INDIFFERENTIABILITY ANALYSIS OF SYMMETRIC KEY CIPHERS

A THESIS SUBMITTED TO

THE INDIAN STATISTICAL INSTITUTE

IN THE SUBJECT OF CRYPTOLOGY

FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

IN THE RESEARCH GROUP APPLIED STATISTICS UNIT.

By

Sayantan Paul

Supervisor: Prof. Mridul Nandi

July 2025

Statement

I declare that the thesis titled "Indifferentiability Analysis of Symmetric Key Ciphers" and the work presented in it are my own, and were produced by my own original research. I confirm that this research was done wholly while in candidature for a doctoral degree at Indian Statistical Institute; that no part of this thesis has previously been submitted for a degree or any other qualification at this institute or any other institution; that wherever I have used or developed on the published work of others, this is always clearly attributed; that wherever I have quoted from the work of others, the source is always clearly cited; that with the exception of such quotations, this thesis is entirely my own work; that I have acknowledged all main sources of help; and that wherever the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself, and have obtained explicit permission from the others to use the joint work in this thesis.

Sayantan Paul

July 11, 2025

Abstract

The thesis presented here analyses the security of certain selected symmetric key ciphers - The ciphers analyzed are the 2 and 3-round Confusion-Diffusion Network, the 3-round Cascade Cipher with two independent keys, and the Feistel Construction with 7 and 8 rounds.

Substitution Permutation Networks (SPNs) are widely used in the design of modern symmetric cryptographic building blocks. Attacks against the 2-round Confusion-Diffusion Network construction have been exhibited by Dodis et al. (2016a) in their Eurocrypt 2016 paper titled 'Indifferentiability of Confusion-Diffusion Networks', and by Da, Xu and Guo (2021b) in their paper 'Sequential Indifferentiability of Confusion-Diffusion Networks'. Both attacks mentioned above were incomplete/erroneous. As part of our first result, we provide a corrected attack on the 2-round NLCDN. Our attack on the 2-round CDN is primitive-construction-sequential, implying that the construction is not secure even in the weaker sequential indifferentiability setting of Mandal, Patarin and Seurin (2012a).

The second part of our first results focuses on Cascade Ciphers. We present an attack on the 3-round cascade construction employing any $2n$-bit to $3n$-bit non-idealized key scheduling function, generalising the heuristic attack based on 'certain' stronger key schedules as described by Guo, Lin and Liu (2016) in 'Revisiting Cascade Ciphers in Indifferentiability Setting'.

Next, as a follow up of the above work, we show that the 3-round Confusion-Diffusion Network construction with linear diffusion layers is indifferentiable from an ideal permutation. This, in conjunction with the previous negative result, shows the tightness of our indifferentiability result.

The final work in this thesis explores the Feistel construction. There have been a series of studies on whether an ideal cipher can be built from a random oracle using a Feistel network. We present a general proof framework that lets us prove the indifferentiability of 7 or more rounds of Feistel. In particular, this is the first indifferentiability proof for 7-round Feistel, and in addition, the 8-round proof is considerably simpler than the previously-known proof.

List of Publications

The following is a list of accepted papers and publications, in order of appearance in the thesis:

1. Nandi, M., Paul, S., Saha, A. (2023). 'Indifferentiability of the Confusion-Diffusion Network and the Cascade Block Cipher' - Published in Codes, Cryptology and Information Security: 4th International Conference, C2SI 2023, 10.1007/978-3-031-33017-9_12

2. Bhaumik, R., Nandi, M., Paul, S., Saha, A. (2024). Indifferentiability of 3-Round Confusion-Diffusion Networks - Published in: Galdi, C., Phan, D.H. (eds) Security and Cryptography for Networks. SCN 2024. Lecture Notes in Computer Science, vol 14974. Springer, Cham. 10.1007/978-3-031-71073-5_7

Acknowledgements

I would like to start off by expressing my deepest gratitude towards my mother, and my brother. They have taught me that I should try to be kind whenever I can, to whoever I can. And that, I believe, is worth more than anything else I've learnt in my life till date.

This thesis would be a blank template if not for my supervisor Prof. Mridul Nandi. He has always been a beacon of encouragement and enthusiasm for me, as I'm certain he has been for all my seniors, peers, and juniors that have worked, or are currently working with him. I have not always been the model subordinate, and I would like to genuinely thank him for the patience he has shown towards me, and I feel truly obliged to have known and worked with a person of his knowledge, intellect, and sheer niceness.

I would be unkind to leave out all my colleagues, who have always been keen to lend me an attentive ear and a helping hand whenever I've needed. I would also like to thank the office staff and other workers in our institute, who play a crucial role in keeping the metaphoric machinery running, of which, I'm undoubtedly a beneficiary. Thank you for all your dedication.

Lastly, I would like to mention my friends. A word of thanks does not seem to do justice to the care and support that I've been bestowed with. Nonetheless, thank you for loving me when it was optional.

Contents

List of Figures

Chapter 1

Introduction

To begin the thesis, we take a brief look at the history of Cryptology, starting off as a practical tool in the human utility belt, evolving to what it is today, a field of cutting edge academic research and real world applications. Subsequently, we attempt to put the thesis into context, and give a brief description of the motivation behind the included works.

1.1 Motivation of the Thesis

In a nutshell, this thesis focuses on how various symmetric key modes perform under the indifferentiability paradigm. Broadly speaking, the idea of indifferentiability aids us in making two types of deductions:

- On the equivalence of different constructions, and
- On the domain extending capabilities of a construction

Rigorously demonstrating that a construction 'does its job securely' is an essential component of cryptographic protocol design. Introduced by Maurer, Renner and Holenstein (2004a), indifferentiability is the natural extension of the indistinguishability security notion, and is based on the ideas from the Universal Composition framework first proposed by Canetti (2000) and on the model by Pfitzmann and Waidner (2004). What makes indifferentiability a stronger security notion than classical indistinguishability is that, in the indifferentiability game, an adversary has access to the primitive as well. Intuitively, this widens the scope of adversarial information, and hence any construction secure in this notion provides 'greater' security.

Equivalence of Constructions. Coron et al. (2005b) famously proved that it is possible to replace a random oracle (taking arbitrary long inputs) by a block cipher-based construction at Crypto '05. Using the indifferentiability framework, Coron et al. (2005b) formally defined the idea of 'indifferentiable construction' of one ideal primitive (e.g, a random oracle) from another ideal primitive (e.g, an ideal block cipher) by exhibiting a block cipher based Merkle-Damgård that mimics a random oracle. Generally speaking, they showed that given a construction $\mathcal{H}$ satisfies this security notion, any scheme that is secure in the former ideal model remains secure in the latter model, when instantiated using $\mathcal{H}$. The other side of the coin, i.e., constructing an ideal cipher from a random oracle, was first proposed by Coron, Patarin and Seurin (2008a) at Crypto '08. They showed that 6 rounds of the Feistel network instantiated with random oracles is indifferentiable from an ideal block cipher. Unfortunately, Holenstein, Künzler and Tessaro (2011a) pointed out a flaw in their simulator, which reverted the equivalence of the Random Oracle Model and the Ideal Cipher Model back into the realm of an open question. Eventually, higher rounds of the Feistel network were shown to achieve this feat of security. Coron et al. (2016) proved the indifferentiability of 14 rounds of Feistel in 2014, and the following year, Dachman-Soled, Katz and Thiruvengadam (2015) showed that the same is true for the 10-round version. The latest results by Dai and Steinberger (2016a) have brought the round complexity of a secure Feistel to 8 rounds.

Domain Extendability. The notion of domain extendability is relevant in many areas of cryptography, such as hash functions, pseudo-random functions, and strong pseudo-random permutations (Luby and Rackoff (1988)). Consider a building block $\mathcal{H}$ defined for a small and fixed bit size domain. Domain extension is essentially the iterative use of $\mathcal{H}$ to extend similar functionalities as that of the building block over arbitrary domain. Domain extenders were initially studied as collision-resistant hash functions by Damgård (1990) and Merkle (1990), as pseudo-random functions by Bellare, Canetti and Krawczyk (1996), as Message Authentication Codes or MACs by Maurer and Sjödin (2005) and An and Bellare (1999), and as universal one-way hash functions by Bellare and Rogaway (1997) and Winternitz (1984). It was Coron et al. (2010) that first described a domain extender for an ideal cipher by exhibiting a construction that is indifferentiable from a $2n$-bit ideal cipher, given an n-bit ideal cipher. From the discussion above on equivalence of constructions, it is worth noting that one could, in theory, use the construction proposed by Coron et al. (2005b) to get a random oracle with n-bit output, and then use the 10-round Feistel by Dachman-Soled, Katz and Thiruvengadam (2015) to obtain an ideal cipher with $2n$-bit input/output. In practice, however, the construction by Coron et al.

(2010) is way more efficient. Taking a step back, it is also pertinent to observe that the results of Coron et al. (2005c) double as a secure 'domain extender' for the random oracle, which is an interesting crossover result.

The literature covered in this thesis makes similar contributions, in terms of results. The (negative) results of the first chapter demonstrate the domain extending (in)capabilities of the 3-round Cascade and the 2-round Confusion-Diffusion network (or, CDN). Firstly, in conjunction with the results on the 4-round Cascade by Guo, Lin and Liu (2016), our Cascade result shows the tightness of the design with respect to the number of rounds required for secure domain extension. Secondly, the 2-round CDN result considered alongside the (positive) results of the second chapter on the indifferentiability security of the 3-round CDN, we see that 3 rounds is necessary and sufficient to build a secure n-bit to $2n$-bit domain extender in the Confusion-Diffusion paradigm. Finally, the (positive) results of chapter 5 on the indifferentiability of 7 rounds of Feistel optimizes the existing literature on building secure $2n$-bit permutations from n-bit random functions by reducing the required number of rounds to 7. Further, the graph theoretic framework employed for the security proof is highly likely to be adaptable towards proving the indifferentiability of 6 rounds of Feistel, which, in tow with the negative result of Coron, Patarin and Seurin (2008a), would close the gap in academic theory with respect to its indifferentiability from an ideal cipher.

1.2 A Brief Look Through History

1.2.1 Pre-Modern Cryptography

The advent of civilisation from hunter gatherers to farming and metallurgical societies, and subsequently to the early dynastic period of kings and conflict necessitated the ability of secrecy. While cryptography took different forms in early civilizations, there is evidence of cryptographic techniques as early as 1900 BCE Egypt - An inscription carved into the main chamber of the tomb of nobleman Khnumhotep II. The hieroglyphics used were different than the 'usual' ones, in a process later termed Symbol Replacement. This, however, wasn't necessarily a secret code. Rather, the form of writing was changed to make it appear more regal sophisticated. In 1500 BCE, a Mesopotamian scribe used cryptography to conceal a formula for pottery glaze. This example is the first known use of cryptography to hide secret information. There has been evidence of use of cryptography in almost every major early civilization. In 500 BCE Greece, the Spartan military used a substitution cipher called the *Scytale*, where a strip of parchment wrapped around a rod would reveal a message when unwound. In 100 BCE, Julius Caesar used

a form of encryption to share secret messages with his army generals at war - What is famously known today as the *Caesar cipher*. Modern examples of the cipher's use range from the Russian army, which employed it as a replacement for more complicated ciphers which their troops had trouble mastering, to lovers exchanging secret messages through the personal advertisement section of The Times (*The Codebreakers*, David Kahn, 1967). On this side of the world, evidence of an ancient treatise on statecraft, political science, economic policy and military strategy called 'Arthashashtra' written by Chanakya (but likely contributed to by multiple authors) has been found dating back to the first century BCE, which describes how assignments were given to spies in "secret writing".

Notable mentions in the Common Era include the use of cryptology by Islamic scholars, notably Al-Kindi, who wrote extensively on cryptanalysis and furthered the field by developing frequency analysis techniques (850 CE), by European monasteries and religious institutions for correspondence, and the famous Voynich manuscript, a mysterious fifteenth-century CE codex belonging to the library of the Holy Roman Emperor Rudolf II which remains undeciphered to this day. In and around 1500 CE, the advent of polyalphabetic ciphers developed Cryptology further, most famously by Leon Battista Alberti and later refined by Johannes Trithemius. In 1553 CE, an improvement on the Caesar cipher was described by Giovan Battista Bellaso, where each letter of the plaintext is encoded with a different Caesar cipher, whose increment is determined by the corresponding letter of another text, the key (a special case of a polyalphabetic substitution cipher). This encryption scheme, albeit easy to understand and implement, remained unbroken for 3 centuries, and earned the title 'Le Chiffrage Indéchiffrable', french for 'The Indecipherable Cipher'). In the 19th century CE, the scheme was misattributed to Blaise de Vigenère, and so acquired its present name, the *Vigenère cipher*. The works of English polymath Charles Babbage (Kasiski's test - later attributed to Friedrich Kasiski) and Dutch linguist Auguste Kerckhoffs (best known for Kerckhoffs' principle) in the 19^{th} century CE advanced the field of Cryptanalysis greatly. The *Hebern rotating machine* created by Edward Hebern in 1917 CE, Illinois marked the first time electrical circuitry was used in a cipher device, as it combined the mechanical parts of a standard typewriter and the electrical parts of the electric typewriter. The World Wars saw another leap in the field with the development of mechanical encryption machines such as the *Enigma machine* by German engineer Arthur Scherbius, and the electro-mechanical device used by British cryptologists to help decipher German Enigma-machine-encrypted secret messages, the *Bombe*, which in turn was developed from a device known as the *Bomba*, designed in Poland at the Biuro Szyfrów (Cipher Bureau) by cryptologist Marian Rejewski.

1.2.2 Modern Cryptography

Modern cryptography hinges on using specialised algorithms that utilise an additional input - the 'key', to encrypt and decrypt information. Broadly speaking, there are two main types of cryptosystems -

1. Symmetric Key Systems: Also known as conventional or single-key encryption, they use the same 'secret' key for encryption and decryption. The (random) choice of said key essentially lends its randomness to the ciphertext. The two main techniques used by Symmetric Key Cryptography are Substitution (e.g., mono/polyalphabetic ciphers, the *One-Time Pad*), and Transposition (e.g., the *Rail Fence* cipher). Encryption is achieved by using one of the two following primitives:

    - *Block Ciphers:* A deterministic, length-preserving system that takes on fixed-length groups of bits (called blocks) as input, and outputs a sequence of blocks of encrypted data in a specific order. The system has separate algorithms for encryption and decryption, albeit, using the same key. Famous examples of block cipher-based cryptosystems include the *Data Encryption Standard* (DES, for short), and the *Advanced Encryption Standard* (AES, for short)
    - *Stream Ciphers:* A system that utilises a pseudo-random cipher stream (called *keystream*), and a unique randomly generated number (called *nonce*), to encrypt the plaintext one bit at a time with the corresponding bit of the keystream to generate the ciphertext stream. Famous examples include the *Rivest Cipher 4*, and the *Salsa20*

2. Assymmetric Key Systems: First proposed in the 1970's by James H. Ellis, a British cryptographer at the UK Government Communications Headquarters (GCHQ), these systems use a pair of keys, called the *public* and *private* keys, one for encryption, and the other for decryption. As the names suggest, the public key is common knowledge, whereas the private key is kept secret by the decrypting entity. Key pairs are generated using hard-to-invert functions called *one-way functions*, and it is impossible to figure out the decryption key just by knowing the encryption key and the cryptographic algorithm. Assymetric systems address two major challenges faced in symmetric cryptography:

    - *Public key encryption,* i.e., the key used for encryption can be publicly shared and parties do not require access to the secret key in order to initiate communication, and
    - *Digital signatures,* i.e., a message signed with the sender's private key and can be

verified by anyone, provided they have access to the sender's public key - This verification proves that the sender is very likely to be the person associated with the public key (since they had access to the private key), and that the signature was prepared for that exact message (since a signature that passes verification with the public key on one message will not pass verification with the public key on other messages)

Examples of well-regarded asymmetric key techniques for varied purposes include the *Diffie–Hellman* key exchange protocol proposed by Whitfield Diffie and Martin Hellman in 1976, the *ElGamal* encryption system described by Taher Elgamal in 1985, and the RSA (Rivest–Shamir–Adleman) encryption system by Ron Rivest, Adi Shamir and Leonard Adleman in 1977

The works in this thesis are on the topic of symmetric key cryptography, specifically, on block ciphers. Going forward, we shall restrict the discussion to block ciphers and their security aspects.

1.3 Block Ciphers

Length-preserving symmetric-key encryption schemes with a fixed-length input—which go by the moniker block cipher in cryptographic parlance—have long established themselves as the fundamental building blocks of symmetric cryptography.

1.3.1 History and Overview

A block cipher E takes a randomly-chosen secret key k and a plaintext block m and outputs a ciphertext block $c = E_k(m)$. If E is a 'strong' block cipher, we'll expect c to be almost uncorrelated with m, and randomly distributed among all possible ciphertext blocks; we'll also expect $c' = E_k(m')$ to be drastically different from c as long as m and m' differ by even a single bit. Notable examples of block ciphers include:

- *Feistel Networks:* Developed in the late 1940's and early 1950's by Horst Feistel, it is based on repeated rounds of substitutions and permutations
- *Lucifer Cipher:* Developed by IBM in the early 1970s, Lucifer was one of the first practical implementations of a block cipher based on Feistel's structure, and was the building block for the fabled DES

- *The Data Encryption Standard:* Developed by IBM based on Horst Feistel's design in the 1970's, DES was submitted to the National Bureau of Standards (now, the NIST) to protect sensitive electronic government data and a modified version was accepted in 1976. In 1977, it became the official Federal Information Processing Standard (FIPS) of the United States

- *The Advanced Encryption Standard:* A variant of the *Rijndael block cipher* developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, AES was the natural successor of DES, and was adopted by the *National Institute of Standards and Technology* (NIST, for short) as its federal encryption standard in 2001

Intuitively, the longer a key is, the more secure is the ciphertext. For example, a brute force attack would be more difficult to mount owing to the larger size of the key set. With modern technology developing exponentially, brute force attacks have become more effective in deciphering ciphertexts yielded by algorithms using smaller length keys. The most famous instance of such vulnerabilities being exploited is the DES, which uses a symmetric-key algorithm and has a key-length of 56 bits. It was broken by differential cryptanalysis in 1990 by Eli Biham and Adi Shamir. In January 1999, *distributed.net* and the *Electronic Frontier Foundation* collaborated to publicly break a DES key in 22 hours and 15 minutes. Other examples of block ciphers being vulnerable to cryptanalytic attacks include the 1987 design Feistel-based *FEAL* which was aimed to replace DES, the Soviet and Russian government standard *GOST*, another Feistel-based design that was discovered to have significant flaws in 2011, and the *Nimbus* block cipher, a 2000 design by Alexis Machado operating on blocks of 64 bits and consisting of 5 rounds of encryption.

The design of strong block ciphers has been a focal point of research in symmetric cryptography for several decades. It was observed early that when designing an algorithm from scratch, the degree of bit-scrambling desirable for a block cipher is easier to achieve on a smaller number of bits than would be practical as block length of a block cipher. Thus it was found convenient to combine such small scrambling functions with an overarching network of weaker functions, so that enough iterations of this network lead to a satisfactory degree of scrambling over the entire state. We'll call the choice of the combining network a design paradigm. Historically, most block cipher designs have subscribed to one of the following four paradigms:

- Feistel networks (named after Horst Feistel) were widely used in early block ciphers like Lucifer Sorkin (1984), DES National Institute of Standards and Technology (1977), Magma National Soviet Bureau of Standards (1989), and FEAL Shimizu and Miyaguchi

(1988) to name a few, and have remained in use through different eras—Camellia Aoki et al. (2001) was proposed in 2001 and Simon Beaulieu et al. (2015) is quite recent;

- Unbalanced Feistel networks (and its generalisations) were used in Khufu and Khafre Merkle (1991), MacGuffin Blaze and Schneier (1995), and Skipjack National Institute of Standards and Technology (1998), among others;

- Lai-Massey networks (named after Xuejia Lai and James Massey)—while less popular than Feistel networks—have found use in the design of several block ciphers such as IDEA Lai and Massey (1991), MESH Nakahara et al. (2004), FOX Junod and Vaudenay (2005), and WIDEA Junod and Macchetti (2009);

- Substitution-Permutation Networks (SPNs) have emerged as the most popular block cipher design paradigm over the last couple of decades, following the remarkable resistance of Rijndael Daemen and Rijmen (2000) (standardised as AES in 2001) to all known cryptanalytic techniques; apart from Rijndael, SPNs have been used in the design of Serpent Biham, Anderson and Knudsen (1998), PRESENT Bogdanov et al. (2007), RECTANGLE Zhang et al. (2015), Kuznyechik National Soviet Bureau of Standards (2015), Kalyna Oliynykov et al. (2015), and many other block ciphers.

Block ciphers account for the bulk of data encryption and data authentication occurring in cryptography and also play a critical role in the design of hash functions Brachtl et al. (1990); Hirose (2006); Lai and Massey (1993); Preneel, Govaerts and Vandewalle (1994). They are typically used in various modes of operation, e.g., *Electronic Code Book* (or ECB), *Cipher Block Chaining* (or CBC), *Cipher Feedback Mode* (or CFB), *Output Feedback Mode* (or OFB), and *Counter Mode* (or CTR) to provide different properties such as confidentiality, integrity, and authenticity. Block ciphers continue to face challenges from advances in cryptanalysis, including attacks on reduced-round versions, side-channel attacks, and potential threats from quantum computing.

1.4 Contributions

1.4.1 The 3 Round Cascade and The 2 Round Confusion-Diffusion Network.

The contribution in this chapter is twofold:

1. The attack on $\mathsf{CDN}_{2,2}$ provided by Dodis et al. Dodis et al. (2016a) is not sequential (Their distinguisher may possibly make queries first to the construction, then to the primitive, and then to the construction again, depending on the primitive query outputs and the diffusion layer). The security proof by Guo et al Da, Xu and Guo (2021a) establishes the sequential indifferentiability Mandal, Patarin and Seurin (2012a) of $\mathsf{CDN}_{3,2}$. However, the corresponding primitive-construction-sequential attack for $\mathsf{CDN}_{2,2}$ given in Da, Xu and Guo (2021a) makes an incorrect assumption about the class of D-boxes, the non-cryptographic diffusion layer of the construction (explained in detail in section 4.2.1). This begs the question if the sequential indifferentiability of $\mathsf{CDN}_{2,2}$ is indeed a tight result, in terms of the number of rounds. We aim to answer this question by exhibiting a distinguisher making at most 10 queries which succeeds against any simulator with an advantage of at least $1 - \mathcal{O}(q_{\mathcal{S}}^2/2^{2n})$, where $q_{\mathcal{S}}$ is the simulator query complexity. This shows that, even with respect to a weaker notion of indifferentiability (sequential), the $\mathsf{CDN}_{2,2}$ construction is not secure. This implies that the other security guarantees that follows from sequential indifferentiability, e.g., correlation intractability Canetti, Goldreich and Halevi (2004), as shown by Mandal, Patarin and Seurin (2012a), also may not hold for $\mathsf{CDN}_{2,2}$.

2. The generalisation made on the key scheduling function by Guo, Lin and Liu (2016) cover a decent range of possible key schedules, the question of indifferentiability is left unanswered for a substantially wider class of key scheduling functions. They also provided the indifferentiability security proof for 4-round Cascade using alternating keys.

   To that end, we close the gap by showing an attack on the 3-round Cascade construction with a generalised $2n$-bit to $3n$-bit non-idealized key schedule. The implication here is that 3-round Cascade cannot be used for the domain extension of key schedules (from $2n$ bits to $3n$ bits), and have to rely on the 4-round construction. Our distinguisher makes 12 queries and succeeds against any simulator with an advantage of at least $1/2 - \mathcal{O}(q_{\mathcal{S}}/2^{2n})$, where $q_{\mathcal{S}}$ is the simulator query complexity.

1.4.2 The 3 Round Confusion-Diffusion Network.

We study the indifferentiability of 3-round CDNs with two independent public permutations over $\mathbb{GF}(2^n)$ in each confusion layer and identical diffusion layers implementing a very simple

$\mathbb{GF}(2^n)$-linear function, defined by

$$(x, y) \mapsto (x \oplus y, x \oplus \alpha \cdot y)$$

for a fixed α different from the 0 or 1 of $\mathbb{GF}(2^n)$.

Given six public permutations $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4, \mathsf{S}_5, \mathsf{S}_6$ over $\mathbb{GF}(2^n)$, we show that $\mathsf{CDN}^{\mathsf{S}_1,\ldots,\mathsf{S}_6}_{,3\times 2}$ is indifferentiable from a public random permutation Π over $\mathbb{GF}(2^{2n})$ by explicitly exhibiting a simulator Sim^{Π} for $\mathsf{S} := (\mathsf{S}_1, \ldots, \mathsf{S}_6)$ which is resistant to indifferentiability attacks, thereby establishing for the first time that 3 rounds are enough to achieve full indifferentiability. (i.e., indifferentiability without sequential assumptions)

Dodis et al. showed that 2-round CDNs with linear d-boxes was not secure via an indifferentiability attack. Further, they studied the indifferentiability of several CDNs with five or more rounds, and referred to the indifferentiability of 3-round CDNs and 4-round CDNs as an open problem Dodis et al. (2016b). Subsequently, Da et al. exhibited an indifferentiability attack Da, Xu and Guo (2021b) on 2-round CDNs with non-linear diffusion layers, also proving the sequential indifferentiability of a 3-round CDN with non-linear layers. The result pertaining to 2-round CDNs in chapter 4 in conjunction with the results of the current chapter show a tightness of sorts, in terms of the number of rounds required for a secure confusion-diffusion network.

While our bound is not good, providing only security up to $2^{n/15}$ queries, we believe these bounds can be improved by assuming suitable combinatorial properties for the P-boxes. Moreover, we point out that such impractical bounds are not uncommon in the study of indifferentiability, as exemplified by the long line of research on the indifferentiability of Feistel networks.

1.4.3 'r' Round Feistel, $r \geq 7$.

We provide a general framework for studying the indifferentiability of 'r' rounds of Feistel, and explicitly prove the indifferentiability security of Feistel Networks with rounds 7 and 8 by exhibiting a simulator that succeeds against any polynomial time adversary with the probability $1 - \mathcal{O}(q^{13}/2^n)$. We also introduce the novel idea of 'line graphs' which equips us for the generalisation to 'r' rounds. Our results bridge the gap in the Feistel literature, with the exception of the 6-round Feistel.

Previous results show the public indifferentiability of the 6-round Feistel (Mandal et al. Mandal, Patarin and Seurin (2012b)). Coron et al. described an attack on the 5-round construction which shows that it is not indifferentiable from an ideal permutation Coron, Patarin and Seurin

(2008b), and also provided a security proof for 6-round Feistel that was famously proved to be erroneous by Holenstein et al. (Holenstein, Künzler and Tessaro (2011b)). The error was due to unaccounted for dependencies in the internal layers of the round functions due to specific types of relations in the outer layers. The general line graph framework provided in this work accounts for such constraints, and enables a simpler viewing of the round function inter-dependencies through graph theoretic techniques.

In terms of positive results, in what has so far been the latest work in this series, Dai and Steinberger (2016b) established the indifferentiability of 8 rounds by optimizing their simulator for 10 rounds from Dai and Steinberger (2015). The indifferentiability of the 6-round Feistel, however, has remained an open question to this day. The next natural step would be to use the 'r' round framework for the indifferentiability of Feistel to prove the security for 6 rounds.

Chapter 2

Literature Survey

The Indifferentiability Framework. The theoretical foundations of indifferentiability were established by Maurer, Renner and Holenstein (2004a), thereby providing a formal basis for evaluating whether a cryptographic construction can securely emulate an ideal primitive, such as a random oracle or a permutation. The authors also showed through their composition theorem that any construction indifferentiable from an ideal primitive also shares it's security properties when instantiated appropriately. This framework significantly advanced the understanding of secure composition in cryptographic systems, and is widely regarded as a watershed moment in modern symmetric key cryptography. A few notable applications of the indifferentiability framework were demonstrated by Coron, Patarin and Seurin (2008b) (equivalence of the Random Oracle model and the Ideal Cipher model), Holenstein, Künzler and Tessaro (2011b) (indifferentiability of the Feistel Network from an ideal cipher), and Coron et al. (2010) (Domain extension of ideal ciphers). Subsequently, a comparatively weaker notion of security called 'Sequential Indifferentiability' was introduced by Mandal, Patarin and Seurin (2012a), where the distinguisher either makes all their primitive queries first, and then their construction queries (making them primitive-construction sequential), or vice-versa.

The Feistel Network. Building on the foundations of indifferentiability, Coron, Patarin and Seurin (2008b) demonstrated how an ideal cipher could be implemented using a Feistel network composed of random functions by proposing a proof for the 6-round construction's indifferentiability from an ideal cipher. However, this result was later shown to contain flaws by Holenstein, Künzler and Tessaro (2010), who also provided a security proof for a much larger round count (18). This correction reinvigorated research into finding minimal secure round constructions.

Mandal, Patarin and Seurin (2012a) showed that a 6-round Feistel network is secure under their relaxed model of sequential indifferentiability. Later, Coron et al. (2016) achieved a full indifferentiability proof for 14 rounds, followed by Dachman-Soled, Katz and Thiruvengadam (2016) for 10 rounds, and Dai and Steinberger (2016b) for 8 rounds. Notably, however, no formal proof existed for the indifferentiability of a 7-round network until the present thesis.

