^{r} \theta_j N - r$ and $E(Y) = 2(i-1) + (q-i)\sum_{j=1}^{r}(1-\theta_j)\prod_{j'\neq j}\theta_j + 2(q-i)\prod_j \theta_j$. All θ_j are random and independent, with expected value θ. Thus, Eq. (4.10) corresponds to $cns \times N^{E(X-Y)}$.

For $r=5$, $N=8$, $q=40$, we computed X and Y in 10 000 such experiments. We obtained an average for $X-Y$ of 2.424 with a standard deviation of 1.522. The exponent of Eq. (4.10) gives 2.442 with these figures. So, assuming that the above experiment models well Iter, the expected value of $\log|\mathsf{Pool}_i|$ should match Eq. (4.10). However, Eq. (4.10) cannot represent well the expected value of $|\mathsf{Pool}_i|$ as exponential with bigger exponents will have a huge impact on the average.

4.2.4.9 Approximation.

For $i \ll N$ we can write $\theta_i = \frac{i}{N}$. By neglecting θ_i^r against θ_i^{r-1}, the complexity is approximated by the maximum of $N^{r\theta N - 2N\theta - r\theta^{r-1}q - r + 2}$. We can easily show that the maximum is reached by $\theta = \theta_c$ with

$$\theta_c = \left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} \left(\frac{N}{q}\right)^{\frac{1}{r-2}}$$

We obtain the complexity

$$T^{\mathsf{Iter}} \approx N^{\frac{(r-2)^2}{r-1}\left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} N \left(\frac{N}{q}\right)^{\frac{1}{r-2}} - r + 2} \tag{4.12}$$

with q **known plaintexts**.

The best complexity is reached with **the full codebook** $q = N^2$ with

$$T^{\mathsf{Iter}} \approx N^{\frac{(r-2)^2}{r-1}\left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} N^{1-\frac{1}{r-2}} - r + 2} \tag{4.13}$$

which is $T^{\mathsf{Iter}} = N^{\frac{(r-2)^2}{r-1}(\beta + o(1)) N^{1-\frac{1}{r-2}}}$ for some $\beta < 1$.

4.2.4.10 Chosen plaintext extension.

Finally, if q is not too close to N^2, a chosen plaintext attack variant consists of fixing the right half of the plaintext as much as possible then guessing F_0 on these points and run the known-plaintext attack on $r-1$ rounds to obtain

$$T^{Iter^*} = N^{\frac{q}{N}-1} T^{Iter}_{r-1} \approx N^{\frac{q}{N}-1+\frac{(r-3)^2}{r-2}\left(\frac{r-3}{(r-1)(r-2)}\right)^{\frac{1}{r-3}} N\left(\frac{N}{q}\right)^{\frac{1}{r-3}}-r+3} \tag{4.14}$$

with q **chosen plaintexts** such that $q \leqslant N^2$.

4.2.4.11 Discussion.

For $N^2 > q > N\frac{r-3}{(r-1)(r-2)}\left(2\frac{(r-3)^2}{(r-2)(r-4)}\right)^{r-3} \sim \frac{Ne^3}{r}2^{r-3}$, we have $T^{Iter^*} < N^{<fracqN-r+2+\frac{r-4}{N}}$ so $T^{Iter^*} < T^{MITM^*}$, so $Iter^*$ **becomes better than** $MITM^*$. Also, for $N \geqslant \frac{(r-3)^{r-2}}{r-1}$, we have $T^{Iter^*} < N^{N-r+2}$ so $Iter^*$ **is faster than exhaustive search on a single round function**.

4.2.4.12 Optimization with larger q.

We easily obtain that this is optimal with

$$T^{Iter^*} \approx N^{(r-3)N^{1-\frac{1}{r-2}}\left(\frac{1}{r-1}\right)^{\frac{1}{r-2}}-r+2} \tag{4.15}$$

for

$$q = \frac{r-3}{r-2}N^{2-\frac{1}{r-2}}\left(\frac{1}{r-1}\right)^{\frac{1}{r-2}}.$$

chosen plaintexts.

4.2.4.13 Variants of $Iter$ and $Iter^*$

Optimized algorithm. We can speed up the algorithm by adding more points in the tuples as soon as we can compute them. Concretely, if one plaintext/ciphertext pair can be "computed" except in one or two rounds, we can deduce the values in the missing rounds and define them in the tuple. Adding x points reduce the number of iterations to define the next pool by N^x.

Abort strategy. Our complexity is not an average complexity but its logarithm has a right average. To avoid having a too high average complexity, we may change the algorithm to make it abort if the pool exceeds a threshold to be defined. For instance, if our theoretical formula predicts a complexity Th, to make sure that the worst case complexity does not exceed $\mathsf{Th} \times N^x$, we set this to the threshold value. This will affect the success probability, which is 100% without the abort strategy, but may be lower for any real number x.

Other improvements. We believe we could improve our algorithms in many ways. For instance, we could take the (M_i, C_i) pairs in an optimized order so that we do not have too many new values appearing in the first and last round functions. This would decrease the number of tuples to consider.

4.2.4.14 Experimental Results

We implemented Algorithm 7 with known plaintext, $r = 5$, $N = 8$, $q = 40$. Our algorithm always ended with a pool limited to a correct set of full tables.

With these parameters, Eq. (4.10) estimates Pool_3 to be the largest with $|\mathsf{Pool}_3| = N^{2.49}$. We checked over ten executions, that $\log_N |\mathsf{Pool}_3|$ has an average of 4.45 and a standard deviation of 0.52. This is quite larger than predicted. More precisely, each partial function in Pool_3 has on average 2.81 defined entries, which is only a bit slightly more than the $N\theta_3 \approx 2.64$ predicted. But adjusting θ_3 to $\frac{2.81}{N}$ in Eq. (4.10) is not enough to explain the high $|\mathsf{Pool}_3|$ which is observed. So, our model for the random variable X may be correct but Y may be overestimated: Iter decimates less than expected. Although we thought Pool_3 would be the largest from our theory, the largest observed pool during our experiment were Pool_4 with logarithmic size with average 5.30. This indicates that our model for Iter is not accurate. At this time, we believe this is mostly due to our tested parameters being too low.

Nevertheless, assuming there is something more fundamental in the Feistel structure which was overlooked in our theoretical analysis, we may wonder what to think of the predicted complexity results. We can see that Eq. (4.11) anticipates a complexity of $T^{\mathsf{Iter}} = 262$ for the above figures. If we change the algorithm to use the abort strategy as soon as the pool becomes bigger than 262, we obtain no successful run of the attack (after running it 10 000 times). Then, using the optimized variant, we could obtain 1.3% of successful runs (over 10 000) with the anticipated complexity. So, we believe that **our anticipated complexities may be achievable with a good success probability**. However, finding a good model for decimation and for the improved algorithm remains an open question.

Of course, using a slightly higher complexity still gives better success probability. For instance, allowing the pools to become N times larger than expected with the improved algorithm leads us to a success rate of 42% (over 100 runs). We summarize our experiments in the table below.