Confusion-Diffusion Networks. Research on Confusion-Diffusion Networks (CDNs), generalising Substitution-Permutation Networks, was pioneered by Dodis et al. (2016a). They proved that CDNs with 2 rounds fail to meet classical indifferentiability standards while constructions with five or more rounds can achieve indifferentiability security with non-linear diffusion layers. Their work left open the question of CDNs with 3 and 4 rounds. Later, Da, Xu and Guo (2021a) investigated CDNs with Non-Linear Diffusion layers (NLCDNs) under sequential indifferentiability, showing that 3 rounds can be secure under this weaker notion while 2 rounds fail. Complementing this, Nandi, Paul and Saha (2023) corrected earlier attacks on 2-round NLCDNs and investigated cascaded block cipher based structures, leading naturally into the present thesis. Finally, building on these results, the current work proves that 3-round CDNs with linear diffusion layers satisfy full indifferentiability, conclusively resolving the previously open case.

Cascade Ciphers. Conceptually introduced by Shannon (1949a), Cascade ciphers were formalized by Even and Goldreich (1985) who also analyzed their structural limitations. Lampe and Seurin (2013) then brought the indifferentiability lens to cascades of ideal ciphers, showing that 2-rounds of cascades are insufficient for cryptographic security. Guo, Lin and Liu (2016) extended this work, proving that 4-round cascades with alternating keys are indifferentiably secure and that the 2 and 3-round constructions are not. This thesis further generalises these findings by presenting an attack on 3-round cascades with arbitrary $2n$-bit to $3n$-bit offline key schedules, reinforcing that at least four rounds are required for classical indifferentiability.

Viewed together, these strands of research delineate the secure design landscape for symmetric key primitives. Feistel constructions now have tight bounds showing the sufficiency of 8 rounds and novel results for the security of 7 rounds. CDNs are fully understood, with 2 rounds proven insecure, and 3 rounds secure under classical indifferentiability. The literature on Cascade ciphers is similarly refined to show that 4 rounds are indeed necessary for security, while 3-round constructions fail. The present thesis thus synthesizes and sharpens these theoretical developments, filling foundational gaps and setting precise thresholds for secure symmetric

cipher design.

Chapter 3

Preliminaries

3.1 General Notations.

We set up the notational structure that will be used in the following sections of the thesis. For convenience, notations common across the thesis are unified in this section, whereas defining notations specific to some particular work(s) is deferred to the respective chapter dedicated to the discussion.

$[k]$	The set of integers $\{1, 2, \ldots, k\}$, $k \in \mathbb{N}$
$[a, b]$	The set of integers $\{a, a+1, \ldots, b\}$, $a \leq b \in \mathbb{N}$
$\{\cdot\}$	A collection of distinct objects (a.k.a Sets)
$(\cdot)$	An ordered collection of objects (a.k.a Tuples)
$\lvert K \rvert$	Size of the set/tuple K
n	Cryptographic security parameter
N	2^n
B	$\{0,1\}^n$
$f,\ g,\ \ldots$	Functions/ Partial Functions
$\mathrm{Dom}(f)$/ $\mathrm{Ran}(f)$	Domain/Range of the function f
$\mathcal{G}(V, E)$	Unlabelled, undirected graph $\mathcal{G}$ with vertex set V and edge set E
$\mathcal{G}(V, E, L)$	Labelled, undirected graph $\mathcal{G}$ with vertex set V, edge set E, and edge label set L
$x[i]$	The i^{th} n-bit block of the bit string x

3.2 Random Functions.

A (probabilistic) function $f : \mathcal{X} \to \mathcal{Y}$ is said to be a random function if for each $x \in \mathcal{X}$ the value of $f(x)$ is chosen uniformly at random from $\mathcal{Y}$. More precisely,

$$\Pr[f(x) = y \mid f(x_1) = y_1, f(x_2) = y_2, \ldots, f(x_q) = y_q] = \frac{1}{|\mathcal{Y}|}$$

where $|\mathcal{Y}|$ is finite and $x \notin \{x_1, \ldots, x_q\}$ and $y, y_1, \ldots, y_q \in \mathcal{Y}$. $\pi : \mathcal{X} \to \mathcal{X}$ is said to be a random permutation if for each $x \in \mathcal{X}$ we have,

$$\Pr[\pi(x) = y \mid \pi(x_1) = y_1, \pi(x_2) = y_2, \ldots, \pi(x_q) = y_q] = \frac{1}{|\mathcal{X}| - q}$$

where $|\mathcal{X}|$ is finite and $x \notin \{x_1, \ldots, x_q\}$, $y_1, \ldots, y_q \in \mathcal{X}$ and $y \in \mathcal{X} \setminus \{y_1, \ldots, y_q\}$.

A random function $f : \mathcal{X} \to \mathcal{Y}$ can be viewed as a function sampled uniformly from the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. A random permutation is similar to a random oracle except that it is a permutation; one can thus view a random permutation $\pi : \mathcal{X} \to \mathcal{X}$ as a permutation chosen uniformly at random from the set of all permutation on $\mathcal{X}$.

3.3 Indifferentiability

3.3.1 Overview

At a high level, there are two common approaches to designing symmetric-key cryptographic systems. The constructive approach is to design cryptographic primitives (such as block ciphers or compression functions) from scratch. The designer creates a circuit or algorithm and claims that it possesses certain cryptographic properties, such as resistance to attacks. However, these claims are typically not mathematically provable; instead, they are supported by heuristic justifications and must withstand prolonged and rigorous scrutiny from competent cryptanalysts to gain acceptance in the cryptographic community. In contrast, in the reductive approach, one builds on previously-designed cryptographic primitives. These primitives, which have undergone extensive analysis and scrutiny, are treated as black-boxes. The designer constructs a new cryptographic system, known as a mode of operation, by utilizing the black-box primitives; such a design is as a norm accompanied by a mathematical reduction-based security proof based on some axiomatic idealized property of the underlying primitive.

The most common type of security proof used in the reductive approach is indistinguishability. Indistinguishability proofs demonstrate that, as long as certain black-box assumptions about

the underlying primitives hold, the constructed system is computationally indistinguishable from an ideal random system with the desired security properties. The concept of indifferentiability, introduced by Maurer, Renner and Holenstein (2004a), offers a framework for 'deeper' security proofs for such constructions.

We use the term oracle to denote an interface which provides an adversary a black-box access to a hidden function. An oracle F providing access to a hidden function ϕ will receive a query x and respond with $\phi(x)$; we simply denote this response as $F(x)$, and say that F provides oracle access to ϕ; wherever there is no scope for confusion, we shall refer to F and ϕ interchangeably. We will also use the standard notation F^G to denote that F has access to the oracle G. The same formalisation can be used to represent oracles that provide several different interfaces, each providing access to a different hidden function, by expanding the domain to include an additional input that indicates which interface to use. For instance, for a hidden permutation π, an oracle can provide access to both π and π^{-1} by accepting an additional input indicating the direction of query (forward or inverse). We provide the relevant formal definitions in the following subsection.

3.3.2 Games for Indifferentiability Security

DEFINITION: Distinguishing Advantage

For two oracles $\mathcal{O}_1$, $\mathcal{O}_0$ and a deterministic adversary $\mathcal{A}$, a distinguishing game is played as follows: the challenger picks a bit b at random and gives $\mathcal{A}$ access to $\mathcal{O}_b$; $\mathcal{A}$ makes a bounded number of adaptive queries to $\mathcal{O}_b$, and outputs a guess bit b'; $\mathcal{A}$ wins if $b' = b$. We define the advantage of $\mathcal{A}$ at distinguishing $\mathcal{O}_1$, $\mathcal{O}_0$ as

$$\mathbf{Adv}^{\mathcal{O}_1,\mathcal{O}_0}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathcal{O}_1} \text{ returns } 1] - \Pr[\mathcal{A}^{\mathcal{O}_0} \text{ returns } 1]\right|.$$

We consider a more general definition of distinguishing games, where the game G can specify additional constraints on the queries made by $\mathcal{A}$. Also, after all the queries have been made but before $\mathcal{A}$ outputs the guess bit, $\mathcal{O}_b$ can reveal some additional information about the hidden function(s) to $\mathcal{A}$. We use the notation $\mathbf{Adv}^{\mathcal{O}_1,\mathcal{O}_0}_{\mathsf{G}}(\mathcal{A})$ to denote the advantage of $\mathcal{A}$ in game G against oracles $\mathcal{O}_1$, $\mathcal{O}_0$.

Indifferentiability. Consider a construction $\mathscr{C}$ with oracle access to a public primitive f trying to emulate an ideal primitive $\mathcal{F}$. In the indifferentiability game (denoted indiff), the real oracle $\mathcal{O}_1$ provides interfaces to $\mathscr{C}$ and f (which in turn can encapsulate several interfaces

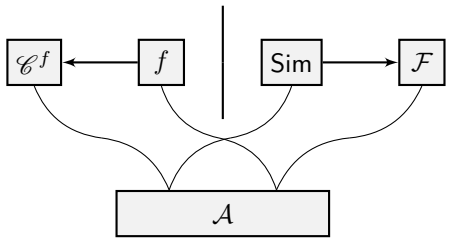

Figure 3.3.1: The distinguishing game of $\mathcal{A}$ in the indifferentiability security game.

as discussed above, like inverse calls), and the ideal oracle $\mathcal{O}_0$ provides interfaces to $\mathcal{F}$ and a simulator Sim, which has oracle access to $\mathcal{F}$, and tries to emulate f. Sim does not see the queries made directly to $\mathcal{F}$ by $\mathcal{A}$.

DEFINITION: (q_S, ϵ)-Indifferentiable

$\mathscr{C}^f$ is said to be (q_S, ϵ)-indifferentiable from $\mathcal{F}$ if there exists a polynomial-time simulator Sim with making at most q_S queries to $\mathcal{F}$, such that for any indiff adversary $\mathcal{A}$ making at most q oracle queries in all,

$$\mathbf{Adv}^{\mathcal{O}_1,\mathcal{O}_0}_{\mathsf{indiff}}(\mathcal{A}) < \epsilon,$$

where $\mathcal{O}_0$ includes access to the particular simulator Sim in question.

Here, q_S and ϵ are functions of q. We simply call $\mathscr{C}^f$ indifferentiable from $\mathcal{F}$ if q_S is bounded above by some polynomial in q and ϵ is a negligible function of the security parameter n for any q. The security parameter we use will be the number of bits needed to represent the output of $\mathcal{F}$, i.e., $\lceil \log_2 |\mathrm{Ran}(\mathcal{F})| \rceil$. In addition to the query complexity, we consider the time complexity of the distinguisher and the simulator. The composition theorem from Maurer, Renner and Holenstein (2004a) states that if $\mathscr{C}$ is indifferentiable from $\mathcal{F}$, then $\mathscr{C}^f$ can securely replace $\mathcal{F}$ in arbitrary (single-stage) contexts. Thus, proving that $\mathscr{C}$ is indifferentiable from $\mathcal{F}$ demonstrates that all the security properties implicit in $\mathcal{F}$ also hold for $\mathscr{C}^f$.

There are, broadly speaking, two main applications of indifferentiability. When f and $\mathcal{F}$ represent different ideal functionalities, indifferentiability can show that a model which assumes the existence of f implies a model which assumes the existence of $\mathcal{F}$. This was seen in action in proving the equivalence of the Random Oracle (RO) model (introduced by Bellare and Rogaway (1993a)) and the Ideal Cipher (IC) model (introduced by Shannon (1949b))—an XOR

of permutations using ICs was shown to be indifferentiable from an RO by Coron, Patarin and Seurin (2008b), and a Feistel network using ROs was shown to be indifferentiable from an IC by Holenstein, Künzler and Tessaro (2011b). When f and $\mathcal{F}$ represent the same ideal functionality but f has a smaller domain, indifferentiability implies that the construction $\mathscr{C}$ can be used for extending the domain of f; this was used by Coron et al. (2010) to demonstrate a domain extender for the IC.

3.3.3 Sequential Indifferentiability.

In the classical indifferentiability setting, the distinguisher doesn't need to make its queries in any particular order with respect to when the construction is queried and when the primitive(s) is queried. A stricter version of the classical indifferentiability is the notion of sequential indifferentiability, introduced by Mandal, Patarin and Seurin (2012a). A sequential distinguisher $\mathcal{D}$ is primitive-construction-sequential if it first makes all its primitive queries (without querying the construction), and then follows to make its construction queries (without querying the primitive), in that order. For a deterministic, sequential adversary $\mathcal{D}$ interacting with oracles $\mathcal{O}_1, \mathcal{O}_2$, its distinguising advantage is defined identically to the case of classical indifferentiability.

DEFINITION: (q_S, ϵ)-Sequentially Indifferentiable

$\mathscr{C}^f$ is said to be (q_S, ϵ)-sequentially indifferentiable from an ideal primitive $\mathcal{F}$ if there exists a polynomial-time simulator Sim with making at most q_S queries to $\mathcal{F}$, such that for any adversary $\mathcal{D}$ making at most q_1 primitive queries and q_2 construction queries (in that order), we have

$$\mathbf{Adv}^{\mathcal{O}_1,\mathcal{O}_0}_{\mathsf{seq\text{-}indiff}}(\mathcal{D}) < \epsilon$$

As in the case of classical indifferentiability, q_S is a function of $q = q_1 + q_2$, and q_S, ϵ are functions of q. Sequential indifferentiability is a weaker form of the classical indifferentiability notion of Maurer, Renner and Holenstein (2004b), in the sense that a sequential adversary acts in a restricted manner as compared to how an adversary might function in the classical indifferentiability setting. It is important to note that any sequential adversarial attack is also applicable in the classical paradigm. This directly implies that if a construction is not sequentially indifferentiable, then it is not differentiable in the classical sense, either.

3.3.4 H-Coefficient Technique

The transcript τ in a distinguishing game is the part of the computation visible to the adversary at the time of choosing its final response. This includes the queries and the responses, and may also include any additional information the oracle chooses to reveal to the adversary at the end of the query-response phase of the game.

We say an oracle $\mathcal{O}$ yields τ to denote the event that $\mathcal{A}$ interacts with $\mathcal{O}$ (the interaction denoted by G) and obtains τ as the transcript. Now, for a transcript to be realised, two things need to happen:

- The adversary needs to make the queries listed in the transcript;
- The game needs to make the corresponding responses.

Of these, the former is deterministic; the latter, probabilistic. For instance, consider a transcript for two queries x_1 and x_2, with outputs y_1 and y_2 respectively. This transcript will be realised only when the following four events occur:

- $\mathcal{A}$ begins by querying x_1 (deterministic, depends only on $\mathcal{A}$);
- $\mathcal{O}$ responds to x_1 with y_1 (probabilistic, depends only on G's randomness after conditioning on first event);
- $\mathcal{A}$, on examining the output y_1, next queries x_2 (deterministic, depends only on $\mathcal{A}$);
- $\mathcal{O}$ responds to x_2 with y_2 (probabilistic, depends only on G's randomness after conditioning on all earlier events).

Thus when we talk of the probability of $\mathcal{O}$ yielding a transcript, we are only concerned with the responses of $\mathcal{O}$, with the assumption that the adversary's queries are consistent with the transcript. For any other adversary, this probability is trivially 0. Thus $\Pr[\mathcal{O} \text{ yields } \tau]$ depends only on τ and the random coin of $\mathcal{O}$, and not on the adversary.

Suppose an adversary $\mathcal{A}$ is trying to distinguish between a real oracle $\mathcal{O}_1$ and an ideal oracle $\mathcal{O}_0$ in a game G. We state below a theorem, due to Patarin Patarin (2009), that we'll later use in our proofs. The name comes from the original paper, where the (scaled) probabilities of a game yielding a transcript were called H-coefficients.

Theorem 1 (H-Coefficient Technique) *Suppose we can define an event* bad *in a game against* $\mathcal{O}_0$*, and we call* τ *good if it can be obtained from* $\mathcal{O}_0$ *without encountering* bad*. Suppose the following hold:*

- $\Pr[\mathsf{bad}] \leq \epsilon_1$.
- *For any good* τ*,*

$$\Pr[\mathcal{O}_1 \text{ yields } \tau] \geq (1 - \epsilon_2) \cdot \Pr[\mathcal{O}_0 \text{ yields } \tau].$$

Then for any adversary $\mathcal{A}$ *we have*

$$\boldsymbol{Adv}_{\mathsf{G}}^{\mathcal{O}_1,\mathcal{O}_0}(\mathcal{A}) \leq \epsilon_1 + \epsilon_2.$$

This technique will be used on multiple occasions in this thesis when proving the indifferentiability security of different constructions.

Chapter 4

The Cascade Block Cipher and the $\mathsf{CDN}_{2,2}$ Construction

4.1 Introduction

Confusion-Diffusion Networks and Existing Results. Substitution Permutation Networks (SPNs) are a class of block ciphers that yield a wn-bit block cipher by iterating three steps, namely, *key addition* (XORing a round key with the wn-bit state), *substitution* (breaking the wn-bit state into w n-bit states and applying an n-bit permutation on each of them), and *permutation* (passing the entire wn-bit state through a key-less permutation). SPNs can be viewed as confusion-diffusion networks, where the "substitution" step is akin to "confusion", and the "permutation" step is viewed as "diffusion". Dodis et al. (2016a) initiated the indifferentiability analysis of SPNs and introduced the notion of *Confusion-Diffusion Networks* (CDNs). CDNs may be viewed as SPNs without the key addition step. Their CDN models are built upon public random (n-bit) primitives, typically called S-boxes, and non-cryptographic permutation layers, typically known as D-boxes. Other than showing an attack on the 2-round CDN construction with a non-idealized key schedule, they proved that, when the D-boxes are non-linear (and hence achieve more diffusion), five rounds are sufficient for indifferentiability, and that the security bounds improve with an increase in the number of rounds. They further showed that when the D-boxes are linear, nine rounds are sufficient for indifferentiability. These results go a long way in establishing that using non-linear diffusion layers leads to better security. Da, Xu and Guo (2021a) continued the CDN exploration by analyzing their security in the sequential indifferentiability paradigm Maurer, Renner and Holenstein (2004b) - a weaker version of

Maurer's indifferentiability setting. They proved the sequential indifferentiability of the 3-round CDN construction (under some moderate conditions on the D-boxes), and exhibit the tightness of their positive result by showing a primitive-construction-sequential attack on the 2-round construction.

Cascade Block Ciphers and Existing Results. An ideal (κ, n)-block cipher $\mathbf{IC}[\kappa, n]$ is a collection of 2^κ independent, random, and efficiently invertible n-bit permutations, indexed by a κ-bit key k. A *Cascade cipher* is a concatenation of block cipher systems. A cascade of l block ciphers is called an l-cascade, and is of the form

$$E_l(k_l, E_{l-1}(k_{l-1}, \ldots, E_2(k_2, E_1(k_1, m)) \ldots)$$

for an input message m. Shannon (1949a) showed that the cascade of l independent ideal (κ, n)-ciphers is a special case of product secrecy system, and is a set of $2^{l\kappa}$ n-bit permutations, but failed to provide additional insight on the nature and structure of the permutations. Even and Goldreich (1985) proved that the set of permutations achieved by l-cascade are not independent and that their behavior could be modeled by conducting only $l \cdot 2^\kappa$ exhaustive experiments. These experiments, however, did not allow the adversary to query the underlying ciphers. Lampe and Seurin (2013) observed that the cascade of two $\mathbf{IC}(\kappa, n)$ with two independent keys was not indifferentiable from $\mathbf{IC}(2\kappa, n)$.

Guo, Lin and Liu (2016) addressed the question of the required conditions and sufficient value of l in the Ideal Cipher Model, under which an l-cascade of $\mathbf{IC}(\kappa, n)$ will be indifferentiable from $\mathbf{IC}(\kappa', n)$, where $\kappa' > \kappa$. They showed that, for an alternating key schedule $KS(k_1, k_2) = (k_1, k_2, k_1, \ldots)$, the 4-cascade construction is indifferentiable from $\mathbf{IC}(2\kappa, n)$ with $n/6$-bit security, whereas the 2-cascade and 3-cascade constructions are not. The existence of the slide attack by Biryukov and Wagner (1999) rendered the authors incapable of considering using the same block cipher for an l-cascade under the alternating key schedule, irrespective of the value of l. The authors, on the request of the EUROCRYPT 2016 referees' panel, further explored the indifferentiability of 3-cascade with the key schedule $(k_1, k_2, k_1 \oplus k_2)$, which side-stepped the attack on 3-cascade with alternating key schedule given by Guo, Lin and Liu (2016). They provided a 10-query distinguisher that succeeds against any simulator. They generalised this attack to cover all key schedules $KS(K) = (k_1, k_2, k_3)$ where $\pi_1(K) = (k_1, k_2)$, $\pi_2(K) = (k_2, k_3)$ and $\pi_3(K) = (k_3, k_1)$ are efficiently computable 2κ-bit permutations, for a 2κ-bit master key K.

4.2 Definition of $\mathsf{CDN}_{w,r}$

NOTATION: General Confusion Diffusion Networks

We write any $x \in \{0,1\}^{wn}$, as $x =: (x[1], x[2], \cdots, x[w])$, with $x[i] \in \{0,1\}^n$ for $i \in [w]$, that is, by $x[i]$ we denote the i-th n-bit block of x.

Fix integers $w, r \in \mathbb{N}$. Consider a collection of rw permutations

$$\mathcal{P} = \{P_{ij} : \{0,1\}^n \to \{0,1\}^n \mid (i,j) \in [r] \times [w]\}$$

and a collection of $r-1$ permutations,

$$\Pi = \{\pi_i : \{0,1\}^{wn} \to \{0,1\}^{wn} \mid i \in [r-1]\}$$

Given $\mathcal{P}$ we define the permutations $P_i : \{0,1\}^{wn} \to \{0,1\}^{wn}$, for $i \in [r]$, as follows:

$$P_i(x)[j] = P_{ij}(x[j]), \quad \forall j \in [w]$$

The confusion diffusion network based on the collections $\mathcal{P}$ and Π, is defined as

$$\mathsf{CDN}_{w,r}^{\mathcal{P},\Pi} := P_1 \circ \pi_1 \circ P_2 \circ \pi_2 \circ \cdots \circ \pi_{r-1} \circ P_r$$

We often refer to this construction as $\mathsf{CDN}_{w,r}$, whenever $\mathcal{P}, \Pi$ is clear from the context.

For the purposes of this thesis, we will be simplifying the notations for $\mathsf{CDN}_{2,2}$ (likewise, for $\mathsf{CDN}_{3,2}$ in the following chapter) with ease of the readers in mind.

NOTATION: $\mathsf{CDN}_{2,2}$

Consider the collection of four permutations from $\{0,1\}^n$ to $\{0,1\}^n$,

$$\mathcal{P} = \{\mathcal{P}_1 := P_{1,1}, \mathcal{P}_2 := P_{1,2}, \mathcal{P}_3 := P_{2,1}, \mathcal{P}_4 := P_{2,2}\}$$

and a permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$, so that

$$\Pi = \{\pi\}$$

Input/output pairs for $\mathcal{P}_i, i \in [4]$ will be represented as $(u_i, v_i), (u'_i, v'_i), (u_i^j, v_i^j)$ etc., and those for the construction C will be denoted as the pair of tuples $((u_1, u_2), (v_3, v_4))$. etc., where $\mathsf{C}(u_1, u_2) = (v_3, v_4)$.

Note that for any particular construction input/output pair $((u_1, u_2), (v_3, v_4))$, we have $\pi(v_1, v_2) = (u_3, u_4)$. These collections of permutations determine the construction $\mathsf{CDN}_{2,2}$, shown in the Fig. 4.2.1.

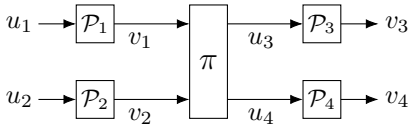

Figure 4.2.1: The $\mathsf{CDN}_{2,2}$ construction.

4.2.1 Attack against $\mathsf{CDN}_{2,2}$

The attack against $\mathsf{CDN}_{2,2}$ proposed in Dodis et al. (2016a) proves that this construction is not indifferentiable in the classical sense. Guo et al. proposed another attack in Da, Xu and Guo (2021a) that they claim proves that this construction is not even sequentially indifferentiable. The attack is as follows:

1. Find v_1 and $v_2 \neq v'_2$ such that $\pi(v_1, v_2)[1] = \pi(v_1, v'_2)[1]$, where $\pi(x)[1]$ denotes the first n-bits of $\pi(x)$
2. Query the right oracles to get $\mathcal{P}_1^{-1}(v_1) \to u_1$, $\mathcal{P}_2^{-1}(v_2) \to u_2$, and $\mathcal{P}_2^{-1}(v'_2) \to u'_2$
3. Query the left construction oracle to get $\mathsf{C}(u_1, u_2) \to (v_3, v_4)$ and $\mathsf{C}(u_1, u'_2) \to (v'_3, v'_4)$
4. Output 1 if and only if $v_3 = v'_3$.

Error of the Attack. Consider a function $\pi(u, v) = (u \oplus v, v)$. Clearly, $\pi(u, v) = \pi(u', v') \implies u = u', v = v'$. Hence, π is a permutation. It is pertinent to note that, for such a permutation π, there does not exist any u and $v \neq v'$ such that $\pi(u, v)[1] = \pi(u, v')[1]$.

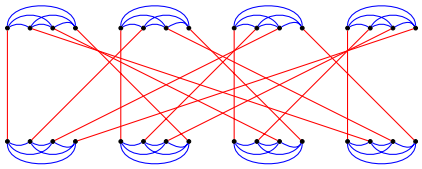

Figure 4.2.2: The top and bottom vertices denote a pair of copies of $\{0,1\}^{2n} = A_1 \sqcup \cdots \sqcup A_{2^n}$, the blue edges are between 2n-bit numbers whose first n-bits match, the red edges denote input-output pairs under π restricted to the first n bits.

In fact, there is an entire class of $2n$-bit permutations for which the above assumption does not hold. Consider a partition of $\{0,1\}^{2n} = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_{2^n}$ where the subset $A_i = \{v \in \{0,1\}^{2n} \mid v[1] = \langle i-1 \rangle_n\}$, where $\langle m \rangle_n$ is the n-bit representation of the integer m. Thus for any i, all elements of A_i have the same first n bits. Now suppose π sends distinct elements of A_i to elements of distinct subsets, i.e. for $v \neq v' \in A_i$, if $\pi(v) \in A_j$ and $\pi(v') \in A_k$, then $j \neq k$. There are exactly $[(2^n)!]^{2^n}$ such permutations π satisfying the above property. For all these permutations, there does not exist $u, v \neq v' \in \{0,1\}^n$, such that $\pi(u,v)[1] = \pi(u,v')[1]$, and hence for which the above attack does not work.

4.2.2 Proposed Attack

For an easy understanding of our attack, we introduce the notion of a functional graph. Consider a bijection $f : \{0,1\}^{2n} \to \{0,1\}^{2n}$. We define the functional graph $\mathcal{G}^f$ for the function f.

DEFINITION: Functional Graph.

Let X, Y be two disjoint copies of $\{0,1\}^{2n}$. The vertex set of $\mathcal{G}^f$ is defined as $X \sqcup Y$, and the edge set of $\mathcal{G}^f$ is defined as follows:

- **Internal Edges:** For $x, x' \in X$, if $x[1] = x'[1]$, then the edge $\{x, x'\} \in \mathcal{G}^f$. For $y, y' \in Y$, if $y[1] = y'[1]$, then the edge $\{y, y'\} \in \mathcal{G}^f$
- **External Edges:** For $x \in X$ and $y \in Y$, if $f(x) = y$, then the edge $\{x, y\} \in \mathcal{G}^f$.

DEFINITION: Alternating Cycle

Let f, $\mathcal{G}^f$ be defined as above. A cycle $C = (a_1, a_2, \ldots, a_k, a_1)$ in $\mathcal{G}^f$ is called an *Alternating Cycle* if, for every internal edge (a_i, a_{i+1}), the edges (a_{i-1}, a_i) and (a_{i+1}, a_{i+2}) are external.

Basic Idea of Our Attack Let $\mathcal{G}^{\text{cons}}$ be the functional graph of the construction permutation. Any alternating cycle in $\mathcal{G}^{\text{cons}}$ represents a constraint, that is a low probability event in the ideal world. Any alternating cycle in the functional graph, $\mathcal{G}^{\pi}$, of π, translates to an equivalent alternating cycle in $\mathcal{G}^{\text{cons}}$, with probability one in the real world. Hence finding an alternating cycle in $\mathcal{G}^{\pi}$ constitutes a valid attack on this construction.

The Distinguisher $\mathcal{D}_{\mathsf{cdn}}$ Since the permutation π is non-idealized, it is available to $\mathcal{D}_{\mathsf{cdn}}$ in totality, and the computation required for finding an alternating cycle in $\mathcal{G}^{\pi}$ can be done efficiently.

Search for either a 4 alternating cycle or 8 alternating cycle in $\mathcal{G}^{\pi}$.
if the 4 alternating cycle (x_0, y_0, y_1, x_1) is found **then**
    Primitive queries to $\mathcal{P}_1^{-1}$: $x_0[1]$.
        Let u_1 be the response received.
    Primitive queries to $\mathcal{P}_2^{-1}$: $x_0[2]$ and $x_1[2]$.
        Let u_2 and u_2' be the respective responses received.
    Construction queries: (u_1, u_2) and (u_1, u_2').
        Let (v_3, v_4) and (v_3', v_4') be the respective responses received.
    if $v_3 = v_3'$ **then**
        return 0.
    else
        return 1.
else if the 8 alternating cycle $(x_0, y_0, y_1, x_1, x_2, y_2, y_3, x_3)$ is found **then**
    Primitive queries to $\mathcal{P}_1^{-1}$: $x_0[1]$ and $x_1[1]$.
        Let u_1^1 and u_1^2 be the respective responses received.
    Primitive queries to $\mathcal{P}_2^{-1}$: $x_0[2], x_1[2], x_2[2]$ and $x_3[2]$.
        Let u_2^1, u_2^2, u_2^3, and u_2^4 be the respective responses received.
    Construction queries: $(u_1^1, u_2^1), (u_1^2, u_2^2), (u_1^2, u_2^3)$ and (u_1^1, u_2^4).
        Let $(v_3^1, v_4^1), (v_3^2, v_4^2), (v_3^3, v_4^3)$ and (v_3^4, v_4^4) be the respective responses received.
    if $v_3^1 = v_3^2$ and $v_3^3 = v_3^4$ **then**
        return 0.
    else
        return 1.