$\mathsf{r} = 5$, $\mathsf{N} = 8$, $\mathsf{q} = 40$				
#runs	success	max $\|\mathsf{Pool}\|$	opt	abort
100	100%	$\mathsf{Th} \times \mathsf{N}^{2.79}$	no	no
10 000	0%		no	Th
1 000	0%		no	$\mathsf{Th} \times \mathsf{N}$
1 000	3%	$\mathsf{Th} \times \mathsf{N}^{1.76}$	no	$\mathsf{Th} \times \mathsf{N}^2$
100	100%	$\mathsf{Th} \times \mathsf{N}^{0.93}$	yes	no
10 000	1%	$\mathsf{Th} \times \mathsf{N}^{-0.29}$	yes	Th
100	42%	$\mathsf{Th} \times \mathsf{N}^{0.59}$	yes	$\mathsf{Th} \times \mathsf{N}$
100	99%	$\mathsf{Th} \times \mathsf{N}^{0.90}$	yes	$\mathsf{Th} \times \mathsf{N}^2$

Here, Th designates the largest pool size as predicted by our theory. The column opt tells whether we used the optimization trick. The abort column indicates when we used the abort strategy, and with which bound. The max $|\mathsf{Pool}|$ column reports the average (logarithmically) of the largest observed pool.[3,4]

We run tests with parameters $\mathsf{r} = 5$, $\mathsf{N} = 10$, and $\mathsf{q} = 40$:

[3]The logarithm of this column is the maximum over each iteration of the average over the runs of the logarithm of the pool size.

[4]The average is computed only includes successful runs, as unsuccessful ones are all on the abort threshold.

$r = 5,\ N = 10,\ q = 40$				
#runs	success	max $\lvert\mathsf{Pool}\rvert$	opt	abort
10 000	0%		no	Th
1 000	0%		no	$\mathsf{Th} \times N$
100	0%		no	$\mathsf{Th} \times N^2$
14	100%	$\mathsf{Th} \times N^{1.40}$	yes	no
10 000	1%	$\mathsf{Th} \times N^{-0.31}$	yes	Th
100	19%	$\mathsf{Th} \times N^{0.60}$	yes	$\mathsf{Th} \times N$
19	68%	$\mathsf{Th} \times N^{1.25}$	yes	$\mathsf{Th} \times N^2$

4.3 Applications

In the standards, the supported domain size of messages in FF1 and FF3* is greater than 100 (i.e. $N^2 \geqslant 100$). For FF1 and FF3*, the best attack is roughly Iter^* for very low N, then MITM^*, then Pool^* for larger N. More precisely, we achieve the following results:

$r = 8$ (FF3*)			$r = 10$ (FF1)		
N	$T^{\mathsf{MITM}^*}[q]$ (4.2)	$T^{\mathsf{Iter}^*}[q]$ (4.15)	N	$T^{\mathsf{MITM}^*}[q]$ (4.2)	$T^{\mathsf{Iter}^*}[q]$ (4.15)
2^1	$2^{6}[\ 2^{2.0}\ ]$	$\mathbf{2^{1}}[\ 2^{2.0}\ ]$	2^1	$2^{8}[\ 2^{2.0}\ ]$	$\mathbf{2^{2}}[\ 2^{2.0}\ ]$
2^2	$2^{21}[\ 2^{4.0}\ ]$	$\mathbf{2^{13}}[\ 2^{4.0}\ ]$	2^2	$2^{29}[\ 2^{4.0}\ ]$	$\mathbf{2^{21}}[\ 2^{4.0}\ ]$
2^3	$2^{58}[\ 2^{5.0}\ ]$	$\mathbf{2^{44}}[\ 2^{5.0}\ ]$	2^3	$2^{82}[\ 2^{5.3}\ ]$	$\mathbf{2^{75}}[\ 2^{5.3}\ ]$
2^4	$2^{147}[\ 2^{6.0}\ ]$	$\mathbf{2^{122}}[\ 2^{6.6}\ ]$	2^4	$2^{211}[\ 2^{6.3}\ ]$	$\mathbf{2^{209}}[\ 2^{6.9}\ ]$
2^5	$2^{356}[\ 2^{7.0}\ ]$	$\mathbf{2^{295}}[\ 2^{8.4}\ ]$	2^5	$2^{516}[\ 2^{7.3}\ ]$	$\mathbf{2^{512}}[\ 2^{8.8}\ ]$
2^6	$2^{837}[\ 2^{8.0}\ ]$	$\mathbf{2^{658}}[\ 2^{10.3}\ ]$	2^6	$2^{1221}[\ 2^{8.3}\ ]$	$\mathbf{2^{1166}}[\ 2^{10.7}\ ]$
2^7	$2^{1926}[\ 2^{9.0}\ ]$	$\mathbf{2^{1401}}[\ 2^{12.1}\ ]$	2^7	$2^{2822}[\ 2^{9.3}\ ]$	$\mathbf{2^{2543}}[\ 2^{12.5}\ ]$
2^8	$2^{4359}[\ 2^{10.0}\ ]$	$\mathbf{2^{2890}}[\ 2^{13.9}\ ]$	2^8	$2^{6407}[\ 2^{10.3}\ ]$	$\mathbf{2^{5383}}[\ 2^{14.4}\ ]$

(Note that the standard requires $N \geqslant 10$ so the first three rows are not appropriate in practice.) We also did the computation without approximations, i.e. by using Eq. (4.11) instead of Eq. (4.12) in Eq. (4.14):

$r = 8$ (FF3*)			$r = 10$ (FF1)		
N	$T^{MITM^*}[q]$ (4.2)	$T^{Iter^*}[q]$ (4.15)	N	$T^{MITM^*}[q]$ (4.2)	$T^{Iter^*}[q]$ (4.15)
2^1	$2^{6}[\ 2^{2.0}\ ]$	$\mathbf{2^{2}}[\ 2^{2.0}\ ]$	2^1	$2^{8}[\ 2^{2.0}\ ]$	$\mathbf{2^{3}}[\ 2^{2.0}\ ]$
2^2	$2^{21}[\ 2^{4.0}\ ]$	$\mathbf{2^{13}}[\ 2^{4.0}\ ]$	2^2	$2^{29}[\ 2^{4.0}\ ]$	$\mathbf{2^{21}}[\ 2^{4.0}\ ]$
2^3	$2^{58}[\ 2^{5.0}\ ]$	$\mathbf{2^{42}}[\ 2^{5.0}\ ]$	2^3	$2^{82}[\ 2^{5.3}\ ]$	$\mathbf{2^{72}}[\ 2^{5.3}\ ]$
2^4	$2^{147}[\ 2^{6.0}\ ]$	$\mathbf{2^{116}}[\ 2^{6.6}\ ]$	2^4	$2^{211}[\ 2^{6.3}\ ]$	$\mathbf{2^{199}}[\ 2^{6.8}\ ]$
2^5	$2^{356}[\ 2^{7.0}\ ]$	$\mathbf{2^{279}}[\ 2^{8.3}\ ]$	2^5	$2^{516}[\ 2^{7.3}\ ]$	$\mathbf{2^{487}}[\ 2^{8.6}\ ]$
2^6	$2^{837}[\ 2^{8.0}\ ]$	$\mathbf{2^{627}}[\ 2^{10.1}\ ]$	2^6	$2^{1221}[\ 2^{8.3}\ ]$	$\mathbf{2^{1115}}[\ 2^{10.5}\ ]$
2^7	$2^{1926}[\ 2^{9.0}\ ]$	$\mathbf{2^{1343}}[\ 2^{12.0}\ ]$	2^7	$2^{2822}[\ 2^{9.3}\ ]$	$\mathbf{2^{2445}}[\ 2^{12.4}\ ]$
2^8	$2^{4359}[\ 2^{10.0}\ ]$	$\mathbf{2^{2788}}[\ 2^{13.8}\ ]$	2^8	$2^{6407}[\ 2^{10.3}\ ]$	$\mathbf{2^{5202}}[\ 2^{14.3}\ ]$

As we can see, the values are not much different.

For instance, for FF3* with $N = 2^3$ (i.e., messages have 6 bits), MITM^* uses $q = 2^5$ pairs (half of the codebook) and search on three points for F_0, the entire (but one point) F_1 and F_2, one bit of F_3 in the encryption direction, and the entire (but one point) F_7 and F_6 and one bit of F_5 in the decryption direction. This is $N^{3+2(N-1)} \times 2^{N-1} = 2^{58}$. (With the same parameters, we have $T^{DV} = 2^{89}$ with the algorithm from App **??**.) With Iter^*, we also use $q = 2^5$ and the pool reaches its critical density for $\theta_c \approx \frac{4.4}{N}$. The complexity is $T^{Iter^*} = 2^{42}$.