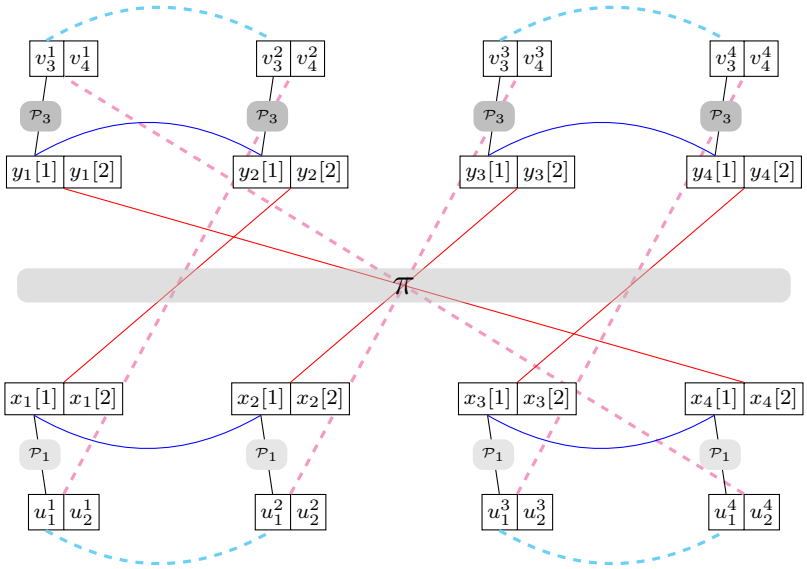

Figure 4.2.3: Attack with an 8 alternating cycle in $\mathcal{G}^{\pi}$

In the real world, given that $\mathcal{D}_{\mathsf{cdn}}$ finds an alternating cycle, it returns 0 with probability 1. In the ideal world, even though the inputs of all construction queries are revealed to the simulator before $\mathcal{D}_{\mathsf{cdn}}$ queries the construction, finding a cycle in $\mathcal{G}^{\text{cons}}$ is still necessary for the simulator to respond to the primitive queries appropriately in order to succeed against this attack. Any simulator making q queries finds a cycle in $\mathcal{G}^{\text{cons}}$ with probability $\mathcal{O}(\frac{q^2}{2^{2n}})$. Thus, for any simulator making $<< 2^n$ queries, the distinguisher $\mathcal{D}_{\mathsf{cdn}}$ can successfully differentiate between the two worlds, with probability close to 1, by just checking whether the alternating cycles of $\mathcal{G}^{\pi}$ has a corresponding isomorphic copy in $\mathcal{G}^{\text{cons}}$.

4.2.3 Analysis of the Attack

Lemma 1 *For any simulator $\mathcal{S}$ which can make q many ideal cipher C-queries, the polynomial-time distinguisher $\mathcal{D}_{\mathsf{cdn}}$ succeeds with advantage*

$$\mathbf{Adv}(\mathcal{D}_{\mathsf{cdn}}) \geq 1 - \frac{q^2}{2^{2n}}$$

Proof 1 *Let $p_1 := \Pr_{\mathsf{re}}\left[\mathcal{D}_{\mathsf{cdn}}^{\mathsf{CDN}_{2,2}^{\Pi,\mathcal{P}},\mathcal{P}} \to 0\right]$ denote the probability that $\mathcal{D}_{\mathsf{cdn}}$ returns 0 when interacting with the construction $\mathsf{CDN}_{2,2}^{\Pi,\mathcal{P}}$ and primitives $\mathcal{P}$, in the real world. Let $p_2 := \Pr_{\mathsf{id}}\left[\mathcal{D}_{\mathsf{cdn}}^{\mathsf{C},\mathcal{S}^{\mathsf{C}}} \to 0\right]$ denote the probability that $\mathcal{D}_{\mathsf{cdn}}$ returns 0 when interacting with ideal cipher C, and simulator*

$\mathcal{S}^{\mathsf{C}}$, *in the ideal world. Thus* $\mathbf{Adv}(\mathcal{D}_{\mathsf{cdn}}) = |p_1 - p_2|$. *In Lemma 2 we show that* $\mathcal{G}^{\pi}$ *will contain either an alternating cycle of length* 4 *or* 8 *and hence* $\mathcal{D}_{\mathsf{cdn}}$ *will find it. Thus* $p_1 = 1$. *In Lemma 3 we show that the probability of the simulator finding a cycle of any length in* $\mathcal{G}^{\mathsf{C}}$, *by making* q *ideal cipher queries is* $q^2/2^{2n}$. *If and only if,* $\mathcal{S}$ *can find an alternating cycle of length* 4 *or* 8 *in* $\mathcal{G}^{\text{cons}} = \mathcal{G}^{\mathsf{C}}$, *can it reply to the distinguisher's primitive queries such that it returns* 0. *Thus* $p_2 \leq \Pr(\mathcal{S}$ *finds a cycle in* $\mathsf{C})$.

$$\begin{aligned}\mathbf{Adv}(\mathcal{D}_{\mathsf{cdn}}) &= \big|p_1 - p_2\big| \\ &\geq 1 - \Pr(\mathcal{S} \textit{ finds a cycle in } \mathsf{C}) \\ &= 1 - \frac{q^2}{2^{2n}}\end{aligned}$$

□

Lemma 2 *For any bijection* $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$, *the functional graph* $\mathcal{G}^{\pi}$ *either contains an alternating cycle of length* 4, *or an alternating cycle of length* 8.

Proof 2 *Let* $\mathcal{G}^{\pi} := \mathcal{G}[X, Y]$. *For any* $x \in X$, *let* $N_X(x) := N(x) \cap X$, *that is the neighbourhood of* x *in* $\mathcal{G}|_X$. *Similarly for any* $y \in Y$, *we write* $N_Y(y) := N(y) \cap Y$. *We extend the definition of* N_X *for any subset* $X' \subseteq X$, *as* $N_X(X') := \bigcup_{x \in X'} N_X(x)$. *For any* $Y' \subseteq Y$, *we define* $N_Y(Y')$ *similarly. Note that for any* $x \in X$ *and* $y \in Y$, $|N_X(x)| = |N_Y(y)| = 2^n - 1$.

We take any $x_0 \in X$, *and let* $y_0 := \pi(x_0)$. *If* $N_X(x_0) \cap \pi^{-1}(N_Y(y_0)) \neq \emptyset$, *then let* $x_1 \in N_X(x_0)$ *and* $y_1 \in N_Y(y_0)$ *be such that* $\pi(x_1) = y_1$, *which implies that* (x_0, y_0, y_1, x_1) *is an alternating cycle in* $\mathcal{G}^{\pi}$, *and we are done. So we assume the contrary, that* $N_X(x_0)$ *and* $\pi^{-1}(N_Y(y_0))$ *are disjoint. Now if there is an internal edge between any two vertices* $x_1, x_2 \in \pi^{-1}(N_Y(y_0))$, *then there exists* $y_1, y_2 \in N_Y(y_0)$ *such that* $\pi(x_1) = y_1$ *and* $\pi(x_2) = y_2$, *which implies* (x_1, y_1, y_2, x_2) *is an alternating cycle in* $\mathcal{G}^{\pi}$. *Similarly, if there is an internal edge between any two points in* $\pi(N_x(x_0))$, *then also we end up with an alternating cycle of length* 4. *So we also assume that* $\pi(N_X(x_0))$ *and* $\pi^{-1}(N_Y(y_0))$ *are independent sets. In that case* $|N_Y(\pi(N_X(x_0)))| = |N_X(\pi^{-1}(N_Y(y_0)))| = |\pi(N_X(\pi^{-1}(N_Y(y_0))))| = (2^n - 1)^2$. *Since* $N_Y(\pi(N_X(x_0))), \pi(N_X(\pi^{-1}(N_Y(y_0)))) \subseteq Y$ *and* $|Y| = 2^{2n}$, *this leads us to the conclusion that* $N_Y(\pi(N_X(x_0))) \cap \pi(N_X(\pi^{-1}(N_Y(y_0)))) \neq \emptyset$. *Let* y_2 *belong to this intersection. Since* $y_2 \in \pi(N_X(\pi^{-1}(N_Y(y_0))))$ *there exists* $y_1 \in N_Y(y_0)$ *and* $x_2 \in N_X(x_1)$, *with* $x_1 := \pi^{-1}(y_1)$, *such that* $\pi(x_2) = y_2$. *Since* $y_2 \in N_Y(\pi(N_X(x_0)))$, *there exists* $x_3 \in N_X(x_0)$ *with* $y_3 = \pi(x_3)$, *such that* $y_2 \in N_Y(y_3)$. *In this case* $(x_0, y_0, y_1, x_1, x_2, y_2, y_3, x_3)$ *form an alternating cycle in* $\mathcal{G}^{\pi}$. □

Lemma 3 *Let* C *be an ideal* $2n$*-bit permutation and let* $\mathcal{S}$ *be any algorithm making at most* q *oracle queries to* C*. Then,*

$$\Pr(\mathcal{S} \textit{ finds an alternating cycle in } \mathcal{P}) \leq \frac{q^2}{2^{2n}}$$

Proof 3 *We denote the transcript of* $\mathcal{S}$ *after* i *sessions as* $\tau_i := \{(x, y) |\ \mathsf{C}(x) = y\}$*. Let* $\mathcal{G}_i$ *be the functional graph of the transcript* τ_i*, which is a subgraph of* $\mathcal{G}^{\mathsf{C}}$*. Assuming that* G_i *is acyclic,* $\Pr(\mathcal{S} \textit{ finds a cycle after the } (i+1)^{th} \textit{ session}) \leq \Pr(y_{i+1}[1] = y_j[1] |\ (\cdot, y_j) \in \tau_i) = \frac{i}{2^n}$*. Then,*

$$\begin{aligned}
\Pr(\mathcal{S} \textit{ finds a cycle in } \mathcal{P}) &= \sum_{i \in [q]} \Pr(\mathcal{S} \textit{ finds a cycle after the } i^{th} \textit{ session}) \\
&\leq q \times \Pr(\mathcal{S} \textit{ finds a cycle after the } q^{th} \textit{ session}) \\
&\leq \frac{q^2}{2^n}
\end{aligned}$$

The second line of the inequality is obtained by applying the Union Bound. □

4.3 Attack on Three Round Cascade Cipher

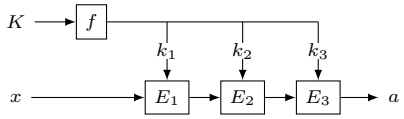

Figure 4.3.1: The 3-Cascade construction for a key schedule f (taking $2n$-bit input K and providing a $3n$-bit output (k_1, k_2, k_3)), and n-bit block ciphers $E_1,\ E_2,\ E_3$.

4.3.1 Preliminiaries

Cascade Block Cipher. For $\ell \geq 2$, let $E_1, \cdots, E_\ell$ be ℓ independent ideal block ciphers, $E_i : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, $i \in [\ell]$. Let $\mathcal{E}^\ell = (E_1, \cdots, E_\ell)$. Let $\mathsf{KS} : \{0,1\}^{k'} \to \{0,1\}^{k\ell}$ be a function, called key-scheduling function, which basically derives ℓ round keys from a master key $K \in \{0,1\}^{k'}$. Then the ℓ-round cascade cipher construction based on E^ℓ and key-scheduling function KS is a block cipher with key space $\{0,1\}^{k'}$ and message space $\{0,1\}^n$, defined as follows:

$$\mathsf{CC}^{\mathcal{E}^\ell}(K, x) := E_\ell(\kappa_\ell, \cdots E_2(\kappa_2, E_1(\kappa_1, x)) \cdots),$$

where $\mathsf{KS}(K) = \kappa^\ell = (\kappa_1, \kappa_2, \cdots, \kappa_\ell) \in (\{0,1\}^k)^\ell$ and $x \in \{0,1\}^n$.

Definition: Left/Right Colliding Keys

A pair of distinct keys $(K := (k_1, k_2, k_3), K' := (k'_1, k'_2, k'_3))$ is said to be *left-colliding* if $k_1 = k'_1$, and are called *right-colliding* if $k_3 = k'_3$. We define the sets of left-colliding and right-colliding key pairs as $\mathcal{C}_\mathsf{L}$ and $\mathcal{C}_\mathsf{R}$ respectively.

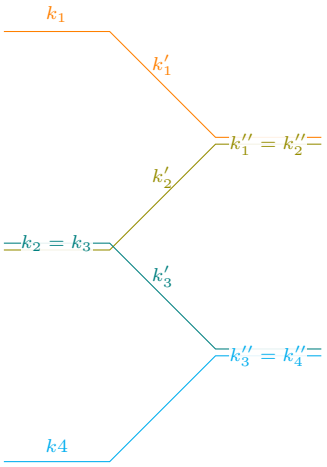

Figure 4.3.2: 4 Chain at left: (K_1, K_2, K_3, K_4), where $K_i = (k_i, k'_i, k''_i)$, satisfying the equalities in the figure.

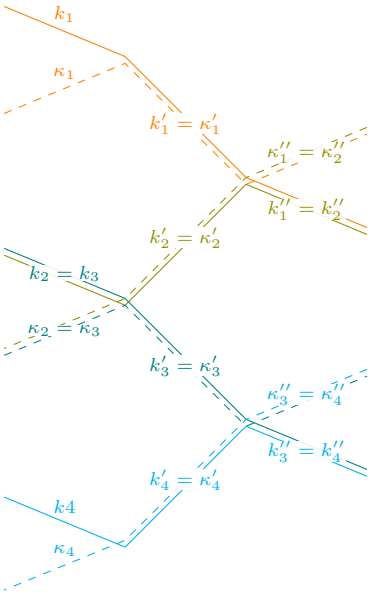

Figure 4.3.3: Double Intersecting 4 Chains: (K_1, K_2, K_3, K_4) and $(\varkappa_1, \varkappa_2, \varkappa_3, \varkappa_4)$, where $K_i = (k_i, k'_i, k''_i)$ and $\varkappa_i = (\kappa_i, \kappa'_i, \kappa''_i)$, satisfying the equalities in the figure.

Definition: 4-Chains (at left)

A 4-tuple (K_1, K_2, K_3, K_4) of keys is said to form *4-chain at left* if

- $(K_1, K_2),\ (K_3, K_4) \in \mathcal{C}_\mathsf{R}$ and
- $(K_2, K_3) \in \mathcal{C}_\mathsf{L}$

We can similarly define a 4-chain (K_1, K_2, K_3, K_4) at right in which (K_1, K_2), $(K_3, K_4) \in \mathcal{C}_\mathsf{L}$ and $(K_2, K_3) \in \mathcal{C}_\mathsf{R}$. See figure 4.3.2 for an illustration of 4-chains. We denote the set $\mathcal{C}_{4\mathsf{L}} := \{(K_1, K_2, K_3, K_4) |\ K_1, K_2, K_3, K_4$ form a 4-chain at left$\}$.

Definition: Double Intersecting Chains

A pair of distinct key tuples $(K_1, K_2, K_3, K_4), (K'_1, K'_2, K'_3, K'_4) \in C_{4\mathsf{L}}$ are said to be *double intersecting* (or simply, *intersecting*) if $K_{j,2} = K'_{j,2},\ \forall\ j \in [4]$, where $K_j =$

$(K_{j,1}, K_{j,2}, K_{j,3}) \in \{0,1\}^{3n}$.

For an intersecting pair of 4-chains (K_1, K_2, K_3, K_4) and $(\kappa_1, \kappa_2, \kappa_3, \kappa_4)$, we denote (k'_1, κ'_1) as the *starting pair* and call (k'_4, κ'_4) the *ending pair*, where $K_1 = (k'_1, \cdot, \cdot)$, $\kappa_1 = (\kappa'_1, \cdot, \cdot)$, $K_4 = (k'_4, \cdot, \cdot)$, and $\kappa_4 = (\kappa'_4, \cdot, \cdot)$ (See figure 4.3.3).

4.3.2 The Distinguisher $\mathcal{D}$

We assume that $\mathcal{D}$ efficiently finds a starting pair (k'_1, κ'_1) and an ending pair (k'_4, κ''_4) such that there are $\mathcal{O}(2^{2n})$ intersecting 4-chains connecting them, the existence of which will be shown in the analysis section. From the above-mentioned set of paths, $\mathcal{D}$ selects a path uniformly and at random. Let (K_1, K_2, K_3, K_4) and $(\kappa_1, \kappa_2, \kappa_3, \kappa_4)$ be the randomly chosen intersecting 4-chains with $F(K_i) = (k'_i, k''_i, k'''_i)$ and $F(\kappa_i) = (\kappa'_i, \kappa''_i, \kappa'''_i)$, $i \in [4]$. $\mathcal{D}$ samples $y_1, y_2 \xleftarrow{\$} \{0,1\}^n$ and executes the following query routine:

Primitive queries to E_1^{-1}: (k'_1, y_1) and (κ'_1, y_1).
    Let x_1, x_2 be the respective responses received.
Construction queries to E: (K_1, x_1) and (κ_1, x_2).
    Let a_1, a_2 be the respective responses received.
Construction queries to E^{-1}: (K_2, a_1) and (κ_2, a_2).
    Let x'_1, x'_2 be the respective responses received.
Construction queries to E: (K_3, x'_1) and (κ_3, x'_2).
    Let a'_1, a'_2 be the respective responses received.
Construction queries to E^{-1}: (K_4, a'_1) and (κ_4, a'_2).
    Let x''_1, x''_2 be the respective responses received.
Primitive query to E_1: (k'_4, x''_1).
    Let y''_1 be the response received.
$b \xleftarrow{\$} \{H, T\}$.
if $b = H$ **then**
    Primitive query to E_1: (κ'_4, x''_2).
else
    $x^* \xleftarrow{\$} \{0,1\}^n$ (If $x^* = x''_2$, sample again)
    Primitive query to E_1: (κ'_4, x^*).
        Let y''_1 be the response received in either case.
if $\left[b = H \text{ and } y''_1 = y''_2\right]$ Or $\left[b = T \text{ and } y''_1 \neq y''_2\right]$ **then**
    Return 0
else
    Return 1

In the Real World, if $b = H$, then $y_1'' = y_2''$ with probability 1, and if $b = T$, then $y_1'' \neq y_2''$ with overwhelming probability. Whereas in the Ideal World, these conditions are satisfied with negligible probability, which will be analyzed in detail in the following section.

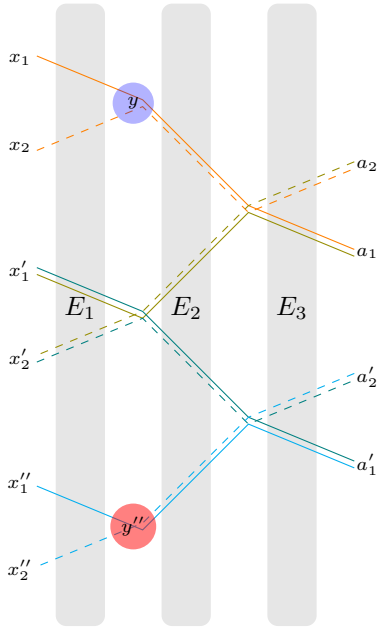

Figure 4.3.4: The Distinguisher samples a pair of Double Intersecting 4-Chains as depicted above, and forces a collision in E_1 at y. Making forward and backward construction queries from x_1 and x_2 with appropriate keys, it obtains x_1'' and x_2'', which, in the real world, must exhibit a collision in E_1 at y''. (The corresponding keys for the primitive and construction queries are implied from the sampled pair of Double Intersecting 4-Chains)

4.3.3 Analysis of Attack

Let $f : A \to B$. A collision pair of f is a pair (a, a') of distinct elements such that $f(a) = f(a')$. Let $\mathcal{C}_f$ denote the set of collision pairs for the function f. We also write $C_f = |\mathcal{C}_f|$.

Lemma 4 *Let $f : A \to B$ be any function where $|A| = k \cdot |B|$, $k \geq 2$. Then,*

$$C_f \geq k(k-1) \cdot |B|$$

Proof 4 *Fix a function f. There is no loss to assume that $B = \{1, 2, \ldots, b\}$ and $|A| = k \times b$. Let $(A_1^f, A_2^f, \ldots, A_b^f)$ be a partition of A such that $A_i^f = f^{-1}(i)$ (which may be the empty set).*

Let $|A_i^f| = a_i,\ i \in [b]$. *Clearly,* $\sum_{i\in[b]} a_i = k \times b$. *Thus,*

$$\begin{aligned}
C_f &= \sum_{i\in[b]} a_i(a_i - 1) \\
&= \left[\sum_{i\in[b]} (a_i)^2 - \sum_{i\in[b]} (a_i)\right] \\
&\geq \left[\frac{1}{b} \times \left(\sum_{i\in[b]} a_i\right)^2 - \sum_{i\in[b]} (a_i)\right] && \textit{[Cauchy-Schwartz Inequality]} \\
&= \left[\frac{(k\times b)^2}{b} - k \times b\right] = k(k-1) \times |B|.
\end{aligned}$$

Note that the equality achieves if and only if $a_i = k\ \forall\ i \in [b]$ (i.e. f is a regular function).

Key Schedule Analysis. Let $F := \{0,1\}^{2n} \to \{0,1\}^{3n}$ be a key scheduling function for the 3-round Cascade cipher

$$E((k,k'),x) = E_3(k_3, E_2(k_2, E_1(k_1, x))), \quad F(k,k') = (k_1,k_2,k_3),$$

E_1, E_2, E_3 are independent n-bit block ciphers and $x \in \{0,1\}^n$. In order to explain how the attack works, we first take a detailed look at the key schedule F. Throughout the section, *keys* will mean the $3n$-bit outputs of the function F. We also assume that F is injective and hence there are exactly 2^{2n} many keys. When F is not injective (suppose $F(K) = F(K')$) a simple attack can be employed by querying the construction with the same input with the keys K and K'. In the ideal world, they yield the same output with negligible probability, whereas this is true with probability 1 in the real world.

Let $F_3 := \{0,1\}^{2n} \to \{0,1\}^n$ be a function defined as $F_3(k,k') = k_3$, where $F(k,k') = (k_1,k_2,k_3)$. Applying lemma 4 to the function F_3, the number of right-colliding key pairs is at least $2^n(2^n-1) \times 2^n$. A similar argument applies to right-colliding key pairs. Hence,

$$|\mathcal{C}_\mathsf{L}| \geq 2^{2n}(2^n - 1), \qquad |\mathcal{C}_\mathsf{R}| \geq 2^{2n}(2^n - 1). \tag{1}$$

Lemma 5 *The number of 4-chains at left (and similarly at right) is at least* $2^{5n}/4$ *for all* $n \geq 2$.

Proof 5 *Let* $\pi_1 := \mathcal{C}_R \to \{0,1\}^n$ *be a function defined as* $\pi_1(K := (k_1,k_2,k_3), K' := (k_1',k_2',k_3)) = k_1$. *Now, consider any collision pair* $((K_2,K_1),(K_3,K_4))$ *of* π_1. *So, by definition of* π_1, (K_2,K_3)

is a left-colliding pair. Hence, (K_1, K_2, K_3, K_4) *is a 4-chain. Thus, the number of 4-chains (using lemma 4 once again, and the Eq.1) is at least* $2^n \times (2^{2n} - 2^n)(2^{2n} - 2^n - 1)$ *which is more than* 2^{5n-2} *as* $n \geq 2$.

So, the above result says that $|\mathcal{C}_{4\mathsf{L}}| \geq 2^{5n}/4$. Once again due to the lower bound of the collision pairs, the number of intersecting 4-chains is at least $2^{6n}/8$.

Lemma 6 *The number of intersecting 4-chains is at least* $(2^{6n} - 2^{5n+3})/2^7$.

Proof 6 *Let* $F_{\circ\circ\circ} := C_{LR} \rightarrow \{0,1\}^{4n}$ *be a function defined as*

$F_{\circ\circ\circ}(K_1, K_2, K_3, K_4) = (F_{|2}(K_1), F_{|2}(K_2), F_{|2}(K_3), F_{|2}(K_4))$, *where* $F_{|2}(K)$ *is defined as above. For sufficiently large* n, $|C_{LR}| \geq 2^{5n-3}$ *[from lemma 5]. Using lemma 4 and noting that* $|\mathrm{Dom}(F_{\circ\circ\circ})| \geq 2^{n-3} \times |\mathrm{Ran}(F_{\circ\circ\circ})|$, *we get that the number of intersecting 4-chains in* $F \geq \frac{2^{n-3}(2^{n-3}-1)}{2} \times |\mathrm{Ran}(F_{\circ\circ\circ})| = \frac{2^{6n}-2^{5n+3}}{2^7} = \mathcal{O}(2^{6n})$.

Note that there can be a maximum of 2^{4n} combinations of starting and ending pairs. By applying the Pigeon Hole Principle, we get that there exists *at least one starting pair* (k, k') *and one ending pair* (k'', k''') such that there are 2^{2n-3} distinct intersecting 4-chains connecting them. It is this information that the distinguisher eventually leverages when executing the attack.

4.3.4 Probability Analysis.

Let $q_{\mathcal{S}}$ be the number of queries allowed to the simulator $\mathcal{S}$. Recall that $\mathcal{D}$ randomly selects an intersecting pair of 4-chains from $\geq 2^{2n-7} - 2^{n-4}$ possible choices (from lemma 6). Let $\mathcal{I}_0$ be the event $[[b = H] \cap [y_1'' = y_2'']] \cup [[b = T] \cap [y_1'' \neq y_2'']]$. Also, let $\mathcal{IW}$ be the event that $\mathcal{D}$ interacts with the Ideal World and $\mathcal{RW}$ be the event that $\mathcal{D}$ interacts with the Real World. Then,

$$\begin{aligned}
\Pr(\mathcal{I}_0|\ \mathcal{RW}) &= \Pr([b = H] \cap [y_1'' = y_2'']|\ \mathcal{RW}) + \Pr([b = T] \cap [y_1'' \neq y_2'']|\ \mathcal{RW}) \\
&= \Pr(b = H).\Pr(E_1(k_4', x_1'') = E_1(\kappa_4', x_2'')|\ \mathcal{RW}) + \\
&\Pr(b = T).[1 - \Pr(E_1(k_4', x_1'') = E_1(\kappa_4', x^*)|\ \mathcal{RW})] \\
&= \frac{1}{2} + \frac{1}{2}[1 - 0] = 1
\end{aligned}$$

In the Ideal World, every intersecting 4-chains of keys connecting the starting pair (k_1', κ_1') and the ending pair (k_4', κ_4') is visible to $\mathcal{S}$. It is but a question of whether $\mathcal{S}$ is able to identify

the exact path chosen by $\mathcal{D}$. Let $\mathcal{P} := \mathcal{P}_{(k'_1,\kappa'_1)\to(k'_4,\kappa'_4)} := \{(\overline{K},\overline{K'})|\ \overline{K},\overline{K'} \in C_{\mathsf{LR}}\}$ be the set of all intersecting 4-chains with starting and ending pairs (k'_1,κ'_1) and (k'_4,κ'_4) respectively, and each $\overline{K}$ is a four-tuple of keys. For an element $(P,P') \in \mathcal{P}$, let $\mathsf{E}(P,x)$ denote the ideal cipher query sequence $E^{-1}(K_4, E(K_3, E^{-1}(K_2, E(K_1,x))))$, where $P = (K_1,K_2,K_3,K_4)$ and $P'[= (K'_1,K'_2,K'_3,K'_4)]$ are a pair of intersecting 4-chains. Let $(P_{\mathcal{D}},P'_{\mathcal{D}}) \in \mathcal{P}$ be the path randomly chosen by $\mathcal{D}$. For simplicity, when $\mathcal{S}$ makes any query of the form $\mathsf{E}(P,x)$ as described above, we will associate 1 query cost with it, instead of 4. Let $X^{\mathcal{S}}$ be a random variable that denotes the path $(P,P') \in \mathcal{P}$ chosen by $\mathcal{S}$. We will assume that if $\mathcal{S}$ observes, for some $(P,P') \in \mathcal{P}$, that $\mathsf{E}(P,x_1) = x''_1$ and $\mathsf{E}(P',x_2) = x''_2$ (or vice-versa), then $\mathcal{S}$ responds to the query $E_1(\kappa'_4,x''_2)$ with the value y''_1. Then,

$[\mathsf{E}(P'_\circ,x_2) = x'_2] \cap [(P_\circ,P'_\circ) \neq (P_{\mathcal{D}},P'_{\mathcal{D}})],\ (P_\circ,P'_\circ) \in \mathcal{P}.$

$$
\begin{aligned}
\Pr(\mathcal{I}_0|\mathcal{IW}) &= \Pr([b=H] \cap [y''_1 = y''_2]|\ \mathcal{IW}) + \Pr([b=T] \cap [y''_1 \neq y''_2]|\ \mathcal{IW}) \\
&\leq \Pr(b=H) \times \Bigg[\Pr\left[\mathsf{E}(P_{\mathcal{D}},x_1) = x''_1,\ \mathsf{E}(P'_{\mathcal{D}},x_2) = x''_2|\ X^{\mathcal{S}} = (P_{\mathcal{D}},P'_{\mathcal{D}})\right] \\
&\quad \times \Pr\left[X^{\mathcal{S}} = (P_{\mathcal{D}},P'_{\mathcal{D}})\right] \\
&\quad +\ \Pr\left[\mathsf{E}(P,x_1) = x''_1,\ \mathsf{E}(P',x_2) = x''_2|\ X^{\mathcal{S}} = (P,P')\right] \\
&\quad \times \Pr\left[X^{\mathcal{S}} = (P,P') \neq (P_{\mathcal{D}},P'_{\mathcal{D}})\right] \Bigg] \\
&\quad +\ \Pr(b=T) \\
&\leq \frac{1}{2}\left[1 \times \frac{q_{\mathcal{S}}}{2^{2n-7}-2^{n-4}}\ +\ q_{\mathcal{S}} \times \frac{1}{2^{2n}} \times \left[1 - \frac{1}{2^{2n-7}-2^{n-4}}\right]\right]\ +\ \frac{1}{2} \\
&= \frac{1}{2}\left[1 + \frac{q_{\mathcal{S}}}{2^{2n-7}-2^{n-4}}\left[1-\frac{1}{2^{2n}}\right]\ +\ \frac{q_{\mathcal{S}}}{2^{2n}}\right]
\end{aligned}
$$

Hence, the indifferentiability advantage of the distinguisher $\mathcal{D}$ against any simulator $\mathcal{S}$ with query bound $q_{\mathcal{S}}$ is

$$
\begin{aligned}
\mathbf{Adv}_{\mathsf{E},\mathcal{S}}(\mathcal{D}) &= |\Pr(\mathcal{I}_0|\mathcal{RW}) - \Pr(\mathcal{I}_0|\mathcal{IW})| \\
&\geq 1 - \frac{1}{2}\left[1 + \frac{q_{\mathcal{S}}}{2^{2n-7}-2^{n-4}}\left[1-\frac{1}{2^{2n}}\right]\ +\ \frac{q_{\mathcal{S}}}{2^{2n}}\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{2} - \left[\frac{q_{\mathcal{S}}}{2^{2n-8} - 2^{n-5}} \left[1 - \frac{1}{2^{2n}} \right] + \frac{q_{\mathcal{S}}}{2^{2n+1}} \right] \\
&\geq \frac{1}{2} - \left[\frac{2^9 \times q_{\mathcal{S}}}{2^{2n}} \left[1 - \frac{1}{2^{2n}} \right] + \frac{q_{\mathcal{S}}}{2^{2n+1}} \right] && [n \geq 5] \\
&= \frac{1}{2} - \left[\frac{2^9 \times q_{\mathcal{S}}}{2^{2n}} - \frac{2^9 \times q_{\mathcal{S}}}{2^{4n}} + \frac{1}{2} \times \frac{q_{\mathcal{S}}}{2^{2n}} \right] \\
&= \frac{1}{2} - \left[\left(2^9 + \frac{1}{2} \right) \frac{q_{\mathcal{S}}}{2^{2n}} - \frac{2^9 \times q_{\mathcal{S}}}{2^{4n}} \right]
\end{aligned}
$$

Theorem 2 *The indiffirentiability advantage of the distinguisher $\mathcal{D}$ against any simulator $\mathcal{S}$ with query bound $q_{\mathcal{S}}$ is given by*

$$\mathbf{Adv}_{\mathsf{E},\mathcal{S}}(\mathcal{D}) \geq \frac{1}{2} - \epsilon$$

Where $\epsilon = \left(2^9 + \frac{1}{2} \right) \frac{q_{\mathcal{S}}}{2^{2n}} - \frac{2^9 \times q_{\mathcal{S}}}{2^{4n}}$.