We may wonder for which N the ciphers offer a 128-bit security. We notice that for $N \leqslant 5$, we have $N^{rN} \gg N^2!$, so FN is likely to be as good as the ideal cipher. However, we have $N^2! < 2^{128}$ so exhaustive search on the codebook is faster than an exhaustive search on 128 bits. For $r = 8$ and $N < 7$, we have $N^{rN} < 2^{128}$ so an exhaustive search on the round functions already shows that we have no 128-bit security. Durak and Vaudenay [DVc] extended it for FF3* with $7 \leqslant N \leqslant 10$ and FF1 with $N = 7$. By doing computations for Iter^*, we extend this to show that **FF3* does not offer a 128-bit security for $N \leqslant 17$**, and **FF1 does not offer a 128-bit security for $N \leqslant 11$**.

The AEZ [HKR17] authenticated encryption method (which is one of the CAESAR 3rd round competitors) features an AEZ-tiny encryption to be used when the messages are very small. AEZ-tiny is a Feistel scheme with AES4 round functions. The number of rounds depends on N. In the table below, we report the four possible configurations of AEZ-tiny and the best value for $\mathsf{T}^{\mathsf{Pool}^*}$.[5] As we can see, **the number of rounds in AEZ-tiny is an overkill**.

N	r	$\mathsf{T}^{\mathsf{MITM}^*}[\mathsf{q}]$ (4.2)	$\mathsf{T}^{\mathsf{Iter}^*}[\mathsf{q}]$ (4.15)
$\mathsf{N} = 2^4$	$\mathsf{r} = 24$	$\mathbf{2^{659}}[2^{7.6}]$	$2^{779}[2^{7.6}]$
$\mathsf{N} = 2^8$	$\mathsf{r} = 16$	$\mathbf{2^{12551}}[2^{11.0}]$	$2^{14235}[2^{14.9}]$
$2^8 < \mathsf{N} < 2^{64}$	$\mathsf{r} = 10$	$\mathsf{N}^{3\mathsf{N}+\mathsf{O}(1)}[5\mathsf{N}]$	$\mathsf{N}^{\mathsf{O}\left(\mathsf{N}^{\frac{7}{8}}\right)}[\mathsf{N}^{2-\frac{1}{8}}]$
$2^{64} \leqslant \mathsf{N} < 2^{128}$	$\mathsf{r} = 8$	$\mathsf{N}^{2\mathsf{N}+\mathsf{O}(1)}[4\mathsf{N}]$	$\mathsf{N}^{\mathsf{O}\left(\mathsf{N}^{\frac{5}{6}}\right)}[\mathsf{N}^{2-\frac{1}{6}}]$

(Note that the big-O notation has no mathematical sense here: it is only here to give a rough estimate.)

It seems that we could reduce r and keep a good security at the same time. Genuinely , we can compute the minimum $\mathsf{r}_{\mathsf{opt}} \geqslant 4$ of the number of rounds for which $\min(\mathsf{T}^{\mathsf{MITM}^*}, \mathsf{T}^{\mathsf{Iter}^*}) \geqslant 2^{\mathsf{s}}$ depending on s and N. To be on the safe page, we computed without using our approximations (i.e., the complexity of the known plaintext part of the attack was computed using Eq. (4.11)). For $\mathsf{s} = 128$ and $\mathsf{s} = 256$, we fetch the following table.[6]

$\mathsf{s} = 128$				$\mathsf{s} = 256$			
N	$\mathsf{r}_{\mathsf{opt}}$	$\mathsf{T}^{\mathsf{MITM}^*}$	$\mathsf{T}^{\mathsf{Iter}^*}$	N	$\mathsf{r}_{\mathsf{opt}}$	$\mathsf{T}^{\mathsf{MITM}^*}$	$\mathsf{T}^{\mathsf{Iter}^*}$
2^1	260	$2^{258.0}$	$\mathbf{2^{128.5}}$	2^1	516	$2^{514.0}$	$\mathbf{2^{256.5}}$
2^2	40	$2^{149.0}$	$\mathbf{2^{129.3}}$	2^2	77	$\mathbf{2^{225.0}}$	$2^{257.6}$
2^3	14	$\mathbf{2^{130.0}}$	$2^{136.5}$	2^3	26	$\mathbf{2^{274.0}}$	$2^{297.8}$
2^4	9	$2^{195.0}$	$\mathbf{2^{155.8}}$	2^4	12	$\mathbf{2^{275.0}}$	$2^{289.1}$
2^5	7	$2^{341.0}$	$\mathbf{2^{187.9}}$	2^5	8	$2^{356.0}$	$\mathbf{2^{279.3}}$
2^6	6	$2^{453.0}$	$\mathbf{2^{236.2}}$	2^6	7	$2^{819.0}$	$\mathbf{2^{415.8}}$
2^7	5	$2^{1016.0}$	$\mathbf{2^{195.4}}$	2^7	6	$2^{1030.0}$	$\mathbf{2^{485.0}}$
2^8	5	$2^{2295.0}$	$\mathbf{2^{370.4}}$	2^8	5	$2^{2295.0}$	$\mathbf{2^{370.4}}$

Even by adding a safety margin, this shows that we do not need many rounds to safely encrypt a byte (that is, $\mathsf{N} = 2^4$) with respect to our best attacks. However,

[5] Again, we computed without using our approximations, i.e. the complexity of the known plaintext part of the attack was computed using Eq. (4.11).

[6] In this table, we computed the value of q suggested by our formulas but rounded in the $\left[\frac{\mathsf{rN}}{2}, \mathsf{N}^2\right]$ interval.

with low r, we should care about other attacks as in Table 4.1. Indeed, for $\oplus$-FN, we recommend never to take $r \leqslant 7$ due to the yo-yo attack [BLP16]. For other FN, we recommend never to take $r \leqslant 5$.

Note that our results for $N \leqslant 2^2$ must be taken with great care. Indeed, $\frac{rN}{2} > N^2$, meaning that we have no unicity of the round functions. In addition to this, $N^2 \log_2 N^2 < s$ i.e. the entropy of the full codebook is already less than the security parameter s.

We plot on Fig. 4.5 the (r, N) parameters for which we have $T^{\mathsf{Pool}^*} = T^{\mathsf{Iter}^*}$. As we can see, for and constant N and a low r, Iter^* is the best attack. The same figure includes the curve corresponding to a 128-bit and a 256-bit security. In Fig. 4.6, we plot complexities for $r = 8$ or $r = 10$ and various ranges of N. The regions for T^{Iter^*} we plot have a minimum for the optimal q and a maximum for $r = \frac{rN}{2}$. The region corresponds to all complexities for $q \in [\frac{rN}{2}, N^2]$.