Illustration of the Attack for a Linear Key Schedule. For the 3-round Cascade Cipher, consider the key schedule $f(k_1, k_2) = (k_1, k_2, k_1 \oplus k_2)$, $k_1, k_2 \in \{0,1\}^n$. Choose $(k_1, k_2), (k_1', k_2') \in \{0,1\}^{2n}$ such that $k_1 \neq k_1'$. Define the 3n-bit keys $K_1, K_2, K_3, K_4, K_1', K_2', K_3', K_4'$ as follows:

- $K_1 = f(k_1, k_2) = (k_1, k_2, k_1 \oplus k_2)$
- $K_2 = f(k_1 \oplus k_2 \oplus k_2', k_2') = (k_1 \oplus k_2 \oplus k_2', k_2', k_1 \oplus k_2)$
- $K_3 = f(k_1 \oplus k_2 \oplus k_2', k_2) = (k_1 \oplus k_2 \oplus k_2', k_2, k_1 \oplus k_2')$
- $K_4 = f(k_1, k_2') = (k_1, k_2', k_1 \oplus k_2')$
- $K_1' = f(k_1', k_2) = (k_1', k_2, k_1' \oplus k_2)$
- $K_2' = f(k_1' \oplus k_2 \oplus k_2', k_2') = (k_1' \oplus k_2 \oplus k_2', k_2', k_1' \oplus k_2)$
- $K_3' = f(k_1' \oplus k_2 \oplus k_2', k_2) = (k_1' \oplus k_2 \oplus k_2', k_2, k_1' \oplus k_2')$
- $K_4' = f(k_1', k_2') = (k_1', k_2', k_1' \oplus k_2')$

We note that both the four-tuples (K_1, K_2, K_3, K_4) and (K_1', K_2', K_3', K_4') form 4-chains, on top of which, the two 4-chains are intersecting. This means, for a fixed starting pair = ending pair $= (k_1, k_1')$, we have 2^{2n} choices for the pair (k_2, k_2'), giving us 2^{2n} intersecting 4-chains with the given starting and ending pair (k_1, k_1'). Thus, for a given (k_1, k_1'), a distinguisher D can select

one of 2^{2n} choices of the pair (k_2, k_2'), define the keys $K_i, K_i',\ i \in [4]$, as described above and proceed in the same fashion as our attack approach for a generic 2n-bit to 3n-bit key schedule.

The 3-round Cascade construction with the linear key schedule given above has already been shown to not be indifferentiable with an Ideal Cipher by Guo et. al., in their 2011 paper 'Revisiting Cascade Ciphers in Indifferentiability Setting'. The two attacks for this key schedule are almost identical (barring the 'coin toss' modification), and the two distinguishers exploit the same limitation in the capabilities of a simulator.

4.4 Conclusion

Our attack on 2-CDN shows that the construction is not indifferentiable in the sequential setting, which is a weaker notion of security as compared to classical indifferentiability. This, combined with Guo et al.'s indifferentiability analysis on 3-CDN Da, Xu and Guo (2021a) solidifies the *exact* number of rounds required for the confusion-diffusion network to be sequentially indifferentiable as 3. This is better than the number of rounds required for full indifferentiability, as found by Dodis et al. Dodis et al. (2016a).

Lampe and Seurin's results on the indifferentiability of the 2-Cascade construction Lampe and Seurin (2013) indicate that the use of underlying sub-keys is pertinent with respect to the question of a cascade of block ciphers being indifferentiable from an ideal block cipher of larger key space. In that regard, our attack on the 3-Cascade construction with a generalised $2n$-bit to $3n$-bit non-idealized key schedule shows that key reuse by considering domain extending bijections *do not* lead to a secure construction in the indifferentiability setting.

Chapter 5

The $\mathsf{CDN}_{3,2}$ Construction

5.1 Introduction

To start the second chapter off, we take a look at Substitution-Permutation Networks, touch briefly on its history, and discuss an variant of SPN's called Confusion-Diffusion Networks.

5.1.1 Substitution-Permutation Networks.

Each round of a Substitution-Permutation network consists of a *substitution layer*, a *permutation layer*, and a round-key addition; the permutation layer is often skipped in the last round. A substitution layer consists of w non-linear invertible public functions applied in parallel; these functions are called *S-boxes*, denote $S_1, \ldots, S_w$. Many common S-boxes have the same width n for the input and the output, and we'll restrict our attention to such S-boxes. n is usually small enough to implement the S-box as a lookup table. The i-th permutation layer consists of a single invertible public function P_i with wn bits of input and output which has no 'cryptographic' strength; we'll call this a P-box. The archetypal P-box is a bit permutation (hence the name), but more generally they can be linear (in $\mathbb{GF}(2^n)$) or simple non-linear functions.

The plaintext is processed through r rounds. In round i, the wn-bit input x of the substitution layer is split into n-bit words as $x_1 \| \ldots \| x_w$, and the output is then computed as $S(x) := S_1(x_1) \| \ldots \| S_w(x_w)$. When $i < r$ this goes into the i-th permutation layer and comes out as $P_i(S(x))$. Finally, it gets added to the round key k_i, so that the final state at the end of the round is $P_i(S(x)) \oplus k_i$. While the permutation layers can be different for different rounds, we assume all the substitution layers to be identical, though we allow the S-boxes within each round to be different. Fig. A.2.1 in Appendix A.2 shows an example of an r-round SPN with

$n = 4$, $w = 8$, and bit-permutations as P-boxes.

While S-boxes and P-boxes are often used as design components in other design paradigms too (e.g., in designing the Feistel round function in DES), SPNs have been found to be extremely effective in achieving the properties of *confusion* and *diffusion* which Shannon Shannon (1949b) noted to be desirable in strong cryptographic systems. Informally, confusion refers to the cryptographically strong scrambling which breaks the correlation between input and output bits, and diffusion refers to the spreading of the effect of each input bit and key bit over a large number of output bits. The structural simplicity which SPNs retain while still introducing sufficient confusion and diffusion in the design has led to its wide popularity among block cipher designers.

5.1.2 Provable Security aspects.

While concrete block cipher designs rely on a large part on competent cryptographers to test their security by throwing at them everything in their arsenal, there is also a parallel stream of research dedicated to finding mathematical proofs that the high-level architecture is somehow inherently weakness-free. Such analysis is typically carried out by making some pseudorandomness assumption on some underlying components and showing by reduction proofs that this pseudorandomness carries over to the block cipher as a whole. Such proofs, when found, imply that the design is potentially that of a good block cipher, as long as the underlying function is strong enough.

Luby and Rackoff Luby and Rackoff (1988) famously showed that a Feistel network can yield a pseudorandom permutation in as few as three rounds when the round function is a pseudorandom function; and a similar analysis was carried out by Vaudenay Vaudenay (1999) for Lai-Massey networks. Later Hoang and Rogaway Hoang and Rogaway (2010) analysed the provable security of several kinds of unbalanced and generalised Feistel networks. What these networks have in common is a round function with large enough input and output so that pseudorandomness assumptions on it makes sense. For SPNs, it was unclear for a while whether such a straightforward reduction makes sense—assuming an entire round to have pseudorandomness makes the analysis trivial for the lack of any combining function as such, and an individual S-box is too small (typically a byte or less) for a straightforward reduction proof to make sense.

In spite of this, several approaches have been suggested for studying the security of SPNs—Naor and Reingold Naor and Reingold (1999) modified a Feistel network into an SPN round with a non-linear P-box and showed it security; Chakraborty and Sarkar Chakraborty and Sarkar (2006) looked at SPNs as a mode of operation; Baignères and Vaudenay Baignères and Vaudenay

(2006) analysed the security of the SPN structure used in AES; Iwata and Kurosawa Iwata and Kurosawa (2001) studied SPNs with the specific P-box used in Serpent. In all of these works, the S-box is assumed to be secret and random. Dodis et al. (2017) first studied the security of SPNs in the public permutation model by modeling the S-boxes as public permutations which the adversary has access to. Their work was extended to show beyond-birthday bound security of SPNs, first for non-linear P-boxes by Cogliati et al. (2018), and later for linear P-boxes by Gao et al. (2020).

Some works have also looked at the security of SPNs using approaches not typical in provable security research. Keliher, Meijer and Tavares (2000) analysed the resistance of SPNs to linear cryptanalysis using approximate linear hulls, Miles and Viola Miles and Viola (2015) adopted a complexity-theoretic approach, and Liu et al. (2023) did a statistical analysis of the t-wise independence of SPNs under various assumptions on S-boxes.

5.1.3 Confusion-Diffusion Networks.

If we view S-boxes as public permutation, a different angle of looking at SPNs is as domain extenders for public permutations. For this purpose, Dodis et al. (2016c) introduced the Confusion-Diffusion paradigm. It may be helpful to mention first that today, the terms confusion and diffusion have slightly different connotations than how Shannon used them—confusion layers now typically consist of strong cryptographic functions of small domain used for localised scrambling, while diffusion layers consist of linear or simple non-linear functions whose role is to spread out the randomness over the entire state. While confusion layers and diffusion layers have played key roles in all the design paradigms discussed earlier, and the term *Confusion-Diffusion Network* was used by Feistel (1970), it was only in 2015 that Dodis et al. (2016c) formalised the notion of a Confusion-Diffusion Network (CDN), which consists of alternating confusion layers and diffusion layers with idealised functionalities.

Dodis et al. saw the CDN as an idealised abstraction of unkeyed SPNs—in their paradigm, confusion layers are made up of parallel calls to public S-boxes, and diffusion layers implement a linear P-box represented by a dense matrix to achieve a high degree of mixing. They showed that 2-round CDNs with linear diffusion layers cannot be indifferentiable, by exhibiting an attack. They then went on to analyse the indifferentiability of several CDNs with five or more rounds, with a passing remark that the indifferentiability of 3-round CDNs and 4-round CDNs remain an open problem. Later, an indifferentiability attack was provided by Da, Xu and Guo (2021b) on 2-round CDNs with non-linear diffusion layers, and they further showed the sequential indifferentiability of a 3-round CDN with non-linear layers. Subsequently, Nandi,

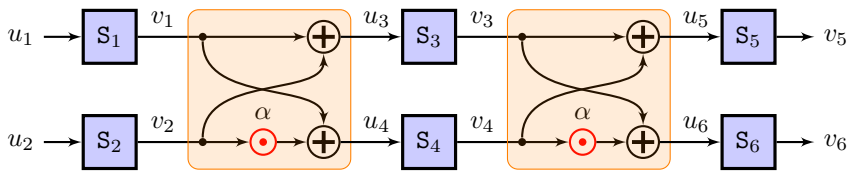

Figure 5.1.1: The $\mathsf{CDN}_{3,2}$ construction. $\odot$ represents multiplication in $\mathbb{GF}(2^n)$, S_1 , ..., S_6 represent six public permutations over $\mathbb{GF}(2^n)$ making up the three confusion layers, and the highlighted zones mark the two linear diffusion layers.

Paul and Saha (2023) pointed out a flaw in this attack, and repaired it to produce a sequential-indifferentiability attack on 2-round CDNs with non-linear layers, thus ending the hope that 2-round CDNs can achieve even a weaker notion of indifferentiability.

5.2 Preliminaries

DEFINITION: Partial Injective Functions

A partial m-bit injective function is a subset $\mathcal{P} \subseteq \{0,1\}^m \times \{0,1\}^m$ such that for any two distinct $(x, y), (x', y') \in \mathcal{P}$, we have $x \neq x'$ and $y \neq y'$.

For two partial m-bit injective functions $\mathcal{P}_1$ and $\mathcal{P}_2$, we say that these are disjointified if we disjointify the sets, i.e., we view $\mathcal{P}_i$ as a subset of $\{i\} \times \{0,1\}^m \times \{0,1\}^m$ for $i = 1, 2$.

The first number 1 or 2 identifies which partial injective function we are considering. Similarly, one can use more identifiers if there are multiple disjointified partial functions. For the sake of notational simplicity, we ignore the symbols which are used for disjointification. On the other hand, we use different reserved fonts to denote elements of disjointifed partial injective functions.

5.3 Main Result

Notation. We briefly recall from Sec. 4.2 the notation we use to describe the CDN we study. $\mathtt{S} = (\mathtt{S}_1, \ldots, \mathtt{S}_6)$ will denote the six public permutations, and $\mathsf{CDN}_{3,2}$ will denote the 3-round CDN built upon them, defined as

$$\mathsf{CDN}_{3,2}(x, y) := (\mathtt{S}_5(\mathtt{S}_3(\mathtt{S}_1(x) \oplus \mathtt{S}_2(y)) \oplus \mathtt{S}_4(\mathtt{S}_1(x) \oplus \alpha \cdot \mathtt{S}_2(y))),$$

$$\mathsf{S}_6(\mathsf{S}_3(\mathsf{S}_1(x) \oplus \mathsf{S}_2(y)) \oplus \alpha \cdot \mathsf{S}_4(\mathsf{S}_1(x) \oplus \alpha \cdot \mathsf{S}_2(y)))).$$

We'll commonly use (u_i, v_i) to denote an input-output pair of S_i for $i = 1, \ldots, 6$.

Main Result. Now we state the main result of this chapter.

Theorem 3 $\mathsf{CDN}_{3,2}$ *is* $(q_1, q_0, q_s, \varepsilon)$*-indifferentiable from a* $2n$*-bit random permutation* Π *with* $\varepsilon = O((q_1 + q_0)^{15}/2^n)$ *and* $q_s = O((q_1 + q_0)^4)$.

We prove Theorem 3 by defining a simulator Sim and showing that for any indifferentiability adversary $\mathcal{A}$ making up to q queries in all, Sim makes at most $O(q^4)$ queries to Π, and

$$\Delta_{\mathcal{A}}\left((\mathsf{S}, \mathsf{CDN}_{3,2}); (\mathsf{Sim}^{\Pi}, \Pi)\right) = O\left(\frac{q^{15}}{2^n}\right).$$

5.3.1 The Simulator

For each $i \in [6]$ we'll use a partial injective function $\mathcal{P}_i$ to keep track of the query-response pairs revealed for S_i. We'll consider $\mathcal{P}_1, \ldots, \mathcal{P}_6$ to be disjointified—an entry $(u_i, v_i) \in \mathsf{S}_i$ will implicitly be represented as (i, u_i, v_i); when there's no scope for confusion we'll not make this disjointification explicit. $\mathcal{P}$ will denote the (disjoint) union of $\mathcal{P}_1, \ldots, \mathcal{P}_6$. We will also consider six disjointified collections of pairs of n-bit variables, $\mathcal{V}_i, i \in [6]$. An entry $(u_i, v_i) \in \mathcal{V}_i$ represents a placeholder for a query-response pair for S_i that may or may not have been revealed yet. Obviously $\mathcal{P}_i \subset \mathcal{V}_i$ for $i \in [6]$, the pairs in $\mathcal{P}_i$ are those variables pairs in $\mathcal{V}_i$ that have been assigned n-bit values. In addition, we'll use a $2n$-bit-to-$2n$-bit partial injective function C to keep track of the query-response pairs revealed for Π.

DEFINITION: Chains

A 6-tuple $((u_1, v_1), \ldots, (u_6, v_6)) \in \mathcal{V}_1 \times \cdots \times \mathcal{V}_6$ is called a *chain* if at least one of the following three conditions is satisfied:

$$1.\ \begin{pmatrix} 1 & 1 \\ 1 & \alpha \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} u_3 \\ u_4 \end{pmatrix},\quad 2.\ \begin{pmatrix} 1 & 1 \\ 1 & \alpha \end{pmatrix} \begin{pmatrix} v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} u_5 \\ u_6 \end{pmatrix},$$

3. $((u_1, u_2), (v_5, v_6)) \in \mathsf{C}$.

A chain will be called a *primitive chain* if either condition 1 or 2 holds, but not 3. A chain is called a *construction chain* if condition 3 holds.

A chain will be called a $\mathcal{P}^3$-*chain* if for one of the following index sets of size 3,

$$I \in \{\{1,2,3\},\{1,2,4\},\{3,5,6\},\{4,5,6\},\{1,2,5\},\{1,2,6\},\{1,5,6\},\{2,5,6\}\}$$

we have the corresponding query-response pairs in the respective partial functions, i.e., $(u_i, v_i) \in \mathcal{P}_i$ for $i \in I$.

A chain will be called a $\mathcal{P}^4$-*chain* if for one of the following index sets of size 4,

$$I \in \{\{1,2,3,4\},\{1,2,5,6\},\{3,4,5,6\}\}$$

we have the corresponding query-response pairs in the respective partial functions, i.e., $(u_i, v_i) \in \mathcal{P}_i$ for $i \in I$.

A $\mathcal{P}^4$-chain that is also a construction chain is called an *almost-*$(\mathcal{P}, \mathsf{C})$*-complete chain.*

A chain will be called a $\mathcal{P}$*-internal chain* if $(u_3, v_3) \in \mathcal{P}_3, (u_4, v_4) \in \mathcal{P}_4$.

A chain is called a $\mathcal{P}$*-complete chain* if all the six query-response pairs are in the respective partial functions, i.e., $(u_i, v_i) \in \mathcal{P}_i$ for all $i \in [6]$.

Chain Completion. Note that every almost-$(\mathcal{P}, \mathsf{C})$-complete chain can be deterministically extended to a complete chain as follows: Suppose $(u_i, v_i) \in \mathcal{P}_i$ for $i \in \{1,2,3,4\}$ and $((u_1, u_2), (v_5, v_6)) \in \mathsf{C}$, then set $\begin{pmatrix} u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & \alpha \end{pmatrix} \begin{pmatrix} v_3 \\ v_4 \end{pmatrix}$. Similarly, if $(u_i, v_i) \in \mathcal{P}_i$ for $i \in \{1,2,5,6\}$, then set

$$\begin{pmatrix} u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & \alpha \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \text{ and } \begin{pmatrix} v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & \alpha \end{pmatrix}^{-1} \begin{pmatrix} u_5 \\ u_6 \end{pmatrix}.$$

Note that any $\mathcal{P}^4$-chain can be converted to an almost-$(\mathcal{P}, \mathsf{C})$-complete chain by just making the corresponding construction query. Also, every $\mathcal{P}^3$-chain can be extended to a complete chain by first converting it into a $\mathcal{P}^4$-chain by making a primitive query (in the ideal world the simulator achieves this by uniformly sampling a random number), converting this $\mathcal{P}^4$-chain into an almost complete chain and then extending it into a complete chain as described above.

Definition: C-consistency, $\mathcal{P}$-completeness

We'll call a $\mathcal{P}$-complete chain $\{(u_1, v_1), \ldots, (u_6, v_6)\}$ C-*consistent* if $((u_1, u_2), (v_5, v_6)) \in \mathsf{C}$.

Conversely, we'll call $((u, u'), (v, v')) \in \mathsf{C}$ $\mathcal{P}$-*complete* if there exists a $\mathcal{P}^4$-chain $((u_1, v_1), \ldots, (u_6, v_6))$ such that $u_1 = u, u_2 = u', v_5 = v, v_6 = v'$. We say C is $\mathcal{P}$-complete if every element of C is $\mathcal{P}$-complete.

We now define certain notions of *saturation* of a transcript. These would help us understand the structure of adversary and simulator views, as will be explained later.

Definition: Consistently Saturated Transcripts

The transcript $(\mathcal{P}, \mathsf{C})$ is called *consistently saturated* if the following conditions hold:

1. All $\mathcal{P}$-complete chains are C-consistent;
2. Every almost-$(\mathcal{P}, \mathsf{C})$-complete chain is a $\mathcal{P}$-complete chain.

If in addition, all $\mathcal{P}^4$-chains are $\mathcal{P}$-complete chains too, then we call $(\mathcal{P}, \mathsf{C})$ a *strong consistently saturated transcript.*

We next define an even stronger notion of saturation. Consider disjointified partial injective functions $\mathcal{P}'_1, \ldots, \mathcal{P}'_6$ and their union $\mathcal{P}'$ such that $\mathcal{P}' \subseteq \mathcal{P}$; the disjointification then implies that $\mathcal{P}'_i \subseteq \mathcal{P}_i$ for $i = 1, \ldots, 6$.

Definition: Preemptively Saturated Transcripts

Given a transcript $(\mathsf{C}, \mathcal{P})$ and $\mathcal{P}' \subseteq \mathcal{P}$, we say that $(\mathsf{C}, \mathcal{P})$ is *preemptively saturated with respect to* $\mathcal{P}'$ if the following conditions hold:

- $(\mathcal{P}', \mathsf{C})$ is consistently saturated;
- C is $\mathcal{P}$-complete;
- Every $(\mathcal{P}')^3$-chain and $\mathcal{P}'$-internal chain is $\mathcal{P}$-complete

DEFINITION: Surely Revealed Construction Queries

A construction query-response pair $((u, u'), (v, v')) \in \mathsf{C}$ is called *surely revealed with respect to* $\mathcal{P}$ if:

1. Either, three of the following four conditions hold: (i) $u \in \mathrm{Dom}(\mathcal{P}_1)$, (ii) $u' \in \mathrm{Dom}(\mathcal{P}_2)$, (iii) $v \in \mathrm{Ran}(\mathcal{P}_5)$, (iv) $v' \in \mathrm{Dom}(\mathcal{P}_6)$;
2. Or, there exists $((u_1, u_1'), (v_1, v_1')) \in \mathsf{C}$ such that one of the following two conditions holds: (i) $u, u_1 \in \mathrm{Dom}(\mathcal{P}_1), u', u_1' \in \mathrm{Dom}(\mathcal{P}_2)$ and either $v = v_1$ or $v' = v_1'$; (ii) $v, v_1 \in \mathrm{Ran}(\mathcal{P}_5), v', v_1' \in \mathrm{Dom}(\mathcal{P}_6)$ and either $u = u_1$ or $u' = u_1'$.

Adversary and Simulator view. Let us denote the transcript maintained by the adversary as $(\mathcal{P}^{\mathcal{A}}, \mathsf{C}^{\mathcal{A}})$. As we have seen above that the adversary can extend its almost-$(\mathcal{P}^{\mathcal{A}}, \mathsf{C}^{\mathcal{A}})$-complete chains into complete chains deterministically, we assume that the adversary transcript is consistently saturated.

The simulator, on the other hand, maintains a transcript $(\mathcal{P}, \mathsf{C}^{\mathcal{S}})$ along with a sub-list of primitive query-response pairs, $\mathcal{P}^{\mathcal{A}|\mathcal{S}} \subseteq \mathcal{P}$ such that the following conditions are satisfied:

- $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ is the smallest super-set of $\mathcal{P}^{\mathcal{A}}$ such that $(\mathcal{P}^{\mathcal{A}|\mathcal{S}}, \mathsf{C}^{\mathcal{S}})$ is strong consistently saturated.
- $(\mathcal{P}, \mathsf{C}^{\mathcal{S}})$ is preemptively saturated with respect to $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$.

We call $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ the *exposed primitive transcript*. $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ is obtained by the transcript recursively by first completing all $(\mathcal{P}^{\mathcal{A}})^4$-chains obtaining a primitive transcript, say $\mathcal{P}^{\mathcal{A}'}$, and then completing all $(\mathcal{P}^{\mathcal{A}'})^4$-chains, and so on. This recursive process is later denoted as $\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}(\mathsf{Q})$, where Q is the query made by the adversary.

DEFINITION: Well-Approximated

Let $(\mathcal{P}^{\mathcal{A}}, \mathsf{C}^{\mathcal{A}})$ be the adversary transcript and $(\mathcal{P}, \mathsf{C}^{\mathcal{S}}, \mathcal{P}^{\mathcal{A}|\mathcal{S}})$ be the simulator transcript, then we say that $\mathsf{C}^{\mathcal{A}}$ *is well-approximated by* $\mathsf{C}^{\mathcal{S}}$ if the following conditions hold:

1. For any $((u_1, u_2), (v_5, v_6)) \in \mathsf{C}^{\mathcal{A}} \setminus \mathsf{C}^{\mathcal{S}}$ both the following conditions hold:
   - $u_1 \notin \mathrm{Dom}(\mathcal{P}_1^{\mathcal{A}|\mathcal{S}})$ or $u_2 \notin \mathrm{Dom}(\mathcal{P}_2^{\mathcal{A}|\mathcal{S}})$;
   - $v_5 \notin \mathrm{Ran}(\mathcal{P}_5^{\mathcal{A}|\mathcal{S}})$ or $v_6 \notin \mathrm{Ran}(\mathcal{P}_6^{\mathcal{A}|\mathcal{S}})$.
2. Constructions chains corresponding to surely revealed queries in both $\mathsf{C}^{\mathcal{A}}$ and $\mathsf{C}^{\mathcal{S}}$ are respectively $\mathcal{P}^{\mathcal{A}}$-complete chains and $\mathcal{P}$-complete chains.

The graph $\mathcal{G}_\pi$. We consider a graph with vertices $M^{((u,u'),(v,v'))} = \begin{pmatrix} u & u' \\ v & v' \end{pmatrix}$ for all $((u,u'),(v,v')) \in \mathsf{C}$ and add an edge between $M^{((u,u'),(v,v'))}$ and $M^{((u_1,u_1'),(v_1,v_1'))}$ if either $u = u_1$, $u' = u_1'$, $v = v_1$, or $v' = v_1'$.

Simulator Description. Our simulator Sim, in effect, preemptively completes the adversary Π queries that have been surely revealed till that point in their interaction. In addition, Sim maintains complete chains in its transcript for every primitive pair in $\mathcal{P}_3^{\mathcal{A}|\mathcal{S}} \times \mathcal{P}_4^{\mathcal{A}|\mathcal{S}}$. Note that such chains are not revealed to the adversary upon completion. The reason for this is two-fold: Firstly, it simplifies the graphical structure that aids Sim in its actions (more precisely, it eliminates the need for maintaining graphs w.r.t primitive pairs in $\mathcal{P}_3^{\mathcal{A}|\mathcal{S}} \times \mathcal{P}_4^{\mathcal{A}|\mathcal{S}}$ as far as component completion is concerned). Secondly, it greatly reduces the algebraic complexities of the diffusion layers (on account of their non-cryptographic nature) that would otherwise make the security analysis very cumbersome.