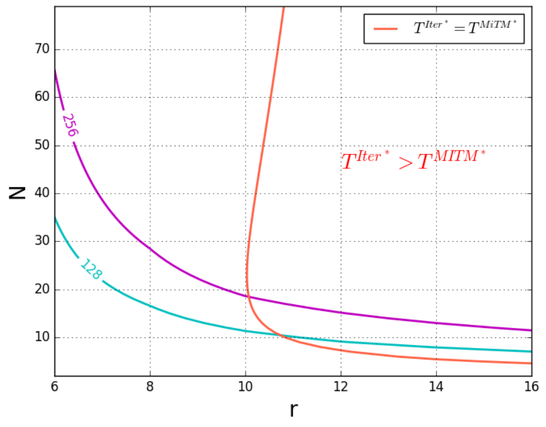
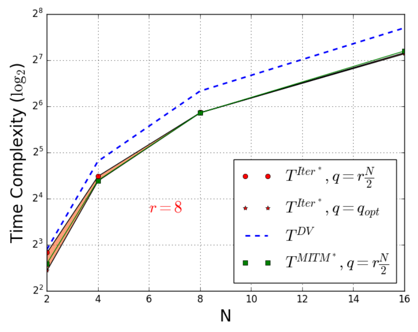


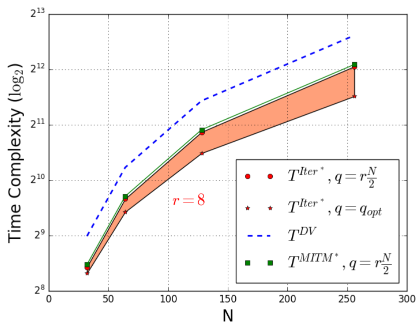
Figure 4.5: Parameters for $T^{\mathsf{Pool}^*} = T^{\mathsf{Iter}^*}$

4.4 Conclusions

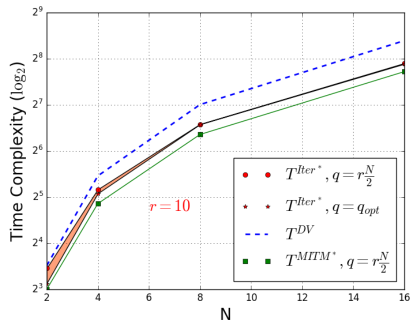
Standard Feistel Networks and its variations have created an active research area since its invention and have been used in construction of many cryptographic systems to a wide extent. The security of FN has been studied for so long and many interesting results have been proposed for cryptanalysis purposes. In this chapter, we considered a very specific type of FN with two branches, secure random round functions, and



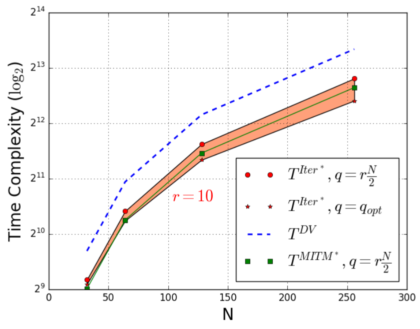
(a) comparison of various attacks $N \leqslant 2^6$ and $r = 8$



(b) comparison of various attacks $N \leqslant 2^9$ and $r = 8$



(c) comparison of various attacks $N \leqslant 2^6$ and $r = 10$



(d) comparison for various attacks $N \leqslant 2^9$ and $r = 10$

Figure 4.6: Time complexity of attacks for different algorithms and various parameters.

modular addition to analyze its security. Additionally, we considered small domains. We started with a brief survey on known and newly derived attacks on this specific FN. Finally, we gave a full recovery of round functions with much smaller complexity than the surveyed attacks. We have given a polynomial attack (in terms of N) on four rounds and some optimized bruteforce attacks for any rounds.

This specific FN with the described properties have been used to build Format-Preserving Encryption and perhaps will inspire many other constructions. However, the security of FN with various properties is not clear (regardless of the significant security analyses mentioned in earlier sections) and has to be investigated more. Our work shows only that a caution should be taken in order to meet the desired security level in the systems.

We give the cryptanalysis of the FF3 standard which is a Feistel-based FPE in the following section.

Chapter 5

Breaking the FF3 Format-Preserving Encryption Standard Over Small Domains

In the previous chapter, we have shown various attacks on generic Feistel Network with two branches, secure random round functions, modular addition, and the small domains. This specific type of FN is used to construct FPE schemes. Indeed, NIST and ANSI published standards including two Feistel-based FPE construction. This chapter gives a total break to FF3 standard over small domains. It is published in ESC'17 [DVa] and the proceedings of Crypto'17 [DVb]. In reaction to this attack, NIST released the following announcement stating that "NIST has concluded that FF3 is no longer suitable as a general-purpose FPE method.": https://beta.csrc.nist.gov/News/2017/Recent-Cryptanalysis-of-FF3. The same FPE algorithm also appears in the ANSI X9.119 standard and ANSI is also working on its update.

5.1 The Variant of FF3 with XOR

In order to illustrate the problem, for a variant of FF3 with $\oplus$ instead of $\boxplus$, we present a trivial attack: Consider an FF3 encryption with a key $\mathsf{K} \in \mathcal{K}$, a tweak $\mathsf{T} = \mathsf{T_L} \| \mathsf{T_R} \in \mathcal{T}$ and domain $\mathcal{X}$. Each round i defines a random function $\mathsf{F}_i = \mathsf{F_K}(\mathsf{T_R} \oplus \mathsf{STR}_2^{32}(i), \cdot)$ for i even ($\mathsf{F}_i = \mathsf{F_K}(\mathsf{T_L} \oplus \mathsf{STR}_2^{32}(i), \cdot)$ for i odd). We use the encryption with an input message $\mathsf{X} = (\mathsf{L}_0, \mathsf{R}_0)$ and output ciphertext $\mathsf{Y} = (\mathsf{L}_w, \mathsf{R}_w)$ with output X_i from each round in Fig. 5.1 (a). We assume that b is even so that $\ell = r$. Now, we take the ciphertext Y from Fig. 5.1 (a) and reverse it into $(\mathsf{L}'_0, \mathsf{R}'_0) = (\mathsf{R}_w, \mathsf{L}_w)$ to encrypt it with a new tweak $\mathsf{T}' = \mathsf{T_R} \oplus \mathsf{STR}_2^{32}(w-1) \| \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(w-1) \in \mathcal{T}$. We show this encryption in Fig. 5.1

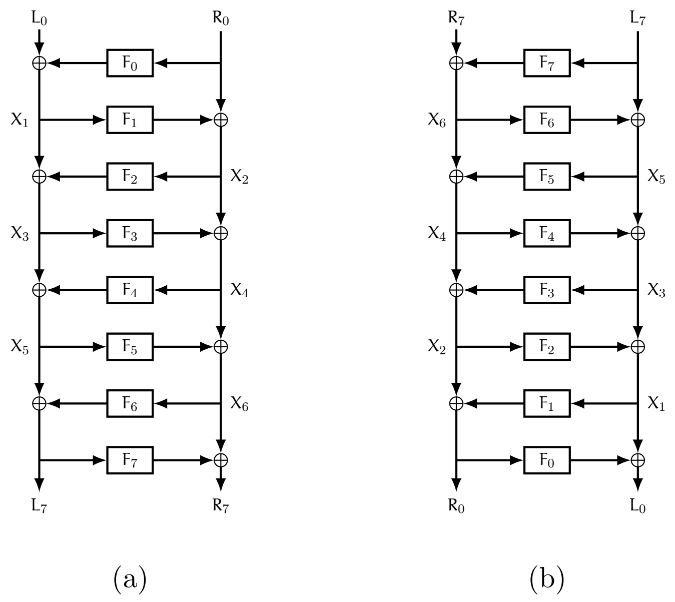


Figure 5.1: Trivial Attack on 8-round FF3 Encryption with $\oplus$ instead of modular addition $\boxplus$.

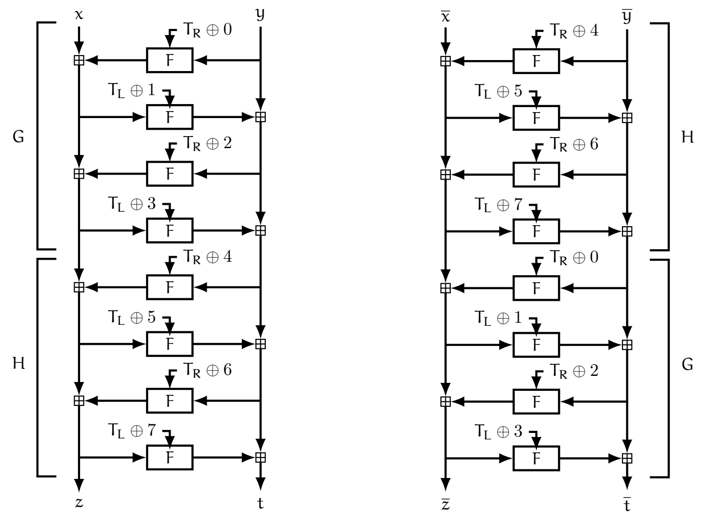
(b). We assume that w is a power of two (Fig. 5.1 uses $w = 8$). With given encryption, we obtain the round functions $\mathsf{F}'_i = \mathsf{F}_{w-1-i}$ as shown on Fig. 5.1 (a). More precisely, the attack works as follows:

- Encrypt $(\mathsf{L}_0, \mathsf{R}_0)$ with the tweak T to get $(\mathsf{L}_w, \mathsf{R}_w)$.
- Encrypt $(\mathsf{R}_w, \mathsf{L}_w)$ with the tweak T' to get $(\mathsf{L}', \mathsf{R}')$.
- If $\mathsf{L}' = \mathsf{R}_0$ and $\mathsf{R}' = \mathsf{L}_0$, output 1. Otherwise, output 0.

The adversary always outputs 1 with $\mathcal{E}_{\mathsf{K}}$. It outputs 1 with $\Pi(\cdot,\cdot)$ with probability $\frac{1}{\mathsf{s}^{\mathsf{b}}}$. Therefore, the advantage is $1 - \frac{1}{\mathsf{s}^{\mathsf{b}}}$.

5.2 Slide Attack on FF3

We developed an attack on 4-round Feistel network in Section 4.2.2 and deploy it as a building block for our chosen-plaintext and chosen-tweak attack to FF3 scheme. Our FF3 attack aims to reconstruct the entire codebook for a challenge tweak for a number of queries which is lower than the size of the brute force codebook attack. The main



(a) FF3 Encryption (b) Slided Encryption

Figure 5.2: FF3 encryption with sliding round functions

idea of the designed FF3 attack takes advantage of the flexibility to change the tweak to permute the round functions.

Consider two functions G and H, where G is a 4-round Feistel scheme using tweakable block cipher F with tweaks $(\mathsf{T_R} \oplus \mathsf{STR}_2^{32}(0), \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(1), \mathsf{T_R} \oplus \mathsf{STR}_2^{32}(2), \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(3))$ and H is a 4-round Feistel scheme using tweakable block cipher F with tweaks $(\mathsf{T_R} \oplus \mathsf{STR}_2^{32}(4), \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(5), \mathsf{T_R} \oplus \mathsf{STR}_2^{32}(6), \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(7))$. In Fig. 5.2, we show two runs of FF3 encryption with tweak $\mathsf{T} = \mathsf{T_L} \| \mathsf{T_R}$ in (a) and tweak $\mathsf{T}' = \mathsf{T_L} \oplus \mathsf{STR}_2^{32}(4) \| \mathsf{T_R} \oplus \mathsf{STR}_2^{32}(4)$ in (b) on two distinct plaintext. We observe that $\mathsf{FF3.E}(\mathsf{K}, \mathsf{T}, \cdot) = \mathsf{H} \circ \mathsf{G}$ and $\mathsf{FF3.E}(\mathsf{K}, \mathsf{T}', \cdot) = \mathsf{G} \circ \mathsf{H}$. For simplicity, we do not explicitly write $\mathsf{STR}_2^{32}(\cdot)$ any longer. Given this permuting ability by setting the tweaks XORed with round functions, we desire to form a "cyclic" behavior of plaintext/ciphertext pairs under two FF3 encryption with sliding G and H.

We pick at random two sets of messages $X = \{xy_0^1, \ldots, xy_0^i, \ldots, xy_0^A\}$ and $\overline{X} = \{\overline{xy}_0^1, \ldots, \overline{xy}_0^i, \ldots, \overline{xy}_0^A\}$ of size A. For each message xy_0^i in X, set $xy_{j+1}^i = \mathsf{Enc}(K, T, xy_j^i)$ with a fixed tweak $T \in \mathcal{T}$ and a fixed key $K \in \mathcal{K}$. We repeat the chain encryption of outputs B times for each message in X. Let XC be the set of chain encryption of elements of X. It contains segments of length B of cycles of $\mathsf{H} \circ \mathsf{G}$. Similarly, for each message

$$\begin{tikzcd}
& \overline{xy}_0^{i'} \arrow[r, bend left, "H"] & xy_{j+1}^{i} \arrow[dr, bend left, "G"] & \\
xy_j^{i} \arrow[ur, bend left, "G"] & & & \overline{xy}_1^{i'} \arrow[dl, bend left, "H"] \\
& \overline{xy}_{j'}^{i'} \arrow[ul, bend left, "H"] & \cdots \arrow[l, bend left] &
\end{tikzcd}$$

Figure 5.3: Circular behavior of plaintext/ciphertext pairs.

$\overline{xy}_0^i$ in $\overline{X}$, set $\overline{xy}_{j+1}^i = \mathsf{Enc}(K, T', \overline{xy}_j^i)$ with the fixed tweak $T' \in \mathcal{T}$ under the same key K. Let $\overline{XC}$ be the set of chain encryption of elements of $\overline{X}$. Apparently, we have $|XC| = AB$ and $|\overline{XC}| = AB$. Given these 2 sets XC and $\overline{XC}$, we attempt to find a collision between XC and $\overline{XC}$ such that $G(xy_j^i) = \overline{xy}_0^{i'}$ or $G(xy_0^i) = \overline{xy}_{j'}^{i'}$ for $1 \leqslant i, i' \leqslant A$ and $1 \leqslant j, j' \leqslant B$. (See Fig. 5.3.) Upon having a table with inputs to G and H, we can apply the known-plaintext recovery attack on 4-round Feistel networks. The concrete algorithm to collect plaintext/ciphertext pairs is given in Algorithm 8.

We now formally prove useful results for the analysis and success probability of the attack in Algorithm 8.

Let Π be a random permutation on $\{0, \dots, N^2 - 1\}$. Let c_k be the number of cycles of length k in Π. The total number of elements in a cycle of length k (for all k) is equal to N^2, meaning that $\sum_{k=1}^{N^2}(kc_k) = N^2$. It is well-known that the expected number of cycles of length k over a random Π is $\mathbb{E}_{\Pi}(c_k) = \frac{1}{k}$. [1]

In what follows, we show two useful results.

Lemma 6. *For a message* xy^i *picked at random, let* $\mathsf{length}(xy^i)$ *be the length of the cycle that contains* xy^i*. For two messages* xy^i *and* $\overline{xy}^{i'}$ *picked at random, let* E_0 *be an event that* xy^i *and* $\overline{xy}^{i'}$ *are in the same cycle. The expected value of* $\mathsf{length}(xy^i)$ *is* $\mathbb{E}_{xy^i,\Pi}[\mathsf{length}(xy^i)] = \frac{N^2+1}{2}$ *and the expected value of* $\mathsf{length}(xy^i)$ *given* E_0 *is* $\mathbb{E}[\mathsf{length}(xy^i)|E_0] = \frac{2N^2+1}{3}$.

[1]The probability that a given point is in a cycle of length exactly k is $\frac{(N^2-1)\cdots(N^2-k+1)}{N^2(N^2-1)\cdots(N^2-k+1)} = \frac{1}{N^2}$. Hence, the expected number of points in a cycle of length k is $1 = \mathbb{E}_{\Pi}(kc_k)$.