The Sim module captures what the simulator does. In case of any primitive query, if the corresponding query-response pair is already set by the simulator then Sim just returns the already set response, and extends the exposed primitive transcript $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ by running the sub-module $\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}$. If however the response is not yet set, then in case of inward queries to $\mathtt{S}_1, \mathtt{S}_2, \mathtt{S}_5^{-1}, \mathtt{S}_6^{-1}$, Sim identifies the component of $\mathcal{G}_\pi$ that is surely revealed by the $\mathsf{Trace}\mathcal{G}_\pi$ procedure, and completes the corresponding construction chains by the $\mathsf{Fill}\mathcal{G}_\pi$ procedure. In case of outward queries to $\mathtt{S}_1^{-1}, \mathtt{S}_2^{-1}, \mathtt{S}_3, \mathtt{S}_3^{-1}, \mathtt{S}_4, \mathtt{S}_4^{-1}, \mathtt{S}_5, \mathtt{S}_6$, if the response is not yet set, then Sim randomly generates the response, say R, and if $\mathcal{P}' = \mathcal{P} \cup \{(\mathsf{Q}, \mathsf{R})\}$, then it identifies all $(\mathcal{P}')^3$-chains and completes them by the $\mathsf{FillPrimChn}$ procedure. See 5.3.1 for details.

$\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}(x)$. This module hardly requires any explanation, and we provide a gist of it here, in words. For an input $x \in \{u_1, \ldots, v_6\}$, this module finds all $\mathcal{P}^4$-chains in $\mathcal{P}$ containing x, completes them, and adds the completed chain to $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ in the entirety of its primitive contents. Each time there are new entries to $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$, the sub-module is run again, and recursively identifies all such $\mathcal{P}^4$ chains till a stagnant $\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ is reached.

Lemma 7 *For the simulator* Sim *described above and any* (q_1, q_0)*-query indifferentiability adversary* $\mathcal{A}$*,*

$$\Delta_{\mathcal{A}}\left((\mathcal{S}, \mathsf{CDN}_{3,2})\ ;\ (\mathsf{Sim}^{\Pi}, \Pi)\right) \leq \left(\frac{(2q_1 + 2q_0)^{15}}{2^n}\right).$$

Figure 5.3.1: The $\mathsf{CDN}_{3,2}$ simulator algorithm

$\mathsf{Sim}(\cdot)$
Input: $\tau_j := \left(\mathsf{C}^{\mathcal{S}}, \mathcal{P} := (\mathcal{P}_1, \ldots, \mathcal{P}_6), \mathcal{P}^{\mathcal{A}|\mathcal{S}}\right),\ x \in \{u_1, v_1, \ldots, u_6, v_6\}$,
Output: x'

if $x = u_1$
  if $u_1 \in \mathrm{Dom}(\mathcal{P}_1)$
    Run $\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}$
    Return v_1
  else
    Run $\mathsf{Trace}_{\mathcal{G}_\pi}(u_1)$
    $\mathsf{Fill}_{\mathcal{G}_\pi}(u_1)$
    Return v_1
  endif
endif

if $x = v_1$
  if $v_1 \in \mathrm{Ran}(\mathcal{P}_1)$
    Run $\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}$
    Return u_1
  else
    Run $\mathsf{FillPtimChn}(v_1)$
    Return u_1
  endif
endif

if $x = u_3$
  if $u_3 \in \mathrm{Dom}(\mathcal{P}_3)$
    Run $\mathsf{Extend}^{\mathcal{P}^{\mathcal{A}|\mathcal{S}}}$
    Return v_3
  else
    Run $\mathsf{FillPtimChn}(u_3)$
    Return v_3
  endif
endif

Theorem 3 follows immediately from Lemma 7.

5.4 Proof Sketch

In this section we provide a sketch of the proof of Lemma 7, using the "Coefficient H" Technique.

The graph $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$. Consider a primitive transcript $\mathcal{P}$. For $(u_3, v_3) \in \mathcal{P}_3$, $(u_4, v_4) \in \mathcal{P}_4$, we can define the 2×2 matrix with n-bit entries as

$$M^{(u_3,v_3),(u_4,v_4)} = \begin{pmatrix} \beta \cdot (\alpha \cdot u_3 \oplus u_4) & \beta \cdot (u_3 \oplus u_4) \\ v_3 \oplus v_4 & v_3 \oplus \alpha \cdot v_4 \end{pmatrix}.$$

Based on this we define

$$V(\mathcal{P}) = \{M^{(u_3,v_3),(u_4,v_4)} : (u_3, v_3) \in \mathcal{P}_3, (u_4, v_4) \in \mathcal{P}_4\}.$$

$\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ is a labeled graph with vertex set $V(\mathcal{P})$. The edge $\{M^{(u_3,v_3),(u_4,v_4)}, M^{(u'_3,v'_3),(u'_4,v'_4)}\}$ is included in $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ if they collide on one of the four entries, i.e., when one of the following holds:

- $\alpha \cdot u_3 \oplus u_4 = \alpha \cdot u'_3 \oplus u'_4$, in which case it has label 1;
- $u_3 \oplus u_4 = u'_3 \oplus u'_4$, in which case it has label 2;

FillPrimChn($\cdot$)
Input: $\tau := \big(\mathsf{C}, \mathcal{P} := (\mathcal{P}_1, \ldots, \mathcal{P}_6), \mathcal{P}^{\mathcal{A}|\mathcal{S}}\big),\ x \in \{v_1, v_2, u_3, v_3, u_4, v_4, u_5, u_6\}$,
$\beta = (1+\alpha)^{-1}$,
Output: $\tau' := \big(\mathsf{C}', \mathcal{P}' := (\mathcal{P}'_1, \ldots, \mathcal{P}'_6)\big)$

$\mathsf{C}' \leftarrow \mathsf{C},\ \mathcal{P}' \leftarrow \mathcal{P}$

if $x = u_3$
  $v_3 \leftarrow\!\!\$\ \{0,1\}^n,\ \mathcal{P}^{\mathcal{A}|\mathcal{S}} \leftarrow \mathcal{P}^{\mathcal{A}|\mathcal{S}} \cup (u_3, v_3)$
  foreach $(u_4, v_4) \in \mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_4$
    if $(u_1, v_1) \in \mathcal{P}_1$ s.t. $\beta \cdot (\alpha \cdot u_3 \oplus u_4) = v_1$
      if $(u_2, v_2) \notin \mathcal{P}_2$ s.t. $\beta \cdot (u_3 \oplus u_4) = v_2$
        $v_2 \leftarrow \beta \cdot (u_3 \oplus u_4)$
        $u_2 \leftarrow\!\!\$\ \{0,1\}^n$
      endif
    elseif $(u_2, v_2) \in \mathcal{P}_2$ s.t. $\beta \cdot (u_3 \oplus u_4) = v_2$
      $v_1 \leftarrow \beta \cdot (\alpha \cdot u_3 \oplus u_4)$
      $u_1 \leftarrow\!\!\$\ \{0,1\}^n$
    else
      $v_1 \leftarrow \beta \cdot (\alpha \cdot u_3 \oplus u_4)$
      $v_2 \leftarrow \beta \cdot (u_3 \oplus u_4)$
      $u_1, u_2 \leftarrow\!\!\$\ \{0,1\}^n$
    endif
    $((u_5, v_5), (u_6, v_6)) \leftarrow \mathsf{SetL3}((u_1, v_1), \ldots, (u_4, v_4))$
    $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
    foreach $i \in \{1, 2, 5, 6\}$
      $\mathcal{P}'_i \leftarrow \mathcal{P}'_i \cup (u_i, v_i)$
  endforeach
  foreach $(u_1, v_1) \in \mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_1, (u_2, v_2) \in \mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_2$
    if $v_1 \oplus v_2 = u_3$
      $u_4 \leftarrow v_1 \oplus \alpha \cdot v_2$
      $v_4 \leftarrow\!\!\$\ \{0,1\}^n$
    $((u_5, v_5), (u_6, v_6)) \leftarrow \mathsf{SetL3}((u_1, v_1), \ldots, (u_4, v_4))$
    $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
    foreach $i \in \{4, 5, 6\}$
      $\mathcal{P}'_i \leftarrow \mathcal{P}'_i \cup (u_i, v_i)$
    endforeach
  endforeach
endif
return $\tau' := (\mathsf{C}', \mathcal{P}')$

cont.

if $x = v_1$
  foreach $(u_2, v_2), (u_3, v_3) \in \mathcal{P}$ s.t. $v_1 \oplus v_2 = u_3$
    $u_4 \leftarrow v_1 \oplus \alpha \cdot v_2$
    $v_4 \leftarrow\!\!\$\ \{0,1\}^n$
    $((u_5, v_5), (u_6, v_6)) \leftarrow \mathsf{SetL3}((u_1, v_1), \ldots, (u_4, v_4))$
    $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(u_4, v_4), (u_5, v_5), (u_6, v_6)\}$
  endforeach
  foreach $(u_2, v_2), (u_4, v_4) \in \mathcal{P}$ s.t. $v_1 \oplus \alpha \cdot v_2 = u_4$
    $u_3 \leftarrow v_1 \oplus v_2$
    $v_3 \leftarrow\!\!\$\ \{0,1\}^n$
    $((u_5, v_5), (u_6, v_6)) \leftarrow \mathsf{SetL3}((u_1, v_1), \ldots, (u_4, v_4))$
    $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(u_3, v_3), (u_5, v_5), (u_6, v_6)\}$
  endforeach
  Return $\tau' := (\mathsf{C}', \mathcal{P}')$

SetL1($\cdot$)
Input: $(u_i, v_i) \in \mathcal{P}_i,\ \forall\ i \in [3, 6]$,
Output: $(u_i, v_i) \in \mathcal{P}_i,\ \forall\ i \in [2]$
$(u_1, u_2) \leftarrow \Pi^{-1}(v_5, v_6)$
$v_1 \leftarrow \beta \cdot (\alpha \cdot u_3 \oplus u_4)$
$v_2 \leftarrow \beta \cdot (u_3 \oplus u_4)$
Return $((u_1, v_1), (u_2, v_2))$

SetL2($\cdot$)
Input: $(u_i, v_i) \in \mathcal{P}_i,\ \forall\ i \in \{1, 2, 5, 6\}$,
Output: $(u_i, v_i) \in \mathcal{P}_i,\ \forall\ i \in [3, 4]$
$u_3 \leftarrow v_1 \oplus v_2$
$u_4 \leftarrow v_1 \oplus \alpha \cdot v_2$
$v_3 \leftarrow \beta \cdot (\alpha \cdot u_5 \oplus u_6)$
$v_4 \leftarrow \beta \cdot (u_5 \oplus u_6)$
Return $((u_3, v_3), (u_4, v_4))$

SetL3 is symmteric to SetL1

$\mathsf{Trace}\mathcal{G}_\pi(\cdot)$
Input: $\tau := \big(\mathsf{C}, \mathcal{P} := (\mathcal{P}_1, \ldots, \mathcal{P}_6), \mathcal{P}^{\mathcal{A}|\mathcal{S}}\big), x \in \{u_1, u_2, v_5, v_6\}$,
Output: $\tau' := (\mathsf{C}', \mathcal{P})$

$\mathsf{C}' \leftarrow \mathsf{C}$

if $x = u_1$
  foreach $(u_2, v_2), (u_5, v_5) \in \mathcal{P}$
    if $(v_5, v_6) = \Pi(u_1, u_2)$
      $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
      Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_6)$
    endif
  endforeach
  foreach $(u_2, v_2), (u_6, v_6) \in \mathcal{P}$
    if $(v_5, v_6) = \Pi(u_1, u_2)$
      $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
      Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_5)$
    endif
  endforeach
  foreach $(u_5, v_5), (u_6, v_6) \in \mathcal{P}$
    if $(u_1, u_2) = \Pi^{-1}(v_5, v_6)$
      $\mathsf{C}' \leftarrow \mathsf{C}' \cup ((u_1, u_2), (v_5, v_6))$
      Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), u_2)$
    endif
  endforeach

cont.

foreach $(u_2, v_2), (u_1', v_1'), (u_2', v_2') \in \mathcal{P}$
  $(v_5, v_6) \leftarrow \Pi(u_1, u_2)$
  $(v_5', v_6') \leftarrow \Pi(u_1', u_2')$
  if $v_5 = v_5'$ or $v_6 = v_6'$
    $\mathsf{C}' \leftarrow \mathsf{C}' \cup \{((u_1, u_2)(v_5, v_6)),$
      $((u_1', u_2')(v_5', v_6'))\}$
    Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_5)$
    Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_6)$
    Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_5')$
    Run $\mathsf{Trace}\mathcal{G}_\pi(\tau' := (\mathsf{C}', \mathcal{P}), v_6')$

- $v_3 \oplus v_4 = v_3' \oplus v_4'$, in which case it has label 5;
- $v_3 \oplus \alpha \cdot v_4 = v_3' \oplus \alpha \cdot v_4'$, in which case it has label 6.

Clearly, $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ is a simple graph (i.e., with no parallel edges and self-loop). We denote the edges having label 1 or 2 as *left edges*, and the edges having label 5 or 6 as *right edges*.

DEFINITION: Valid Ordered

We call $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ *valid ordered* if for every connected component $C = \{M_1, \ldots, M_s\}$ there exists $\delta_1, \cdots, \delta_s \in \{1, -1\}$ such that for all $1 \le i \le s$,

- if $\delta_i = +1$ there is no left edge between M_i and M_j, for all $j < i$;
- if $\delta_i = -1$ there is no right edge between M_i and M_j, for all $j < i$.

We call $((M_1, \delta_1), \ldots, (M_s, \delta_s))$ *valid ordering* of C.

In other words, the vertices of a connected component C can be ordered as $M_1, \ldots, M_s$ such that all M_i can have either only left edges or only right edges with all other previous vertices. Note that if $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ is valid ordered then every subgraph of $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ is valid ordered, as the

$\mathsf{Fill}\mathcal{G}_\pi(\cdot)$
Input: $\tau := \big(\mathsf{C}, \mathcal{P} := (\mathcal{P}_1, \ldots, \mathcal{P}_6), \mathcal{P}^{\mathcal{A}|\mathcal{S}}\big),\ C \subseteq \mathcal{G}_\pi,$
Output: $\tau' := \big(\mathsf{C}', \mathcal{P}' := (\mathcal{P}'_1, \ldots, \mathcal{P}'_6)\big),\ \mathcal{P}^{\mathcal{A}|\mathcal{S}'}$

$\mathsf{C}' \leftarrow \mathsf{C},\ \mathcal{P}' \leftarrow \mathcal{P},\ \mathcal{P}^{\mathcal{A}|\mathcal{S}'} \leftarrow \mathcal{P}^{\mathcal{A}|\mathcal{S}}$

foreach $u_1, u_2 \in \mathsf{C}$
  if $u_1 \notin \mathrm{Dom}(\mathcal{P}_1) : v_1 \leftarrow\!\$\ \{0,1\}^n$ **endif**
  if $u_2 \notin \mathrm{Dom}(\mathcal{P}_2) : v_2 \leftarrow\!\$\ \{0,1\}^n$ **endif**
  $\mathcal{P}^{\mathcal{A}|\mathcal{S}'} \leftarrow \mathcal{P}^{\mathcal{A}|\mathcal{S}'} \cup \{(u_1, v_1), (u_2, v_2)\}$
  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(u_1, v_1), (u_2, v_2)\}$
endforeach
foreach $v_5, v_6 \in \mathsf{C}$
  if $v_5 \notin \mathrm{Ran}(\mathcal{P}_5) : u_5 \leftarrow\!\$\ \{0,1\}^n$ **endif**
  if $v_6 \notin \mathrm{Ran}(\mathcal{P}_6) : u_6 \leftarrow\!\$\ \{0,1\}^n$ **endif**
  $\mathcal{P}^{\mathcal{A}|\mathcal{S}'} \leftarrow \mathcal{P}^{\mathcal{A}|\mathcal{S}'} \cup \{(u_5, v_5), (u_6, v_6)\}$
  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(u_5, v_5), (u_6, v_6)\}$
endforeach

cont.

foreach $M = ((u_1, u_2), (v_5, v_6)) \in \mathsf{C}$
  $((u_3, v_3), (u_4, v_4))$
    $\leftarrow \mathsf{SetL2}\Big(\big(u_1, \mathcal{P}_1(u_1)\big), \big(u_2, \mathcal{P}_2(u_2)\big), \big(\mathcal{P}_5^{-1}(v_5), v_5\big)\big(\mathcal{P}_6^{-1}(v_6), v_6\big)\Big)$
  $\mathcal{P}^{\mathcal{A}|\mathcal{S}'} \leftarrow \mathcal{P}^{\mathcal{A}|\mathcal{S}'} \cup \{(u_3, v_3), (u_4, v_4)\}$
  $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(u_3, v_3), (u_4, v_4)\}$
endforeach
Return $\tau' := (\mathsf{C}', \mathcal{P}'),\ \mathcal{P}^{\mathcal{A}|\mathcal{S}'}$

restricted ordering would work.

DEFINITION: Good Transcript

A transcript $(\mathcal{C}, \mathcal{P})$, preemptively saturated with respect to $\mathcal{P}' \subseteq \mathcal{P}$, is called a *good* transcript if both $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ and $\mathcal{P}_\pi$ are valid ordered.

Description of Bad Events. Let the simulator transcript after i query sessions be $(\mathsf{C}^{\mathcal{S}}, \mathcal{P}, \mathcal{P}^{\mathcal{A}|\mathcal{S}})$. Let $\mathsf{C}^{\mathcal{A}}$ be the list of construction queries made by the adversary. After a query to the simulator or to Π is resolved, let the updated simulator transcript be $(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}', \mathcal{P}^{\mathcal{A}|\mathcal{S}'})$. Before describing our bad events, we first identify three distinct types of simulator queries (so that all the rest are symmetric to one of the three):

1. $\mathcal{A}$ queries u_1 to S_1, and Sim responds with v_1;
2. $\mathcal{A}$ queries v_1 to S_1^{-1}, and Sim responds with u_1;
3. $\mathcal{A}$ queries u_3 to S_3, and Sim responds with v_3.

In each of these cases, if the query comes from a stored query-response pair in $\mathcal{P}$, then the response is deterministic, and no bad events occur. So we can assume for the purpose of the bad events that the queries are 'fresh', in which case following the query Sim adds a primitive input-output pair, and possibly a collection of $\mathcal{P}$-complete chains to its transcript. This can lead to the following bad events:

1. **Injectivity Loss in $\mathcal{P}$:** For some $(u_i, v_i) \in \mathcal{P}'_i \setminus \mathcal{P}_i$, $\exists\ (u'_i, v_i)$ or $(u_i, v'_i) \in \mathcal{P}_i$ where $u_i \neq u'_i, v_i \neq v'_i$;

2. **Accidental Equality in Diffusion Layer:** $\exists$ a $\mathcal{P}^3$ or $\mathcal{P}^4$-chain in $(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}')$ that is not complete in $\mathcal{P}'$;

3. **Hitting in $\mathcal{G}^{\mathsf{Int}}$:** For a query of the form v_1, v_2, u_3, or u_4 (i.e., from the first diffusion layer), $\exists\ M \in \mathcal{G}^{\mathsf{Int}}(\mathcal{P}') \setminus \mathcal{G}^{\mathsf{Int}}(\mathcal{P}),\ M' \in \mathcal{G}^{\mathsf{Int}}(\mathcal{P}')$ such that M, M' are connected by a 5 or 6 labelled edge (Similarly for a query of the form v_3, v_4, u_5 or u_6, or the second diffusion layer);

4. **Hitting in Π:** For some forward Π query $((u_1, u_2), (v_5, v_6)) \in (\mathsf{C}^{\mathcal{S}'} \cup \mathsf{C}^{\mathcal{A}'}) \setminus (\mathsf{C}^{\mathcal{S}} \cup \mathsf{C}^{\mathcal{A}})$, $\exists\ v'_5, v'_6 \in \mathrm{Ran}(\mathcal{P})$ or $\exists\ (v'_5, \cdot), (\cdot, v'_6) \in \mathrm{Ran}(\mathsf{C}^{\mathcal{S}'} \cup \mathsf{C}^{\mathcal{A}'}) \setminus \{(v_5, v_6)\}$ such that either $v_5 = v'_5$ or $v_6 = v'_6$ (Similarly for some backward Π query);

We write bad to denote that one of the above bad events has occurred. Then we can show the following, the proof of which is given in Appendix A.1.

Lemma 8 *For an adversary $\mathcal{A}$ making at most q combined queries,*

$$\Pr(\mathsf{bad}) \leq \frac{2^{14} q^{15}}{2^n}.$$

Lemma 9 *Suppose $(\mathsf{C}^{\mathcal{A}}, \mathcal{P}^{\mathcal{A}})$ is the consistently saturated adversary transcript and the simulator transcript, $(\mathsf{C}^{\mathcal{S}}, \mathcal{P})$, preemptively saturated with respect to $\mathcal{P}^{\mathcal{A}|\mathcal{S}} \subseteq \mathcal{P}$, is a good transcript such that $\mathsf{C}^{\mathcal{A}}$ is well-approximated by $\mathsf{C}^{\mathcal{S}}$. Suppose the adversary queries Q and gets the response R. Let $(\mathsf{C}^{\mathcal{A}'}, \mathcal{P}^{\mathcal{A}'}) = (\mathsf{C}^{\mathcal{A}}, \mathcal{P}^{\mathcal{A}}) \cup (\mathsf{Q}, \mathsf{R})$, and $(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}', \mathcal{P}^{\mathcal{A}|\mathcal{S}'}) = \mathsf{Sim}(\mathsf{C}^{\mathcal{S}}, \mathcal{P}, \mathcal{P}^{\mathcal{A}|\mathcal{S}}, \mathsf{Q})$ and assume bad does not occur on query Q. Then $(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}')$ is preemptively saturated w.r.t. $(\mathcal{P}^{\mathcal{A}|\mathcal{S}})'$ and is a good transcript. Also $\mathsf{C}^{\mathcal{A}'}$ is well-approximated by $\mathsf{C}^{\mathcal{S}'}$.*

Proof 7 *Keeping in mind that $\mathsf{C}^{\mathcal{A}}$ is well approximated by $\mathsf{C}^{\mathcal{S}}$, we make the following observations upon an adversary query.*

$\mathsf{C}^{\mathcal{A}'}$ is well approximated by $\mathsf{C}^{\mathcal{S}}$. *Consider a non-redundant Π query made by $\mathcal{A}$ with input (u_1, u_2) and output (v_5, v_6). Without loss of generality, we assume that it is a forward query. Since we assume that no bad events occur, we have $(u_5, v_5), (u_6, v_6) \notin \mathcal{P}^{\mathcal{A}}$, and there doesn't exist $((u'_1, u'_2), (v'_5, v'_6)) \in \mathsf{C}^{\mathcal{A}} \cup \mathsf{C}^{\mathcal{S}}$ such that $v_5 = v'_5$ or $v_6 = v'_6$. Hence, $\mathsf{C}^{\mathcal{A}} \cup ((u_1, u_2), (v_5, v_6))$ is also well approximated by $\mathsf{C}^{\mathcal{S}}$.*

$(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}')$ **is preemptively saturated w.r.t** $\mathcal{P}^{\mathcal{A}'}$**.** *Since* $(\mathsf{C}^{\mathcal{S}}, \mathcal{P})$ *is preemptively saturated w.r.t* $\mathcal{P}^{\mathcal{A}}$*, each entry in* $\mathcal{P}^{\mathcal{A}'} \setminus \mathcal{P}^{\mathcal{A}}$ *belongs to a* $\mathcal{P}$*-complete line already, given that no bad events have occurred. Since* $\mathcal{P}$ *already contains each element in* $\mathcal{P}^{\mathcal{A}'} \setminus \mathcal{P}^{\mathcal{A}}$, Sim *can identify exactly the primitive pairs to be added to* $\mathcal{P}^{\mathcal{A}}$ *(i.e., the primitive pairs that* $\mathcal{A}$ *can calculate without querying* Sim*). Once* $\mathcal{P}^{\mathcal{A}'} \setminus \mathcal{P}^{\mathcal{A}}$*, is correctly captured, the fact that* $(\mathsf{C}^{\mathcal{S}'}, \mathcal{P}')$ *is preemptively saturated w.r.t* $\mathcal{P}^{\mathcal{A}'}$ *is maintained by the definition of* Sim*.*

$\mathcal{G}^{\mathsf{Int}}(\mathcal{P}')$ **is valid-ordered.** *Since the event* $\mathcal{E}H_{\mathsf{Q}}$ *does not occur, then for any* $(\mathsf{C}, c), (\mathsf{d}, d) \in ((\mathcal{P}' \setminus \mathcal{P}) \times \mathcal{P}) \cup (\mathcal{P} \times (\mathcal{P}' \setminus \mathcal{P}))$*, either no* 12*-edge or no* 56*-edge is incident on* $M^{(u_3,v_3),(u_4,v_4)}$*. Hence the valid ordering of* $\mathcal{G}^{\mathsf{Int}}(\mathcal{P})$ *can be extended to a valid ordering of* $\mathcal{G}^{\mathsf{Int}}(\mathcal{P}')$ *by assigning the direction* $+1$ *to the new vertex of* $M^{(u_3,v_3),(u_4,v_4)}$*, if no* 12*-edge is incident on it, or assigning the* -1 *to it if no* 56*-edge is incident.*

Finally, we can show the entropy matches in the ideal and real worlds. Since in the ideal world, all simulator samplings are done with replacement, we have the following lemma.

Lemma 10

$$\frac{\Pr\left((\mathsf{p}, \mathsf{CDN}^{\mathsf{p}}_{3,2}) = (f, P)\right)}{\Pr\left((\mathsf{Sim}^{\Pi}_{q}, \Pi_q) = (f, P)\right)} \;\geq\; 1.$$

Thus by H-coefficient technique we have our main result.

5.5 Conclusion

In this chapter we showed that 3-round CDNs with two S-boxes in each layer can be indifferentiable with linear diffusion layers without assuming special combinatorial properties on said diffusion layers. Possible future work can extend this to more S-boxes in each layer, and to improve the indifferentiability bound for specially designed diffusion layers.

Chapter 6

The Feistel Construction

6.1 Introduction

Motivation of Ideal Permutation. We commonly see proofs in *ideal models* (e.g., *random oracle model* introduced by Bellare and Rogaway (1993b) and the *ideal cipher model* introduced by Shannon (1949b)), where the unkeyed primitive is replaced by a function with certain idealized properties; while such proofs do not provide the reduction guarantees of proof in the standard models, they are still seen as a commonly accepted framework of arguing that the mode does not have any inherent structural weaknesses.

An ideal cipher is a function that, given a key space $\mathcal{K}$ and a domain Dom, behaves like a family of independent random permutations over Dom indexed by the keys from $\mathcal{K}$. Thus, for a fixed key, an ideal cipher boils down to an ideal permutation or random permutation model. One question of interest to researchers has been whether a proof using random oracles and a proof using ideal permutation are two different assumptions, or whether instead, they are equivalent.

The former of these questions was settled by Coron et al. (2005a), who showed that an ideal cipher is enough to build a random oracle, by building a hash function from an ideal cipher that is indifferentiable from a random oracle. The latter, however, proved to be trickier and led to a long and interesting line of research, the history of which we briefly recap below.

Indifferentiability of Feistel Permutation. The Feistel mode, first used by Horst Feistel as a design component of Lucifer Feistel (1973), is a widely popular transformation for building a permutation from a sequence of functions known as the *round functions*. As such, it is a natural choice for building an ideal cipher out of a random oracle, and was first studied in this

context by Coron, Patarin and Seurin (2008b), who also remarked that it is enough to build a random permutation from a random oracle, because we can always turn this into an ideal cipher by inserting the key as an additional input to the random oracle (which makes the round functions, and thus the permutation, keyed).

An *r-round Feistel network* Ψ_r is constructed by composing r *Feistel permutations* Ψ, each built from a single round function f over $\mathbb{GF}(2^n)$ as follows: $\Psi^f(x, y) := (y, x + f(y))$. The (independent) round functions can be implemented from the random oracle by inserting the round number as an additional input. The problem of building a random permutation from a random oracle using a Feistel network thus reduces to choosing an appropriate number of round r, building a simulator for Ψ_r, and demonstrating that this simulator achieves indifferentiability.

The authors Coron, Patarin and Seurin (2008b) showed that five rounds of Feistel are not enough to build a random permutation, by demonstrating an indifferentiability attack on Ψ_5 that works against any simulator. To complement this, they also produced a proof of indifferentiability for Ψ_6 with a simulator that detects and preemptively completes 3-*chains*. Later, Seurin in his PhD thesis Seurin (2009) provided a simpler proof of indifferentiability for Ψ_{10}, using a simulator similar to that in Coron, Patarin and Seurin (2008b). Unfortunately, on closer scrutiny by Holenstein, Künzler and Tessaro (2010), both proofs were revealed to be flawed, and this set off a series of follow-up works on this topic. In Holenstein, Künzler and Tessaro (2010) the authors also demonstrated an attack against the simulator of Coron, Patarin and Seurin (2008b), thus showing that this simulator is too simple to fool an indifferentiability adversary; they went on to describe a stronger attack on Ψ_6 that works against a large class of "natural" simulators, thus casting doubts on whether six rounds is at all enough to achieve indifferentiability. Later, Seurin himself found an attack Seurin (2015) on Ψ_{10} against the simulator in Seurin (2009).

On the proof side, it seemed prudent to start with less ambitious targets. An early version of Holenstein, Künzler and Tessaro (2010) proposed a simulator for Ψ_{18} that achieved indifferentiability, and in a later revision this was changed to a simulator for Ψ_{14}. Next, two concurrent works by Dai and Steinberger (2015) and Dachman-Soled, Katz and Thiruvengadam (2016) both proved indifferentiability of Ψ_{10}, the former by repairing the flawed Ψ_{10} simulator from Seurin (2009), and the latter by modifying the Ψ_{14} simulator from Holenstein, Künzler and Tessaro (2010). Finally, in what has so far been the latest work in this series, Dai and Steinberger (2016b) established the indifferentiability of Ψ_8 by optimizing their simulator for Ψ_{10} from Dai and Steinberger (2015). The indifferentiability of Ψ_6, however, has remained an open question to this day.