Algorithm 8: FF3 Attack

```
    Input : a tweak bit string $\mathsf{T}$ such that $|\mathsf{T}| = 64$, a key $\mathsf{K}$
 1  $\mathsf{T_L} || \mathsf{T_R} \leftarrow \mathsf{T}$
 2  $\mathsf{T}' \leftarrow \mathsf{T_L} \oplus 4 || \mathsf{T_R} \oplus 4$
 3  foreach $i = 1 \cdots A$ do
 4      pick $xy_0^i$ and $\overline{xy}_0^i$
 5      foreach $j = 1 \cdots B$ do
 6          $xy_j^i = \mathsf{FF3.E}(K, T, xy_{j-1}^i)$
 7          $\overline{xy}_j^i = \mathsf{FF3.E}(K, T', \overline{xy}_{j-1}^i)$
 8      end
 9  end
10  foreach $i, i' = 1 \cdots A$ do
11      foreach $j = 0 \cdots B - M - 1$ do
12          // assume that $G(xy_j^i) = \overline{xy}_0^{i'}$
13          run attack on $G$ with samples $G(xy_{j+k}^i) = \overline{xy}_k^{i'}$ for $k = 0 \cdots B - j$
14          if succeeded, run attack on $H$ with samples $H(G(xy_k^i)) = xy_{k+1}^i$ for
            $k = 0 \cdots B - 1$
15      end
16      foreach $j = 0 \cdots B - M - 1$ do
17          // assume that $G(xy_0^i) = \overline{xy}_{j'}^{i'}$
18          run attack on $G$ with samples $G(xy_k^i) = \overline{xy}_{j+k}^{i'}$ for $k = 0 \cdots B - j$
19          if succeeded, run attack on $H$ with samples $H(G(xy_k^i)) = xy_{k+1}^i$ for
            $k = 0 \cdots B - 1$
20      end
21  end
```

Proof. We use the same notation for c_k as above.

$$\mathbb{E}_{xy^i,\Pi}[\mathsf{length}(xy)] = \mathbb{E}_{xy^i,\Pi}\left[\sum_{k=1}^{N^2} kc_k \frac{k}{N^2}\right] = \sum_{k=1}^{N^2} \mathbb{E}[c_k]\frac{k^2}{N^2} = \sum_{k=1}^{N^2} \frac{k}{N^2} = \frac{N^2+1}{2}$$

We first observe that for any messages xy^i and $\overline{xy}^{i'}$, being in the same cycle of every possible length occurs with probability $\frac{1}{2}$. Then,

$$\Pr[E_0] = \mathbb{E}_\Pi\left[\sum_{k=1}^{N^2} c_k\left(\frac{k}{N^2}\right)^2\right] = \sum_{k=1}^{N^2}\frac{k}{N^4} = \frac{1}{2} + \frac{1}{2N^2} \approx \frac{1}{2}$$

$$\begin{aligned}\mathbb{E}[length(xy^i)|E_0] &= \mathbb{E}_\Pi\left[\sum_{k=1}^{N^2} kc_k\left(\frac{k^2}{N^4}\right)\frac{1}{\Pr(E_0)}\right] = \frac{\sum_{k=1}^{N^2}\frac{k^2}{N^4}}{\Pr(E_0)} \\ &= \frac{2N^2}{N^2+1}\times\frac{(N^2+1)(2N^2+1)}{6N^2} = \frac{2N^2+1}{3}\end{aligned}$$

□

This means that if we pick xy^i and $\overline{xy}^{i'}$ at random and let $xy^j = G^{-1}(\overline{xy}^{i'})$ then xy^i and $\overline{xy}^{i'}$ are in the same cycle with probability close to $\frac{1}{2}$ and we will observe Fig. 5.3. One problem is that the cycle is typically long, i.e. $\frac{2N^2}{3}$ as shown in Lemma 6, but we want that two segments of length B, starting from xy^i and $\overline{xy}^{i'}$, intersect on at least M points. Therefore, we need the probability of two segments overlapping in a cycle of length k on at least M points.

Lemma 7. *Let two segments* $xy^i–\Pi(xy^i)–\Pi^2(xy^i)–\cdots–\Pi^B(xy^i)$ *and* $\overline{xy}^{i'}–\Pi(\overline{xy}^{i'})–\Pi^2(\overline{xy}^{i'})–\cdots–\Pi^B(\overline{xy}^{i'})$ *overlap in a given cycle of length* k *on at least* M *points be the event* E_1^k. *Let* E_1 *be the union of all* E_1^k *for every possible length of* k. *The probability that* E_1 *occurs is equivalent to* $\frac{2(B-M)}{N^2}$ *for* $M = o(N^2)$.

Proof. We use the same notation for c_k as above.

$$\begin{aligned}\Pr[E_1] &= \mathbb{E}_\Pi\left[\sum_{k=M}^{N^2} c_k \Pr[E_1^k]\right] = \mathbb{E}_\Pi\left[\sum_{k=M}^{N^2} c_k\frac{k}{N^2}\frac{min\{k, 2(B-M)+1\}}{N^2}\right] \\ &\sim \frac{2(B-M)}{N^2} \text{ for } M = o(N^2)\end{aligned}$$

$\square$

The probability of success of our FF3 attack depends on $\Pr[E_1]$ and on the success probability of our 4-round recovery attack on Feistel network. More clearly,

$$p_{\mathsf{success}} = \left(1 - (1 - \Pr[E_1])^{A^2}\right) p^{\mathsf{Feistel}}_{\mathsf{success}}$$

which is equivalent to $\left(1 - e^{\frac{-2(B-M)A^2}{N^2}}\right) p^{\mathsf{Feistel}}_{\mathsf{success}}$. Thus, we need $A^2(B-M) \approx N^2$ to obtain a constant p_{success}. We can neglect the cost of the attack on H as we have plenty of samples and we only run it once G is recovered.

Our attack has $2AB$ data complexity. The time complexity is A^2B times the complexity of 4-round recovery attack on Feistel network. To minimize the data complexity $2AB$ with $A^2(B-M) = N^2$ and $B \geqslant M$, we set $B = 2M$, then $A = \frac{N}{\sqrt{M}}$. Therefore, **we have data complexity of FF3 attack as $4N\sqrt{M}$ and time complexity as $2N^2$ times the complexity of 4-round recovery attack on Feistel network and $p_{\mathsf{success}} \approx 1 - e^{-p^{\mathsf{Feistel}}_{\mathsf{success}}}$.**

We fully implemented the attack, but to test its success probability, we could skip some parts of the running time we knew the attack would fail. Namely, in Algorithm 4, we can identify directly which segments overlap (using the key) and proceed directly to the 4-round Feistel attack on the right pair of segments. We show on Table 5.1 the experimental probability of success of the whole attack following the strategies S_j, $j = 1, \dots, 4$. The probability was computed for 10,000 executions. [2] We also took the executions collecting less than M samples, as long as they succeed to recover all tables. Curiously, the $N \leqslant 4$ and $\lambda = 1$ cases seem to take M too low to be able to find cycles. As we can see, the success probability is pretty good (18%–77% for $8 \leqslant N \leqslant 512$) for $\lambda = 1$ and the strategy S_2 collecting the largest connected components in G'.

We conclude that the full attack succeeds with good probability.

[2]Executions of the attack on the 4-round Feistel scheme which we used to fill our Tables 4.2, 4.3 from Chapter 4, are precisely those getting the M samples in this experiment. For some rows with M too large, no experiments collected M pairwise different messages they are thus not reported in the previous table. Nevertheless, our attack may still work, even though we collect less than M samples. This is why they appear on Table 5.1.