Our Main Result. In this work, we show a proof approach for the indifferentiability of r rounds of Feistel for $r \geq 7$. Concretely, we descibe simulators Sim_8 for $r = 8$ and Sim_7 for $r = 7$, which achieve the following bounds.

Theorem 4 (Main Theorem) *For the simulators* Sim_8 *and* Sim_7,

$$\boldsymbol{Adv}^{\Psi^f_8}_{\Pi\text{-}\mathit{indiff}}(\mathcal{A} \mid \mathsf{Sim}_8) \leq \mathcal{O}(q^{13}/2^n),$$

$$\boldsymbol{Adv}^{\Psi^f_7}_{\Pi\text{-}\mathit{indiff}}(\mathcal{A} \mid \mathsf{Sim}_7) \leq \mathcal{O}(q^{13}/2^n).$$

Related Results. Feistel networks have been the subject of a host of studies in the context of various security notions. Luby and Rackoff Luby and Rackoff (1988) first showed the indistinguishability of Ψ_3 from a random permutation in the chosen-plaintext setting up to the *birthday bound* (for no. of queries not exceeding the square root of the round functions' range-size). Later works in this direction include improved indistinguishability proofs Maurer and Pietrzak (2003); Patarin (2003) and a characterisation of Feistel networks with repeated round functions Nandi (2010).

Mandal, Patarin and Seurin (2012b) showed the *public indifferentiability* of Ψ_6, which is a modified indifferentiability game where the simulator can see all the construction queries by the adversary. Coron et al. (2010) showed the indifferentiability up to the birthday bound of a modification of the Feistel network that uses tweakable random permutations as round functions, and later Bhaumik, Nandi and Raychaudhuri (2021) improved this bound.

Roadmap of the chapter. Firstly, we introduce a chain structure among the underlying primitive query-response pairs, which captures all the interdependencies among these pairs as necessitated by real-world realisability. Following that, we describe certain desirable properties that the simulator intends to maintain in its transcript, which in turn, ensure the correctness of the simulator and prevents the adversary from mounting attacks that succeed with negligible probability. Finally, we illustrate a simulator which maintains said desirable properties by updating its transcript as required, following each adversary query to the simulator.

6.2 Preliminaries

6.2.1 Feistel Mode

Given a function $f : \mathsf{B} \to \mathsf{B}$, the single-round *Feistel permutation* Ψ_f based on f acting over B^2 is defined as

$$\Psi^f(x, y) := (y, x + f(y)).$$

It is easy to see that this is a permutation, with inverse defined as $\Psi^{f-1}(y, z) = (z + f(y), y)$.

Given r functions $f_1, \ldots, f_r : \mathsf{B} \to \mathsf{B}$, the r-round Feistel permutation $\Psi_r^{f_1,\ldots,f_r}$ based on $f_1, \ldots, f_r$ is defined as the sequential composition of the single-round Feistel permutations $\Psi^{f_1}, \ldots, \Psi^{f_r}$, i.e.,

$$\Psi_r^{f_1,\ldots,f_r} := \Psi^{f_r} \circ \ldots \circ \Psi^{f_1}.$$

As a shorthand, we can replace $f_1, \ldots, f_r$ by a single function f which takes an additional index i from $[r]$ as input and behaves like the corresponding f_i. This allows us to denote r-round Feistel permutation based on f as simply Ψ_r^f.

6.2.2 Minimal Fixed Point Set

Suppose elements of X are sets. To each function $f : D \to X$, $g : D^2 \to X$, we associate $f^\cup, g^\cup : \mathscr{P}(D) \to X$ mapping $A \subseteq D$ to $\bigcup_{u \in D} f(u)$ and $\bigcup_{u,u' \in D} g(u, u')$ respectively. We sometimes abuse the notation f or g to denote $f^\cup$ or $g^\cup$ respectively, which would be clear from the nature of the input of the function. We also write $g(u, u)$ as $g(u)$. We call g a symmetric function if $g(u, v) = g(v, u)$ for all u, v.

DEFINITION: Increasing Function and Fixed Point

We call $f : X \to X$ *increasing* if for all $A \in X$, $A \subseteq f(A)$.

We call a set A fixed point for f if $f(A) = A$. We write $A_* := f_*(A)$ to denote the smallest fixed point set for f containing A.

If f is an increasing function defined over a finite set X then for any $A \in X$, then

$$A_* := f^{(m)}(A)$$

for some integer $m > 0$ for which $f^{(m)}(A) = f^{(m+1)}(A)$ where $f^{(m)}$ denotes the self-composition

of f applied m times. Such an m exists as f is an increasing function and X is a finite set. [1]

Closed Set of Points and Lines. We fix a set D whose elements are called points. For $g : D^2 \to \mathscr{P}(D)$ (called a point function), the function $g^+(V) = V \cup g^{\cup}(V)$ is called *accumulating function* based on g. A set A is called closed w.r.t. g or g-closed if for all $u, v \in A$, $g(u, v) \subseteq A$ (equivalently, $g^+(A) = A$, a fixed point set of g^+). We write $g_*(A) := g_*^+(A)$ to denote the smallest g-closed set containing A (equivalently, smallest fixed point set of g^+ containing A).

Line Function. Let us fix a function $\mathsf{Bd} : \mathscr{L} \to \mathscr{P}(D)$ and we call elements of $\mathscr{L}$ lines and elements of $\mathsf{Bd}(\ell)$ *boundary points of the line* ℓ. A function $\mathsf{L} : D^2 \to \mathscr{P}(\mathscr{L})$ is called *line function* if for all u, v, $\mathsf{V}(u, v) := \mathsf{Bd}\big(\mathsf{L}(u, v)\big)$ contains u and v. The point function V maps to all boundary points of the lines mapped by the line function. We write $\mathsf{L}(\mathsf{V}_*(A))$ as $\mathsf{L}_*(A)$.

An algorithm for Computing Smallest Fixed Point Set. Suppose a subroutine $S(v, v')$ computes $\mathsf{L}(v, v')$ for all $v, v' \in D$. Now we define an algorithm $S_*(u)$ which computes $\mathsf{L}_*(\{u\})$ and $\mathsf{V}_*(\{u\})$.

1. S_* maintains global variables initialized as $\texttt{ToVisit} \leftarrow \{u\}$, $\texttt{Visited} \leftarrow \emptyset$ and $L \leftarrow \emptyset$. Let V be always identical to $\texttt{ToVisit} \cup \texttt{Visited}$.
2. While $\texttt{ToVisit} \neq \emptyset$ it runs the following step (3).
3. Computes $\Delta S \leftarrow \bigcup_{\substack{v \in V \\ w \in \texttt{ToVisit}}} S(v, w)$ and updates the following:
   (a) $L \leftarrow L \cup \Delta S$;
   (b) $\Delta V \leftarrow \mathsf{Bd}(\Delta S)$;
   (c) $\texttt{Visited} \leftarrow \texttt{Visited} \cup \texttt{ToVisit}$;
   (d) $\texttt{ToVisit} \leftarrow \Delta V \setminus V$;
4. Finally, it returns (V, L).

It is easy to see that if $S_*(\{u\})$ returns (L, V) then $\mathsf{L}_*(\{u\}) = L$ and $\mathsf{V}_*(\{u\}) = V$.

[1] By induction on n, we show that $f^{(n)}(A) \subseteq B$ for all n where B is any fixed point set containing A. Hence A_* is the smallest fixed point (which contains A).

Well Approximation of $\mathsf{V}_*, \mathsf{L}_*$. Let $\mathscr{L}_w, \mathscr{L}_d \subseteq \mathscr{L}$ and there is a function $d : \mathscr{L}_w \to \mathscr{L}_d$ such that for all $\ell \in \mathscr{L}_w$, $\mathsf{Bd}(d_\ell) \subseteq \mathsf{Bd}(\ell)$ with $|\mathsf{Bd}(\ell) \setminus \mathsf{Bd}(d_\ell)| = 1$ where $d(\ell) := d_\ell$. We fix the collections $\mathscr{L}_w$ and $\mathscr{L}_d$, and the function $d(\cdot)$. We call elements of $\mathscr{L}_w$ (resp. $\mathscr{L}_d$) maximal wrap-around (resp. direct) lines. The maximal direct line d_ℓ associated with a maximal wrap-around line ℓ may have fewer boundary points.

DEFINITION: Approximation

A collection of lines $\mathscr{L}_1$ is called an *approximation* of $\mathscr{L}_2$ if

1. for every $\ell \in \mathscr{L}_1 \setminus \mathscr{L}_2$, we have $d_\ell \in \mathscr{L}_2$ and
2. conversely, for every $\ell' \in \mathscr{L}_2 \setminus \mathscr{L}_1$, we have $\ell' = d_\ell$ for some $\ell \in \mathscr{L}_1$.

For any $\ell \in \mathscr{L}_w$, we call the point $v \in \mathsf{Bd}(\ell) \setminus \mathsf{Bd}(d_\ell)$ leaf w.r.t. L if $\mathsf{L}(v) = \{\ell\}$ and for all $w \neq v$, $\mathsf{L}(w, v) = \emptyset$. In this case, we call ℓ leaf line.

DEFINITION: Well-Approximation

Let $(\mathsf{L}', \mathsf{V}')$ and (L, V) be pairs of associated line and point functions. We call (L, V) is a *well-approximation* of $(\mathsf{L}', \mathsf{V}')$ if, for all $L := \mathsf{L}(u, v) \neq \mathsf{L}'(u, v) := L'$, L is an approximation of L' and all $\ell \in L \setminus L'$ is a leaf line.

It is straightforward to see the following result.

Lemma 11 *If (L, V) is an well-approximation of $(\mathsf{L}', \mathsf{V}')$ then for all u, $\mathsf{L}_*(\{u\})$ is an approximation of $\mathsf{L}'_*(\{u\})$.*

6.2.3 View and Transcript

Let $\mathcal{A}$ be a deterministic algorithm interacting with either $\mathsf{RW} := (\Psi^{\mathsf{RF}}, \mathsf{RF})$ or $\mathsf{IW} := (\Pi, \mathsf{Sim}^{\Pi})$ where Sim is a simulator and RF is a real-world primitive oracle which is based on a random function RF (abusing notation). We call the first oracle construction interface, that grants access to both forward and backward queries. We record a backward construction query $y \in \mathsf{B}^2$ with a response $x \in \mathsf{B}^2$ as (x, y) (i.e., in the forward direction only). The second oracle, called primitive interface, however, may return additional query responses (there is no loss as it can only increase the advantage to the distinguisher) which must include a pair $(x, \hat{x})$, where x is the current query and $\hat{x}$ represents the primitive response of x. In addition to the pair, it also responds to some additional queries based on the current transcript. Thus, all responses collectively to $\mathcal{A}$ are

represented by a pair

$$\delta := (P, F)$$

where

- P is a $2n$-bit partial injective function[2], called *construction view* (representing query responses of the construction interface), and
- F is a B_{prim} to B partial function, called *primitive view* (representing query-responses of primitive interface).

We call all such pairs $\delta = (P, F)$ **views**. We also assume that after q queries, *the distinguisher reveals all construction queries* to the primitive interface, and a post-query phase view $\delta^* = (P^*, F^*)$ is obtained. So, the second oracle or primitive interface (both for the real and ideal world) returns a view. Revealing more information to the distinguisher and revealing all construction queries by the distinguisher to the primitive interface after all designated queries are over can only increase the distinguishing advantage.

DEFINITION: View and Transcript

Let $\delta^t := (\delta P^t,\ \delta F^t)$ denote the collection of all query responses obtained on t^{th} query x_t. Then the view of $\mathcal{A}$ at time $t \in [q]$, where q is the number of queries by $\mathcal{A}$, is defined as

$$\mathcal{V}_t(\mathcal{A}^{\mathcal{O}}) := \tau^t := (P^t := \bigcup_{j \le t} \delta P^j,\ F^t := \bigcup_{j \le t} \delta F^j).$$

We define the *transcript* of $\mathcal{A}$ as

$$\textsc{Transc}(\mathcal{A}^{\mathcal{O}}) = \tilde{\tau} := (x_1, \tau^1, x_2, \tau^2, \ldots, x_q, \tau^q, \tau^*), \quad \tau^* = (P^q \cup P^*,\ F^q \cup F^*)$$

where $\delta^* = (P^*, F^*)$ is the post-query phase view.

6.3 Lines, Segments and Views for Feistel Construction

In this section, we introduce lines, segments and different types of views for a general r-round Feistel construction where $r \ge 7$. Sometimes some terminologies are based on the cases where

[2] A partial function f from D to R, denoted as $F : D \dashrightarrow R$, is a subset of $D \times R$ such that for all $(x, y), (x, z) \in f$ we have $y = z$ (and the common value is denoted as $f(x)$. Let $\text{Dom}(f) = \{x \in D : \exists y, (x, y)\}$ and for all $x \notin \text{Dom}(F)$.

$r = 2k$ (even umber of rounds) or $r = 2k - 1$ (odd number of rounds) for some integer k.

Notation on Point Tuple. We call the set $[0..r+1]$ shore number set and $\{i\} \times \mathsf{B}$ is the set of all points placed at shore i. The point set $[0..r+1] \times \mathsf{B}$ is also regarded as the block set B, implicitly incorporating shore numbers through the suffixes used in the notation. For instance, ℓ_i, ℓ'_i, etc., denote elements of both B and $\{i\} \times \mathsf{B}$, discernible from the context. A tuple $\alpha = (\alpha_1, \ldots, \alpha_s)$ is called *non-repeating* if α_i's are distinct, and in this case, we write the set of all elements in the tuple as (α) (also as α, abusing notation). We call a non-repeating α passes through x if $x \in \alpha$. For a collection of non-repeating tuples $\mathscr{A}$, we write $(\mathscr{A}) = \cup_{\alpha \in \mathscr{A}} (\alpha)$. Let $i, j \in [0..r+1]$. We define a non-repeating tuple of contiguous shore numbers and contiguous point tuple as follows:

(direct) $i < j$**:** $(i..j) = (i, i+1, \ldots, j)$, $\ell_{(i..j)} = (\ell_i, \ell_{i+1}, \ldots, \ell_j)$.

(wrap-around) $r \geq i > j \geq 1$**:** $(i..j) = (i, i{+}1, \ldots, r{+}1, 0, 1, \ldots, j)$, $\ell_{(i..j)} = (\ell_i, \ldots, \ell_{r+1}, \ell_0, \ell_1, \ldots, \ell_j)$.

We write $(i..j)^c := [0..r{+}1] \backslash (i..j)$. If $\ell := \ell_{(i..j)}$ is a contiguous point tuple we write $\mathsf{sh}(\ell) := (i..j)$ and ℓ_k to denote the k shore point of ℓ, $k \in (i..j)$. We write $\mathsf{Bd}(\ell) = \{\ell_i, \ell_j\}$ and we extend $\mathsf{Bd}(\mathscr{L}) := \cup_{\ell \in \mathscr{L}} \mathsf{Bd}(\ell)$ for a collection of contiguous point tuples $\mathscr{L}$. We call ℓ *complete* if $\mathsf{sh}(\ell) = (0..r+1)$. For any contiguous shore tuple $(i..j)$ and $\ell_{(i..j)}$ we call

$$\textsc{int}(i..j) = \big((i..j) \cap [r]\big) \setminus \{i, j\}, \quad \textsc{int}(\ell) := \{\ell_k \ : \ k \in \textsc{int}(i..j)\}$$

the set of interior primitive shore numbers and points respectively. If $(i'..j') \subseteq (i..j)$, we also write $\ell_{(i'..j')}$ to denote the sub-tuple restricted at the tuple of contiguous shore numbers $(i'..j')$. Two contiguous point tuples ℓ and ℓ' are said to *meet* at shore i (resp. at point ℓ_i) if $\ell \cap \ell' = \{\ell_i\}$. In this case, ℓ_i is called the *meeting point*. We say that ℓ passes through a point v if $v \in (\ell)$. Similarly, we define ℓ passing through more than one point. With respect to this work, we restrict ourselves to a class for which no two distinct point tuples intersect at more than one point.

Segments and Lines. Let $\tau = (P, F)$ be a view throughout this section. A point $v \in \mathsf{DF}$ is called a F-defined point. All points of $([r] \times \mathsf{B}) \setminus \mathsf{DF}$ are called F-undefined points.

DEFINITION: Segments and Lines

A contiguous point tuple $\ell_{(i..j)}$ with $|i - j| > 1$, is called a $(i..j)$ (or $(i..*)$ or $(*..j)$) τ-segment (we skip $(i..j)$ or τ or both whenever it is not relevant or does not lead to any ambiguity) if

$F(\ell_k) = \ell_{k-1} \oplus \ell_{k+1},\ \forall\ k \in \text{INT}(i..j),\ P(\ell_0, \ell_1) = (\ell_r, \ell_{r+1})$ for a wrap-around $(i..j)$.

The segment $\ell_{(i..j)}$ is called direct (resp. wrap-around) if $(i..j)$ is direct (resp. wrap-around).

A segment $\ell_{(i..j)}$ is called *line* if $\ell_i, \ell_j \notin \mathsf{DF}$, otherwise we call non-maximal segment. The elements of $\mathsf{Bd}(\ell) := \{\ell_i, \ell_j\}$ are called boundary points of ℓ.

A segment captures the computation of Feistel. For example, if $\Psi^F(\ell_0, \ell_1) = (\ell_r, \ell_{r+1})$ where $\ell_1, \ell_2, \ldots, \ell_r$ are the r consecutive inputs of the underlying primitive functions then $(\ell_0, \ell_1, \ldots, \ell_{r+1})$ is a complete line. In the case of a partial computation, a line captures the maximal possible computation of Ψ^F can be made. Suppose $\ell_{(i..j)}$ is a direct line then

$$\Psi^F_{(i..j)}(\ell_i, \ell_{i+1}) = (\ell_{j-1}, \ell_j).$$

In case of wrap-around line $\ell_{(j..i)}$, we have

$$\begin{aligned} \Psi^F_{(j..r+1)}(\ell_j, \ell_{j+1}) &= (\ell_r, \ell_{r+1}), \\ \Psi^F_{(0..i)}(\ell_0, \ell_1) &= (\ell_{i-1}, \ell_i), \\ \Psi^F(\ell_0, \ell_1) &= (\ell_r, \ell_{r+1}). \end{aligned}$$

Hence, we should have $\Psi^F_{(i-1..j+1)}(\ell_{i-1}, \ell_i) = (\ell_j, \ell_{j+1})$. However, the internal computation of this is unknown from the given view.

Value Points and Boundary Points. We call a segment ℓ Gen-k (resp. Gen-k+ or Gen-k−) if the number of F-defined points in ℓ is k (resp. at least k or at most k). We call ℓ Gen-i/j line (and similarly extend for more than two choices) if it is either Gen-i or Gen-j line. Let $\mathscr{L}(\tau)$ (resp. $\mathscr{S}(\tau)$) be the collection of all lines (resp. segments) excluding all Gen-1 direct line (resp. segment) $\ell_{(i..i+2)}$ such that both ℓ_i, ℓ_{i+2} are not boundary points of a wrap-around or Gen-2 segment. The elements of $\text{VAL}(\tau) := \{\mathscr{L}(\tau)\} \cup \mathsf{DF}$ (resp. $\mathsf{Bd}(\tau) := \mathsf{Bd}(\mathscr{L}(\tau))$) are called value points of τ or τ-value points (resp. boundary points or τ-boundary points). Note that $\mathsf{Bd}(\tau) \subseteq \text{VAL}(\tau) \setminus \mathsf{DF}$ and if $v \in \text{VAL}(\tau) \setminus (\mathsf{Bd}(\tau) \cup \mathsf{DF})$ then it must be from shore 0 or $r+1$ (lying on a wrap-around line, not as a boundary point).

Line-class Notation. We use WA, dir, Gen-x, x $\in \{k, k+, k-\}$, $(i..j)$ in the suffix of $\mathscr{L}(\tau)$ to denote the corresponding class of lines.[3] For example, $\mathscr{L}_{\mathsf{Gen\text{-}2+,dir}}(\tau)$ denotes the collection of all Gen-2+ direct lines. Given a set of boundary points Bd, we write Bd_i (resp. $\mathsf{Bd}_{i/j}$) to denote the set of all boundary points from Bd with shore i (resp. i or j). For example, $\mathsf{Bd}_{1/r}(\tau)$ denotes the set of all boundary points of τ at shore 1 or r.

NOTATION: Line Collection Through a Point and Their Boundary Points.

$\mathscr{L}_{|v}$ denotes the set of all lines of $\mathscr{L}$ passing through v. We write $\text{OP}_{\mathscr{L}}(v) := \mathsf{Bd}(\mathscr{L}_{|v})\backslash\{v\}$.

For a set of boundary points B, we write $\mathscr{L}(u \bowtie B)$ to denote the set of all lines passing through u and v, for all $v \in B$. Note, $\mathscr{L}_{|_v}$ is identical to $\mathscr{L}(v \bowtie \text{OP}_{\mathscr{L}}(v))$.

Fresh Point. A point v is $\mathscr{L}$-fresh (resp. fresh) if $|\mathscr{L}_{|v}| = 1$ (resp. $|\mathscr{L}(\tau)_{|v}| = 1$).

Meeting at Boundary (Intersection). If segments ℓ and ℓ' meet at shore i such that ℓ_i is a boundary point then we denote $\ell \curlywedge_i \ell'$ or $\ell \curlywedge_{\ell_i} \ell'$ and we call ℓ, ℓ' intersect at shore i (the meeting point ℓ_i is also called intersection point). We also write $\ell \curlywedge \ell'$ to denote that $\ell \curlywedge_i \ell'$ for some $i \in [r]$. A collection of lines $\mathscr{L}$ is called *untangled* if two lines from $\mathscr{L}$ can only meet at an F-defined point.

Forcing and Mixed Forcing Intersection. A 4-set of lines

$$L := \{\ell_{((r-1)/r..1)}, k_{(3..1)}, k'_{(3..1)}, \ell'_{((r-1)/r..1)}\}$$

such that (i) $\ell \curlywedge_1 k \curlywedge_3 k' \curlywedge_1 \ell'$, and (ii) $\ell_0 \oplus \ell'_0 = k_0 \oplus k'_0 \oplus k_4 \oplus k'_4$ is called a forcing intersection set at shore 1. We similarly define the forcing intersection at shore r. Let $\mathfrak{L}_{\mathsf{FI}}$ be the collection of all forcing intersection sets. A 3-set of lines $L' := \{\ell_{((r-1)/r..1)}, k_{(3..1)}, \ell'_{(3..i)}\}$ for $i \in [6..1]$ $\ell \curlywedge_1 k \curlywedge_3 \ell'$ and $\ell_0 \oplus \ell'_4 = k_0 \oplus k_4$ is called mixed forcing intersection set at shore 1. We similarly define it for shore r and let $\mathfrak{L}_{\mathsf{MFI}}$ be the collection of all forcing intersection sets.

The lines ℓ, ℓ' described above in L, L' are called side lines. Let $\ell \curlywedge_u k \curlywedge k' \curlywedge \ell'$ (or forced mixed intersection $\ell \curlywedge_u k \curlywedge \ell'$) and $\mathscr{L}$ be a collection of lines such that $\ell' \in \mathscr{L}, \ell \notin \mathscr{L}$ then we call ℓ 2-step line at u for $\mathscr{L}$.

[3] For line collection $\mathscr{L}(\tau)$ etc., we skip τ whenever there is no ambiguity (e.g., no specific line collection $\mathscr{L}$ is present or not more than one view is present in the same context).

6.3.1 Types of Views

Feistel View. For $(i'..j') \subseteq (i..j)$ and a segment $\ell_{(i..j)}$, $\ell_{(i'..j')}$ is said to be extended to $\ell_{(i..j)}$ (and $\ell_{(i'..j')}$ is said to be a sub-segment of ℓ). It is easy to see that every direct segment can be extended to a unique line (the longest segment containing the segment). So, for any two contiguous shored u, v with at least one F-sampled point, there is a unique line, denoted as $\overline{uv}$, containing u, v. A segment ℓ is called τ-complete if it extends to a complete line, otherwise, it is called τ-incomplete. It is also easy to see that a wrap-around segment cannot be extended to multiple lines. However, it is not guaranteed that every wrap-around segment can be extended to a line. Whenever a wrap-around segment ℓ is extended to a line we denote it as $\overline{\ell}$.

DEFINITION: Feistel view

We call a view *Feistel* if every Gen-2 wrap-around segment ℓ is extended to a unique line $\overline{\ell}$. This unique line is either a wrap-around or a complete line.

If τ is Feistel then ℓ can be extended to a unique wrap-around line or a complete line, denoted as $\overline{\ell}$. For a collection of segments $\mathscr{S} \subseteq \mathscr{S}(\tau)$, we write

$$\overline{\mathscr{S}^\tau} := \overline{\mathscr{S}^\tau} = \{\overline{\ell} \ : \ \ell \in \mathscr{S}\}.$$

Lemma 12 *If τ is not Feistel view then there is a Gen-(r - 2) wrap-around τ-incomplete segment. Equivalently, if every incomplete τ-segment has at most $(r-3)$ F-defined points, the view is Feistel.*

Proof 8 *Let $\ell_{(i..j)}$ be a wrap-around segment with $r-3$ F-defined point and $\ell_i, \ell_j \in \mathsf{DF}$ and so $i = j + r - 4$. Now, $\ell_{i-1} = F(\ell_i) \oplus \ell_{i+1}, \ell_{j+1} := F(\ell_j) \oplus \ell_{j-1} \notin \mathsf{DF}$. Hence, $\ell_{(i-1..j+1)}$ is a wrap-around line. Hence τ is a Feistel view.* □

Internally Complete View. Contiguous points u_i, u_{i+1} (similarly for three or more contiguous points) are called *completed* if there is a complete line passing through u_i, u_{i+1}. Let $\mathscr{L}_{k/k+1}(\tau)$ (resp. $\mathscr{L}_{k,k+1}(\tau)$) consist of all direct lines passing through $u \in \mathsf{DF}_k \cup \mathsf{DF}_{k+1}$ (resp. $u \in \mathsf{DF}_k$ and $v \in \mathsf{DF}_{k+1}$.

1. A view is called $(k, k+1)$-complete if every pair $(u_k, u_{k+1}) \in \mathsf{DF}_k \times \mathsf{DF}_{k+1}$ is completed (i.e., $\ell \in \mathscr{L}_{k,k+1}$ is a complete). A view is called $(k, k+1)$-semi-complete if any τ-incomplete line ℓ passing through $u_k, u_{k+1} \in \mathsf{DF}$ is a $(k-1..k+2)$ line. Equivalently, $\ell \in \mathscr{L}_{k,k+1}$ is either a complete line or $(k-1..k+2)$ line.

2. A view is called $(k-1, k, k+1)$-complete if every triple $(u_{k-1}, u_k, u_{k+1}) \in \mathsf{DF}_{k-1} \times \mathsf{DF}_k \times \mathsf{DF}_{k+1}$ is completed. In this case, every incomplete line $\ell \in \mathscr{L}_{k-1,k}$ (resp. $\ell \in \mathscr{L}_{k,k+1}$) is

a $(*..k+1)$ line (resp. $(k-1..*)$ line.),

A view is called internally complete if it is either $(k, k+1)$-complete (when $r = 2k$) or $(k-1, k, k+1)$-complete (when $r = 2k-1$). For even r, the internally semi-complete view is a $(k, k+1)$ semi-complete view.

Boundary Fresh View. A $(0..*)$ or $(*..r+1)$ (resp. $(r..*)$ or $(*..1)$) Gen-2+ τ-incomplete line is called direct (resp. wrap-around) boundary line. Let $\mathscr{L}_{\mathsf{dir\_BDRY}}(\tau)$ and $\mathscr{L}_{wa\_\mathrm{BDRY}}(\tau)$ be the set of all direct and wrap-around boundary lines, respectively. Let

$$\mathsf{Rev}(\tau) = \mathsf{Bd}_{1/r}(\mathscr{L}_{wa\_\mathrm{BDRY}}(\tau))$$

and points of the above set are called revealed-boundary points. Let $\mathscr{L}_{(i..1)\curvearrowleft_{\mathsf{Rev}}}$ (resp. $\mathscr{L}_{(r..i)\curvearrowleft_{\mathsf{Rev}}}$) be the collection of all $\ell_{(i..1)}$ (resp. $\ell_{(r..i)}$) lines intersecting at a revealed-boundary point at shore 1 (resp. r).

DEFINITION: Boundary Extendable Lines and Boundary-Fresh View.

A $(0..*)$ (resp $(*..r+1)$) direct line is called τ-extendable if $(\ell_0, \ell_1) \in \mathrm{Dom}(P)$ (resp. $(\ell_r, \ell_{r+1}) \in \mathrm{Ran}(P)$). Let $\mathscr{L}_{\mathsf{dir\_BDRY}*}(\tau)$ be the set of all τ-extendable direct boundary lines.[a]

We call τ *boundary extendable* if $\mathscr{L}_{\mathsf{dir\_BDRY}*}(\tau) = \mathscr{L}_{\mathsf{dir\_BDRY}}(\tau)$.