N	M	λ	A	B	#run	Pr[succ, S_1]	Pr[succ, S_2]	Pr[succ, S_3]	Pr[succ, S_4]
2	2	0.71	1	4	10000	0.00 %	0.00 %	0.00 %	0.00 %
4	5	0.56	2	10	10000	0.00 %	0.00 %	0.00 %	0.00 %
8	15	0.53	2	30	10000	0.00 %	0.00 %	0.00 %	0.00 %
16	46	0.51	2	92	10000	0.00 %	0.00 %	0.00 %	0.00 %
32	144	0.50	2	288	10000	0.03 %	0.47 %	1.38 %	1.38 %
64	457	0.50	3	914	10000	0.01 %	1.61 %	5.08 %	5.12 %
128	1449	0.50	3	2898	10000	0.00 %	1.51 %	5.25 %	5.73 %
256	4598	0.50	3	9196	10000	0.00 %	0.52 %	3.55 %	4.59 %
512	14597	0.50	3	29194	9996	0.00 %	0.19 %	1.85 %	3.10 %
2	3	1.06	1	6	10000	0.00 %	0.00 %	0.00 %	0.00 %
4	8	0.89	1	16	10000	0.03 %	0.03 %	0.48 %	0.48 %
8	23	0.81	2	46	10000	2.64 %	1.54 %	3.29 %	3.30 %
16	73	0.81	2	146	10000	7.32 %	15.34 %	21.04 %	21.05 %
32	230	0.80	2	460	10000	7.38 %	30.84 %	41.19 %	41.19 %
64	730	0.80	2	1460	10000	5.90 %	39.58 %	50.78 %	50.73 %
128	2318	0.80	2	4636	10000	1.69 %	41.36 %	53.14 %	53.16 %
256	7357	0.80	3	14714	10016	0.68 %	54.52 %	71.68 %	72.12 %
512	23355	0.80	3	46710	831	0.00 %	50.66 %	68.95 %	69.92 %
2	3	1.06	1	6	10000	0.00 %	0.00 %	0.00 %	0.00 %
4	9	1.00	1	18	10000	1.18 %	1.40 %	2.84 %	2.84 %
8	29	1.02	2	58	10000	17.24 %	17.99 %	21.46 %	21.46 %
16	91	1.01	2	182	10000	20.15 %	35.35 %	38.85 %	38.85 %
32	288	1.00	2	576	10000	22.01 %	45.89 %	48.29 %	48.24 %
64	913	1.00	2	1826	10000	28.20 %	54.14 %	54.41 %	54.15 %
128	2897	1.00	2	5794	10000	26.24 %	56.85 %	55.14 %	54.65 %
256	9196	1.00	2	18392	10000	28.09 %	55.87 %	54.64 %	54.14 %
512	29193	1.00	3	58386	609	35.14 %	75.86 %	74.55 %	73.56 %
2	6	2.12	1	12	10000	12.20 %	12.20 %	12.20 %	12.20 %
4	18	2.00	1	36	10000	14.15 %	15.62 %	16.48 %	16.48 %
8	58	2.03	1	116	10000	12.96 %	13.92 %	14.40 %	14.40 %
16	182	2.01	1	364	10000	6.10 %	7.37 %	7.65 %	7.65 %
32	575	2.00	1	1150	10000	2.20 %	3.62 %	3.80 %	3.80 %
64	1825	2.00	2	3650	10000	2.80 %	5.59 %	6.34 %	6.32 %
128	5793	2.00	2	11586	2989	2.24 %	4.05 %	4.35 %	4.32 %
256	18391	2.00	2	36782	188	1.60 %	4.26 %	4.26 %	4.26 %
512	58386	2.00	2	116772	11	9.09 %	9.09 %	9.09 %	9.09 %
2	9	3.18	1	18	10000	12.38 %	12.38 %	12.38 %	12.38 %
4	27	3.01	1	54	10000	13.92 %	15.62 %	16.46 %	16.46 %
8	86	3.02	1	172	10000	12.79 %	13.95 %	14.31 %	14.31 %
16	272	3.01	1	544	10000	5.13 %	6.56 %	6.91 %	6.91 %
32	863	3.00	1	1726	10000	2.04 %	3.25 %	3.47 %	3.46 %
64	2737	3.00	1	5474	9561	1.22 %	2.20 %	2.42 %	2.43 %
128	8689	3.00	1	17378	439	0.23 %	0.68 %	0.91 %	0.91 %
256	27586	3.00	2	55172	11	0.00 %	0.00 %	0.00 %	0.00 %
512	87579	3.00	2	175158	2	0.00 %	0.00 %	0.00 %	0.00 %

Table 5.1: Experimental probability of success in the FF3 attack for various parameters using strategy S_j

5.3 Repairing FF3

As a quick fix, we can suggest changing the length of the tweak in FF3 so that the adversary has no longer control on what is XORed to the round index. The same should hold if some other part of the tweak is XORed to a counter in a CBC mode, as proposed by the authors of the construction [BPS]. We obtain a scheme with a shorter tweak, to which we concatenate the round index instead of XORing it.

The original Luby-Rackoff results [LR88] was extended following this idea by Black and Rogaway [BR02], but the obtained security result is quite weak as we can only prove that for a number of queries $q \ll \sqrt{N}$, the cipher resists to chosen-plaintext attacks, even with only three rounds. By similarly extending the results by Patarin [Pat10], we can obtain that for $q \ll N$, the cipher resists to chosen-plaintext and ciphertext attacks, even with only six rounds. However, this says nothing in the case $q \sim N^{\frac{3}{2}}$ which is the case of our 4-round attack.

5.4 Conclusion

We took the NIST standard FF3 and investigated its security on small domain sizes. We started exploiting that we can permute the round functions due to a bad domain separation in the tweak scheme which uses an XOR with the round index. This permutation leads us to develop a slide attack on FF3, based on our own design for 4-round Feistel schemes attack that works with known plaintexts/ciphertexts. Our FF3 attack works with chosen plaintexts and two tweaks. It improves the recent results from Bellare et al. [BHT16] on data and time complexity to break FF3. Unlike the work by Bellare et al., we focused on a more traditional approach to recover the decryption function for FF3. Our results clearly show that the intended security of FF3 standard has not met for 128-bit of security for small domains. ANSI X9.119 is currently considering adopting FF3 with our proposed repair. NIST may update its standard.

References

[AB96] Ross Anderson and Eli Biham. Two practical and provably secure block ciphers: Bear and lion. In Dieter Gollmann, editor, *Fast Software Encryption: Third International Workshop Cambridge, UK, February 21–23 1996 Proceedings*, volume 1029, pages 113–120, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[ABE^{+}13] Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, and Ramarathnam Venkatesan. Orthogonal security with cipherbase. In *CIDR*, 2013.

[AES01] *The Advanced Encryption Standard (AES).* National Institute of Standards and Technology, 2001.

[AIK^{+}00] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis. In Douglas R. Stinson and Stafford E. Tavares, editors, *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000, Proceedings*, volume 2012, pages 39–56. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[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. ACM, 2004.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. *Deterministic and Efficiently Searchable Encryption*, pages 535–552. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. pages 224–241, 2009.

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

[BDV16] F. Betül Durak and Serge Vaudenay. *Circular Security Reconsidered*, pages 3–19. Springer International Publishing, Cham, 2016.

[BHT16] Mihir Bellare, Viet Tung Hoang, and Stefano Tessaro. Message-recovery attacks on Feistel-based Format Preserving Encryption. In *23th CCS Proceedings*, 2016.

[BLP16] Alex Biryukov, Gaëtan Leurent, and Léo Perrin. Cryptanalysis of feistel networks with secret round functions. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015: 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566, pages 102–121. Springer International Publishing, 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. pages 563–594, 2015.

[BP15] Alex Biryukov and Léo Perrin. On reverse-engineering S-boxes with hidden design criteria or structure. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215, pages 116–140. Springer International Publishing, 2015.