τ is called *boundary-fresh* if

BF1: $\mathscr{L}_{wa\_\mathrm{BDRY}} = \overline{\mathscr{L}}_{\mathsf{dir\_BDRY}*}$ (every non-maximal $(r-1..2)$ segment is τ-complete) and

BF2: every revealed-boundary point v lies on exactly one boundary wrap-around line, denoted as $\ell(v)$. So,

$$\mathscr{L}_{wa\_\mathrm{BDRY}}(\tau)_{|v} = \{\ell(v)\}, \quad \mathrm{OP}_{\mathscr{L}_{wa\_\mathrm{BDRY}}}(v) = \mathsf{Bd}(\ell(v)) \setminus \{v\}.$$

[a] Note that $\mathscr{L}_{wa\_\mathrm{BDRY}}(\tau) \subseteq \overline{\mathscr{L}}_{\mathsf{dir\_BDRY}*}(\tau)$. For example, a $(p..r+1)$ direct line $(p < r)$ can be extended to a $(p..1)$ boundary line or a $(p..2/\cdots/p-2)$ non-boundary line.

Good View.

$r = 2k$**:** A view is called good (resp. semi-good) if it is boundary-fresh, Feistel and $(k, k+1)$-complete (resp. $(k, k+1)$-semi-complete).

$r = 2k-1$**:** A view is called good if it is boundary-fresh, Feistel and $(k-1, k, k+1)$-complete.

6.3.2 Extensions and Completion of Views

A view $\tau' = (P', F')$ is called an *extension* of $\tau = (P, F)$, denoted $\tau \subseteq \tau'$, if

$$P \subseteq P', \quad F \subseteq F'.$$

DEFINITION: Completion of Collection of Lines

Let $\mathscr{L}$ be a collection of lines for a view $\tau = (P, F)$. Let $\tau' = (P, F' := C \cup F)$ be a view.

- We say that $\mathscr{L}$ *is completed by* C if every $\ell \in \mathscr{L}$ is τ'-complete.
- Moreover, we call C or τ' $\mathscr{L}$-completion if $\mathscr{L}$ is completed by C, and C contains only those elements of the primitive view required to define $\overline{\ell}$ for all $\ell \in \mathscr{L}$. If $\mathscr{L} = \mathscr{L}_{(k-1..k+2)}$ then $\mathscr{L}$-completion is also called $(k, k+1)$-completion (or an internal completion when $r = 2k$).

We write $\mathcal{N} := \text{NEW}(\tau \to \tau')$ to denote the set of all segments $\ell \in \mathscr{S}(\tau')$ (called new segments) such that no sub-segments of ℓ is a τ-segment.

We call τ' $\mathscr{L}+(k, k+1)$ completion (or $\mathscr{L}$-internal completion for $r = 2k$) of τ if there is a view τ'' (intermediate view) which is $\mathscr{L}$ completion of τ and τ' is $(k, k+1)$-completion of τ''.

Different Types of Extension of Lines Let τ' be an extension of τ and a τ-line ℓ is extended to ℓ' in τ'. We call ℓ has one-step extension if $\delta := |\ell' \setminus \ell| = 1$ or 2, with $\text{INT}(\ell') \setminus \text{INT}(\ell) \subseteq \mathsf{Bd}(\ell)$. It is called one-sided if $\delta = 1$, or both-sided if $\delta = 2$.

A wrap-around line $\ell_{(i..1)}$ or $\ell_{(r..i)}$ undergoes two-step extension if ℓ is extended to $(i..3)$ or $(r-2..i)$ line. A direct $(3..i)$ or $(i..r-2)$ line undergoes two-step extension if the line is extended to $(1..i)$ or $(i..r)$ line.

DEFINITION: Valid collection of lines

Let $\mathscr{L}_1 \subseteq \mathscr{L}(\tau) \setminus \mathscr{L}$ be all lines sharing at least one common boundary point with a given collection of lines $\mathscr{L}$. Let $B := \mathsf{Bd}(\mathscr{L})$. Suppose we have the following:

1. $\mathscr{L}_1$ doesn't have any $(3..1)$ or $(r..r-2)$ or $(r-1..2)$ line.
2. If $\ell \in \mathscr{L}_1$ with $\mathsf{Bd}(\ell) \subseteq B$ then it is either a direct line or a Gen-0 line.
3. If $\ell \in \mathscr{L}_1$ is a boundary wrap-around line with v as a revealed-boundary point then

$v \notin B$.

4. If a shore-$1/r$ intersection of Gen-1+ wrap-around lines $\ell \curlywedge_u \ell'$ with $u \notin \mathsf{Bd}(\mathscr{L})$ then both ℓ, ℓ' cannot be in $\mathscr{L}_1$.

5. A 2-step wrap-around line w.r.t. $\mathscr{L}$ is a Gen-0 line.

6. If ℓ, m are 2-step intersecting lines w.r.t. $\mathscr{L}$ then ℓ, m must intersect at the 2-step point only. If a 2-step line ℓ intersects at $m \in \mathscr{L}_1$ then it must intersect at a 2-step point.

7. For $r = 2k$: If $\ell \in \mathscr{L}_1$ is a $(k-1..*)$ Gen-1+ direct line then $\ell_{k-1} \notin B$.

8. For $r = 2k-1$: If $\ell \in \mathscr{L}_1$ is a $(k-2..*)$ Gen-2+ direct line then $\ell_{k-2} \notin B$.

We call $\mathscr{L}$ *valid* if it satisfies the above conditions. We say that $\mathscr{L}$ completes u if $u \in \mathsf{Bd}(\mathscr{L})$. The smallest collection of τ-lines which is valid and completes u is denoted as $\mathscr{L}(u \mid \tau)$.

It is easy to see that $\mathscr{L}(u \mid \tau)$ exists and it is unique (Namely, it is the intersection of all collections of lines which are valid. The complete collection is a trivial valid collection and hence the collection of valid line collections is not empty). Suppose τ' is $\mathscr{L}$-completion of τ for some $\mathscr{L}$ (Note that $\mathscr{L}_1$ is not completed). Moreover, assume that a 2-step line for $\mathscr{L}$ undergoes 2-step extension. If $\mathscr{L}$ is not valid then τ' is not good. This can be verified by considering 1-7 points of a valid collection one by one.

1. Suppose point 1 is not satisfied then say $\ell_{(3..1)}$ is extended in τ'. However, then ℓ must be τ'-complete. A similar argument holds for the other choices of lines.

2. Suppose point 2 or 3 is not satisfied then τ' does not satisfy BF1.

3. Violation of point 4 leads to a violation of BF2.

4. Violation of point 5 leads to a violation of BF1 (as the number of F-defined points are increased by 2).

5. Violation of point 6 leads to a violation of BF2.

6. Violation of points 7 or 8 leads to the violation of internally complete property.

DEFINITION: simple extension

Let $\tau' = (P, F' := F \sqcup C)$ be a $\mathscr{L}$-completion of τ such that the following holds:

S1 All $\ell \in \mathscr{L}$ is extended to a complete line ℓ' such that $(\ell' \setminus \ell) \cap \text{VAL}(\tau) = \emptyset$ (we call ℓ freshly extended). This implies that $\text{Dom}(C) \cap \text{VAL}(\tau) = \mathsf{Bd}(\mathscr{L})$.

S2 Every τ-line $\ell \in \mathscr{L}_1$ is freshly and one or two steps extended to ℓ' in τ' (i.e., $(\ell' \setminus \ell) \cap \text{VAL}(\tau) = \emptyset$).

S3 $\ell \in \mathscr{L}_1$ has two steps extension if and only if it is a 2-step line for $\mathscr{L}$.

S4 When $r = 2k - 1$, there is no new $(k-2..k+2)$ segment.

S5 When $r = 2k$ (even), every new segment $\ell_{(k-1..k+2)}$ is a $(k-1..k+2)$ line.

We call τ' a simple $\mathscr{L}$-completion. When $r = 2k$ and τ' is a simple $\mathscr{L}_{(k-1..k+2)}(\tau)$-completion, we call it a simple internal extension.

Let τ' be $\mathscr{L} + (k, k+1)$ completion of τ with an intermediate view τ''. We call it a simple extension if τ'' is a simple $\mathscr{L}$ completion extension of τ, and τ' is a simple internal extension of τ''.

An extended view $\tau'' = (P \cup \{((a,b),(c,d))\}, F)$ is called a simple construction view extension if $b \notin \text{VAL}(\tau)$ (for a backward query) or $c \notin \text{VAL}(\tau)$ (for a forward query). So, at least one of b and c is not a τ-value point.

The following lemma is straightforward from the definition of simple extension as defined above.

Lemma 13 $r = 2k$**:** *Let τ be a good view and τ' be a simple $\mathscr{L}$-completion of τ for an valid collection $\mathscr{L}$, then τ' is a semi-good view. Let τ' be a semi-good view and τ'' be a simple internal extension of τ, then τ' is a good view. Hence, for any good view τ and an valid $\mathscr{L}$, if τ'' is a simple $\mathscr{L} + (k, k+1)$ completion then τ'' is a good view.*

$r = 2k - 1$**:** *Let τ be a good view and τ' be a simple $\mathscr{L}$-completion of τ for an valid collection $\mathscr{L}$, then τ' is a good view.*

If τ is a good view and τ' is a simple construction view extension then τ' is also a good view.

DEFINITION: Boundary BAD Random Permutation.

Let $\tau = (P, F)$ be a good view, $\mathscr{L}$ be a collection of lines and u be a F-undefined point. Let $\mathrm{Dom}' = \mathsf{DF} \cup \mathsf{Bd}(\mathscr{L}) \cup \{u\}$. A P-conditioned random permutation Π (i.e., $\Pi \supseteq P$) is called boundary bad w.r.t. $(\tau, \mathscr{L}, u)$ if one of the following holds:

case $v_{r-1} \in \mathrm{Dom}', v_r \in \mathsf{DF}, (v_r, v_{r+1} := v_{r-1} \oplus F(v_r)) \notin \mathrm{Ran}(P)$: $\Pi_2^{-1}(v_r, v_{r+1}) \in \textsc{val} \cup \{u\}$;

case $v_2 \in \mathrm{Dom}', v_1 \in \mathsf{DF}, (v_0 := F(v_1) \oplus v_2, v_1) \notin \mathrm{Ran}(P)$: $\Pi_1(v_0, v_1) \in \textsc{val} \cup \{u\}$;

case $u_1, v_1 \in \mathsf{DF}, u_2, v_2 \in \mathrm{Dom}', (u_0 := F(u_1) \oplus u_2, u_1) \notin \mathrm{Dom}(P)$: $\Pi_1(u_0, u_1) = \Pi_1(v_0, v_1)$;

case $u_r, v_r \in \mathsf{DF}, u_{r-1}, v_{r-1} \in \mathrm{Dom}', (u_r, u_{r+1} := u_{r-1} \oplus F(u_r)) \notin \mathrm{Ran}(P)$: $\Pi_2^{-1}(u_r, u_{r+1}) = \Pi_2^{-1}(v_r, v_{r+1})$;

6.4 Indifferentiability Analysis of r round Feistel

Let τ^t denote the view at time t and $\tau^0 = \emptyset$ which is a vacuously good view. Now, at time t, an adversary $\mathcal{A}$ makes a query x_t which is a deterministic function of τ^{t-1} (assuming the adversary is deterministic).

DEFINITION: Good Sequence of Transcripts

We call $(\tau^1, \ldots, \tau^q, \tau^*)$ good if τ^t is $\mathscr{L}_t$+ simple completion of τ^{t-1} where $\mathscr{L}_t = \mathscr{L}(x_t \mid \tau^{t-1})$ and τ^* is $\mathscr{L}_{\textsc{wa}}(\tau^q)$ simple completion.

Suppose $\tau = (P, F)$ is a good view and $\tau^* = (P, F^*)$ is a completion of all wrap-around lines (also called P-completion). Then, $P \subseteq \Psi^{F^*}$. For a good $\tau := (\tau^1, \ldots, \tau^q, \tau^* = (P^*, F^*))$, all τ^t are good view and τ^* is also a good view containing no wrap-around lines (i.e., $P \subseteq \Psi^{F^*}$). For any such good transcript τ, we have

$$\Pr_{\mathsf{RW}}(\tau) = 2^{-n|F^*|}$$

as the transcript is completely determined by F^*. Now we compute the probability of realizing a good transcript in the ideal world. Let Sim^{Π} be a simulator which returns δ^t for all t and δ^* in the final phase. Suppose Sim has primitive view F^{t-1} and on query x_t it works as follows:

Find Phase: $\mathsf{Sim}^{\Pi}(x_t \mid F^{t-1})$ finds a collection of lines $\mathscr{L}_t$. It extends F^{t-1} to F^t just by completion of $\mathscr{L}_t$. It Returns $\delta^t = F^t \setminus F^{t-1}$ to the adversary.

Final Phase: In the final phase in which P^t is revealed to the simulator, it also extends F^q to F^* by completing all lines corresponding to P^t. It returns $\delta^* = F^* \setminus F^q$.

DEFINITION: Good Simulator

A simulator is called $(\varepsilon_1, \varepsilon_2)$ good if it satisfies the following:
(i) $\mathscr{L}_t = \mathscr{L}(x_t \mid \tau^{t-1})$ for all t whenever the random permutation is not boundary bad (and the probability that a random permutation is a bad boundary is at most ε_1).
(ii) it defines a simple $\mathscr{L}_t$-completion for all t with probability at least $1 - \varepsilon_2$.
(ii) It samples $s = |F^*| - 2|P^q|$ many points randomly during the updation of the primitive view including the final phase.

Bad Event for the Ideal World. We call bad_1 holds if Π is a bad boundary random permutation at any point of time in the query response process. We call bad_2 holds if the simulator does not make simple completion at any point of time. For a good (ϵ_1, ϵ_2) simulator, we have $\Pr(\mathsf{bad}_1) \leq \epsilon, \Pr(\mathsf{bad}_2) \leq \epsilon_2$. If it is not bad then we have seen that $\mathscr{L}_t = \mathscr{L}(x_t \mid \tau^{t-1})$ for all t. Thus, the simulator finds $\mathscr{L}(x_t \mid \tau^{t-1})$ for all t correctly with probability at least $1 - \epsilon_1$. Moreover, all completions are simple and hence the transcript is good. So a transcript is bad in an ideal world and happens with probability at most $\epsilon_+ \epsilon_2$.

Now, for any good transcript $\tau := (\tau^1, \ldots, \tau^q, \tau^*)$,

$$\Pr_{\mathsf{RW}}(\tau) = (2^{-n|F^*|}, \quad \Pr_{\mathsf{IW}}(\tau) = (2^{-2n})_{|P^*|} \times 2^{-n|S|}$$

where S is the points in which it samples during the updation of the primitive view including the final phase. If the simulator is good then $|S| = |F^*| - 2|P^*|$. Hence, we apply the H-technique and we obtain our main result.

Theorem 5 *If there is a (ϵ_1, ϵ_2) good simulator, then $(\tau^1, \ldots, \tau^q, \tau^*)$ is good with probability $1 - q(q^4\epsilon_1 + \epsilon_2)$.*

We prove this result particularly only for eight rounds at the end of Subsect. 6.4.1. The statement can be proved analogously for r rounds.

6.4.1 Simulator for 8 Rounds

Simulation Closed Line Collection.

1. For a shore 2 F-undefined point u, $\mathsf{L}_1(u) := \mathscr{L}_{(8..2)\nwarrow_{\mathsf{Rev}}}(\tau)_{|u}$ is a collection of (8..2) lines. If $\ell \in \mathsf{L}_1(u)$ then ℓ is a (8..2) line such that $\ell_2 = u$ and $\ell \nwarrow_8 \ell'$ for a boundary wrap-around line. So, if u happens to be a F-defined point then ℓ is extended to a boundary wrap-around line and hence boundary fresh property gets violated. So we must complete ℓ (and also ℓ'). Similarly, for a shore 7 boundary point u, we define $\mathsf{L}_1(u) := \mathscr{L}_{(7..1)\nwarrow_{\mathsf{Rev}}}(\tau)_{|u}$.

2. For a shore 2 or 7 F-undefined point u, $\mathsf{L}_2(u) = \mathscr{L}_{(7..2)}(\tau)_{|u}$, a collection of (7..2) lines. Every $\ell \in \mathsf{L}_2(u)$ is (7..2) line with $u = \ell_2$ or ℓ_7. Hence, if u happens to be F-defined then the boundary fresh property gets violated. This justifies the definition of $\mathsf{L}_2(u)$ that needs to be completed whenever u is completed. We finally define $\mathsf{L}(u) = \mathsf{L}_1(u) \cup \mathsf{L}_2(u)$, a collection of wrap-around lines with one boundary shore 2 or 7.

3. For a revealed boundary point u (at shore 1 or 8), we define $\mathsf{L}(u) = \{\ell(u)\}$, the only wrap-around boundary line passing through ℓ (whenever τ is boundary-fresh). Note that whenever u is completed, the boundary fresh property would be violated, and so $\ell(u)$ must be completed too.

4. For a shore 4 or 5 F-undefined u, let $\mathsf{V}(u) = \textsc{op}_{\mathscr{L}_{4/5}(\tau)_{|u}}(u) \setminus \mathsf{B}_{0/9}$. Note that any wrap-around line $\ell \in \mathscr{L}_{4/5}(\tau)_{|u}$ is a (4..1) (resp. (8..5)) line if u is at shore 4 (resp. 5). Let $\mathsf{L}(u)$ consists of all lines from $\mathscr{L}_{4/5}(\tau)_{|u}$ except that all wrap-around (4..1) (resp. (8..5)) lines are replaced by (4..9) (resp. (5..9)) direct lines. If $\ell \in \mathsf{L}(u)$ and u happens to be completed then the segment ℓ has F-defined points both at shore 4 and 5. So we must complete ℓ to keep (4,5)-complete property.

5. If u is F-undefined and u' is a boundary point, both at shore 2, we define

$$\mathsf{L}(u,u') = \{\ell_{(8..2)} \ : \ \ell'_{(8..2)} \nwarrow_8 \ell, \ \{\ell_2, \ell'_2\} = \{u, u'\}\}.$$

   Let $\mathsf{V}(u,u')$ be the set of all ℓ_8 points. We similarly define $\mathsf{L}(u,u')$ and $\mathsf{V}(u,u')$ for shore 7 points (u,u'). For any pair of distinct points, we define

$$\mathsf{L}(u,u') = \{\ell \in \mathscr{L}_{\text{WA},\mathsf{Gen}\text{-}1} \ : \ \mathsf{Bd}(\ell) = \{u,u'\}\}.$$

6. For all other cases, we define $\mathsf{L}(u) = \mathsf{L}(u,u') = \emptyset$. For all u, we define $\mathsf{V}(u) = \textsc{op}_{\mathsf{L}(u)}(u)$.

Well Approximation of $\mathsf{V}(u)$, $\mathsf{V}(u,u')$, $\mathsf{L}(u)$ and $\mathsf{L}(u,u')$ by Simulator. Let $\tau = (P,F)$ be a good view and let Π be a P-conditioned random permutation. We define $\mathsf{Sim}^{\Pi}_{find}(u \mid F)$ as follows for a boundary point u with shore i and we define $\mathsf{Sim}^{\Pi}_{find}(u,u'|F)$ for boundary points u,u' of shore 2 or 7 as follows:

$\underline{\mathsf{Sim}^{\Pi}_{find}(u \mid F)}$:

$i=1$**:** For all $u_7,u_8 \in \mathsf{DF}$ check whether $\Pi^{-1}(u_8, u_7 \oplus F(u_8)) = (u_0,u)$ for some $u_0 \in \mathsf{B}$. In such a case $\ell(u) = (v_6,v_7,v_8,v_9,v_0,u)$, where $v_6 = v_8 \oplus F(v_7)$ and $v_9 = v_7 \oplus F(v_8)$.

$i=2$**:** Maintain two collection of lines, initialized as $\mathscr{L}_1 \leftarrow \emptyset, \mathscr{L}_2 \leftarrow \emptyset$.

For every $u_1,v_1,v_2 \in \mathsf{DF}$, if $\Pi_1(u \oplus F(u_1),u_1) = \Pi_1(v_2 \oplus F(v_1),v_1) = u_8$, set

$$\mathscr{L}_1 \leftarrow \mathscr{L}_1 \cup (u_8,u_9,u_0,u_1,u),$$

where $u_9 = \Pi_2(u \oplus F(u_1),u_1)$ and $u_0 = u \oplus F(u_1)$.

For every $u_1,u_8 \in \mathsf{DF}$, check if $\Pi(u \oplus F(u_1),u_1) = (u_8,u_9)$ for some $u_9 \in \mathsf{B}$. If so,

$$\mathscr{L}_2 \leftarrow \mathscr{L}_2 \cup (u_7,u_8,u_9,u_0,u_1,u),$$

where $u_7 = u_9 \oplus F(u_8)$, $u_0 = u \oplus F(u_1)$.

Finally, $\mathscr{L} = \mathscr{L}_1 \cup \mathscr{L}_2$.

$i=4$**:** Maintain collection of lines initialized as $\mathscr{L},\mathscr{L}' \leftarrow \emptyset$.

For every $u_5,u_6 \in \mathsf{DF}$, where $u_6 = u \oplus F(u_5)$ and $u_7 := u_5 \cup F(u_6) \notin \mathsf{DF}$,

$$\mathscr{L} \leftarrow \mathscr{L} \cup \{(u,u_5,u_6,u_7)\}, \quad \mathscr{L}' \leftarrow \mathscr{L}' \cup \{(u,u_5,u_6,u_7)\}$$

For every $u_5,u_6,u_7 \in \mathsf{DF}$, where $u_6 = u \oplus F(u_5)$, $u_7 = u_5 \oplus F(u_6)$, and $u_8 := u_6 \cup F(u_7) \notin \mathsf{DF}$,

$$\mathscr{L} \leftarrow \mathscr{L} \cup \{(u,u_5,u_6,u_7,u_8)\}, \quad \mathscr{L}' \leftarrow \mathscr{L}' \cup \{(u,u_5,u_6,u_7,u_8)\}$$

For every $u_5,u_6,u_7,u_8 \in \mathsf{DF}$, where $u_6 = u \oplus F(u_5)$, $u_7 = u_5 \oplus F(u_6)$, $u_8 =$

$6 \oplus F(u_7)$,

$$\mathscr{L} \leftarrow \mathscr{L} \cup \{(u, u_5, u_6, u_7, u_8, u_9, u_0, u_1)\},$$

$$\mathscr{L}' \leftarrow \mathscr{L}' \cup \{(u, u_5, u_6, u_7, u_8, u_9)\}$$

where $u_9 = u_7 \cup F(u_8)$ and $(u_0, u_1) = \Pi^{-1}(u_8, u_9)$. [a]

$i = 5, 7, 8$**:** Symmetric to $i = 4, 2, 1$, respectively.

If $i \neq 4, 5$, return $\mathsf{Sim}^{\Pi}_{findV}(u|F) := \text{OP}_{\mathscr{L}}(u)$ and $\mathsf{Sim}^{\Pi}_{findL}(u|F) := \mathscr{L}$.
If $i = 4, 5$, return $\text{OP}_{\mathscr{L}}(u)$ and $\mathscr{L}'$.

$\underline{\mathsf{Sim}^{\Pi}_{find}(u, u'|F)}$ at shore i.

Maintain a collection of lines $\mathscr{L}$ and points V initialized as $\mathscr{L}, V \leftarrow \emptyset$.

$i = 2$ For every $v_1, v_1' \in \mathsf{DF}$,

$$\Pi_1(u \oplus F(v_1), v_1) = v_8 := v_8' := \Pi_1(u' \oplus F(v_1'), v_1')$$

$$\Rightarrow \mathscr{L} \leftarrow \mathscr{L} \cup \{(v_8, v_9, v_0, v_1, u), (v_8, v_9', v_0', v_1', u')\}, V \leftarrow V \cup \{v_8\}$$

where $v_0 = u \oplus F(v_1)$, $v_0 = u' \oplus F(v_1')$, $(v_8, v_9) = \Pi(v_0, v_1)$ and $(v_8, v_9') = \Pi(v_0', v_1')$.

$i = 7$ Symmetric definition for shore 7.

Return $\mathsf{Sim}^{\Pi}_{findV}(u, u'|F) := V$, $\mathsf{Sim}^{\Pi}_{findL}(u, u'|F) := \mathscr{L}$.

[a] Note that if $(u_8, u_9) \notin \text{Ran}(P)$ then $(u, u_5, \ldots, u_9) \in \mathsf{Ext}(u)$, otherwise $(u, u_5, \ldots, u_1) \in \mathsf{Ext}(u)$. However, the simulator Sim has no way to know whether $(u_8, u_9) \in \text{Ran}(P)$ or not.

Lemma 14 (Correctness of Sim at Each Step) *Let $\tau = (P, F)$ be a good view and u is a F-undefined point. Let $\mathscr{L} := \mathscr{L}_\tau(u)$ and Suppose a P-conditioned Π is not boundary bad w.r.t. (τ, u). Then, $(\mathsf{Sim}^{\Pi}_{findV}, \mathsf{Sim}^{\Pi}_{findL})$ is a well-approximation of (V, L).*

Moreover we have that,

$$\Pr[P\text{-conditioned } \Pi \text{ is not boundary bad w.r.t. } (\tau, u)] \geq 1 - \mathcal{O}(q^8/2^n)$$

The proof is given in Supplementary Material B.1.

Description of Simulation Closed Completion. Let τ be a good view. Let $\mathscr{L}$ be a simulation closed collection of lines and let

$$\textsc{samp}(\mathscr{L}) = \{(\ell, k) \ : \ \ell_{(i..j)} \in \mathscr{L},\ k \in [0..9] \setminus \big(\{i-1, j+1\} \cup \textsc{int}(\ell)\big)\}.$$

Every direct line $\ell \in \mathscr{L}$ either passes through $u \in \mathsf{DF}_4$ (ℓ is called forward) or $u \in \mathsf{DF}_5$ (ℓ is called backward) but not both. Let $\textsc{cons\_samp}(\mathscr{L})$ be the set of all pairs $(\ell, k) \in \textsc{samp}(\mathscr{L})$ such that $k \in \{8, 9\}$ for a forward line ℓ or $k \in \{0, 1\}$ for a backward line ℓ. Let $\textsc{sim\_samp}(\mathscr{L}) = \textsc{samp}(\mathscr{L}) \setminus \textsc{cons\_samp}(\mathscr{L})$. Let $\mathscr{L}_1 \subseteq \mathscr{L}(\tau) \setminus \mathscr{L}$ be all lines sharing a common at least one common boundary point with a given collection of lines $\mathscr{L}$. Let $\mathscr{L}_2$ be all other lines (no common boundary points). We define

$$\begin{aligned}
\textsc{set}(\mathscr{L}) &:= \{(i-1, \ell), (j+1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}\} \\
\textsc{set}_{1-}(\mathscr{L}) &:= \{(i-1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_i \in \mathsf{Bd}(\mathscr{L})\} \\
\textsc{set}_{1+}(\mathscr{L}) &:= \{(j+1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_j \in \mathsf{Bd}(\mathscr{L})\}.
\end{aligned}$$

We define $\textsc{set}_1(\mathscr{L}) = \textsc{set}_{1-}(\mathscr{L}) \cup \textsc{set}_{1+}(\mathscr{L})$. So, for all $\ell \in \mathscr{L}$, (ℓ, k) is either from $\textsc{set}(\mathscr{L})$ or from $\textsc{samp}(\mathscr{L})$ or $\ell_k \in \mathsf{DF}$. So after completion, ℓ_k is F'-defined for all $k \in [8]$. We now sample elements from B randomly for each element $\textsc{sim\_samp}(\mathscr{L})$:

1. $\hat{u} \leftarrow\!\$\ \mathsf{B}, \forall u \in \mathsf{Bd}(\mathscr{L})$.
2. $\ell_k \leftarrow\!\$\ \mathsf{B}, \forall (\ell, k) \in \textsc{sim\_samp}(\mathscr{L}) \setminus \mathsf{Bd}(\mathscr{L})$.
3. For a forward ℓ, define $(\ell_8, \ell_9) \leftarrow \Pi(\ell_0, \ell_1)$. Similarly, for a backward ℓ, define $(\ell_0, \ell_1) \leftarrow \Pi^{-1}(\ell_8, \ell_9)$.

For all $(\ell, i-1) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})$, we define $\ell_{i-1} = \hat{\ell}_i \oplus \ell_{i+1}$ and $\ell_{j+1} = \hat{\ell}_j \oplus \ell_{j-1}$. We define $F' = F \cup \{(\ell_i, \ell_{i-1} \oplus \ell_{i+1}), \forall i \in [8], \ell \in \mathscr{L}\}$. This defines $\mathscr{L}$-completion.

Definition: Bad Sampling.

We call above sampling GOOD (otherwise, we call BAD) if the following holds:

good_1: All ℓ_k for $(\ell, k) \in \textsc{samp}(\mathscr{L}) \setminus \mathsf{Bd}(\mathscr{L})$ are distinct and different from $\textsc{val}(\tau)$.

good_2: All ℓ_k for $(\ell, k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})$ are distinct and different from $\textsc{val}(\tau)$.

good_3: For all $(\ell, k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})$, $(\ell', m) \in \textsc{samp}(\mathscr{L})$, $\ell_k \neq \ell'_m$.

good_4: If $\ell_{(3..6)}$ is a new segment then both $\ell_3, \ell_6 \notin \mathsf{DF}'$.