[BPS] Eric Brier, Thomas Peyrin, and Jacques Stern. BPS: a Format-Preserving Encryption Proposal. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf.

[BR02] John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In Bart Preneel, editor, *Topics in Cryptology — CT-RSA 2002: The Cryptographers' Track at the RSA Conference 2002 San Jose, CA, USA, February 18–22, 2002 Proceedings*, volume 2271, pages 114–130, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[BRRS09] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography: 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867, pages 295–312. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[BRS] Mihir Bellare, Phillip Rogaway, and Terence Spies. The FFX mode of operation for format-preserving encryption. Draft 1.1. Submission to NIST, Feb. 2010. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf.

[BS97] Michael Brightwell and Harry E. Smith. Using Datatype-Preserving Encryption To Enchance Data Warehouse Security. Available at: http://csrc.nist.gov/nissc/1997/proceedings/141.pdf, 1997.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 668–679. ACM, 2015.

[CJJ$^+$14] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS*, volume 14, pages 23–26, 2014.

[CLWW16] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, 2016. To appear.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251 – 280, 1990.

[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*, CCS '16, pages 1155–1166, New York, NY, USA, 2016. ACM.

[DDKS12] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417, pages 719–740, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[DDKS15] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. New attacks on Feistel structures with improved memory complexities. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215, pages 433–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[DES] *Data Encryption Standard, National Bureau of Standards, NBS FIPS PUB 46, January 1977.* National Bureau of Standards, U.S. Department of Commerce.

[DH77] W. Diffie and M. E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, June 1977.

[DVa] F. Betül Durak and Serge Vaudenay. Breaking the FF3 format preserving encryption. Proceedings of ESC 2017: `https://www.cryptolux.org/mediawiki-esc2017/images/8/83/Proceedings_esc2017.pdf`.

[DVb] F. Betül Durak and Serge Vaudenay. Breaking the FF3 format preserving encryption standard over small domain. To appear in CRYPTO 2017.

[DVc] F. Betül Durak and Serge Vaudenay. Cryptanalysis of feistel networks. To appear.

[enc15] encrypted-bigquery-client. `https://github.com/google/encrypted-bigquery-client`, 2015.

[ER59] Paul Erdős and Alfred Renyi. *On Random Graphs I*, pages 290–297. Publicationes Mathematicae, 1959.

[Fer99] Bruce Schneier; John Kelsey; Doug Whiting; David Wagner; Chris Hall; Niels Ferguson. *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*. John Wiley & Sons, New York City, 1999.

[GHH$^+$14] Patrick Grofig, Isabelle Hang, Martin Härterich, Florian Kerschbaum, Mathias Kohler, Andreas Schaad, Axel Schröpfer, and Walter Tighzert. Privacy by encrypted databases. In *Privacy Technologies and Policy - Second Annual Privacy Forum, APF 2014, Athens, Greece, May 20-21, 2014. Proceedings*, pages 56–69, 2014.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996.

[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.

[HKR17] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. AEZ v5: Authenticated encryption by enciphering, 2017.

[HR10] Viet Tung Hoang and Phillip Rogaway. On generalized Feistel networks. In *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223, pages 613–630, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[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.

[IS13] Takanori Isobe and Kyoji Shibutani. Generic key recovery attack on Feistel scheme. In *Advances in Cryptology - ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269, pages 464–485, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Ker15] Florian Kerschbaum. Frequency-hiding order-preserving encryption. pages 656–667, 2015.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 965–976, New York, NY, USA, 2012. ACM.

[KS14] Florian Kerschbaum and Axel Schröpfer. Optimal average-complexity ideal-security order-preserving encryption. pages 275–286, 2014.

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

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, 2011.

[Luc96] Stefan Lucks. Faster luby-rackoff ciphers. In Dieter Gollmann, editor, *Fast Software Encryption: Third International Workshop Cambridge, UK, February 21–23 1996 Proceedings*, volume 1039, pages 189–203, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[MCO$^+$15] Charalampos Mavroforakis, Nathan Chenette, Adam O'Neill, George Kollios, and Ran Canetti. Modular order-preserving encryption, revisited. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 763–777, 2015.

[NIS16] *Recommendation for Block Cipher Modes of Operation: Methods for Format Preserving Encryption.* National Institute of Standards and Technology, 2016.

[NKW15] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. pages 644–655, 2015.

[NPG14] Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. Dynamic searchable encryption via blind storage. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 639–654, Washington, DC, USA, 2014. IEEE Computer Society.

[NVP13] Valérie Nachef, Emmanuel Volte, and Jacques Patarin. Differential attacks on generalized Feistel schemes. In *Cryptology and Network Security: 12th International Conference, CANS 2013, Paraty, Brazil, November 20-22. 2013. Proceedings*, volume 8257, pages 1–19. Springer International Publishing, 2013.

[Pat92] Jacques Patarin. New results on pseudorandom permutation generators based on the DES scheme. In *Advances in Cryptology — CRYPTO '91: Proceedings*, volume 576, pages 301–312, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[Pat08] Jacques Patarin. Generic attacks on Feistel schemes. http://eprint.iacr.org/2008/036, 2008.

[Pat10] Jacques Patarin. Security of balanced and unbalanced feistel schemes with linear non equalities. http://eprint.iacr.org/2010/293, 2010.

[PLZ13] Raluca A. Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. pages 463–477, 2013.

[PNB06] Jacques Patarin, Valérie Nachef, and Côme Berbain. Generic attacks on unbalanced Feistel schemes with contracting functions. In *Advances in Cryptology – ASIACRYPT 2006: 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006. Proceedings*, volume 4284, pages 396–411, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[PRZB11] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*, pages 85–100, 2011.

[PW16] David Pouliot and Charles V Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1341–1352. ACM, 2016.

[Riv95] Ronald L. Rivest. The RC5 encryption algorithm. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings*, volume 1008, pages 86–96, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[Rog] Phillip Rogaway. A Synopsis of Format Preserving Encryption. http://web.cs.ucdavis.edu/~rogaway/papers/synopsis.pdf.

[Sal95] A. I. Saltykov. The number of components in a random bipartite graph. *Discrete Mathematics Applications*, 5:515–523, 1995.

[Sch94] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption: Cambridge Security Workshop Cambridge, U. K., December 9–11,1993 Proceedings*, volume 809, pages 191–204, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[SCSL11] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious ram with o((logn)3) worst-case cost. In *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'11, pages 197–214, Berlin, Heidelberg, 2011. Springer-Verlag.

[SK96] Bruce Schneier and John Kelsey. Unbalanced feistel networks and block cipher design. In *Proceedings of the Third International Workshop on Fast Software Encryption*, volume 1039, pages 121–144, London, UK, 1996. Springer-Verlag.

[SM88] Akihiro Shimizu and Shoji Miyaguchi. Fast data encipherment algorithm FEAL. In *Advances in Cryptology — EUROCRYPT' 87: Workshop on the Theory and Application of Cryptographic Techniques Amsterdam, The Netherlands, April 13–15, 1987 Proceedings*, volume 304, pages 267–278, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[Spi08] Terence Spies. Format preserving encryption. Unpublished white paper, available at: https://www.voltage.com/wp-content/uploads/Voltage-Security-WhitePaper-Format-Preserving-Encryption.pdf, 2008.

[SPS14] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *NDSS*, volume 14, pages 23–26, 2014.

[SSS11] Emil Stefanov, Elaine Shi, and Dawn Song. Towards practical oblivious RAM. *CoRR*, abs/1106.3652, 2011.

[SvDS^{+}13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 299–310, New York, NY, USA, 2013. ACM.

[SWP00a] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, pages 44–, Washington, DC, USA, 2000. IEEE Computer Society.

[SWP00b] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. pages 44–55, 2000.

[ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. *IACR Cryptology ePrint Archive*, 2016:172, 2016.