We define the bad events corresponding to the sampling procedure as the complements of the good events defined above:

$$\mathsf{bad}_i = \mathsf{good}_i^c, i = 1, 2, 3, 4$$

Lemma 15 *If the sampling defined above is* GOOD *then the completion is a simple extension. Similarly, if P-conditioned random permutation Π is boundary good w.r.t. $(\tau, (x, y, \pm))$ then a construction query completion is simple.*

Proof Sketch. This can be very easily seen from the fact that, good_1 implies S1, $\mathsf{good}_1 \wedge \mathsf{good}_2 \wedge \mathsf{good}_3$ implies S2. S3 is vacuously true since there are no two-step lines in case of eight rounds, as τ is $(4, 5)$-complete. Finally good_4 implies S5. □

(4,5)-Completion. Let $\tau = (P, F)$ be a semi-complete view. Let $\mathscr{L} := \mathscr{L}_{(3..6)}$ be the collection of all (3..6) lines and $\textsc{samp}(\mathscr{L}) = \mathscr{L} \times ([0..9] \setminus \{2, 4, 5, 7\})$. Let $\textsc{cons\_samp}(\mathscr{L}) = \mathscr{L} \times \{8, 9\}$ and $\textsc{sim\_samp}(\mathscr{L}) = \textsc{samp}(\mathscr{L}) \setminus \textsc{cons\_samp}(\mathscr{L}) = \mathscr{L} \times \{0, 1, 3, 6\}$. Let $\mathscr{L}_1 \subseteq \mathscr{L}(\tau) \setminus \mathscr{L}$ be all lines sharing at least one common boundary point (equivalently, exactly one common boundary point) with $\mathscr{L}$. Let $\mathscr{L}_2$ be all other lines (no common boundary points). We define

$$\begin{aligned}
\textsc{set}(\mathscr{L}) &:= \{(i-1, \ell), (j+1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}\} \\
\textsc{set}_{1-}(\mathscr{L}) &:= \{(i-1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_i \in \mathsf{Bd}(\mathscr{L})\} \\
\textsc{set}_{1+}(\mathscr{L}) &:= \{(j+1, \ell) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_j \in \mathsf{Bd}(\mathscr{L})\}.
\end{aligned}$$

We define $\textsc{set}_1(\mathscr{L}) = \textsc{set}_{1-}(\mathscr{L}) \cup \textsc{set}_{1+}(\mathscr{L})$. We now sample elements from B randomly for each element $\textsc{sim\_samp}(\mathscr{L})$:

1. $\hat{u} \leftarrow\$ \mathsf{B}, \forall u \in \mathsf{Bd}(\mathscr{L})$.
2. $\ell_k \leftarrow\$ \mathsf{B}, \forall (\ell, k) \in \textsc{sim\_samp}(\mathscr{L}) \setminus \mathsf{Bd}(\mathscr{L})$.
3. For all ℓ, define $(\ell_8, \ell_9) \leftarrow \Pi(\ell_0, \ell_1)$.

For all $(i-1, \ell) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})$, we define $\ell_{i-1} = \hat{\ell}_i \oplus \ell_{i+1}$ and $\ell_{j+1} = \hat{\ell}_j \oplus \ell_{j-1}$. We define $F' = F \cup \{(\ell_i, \ell_{i-1} \oplus \ell_{i+1}), \forall i \in [8], \ell \in \mathscr{L}\}$. This defines $\mathscr{L}$-completion.

DEFINITION: Bad Sampling for (4,5)-Completion

We call the above sampling GOOD (Otherwise, called BAD) if the following conditions hold:

good_1: All ℓ_k for $(\ell, k) \in \text{SIM\_SAMP}(\mathscr{L}) \setminus \mathsf{Bd}(\ell)$ are distinct and different from $\text{VAL}(\tau)$

good_2: All ℓ_k for $(\ell, k) \in \text{SET}(\mathscr{L}) \cup \text{SET}_1(\mathscr{L})$ are distinct and different from $\text{VAL}(\tau)$

We define the bad events corresponding to the sampling procedure as the complements of the good events defined above:

$$\mathsf{bad}_i = \mathsf{good}_i^c, i = 1, 2$$

Lemma 16 *If the sampling defined above for (4,5)-completion is GOOD, then the completion is a simple extension.*

Here, as before, S1 and S2 follows from $\mathsf{good}_1 \wedge \mathsf{good}_2$.

Lemma 17 $\Pr(\textit{sampling is BAD}) \leq \mathcal{O}(q^6/2^n)$.

The proof is given in Supplementary Material B.2.

Proof of Theorem 5 for 8 rounds. From Lemma 14 we have that whenever P-conditioned random permutation Π is not boundary bad $(\mathsf{Sim}_{findL}, \mathsf{Sim}_{findV})$ is a well-approximation of (L, V) Now since Sim_{findL} and Sim_{findV} find at least one new line and at least one new vertex every time it is run, it can be executed at most q^4 times until it finds the minimal fixed point set. The collection of lines obtained in each such execution is a well-approximation of the corresponding execution of L, implying by Lemma 11, that $\mathsf{Sim}_{findL*}(x_t)$ is an approximation of $\mathsf{L}_*(x_t)$. Since $\mathsf{L}_*(x_t)$ is a valid collection of lines, so is its approximation $\mathsf{Sim}_{findL*}(x_t)$. Now by Lemma 15, for good sampling the $\mathsf{Sim}_{findL*}(x_t)$-extension is a simple extension. Now, by Lemma 13 we have that the simple $\mathsf{Sim}_{findL*}(x_t|\tau_{t-1})$-extension τ' is semi-good. Now τ_t will be the $(4, 5)$-completion of τ', which is good by Lemma 13 if the sampling is good, since then the $(4, 5)$-completion is a simple extension by Lemma 16. □

6.4.2 Simulator for 7 Rounds

Forced-Intersection and Mixed Forced-Intersection Lines.

1. Let $\mathfrak{L}_{\mathsf{FI}}$ be the collection of all size 4 sets of lines:

   - $\{\ell_{(6/7..1)}, \ell'_{(3..1)}, \ell''_{(3..1)}, \ell'''_{(6/7..1)}\}$ such that $\ell \leftthreetimes_1 \ell' \leftthreetimes_3 \ell'' \leftthreetimes_1 \ell'''$ and $\ell_0 \oplus \ell'_0 \oplus \ell''_0 \oplus \ell'''_0 \oplus \ell'_4 \oplus \ell''_4 = 0^n$. Moreover, if ℓ (resp., ℓ''') is a (7..1) line, then ℓ_7 (resp., ℓ'''_7) is a revealed boundary point.
   - We similarly define for the other side boundary point. $\{\ell_{(7..2/1)}, \ell'_{(7..5)}, \ell''_{(7..5)}, \ell'''_{(7..2/1)}\}$ such that $\ell \leftthreetimes_7 \ell' \leftthreetimes_5 \ell'' \leftthreetimes_7 \ell'''$ and $\ell_8 \oplus \ell'_8 \oplus \ell''_8 \oplus \ell'''_8 \oplus \ell'_4 \oplus \ell''_4 = 0^n$. Moroever, if ℓ (resp., ℓ''') is a (7..1) line then ℓ_1 (resp., ℓ'''_1) is a revealed boundary point.

   and the following size 8 sets:

   - $\{\ell^{\dagger}_{(7..1/2)}, \ell^{\dagger\dagger}_{(7..5)}, \ell^{\dagger\dagger\dagger}_{(7..5)}, \ell_{(7..1)}, \ell'_{(7..1)}, \ell^{\ddagger\ddagger\ddagger}_{(7..5)}, \ell^{\ddagger\ddagger}_{(7..5)}, \ell^{\ddagger}_{(7..1/2)}\}$ such that $\ell^{\dagger} \leftthreetimes_7 \ell^{\dagger\dagger} \leftthreetimes_5 \ell^{\dagger\dagger\dagger} \leftthreetimes_7 \ell \leftthreetimes_1 \ell' \leftthreetimes_7 \ell^{\ddagger\ddagger\ddagger} \leftthreetimes_5 \ell^{\ddagger\ddagger} \leftthreetimes_7 \ell^{\ddagger}$ and $\ell_8 \oplus \ell^{\dagger}_8 \oplus \ell^{\dagger\dagger}_8 \oplus \ell^{\dagger\dagger\dagger}_8 \oplus \ell^{\dagger\dagger}_4 \oplus \ell^{\dagger\dagger\dagger}_4 = \ell'_8 \oplus \ell^{\ddagger}_8 \oplus \ell^{\ddagger\ddagger}_8 \oplus \ell^{\ddagger\ddagger\ddagger}_8 \oplus \ell^{\ddagger\ddagger}_4 \oplus \ell^{\ddagger\ddagger\ddagger}_4 = 0^n$. Moreover, if $\ell^{\dagger}$ (resp., $\ell^{\ddagger}$) is a (7..1) line then $\ell^{\dagger} \in \mathscr{L}_{(7..1)\leftthreetimes_{\mathsf{Rev}(1)}}$ (resp., $\ell^{\ddagger} \in \mathscr{L}_{(7..1)\leftthreetimes_{\mathsf{Rev}(1)}}$).
   - $\{\ell^{\dagger}_{(6/7..1)}, \ell^{\dagger\dagger}_{(3..1)}, \ell^{\dagger\dagger\dagger}_{(3..1)}, \ell_{(7..1)}, \ell'_{(7..1)}, \ell^{\ddagger\ddagger\ddagger}_{(3..1)}, \ell^{\ddagger\ddagger}_{(3..1)}, \ell^{\ddagger}_{(6/7..1)}\}$ such that $\ell^{\dagger} \leftthreetimes_1 \ell^{\dagger\dagger} \leftthreetimes_3 \ell^{\dagger\dagger\dagger} \leftthreetimes_1 \ell \leftthreetimes_1 \ell' \leftthreetimes_7 \ell^{\ddagger\ddagger\ddagger} \leftthreetimes_5 \ell^{\ddagger\ddagger} \leftthreetimes_7 \ell^{\ddagger}$ and $\ell_0 \oplus \ell^{\dagger}_0 \oplus \ell^{\dagger\dagger}_0 \oplus \ell^{\dagger\dagger\dagger}_0 \oplus \ell^{\dagger\dagger}_4 \oplus \ell^{\dagger\dagger\dagger}_4 = \ell'_0 \oplus \ell^{\ddagger}_0 \oplus \ell^{\ddagger\ddagger}_0 \oplus \ell^{\ddagger\ddagger\ddagger}_0 \oplus \ell^{\ddagger\ddagger}_4 \oplus \ell^{\ddagger\ddagger\ddagger}_4 = 0^n$. Moreover, if $\ell^{\dagger}$ (resp., $\ell^{\ddagger}$) is a (7..1) line then $\ell^{\dagger} \in \mathscr{L}_{(7..1)\leftthreetimes_{\mathsf{Rev}(1)}}$ (resp., $\ell^{\ddagger} \in \mathscr{L}_{(7..1)\leftthreetimes_{\mathsf{Rev}(7)}}$).

   Let $\mathscr{L}_{\mathsf{FI}} := \bigcup_{L \in \mathfrak{L}_{\mathsf{FI}}} L$.

2. Let $\mathfrak{L}_{\mathsf{MFI}}$ be the collection of all size 3 sets of lines:

   - $\{\ell_{(6..1)}, \ell'_{(3..1)}, \ell''_{(3..i)}\},\ i \in \{6, \ldots, 9, 1\}$ such that $\ell \leftthreetimes_1 \ell' \leftthreetimes_3 \ell''$ and $\ell_0 \oplus \ell'_0 = \ell'_4 \oplus \ell''_4$.
   - $\{\ell_{(7..2)}, \ell'_{(7..5)}, \ell''_{(j..5)}\},\ j \in \{1, 2, 3, 7\}$ such that $\ell \leftthreetimes_7 \ell' \leftthreetimes_5 \ell''$ and $\ell_7 \oplus \ell'_7 = \ell'_4 \oplus \ell''_4$.

   Let $\mathscr{L}_{\mathsf{MFI}} := \bigcup_{L \in \mathfrak{L}_{\mathsf{MFI}}} L$.

A line $\ell_{(6/7..1)}$ (resp., $\ell_{(7..1/2)}$) is said to have a forced-intersection/mixed forced-intersection structure at shore 1 (resp., 7) if there exist lines $\ell'_{(3..1)}, \ell''_{(3..1)}, \ell'''_{(6/7..1)}$ ($\ell'_{(7..5)}, \ell''_{(7..5)}, \ell'''_{(7..1/2)}$) such that $\ell_0 \oplus \ell'''_0 = \ell'_0 \oplus \ell''_0 \oplus \ell'_4 \oplus \ell''_4$ (resp., $\ell_7 \oplus \ell'''_7 = \ell'_7 \oplus \ell''_7 \oplus \ell'_4 \oplus \ell''_4$)

Simulation Closed Line Collection.

1. For a shore 2 or 6 F-undefined point u, we define $\mathsf{L}(u) := \mathsf{L}_1(u) \cup \mathsf{L}_2(u)$, where $\mathsf{L}_i(u)$ are defined as follows:

   (i) $\mathsf{L}_1(u) = \mathscr{L}_{(6..2)}(\tau)_{|u}$, a collection of (6..2) lines.

   (ii) If u is a shore 2 point, $\mathsf{L}_2(u) := \mathscr{L}_{(7..2)\nwarrow_{\mathsf{Rev}}}(\tau)_{|u}$, a collection of (7..2) lines incident on revealed boundary points. Similarly, for a shore 6 boundary point u, $\mathsf{L}_2(u) := \mathscr{L}_{(6..1)\nwarrow_{\mathsf{Rev}}}(\tau)_{|u}$.

2. For a revealed boundary point u (at shore 1 or 7), we define $\mathsf{L}(u) = \{\ell(u)\}$ is a collection wrap-around boundary lines. For any other F-undefined shore 1 point u, $\mathsf{L}(u) = \emptyset$.

3. For a shore 3 or 5 F-undefined u, let $\mathsf{L}(u) = \mathscr{L}_{34/45}(\tau)_{|u}$.

4. For F-undefined boundary points, u in shore 6 and v in a shore 1, let $\mathfrak{L}_{\mathsf{FI}}(u,v) = \{L \in \mathfrak{L}_{\mathsf{FI}} : \ell_{(6..1)} \in L, \ell_6 = u, \ell_1 = v\}$, and define

$$\mathsf{L}(u,v) = \bigcup_{L \in \mathfrak{L}(u,v)} L$$

   We similarly define $\mathsf{L}_{\mathsf{FI}}(u,v)$ for a pair of F-undefined points (u,v), where u is in shore 7 and v is in shore 2.

5. For F-undefined boundary points, u in shore 7 and v in a shore 1, let $\mathfrak{L}_{\mathsf{FI}}(u,v) = \{L \in \mathfrak{L}_{\mathsf{FI}} : \ell_{(7..1)} \in L, \ell_7 = u, \ell_1 = v\}$, and define

$$\mathsf{L}(u,v) = \bigcup_{L \in \mathfrak{L}_{\mathsf{FI}}(u,v)} L$$

6. If u is F-undefined and u' a boundary point, both at shore 2, we define

$$\mathsf{L}_1(u,u') = \{\ell_{(7..2)} \ : \ \ell'_{(7..2)} \nwarrow_7 \ell, \ \{\ell_2, \ell'_2\} = \{u, u'\}\}.$$

   We also define $\mathfrak{L}_{\mathsf{FI}}(u,u') := \{L \in \mathfrak{L}_{\mathsf{FI}} : L = \{\ell^{\dagger}_{(7..2)}, \ell^{\dagger\dagger}_{(7..5)}, \ell^{\dagger\dagger\dagger}_{(7..5)}, \ell_{(7..1)}, \ell'_{(7..1)}, \ell^{\ddagger\ddagger\ddagger}_{(3..1)}, \ell^{\ddagger\ddagger}_{(7..5)}, \ell^{\ddagger}_{(7..2)}\}, \ell^{\dagger}_2 = u, \ell^{\ddagger}_2 = u'\}$ Then

$$\mathsf{L}_2(u,u') = \bigcup_{L \in \mathfrak{L}_{\mathsf{FI}}(u,u')} L$$

   Finally, define $\mathsf{L}(u,u') = \mathsf{L}_1(u,u') \cup \mathsf{L}_2(u,u')$.

We similarly define $\mathsf{L}(u,u')$ and $\mathsf{V}(u,u')$ for shore 6 points (u,u').

7. For F-undefined pair of points $(u,u') \in \mathsf{Rev}_7(\tau)^{[2]} \setminus \mathsf{Bd}_6(\mathscr{L}(\tau))^{[2]}$, we define, $\mathfrak{L}_{\mathsf{FI}}(u,u') := \big\{L \in \mathfrak{L}_{\mathsf{FI}} : L = \{\ell^{\dagger}_{(7..1)}, \ell^{\dagger\dagger}_{(3..1)}, \ell^{\dagger\dagger\dagger}_{(3..1)}, \ell_{(7..1)}, \ell'_{(7..1)}, \ell^{\ddagger\ddagger\ddagger}_{(3..1)}, \ell^{\ddagger\ddagger}_{(3..1)}, \ell^{\ddagger}_{(6/7..1)}\}, \ell^{\dagger}_{6/7} = u, \ell^{\ddagger}_{6/7} = u'\big\}$. Then

$$\mathsf{L}(u,u') = \bigcup_{L \in \mathfrak{L}_{\mathsf{FI}}(u,u')} L$$

We define $\mathsf{L}(u,u')$ analogously for $(u,u') \in \mathsf{Rev}_1(\tau)^{[2]} \setminus \mathsf{Bd}_2(\mathscr{L}(\tau))^{[2]}$.

8. For an F-undefined point u in shore 2 and a boundary point u' in shore 3, we define

$$\mathsf{L}_{\mathsf{MFI}}(u,u') = \{\ell'_{(3..5)} \ : \ \ell_{(7..2)} \curvearrowleft_7 k_{(7..5)} \curvearrowleft_5 \ell', \ \{\ell_2, \ell'_3\} = \{u,u'\}\}.$$

We similarly define $\mathsf{L}_{\mathsf{MFI}}(u,u')$ for a pair of points (u,u'), where u is a shore 6 point and u' is a shore 5 point. (Similarly defined when u' is a shore 1/2 point)

9. For any pair of distinct points we define $\mathsf{L}(u,u') = \{\ell \in \mathscr{L}_{\mathrm{WA},\mathsf{Gen\text{-}1}} \ : \ \mathsf{Bd}(\ell) = \{u,u'\}\}$.

10. For any pair of distinct boundary points u,u' from shores 3 and 5 respectively, we define $\mathsf{L}(u,u') = \{\ell \in \mathscr{L}_{\mathsf{dir},\mathsf{Gen\text{-}1}} \ : \ \mathsf{Bd}(\ell) = \{u,u'\}\}$

For all other cases, we define $\mathsf{L}(u) = \mathsf{L}(u,u') = \emptyset$. For all u, we define $\mathsf{V}(u) = \mathrm{OP}_{\mathsf{L}(u)}(u)$

Computation of $\mathsf{V}(u)$, $\mathsf{V}(u,u')$, $\mathsf{L}(u)$ and $\mathsf{L}(u,u')$ by Simulator. Let $\tau = (P,F)$ be a good view. We define $\mathsf{Sim}^{\Pi}_{find}(u \mid F)$ as follows for a boundary point u with shore i:

It is important to note that for an input u from shore i, other than the forced-intersection/mixed forced-intersection cases, computation of $\mathsf{V}(u), \mathsf{L}(u)$ for 7 rounds is analogous to their computation for 8 rounds. These include:

1. For $i = 1$, finding the unique $\mathsf{Gen\text{-}2+}$ line with boundary point u
2. For $i = 2$,
   - Finding all pairs of lines $\ell_{(7,2)} \curvearrowleft_7 \ell'_{(7..i)}, \ i \in [3,5]$ such that $\ell_2 = u$
   - Finding all lines $\ell_{(6..2)}$ such that $\ell_2 = u$
3. For $i = 3$, finding all direct lines $\ell_{3..j}, \ j \in [7,8]$ such that $\ell_3 = u$
4. Symmetrically, for $i = 5, 6, 7$

It is easy to see that these lines can be identified by the Simulator in much the same way with minor adjustment to the shore numbers, the details for which are omitted. We now describe the computation of $\mathsf{L}(u,u'),\mathsf{V}(u,u')$ for a pair of inputs u,u' from shores i,j respectively. Like in case of the single input, (a part of) the computation of $\mathsf{V}(u,u'),\mathsf{L}(u,u')$ is analogous to 8 rounds, namely, for $i=j=2$ (resp., $i=j=6$), finding Gen-1 lines $\ell_{(7..2)} \curlywedge_7 \ell'_{(7..2)}$ such that $\ell_2 = u, \ell'_2 = u'$ (resp., all Gen-1 lines $\ell_{(6..1)} \curlywedge_1 \ell'_{(6..1)}$ such that $\ell_6 = u, \ell'_6 = u'$), the details for which are omitted. We now describe the additional computation steps for V,L for a pair of inputs. We will assume that any Gen-2+ direct or wrap-around line is already in possession of the simulator, hence no computation is required on that end and the information available to the simulator can be readily used.

$\underline{\mathsf{Sim}^{\Pi}_{find}(u,u'|F)}$ at shore i.

Maintain a collection of lines $\mathscr{L}$ and points V initialized as $\mathscr{L}, V \leftarrow \emptyset$.

$i=2, j=7$**.** For every $v_1 \in \mathsf{DF}$, check if there exist lines $\ell_{(7..5)} \curlywedge_5 \ell'_{(7..5)}$ such that

Case 1:

- $\Pi_1(F(v_1)\oplus u, v_1) = \ell_7$, and
- $\Pi_2^{-1}(\ell'_7, v'_8) \in \mathsf{DF}$, where $v_8 = \ell_8 \oplus \ell'_8 \oplus \Pi_2(F(v_1)\oplus u, v_1) \oplus \ell_4 \oplus \ell'_4$

If found, then $\mathscr{L} \leftarrow \mathscr{L} \cup \{(v_7, v_8, v_0, v_1, u), (v'_7, v'_8, v'_0, v'_1, u')\}$, $V \leftarrow V \cup \{v'_2, v_7, v'_7\}$, where $v_0 = F(v_1)\oplus u, (v_7, v_8) = \Pi(v_0, v_1), v'_7 = \ell'_7, (v'_0, v'_1) = \Pi^{-1}(v'_7, v'_8), v'_2 = v'_0 \oplus F(v'_1)$

Case 2:

- $\Pi_1(F(v_1)\oplus u, v_1) = \ell_7$, and
- $\Pi_2^{-1}(\ell'_7, v'_8) = \ell''_1$ for some Gen-2+ wrap-around line $\ell''_{(i..1)}, i \in [3,5]$, where $v'_8 = \ell_8 \oplus \ell'_8 \oplus \ell_4 \oplus \ell'_4 \oplus \Pi_2(F(v_1)\oplus u, v_1)$

If found, then $\mathscr{L} \leftarrow \mathscr{L} \cup \{(v_7, v_8, v_0, v_1, u), (v'_7, v'_8, v'_0, u')\}$, $V \leftarrow V \cup \{v'_1, v_7, v'_7\}$, where $v_0 = F(v_1)\oplus u, (v_7, v_8) = \Pi(v_0, v_1), v'_7 = \ell'_7, (v'_0, v'_1) = \Pi^{-1}(v'_7, v'_8)$

$i=6, j=1$**.** Identical to the above case

$i = j = 2$. For every $v_1, v_1' \in \mathsf{DF}$, check if there exist lines $\ell_{(7..5)} \nearrow_5 \ell'_{(7..5)}, k_{(7..5)} \nearrow_5 k'_{(7..5)}$ such that

- $\Pi_1(F(v_1) \oplus u, v_1) = \ell_7$, $\Pi_1(F(v_1') \oplus u', v_1') = k_7$
- $\Pi_2^{-1}(\ell_7', u_8') = \Pi_2^{-1}(k_7', v_8')$, where $u_8' = \ell_8 \oplus \ell_8' \oplus \ell_4 \oplus \ell_4' \oplus \Pi_2(F(v_1) \oplus u, v_1)$, and $v_8' = k_8 \oplus k_8' \oplus k_4 \oplus k_4' \oplus \Pi_2(F(v_1') \oplus u', v_1')$

If found, then $\mathscr{L} \leftarrow \mathscr{L} \cup \{(v_7, v_8, v_0, v_1, u), (v_7', v_8', v_0', u'), (w_7, w_8, w_0, w_1), (w_7', w_8', w_0', w_1)\}$, $V \leftarrow V \cup \{v_7, v_7', w_1, w_7, w_7'\}$, where $v_0 = F(v_1) \oplus u, (v_7, v_8) = \Pi(v_0, v_1), v_0' = F(v_1') \oplus u', (v_7', v_8') = \Pi(v_0', v_1'), w_7 = \ell_7', w_7' = k_7', (w_0, w_1) = \Pi^{-1}(\ell_7', u_8'), (w_0', w_1') = \Pi^{-1}(k_7', v_8')$

$i = j = 6$. Identical to the above case.

$i = 2, j \in \{1, 2, 3, 7\}$ For every $v_1 \in \mathsf{DF}$, check if there exist lines $\ell_{(7..5)} \nearrow_5 \ell'_{(j..5)}$ such that

- $\Pi_1(F(v_1) \oplus u) = \ell_7$, and
- $\ell_8 \oplus v_8 = \ell_4 \oplus \ell_4'$, where $v_8 = \Pi_2(F(v_1) \oplus u)$

If found, then $\mathscr{L} \leftarrow \mathscr{L} \cup \{(v_7, v_8, v_0, v_1, u), (v_j', \ldots, u')\}$, $V \leftarrow V \cup \{v_7\}$, where $(v_7, v_8) = \Pi(F(v_1) \oplus u), v_j' = \ell'_{op(u')}$

$i = 6, j \in \{1, 5, 6, 7\}$. Identical to the above case.

Return $\mathsf{Sim}^{\Pi}_{findV}(u, u'|F) := V$, $\mathsf{Sim}^{\Pi}_{findL}(u, u'|F) := \mathscr{L}$.

Lemma 18 (Correctness of Sim at Each Step) *Let $\tau = (P, F)$ be a good view and u is a F-undefined point. Let $\mathscr{L} := \mathscr{L}_\tau(u)$ and Suppose a P-conditioned Π is not boundary bad w.r.t. (τ, u). Then, we have $(\mathsf{Sim}^{\Pi}_{findV}, \mathsf{Sim}^{\Pi}_{findL})$ is a well-approximation of (V, L).*

Moroever, we have

$$\Pr[P\text{-conditioned } \Pi \text{ is not boundary bad w.r.t. } (\tau, u)] \geq 1 - \varepsilon_1$$

where $\varepsilon_1 = \mathcal{O}(q^8/2^n)$.

The proof is given in Supplementary Material B.3.

Description of Simulation Closed Completion. Let τ be a good view and let $\mathscr{L}$ be a simulation closed collection of lines. Every direct line $\ell \in \mathscr{L}$ either passes through $(u_3, u_4) \in \mathsf{DF}_3 \times \mathsf{DF}_4$ (ℓ is called forward) or $(u_4, u_5) \in \mathsf{DF}_4 \times \mathsf{DF}_5$ (ℓ is called backward) but not both. Let $\textsc{cons\_samp}(\mathscr{L})$ be the set of all pairs $(\ell, k) \in \textsc{samp}(\mathscr{L})$ such that $k \in \{7, 8\}$ for a forward line ℓ or $k \in \{0, 1\}$ for a backward line ℓ. We describe the sampling/setting function for different categories of lines in $\mathscr{L}$ below.

We defer the detailed description of the completion to Supplementary Material B.5.

We now sample elements from B randomly for each element $\textsc{sim\_samp}(\mathscr{L})$:

1. $\hat{u} \leftarrow\!\$\ \mathsf{B}, \forall u \in \mathsf{Bd}(\mathscr{L}) \cup \textsc{setsamp}(\mathscr{L})$.
2. $\ell_k \leftarrow\!\$\ \mathsf{B}, \forall (\ell, k) \in \textsc{sim\_samp}(\mathscr{L}) \setminus \mathsf{Bd}(\mathscr{L})$.
3. For a forward ℓ, define $(\ell_7, \ell_8) \leftarrow \Pi(\ell_0, \ell_1)$. Similarly, for a backward ℓ, define $(\ell_0, \ell_1) \leftarrow \Pi^{-1}(\ell_7, \ell_8)$.

This concludes our sampling definition that our simulator employs for any $\mathscr{L}$-completion.

Definition: Bad Sampling

We call the above sampling *good* (Otherwise called *bad*) if the following conditions hold:

good_1: All ℓ_k for $(\ell, k) \in \textsc{samp}(\mathscr{L}) \setminus \mathsf{Bd}(\mathscr{L})$ are distinct and different from $\textsc{val}(\tau) \cup \textsc{setsamp}(\mathscr{L})$.

good_2: All ℓ_k for $(\ell, k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_{1/2}(\mathscr{L})$ are different from $\textsc{val}(\tau) \cup \textsc{samp}(\mathscr{L}) \cup \textsc{setsamp}(\mathscr{L})$. Further, if $\ell_k = \ell'_k$ for some $(\ell, k) \neq (\ell', k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_{1/2}(\mathscr{L})$, then either $(\ell, \cdot, \ell') \in \mathfrak{L}_{\mathsf{MFI}}$ or $(\ell, \cdot, \cdot, \ell') \in \mathfrak{L}_{\mathsf{FI}}$.

good_3: If $(\ell, k) = (\ell', k) \in \textsc{setsamp}(\mathscr{L})$, then $(\ell, \cdot, \cdot, \ell') \in \mathfrak{L}_{\mathsf{FI}}$.

good_4: No $\ell_{(2..6)}$ segment exists in $\mathscr{L}(\tau')$.

We define the bad events corresponding to the sampling procedure as the complements of the good events defined above:

$$\mathsf{bad}_i = \mathsf{good}_i^c, i = 1, 2, 3, 4$$

Lemma 19 *If the sampling defined above is* GOOD *then the completion is a simple extension.*

Similarly, if P-conditioned Π is boundary good w.r.t. $(\tau, (x, y, \pm))$ then a construction query completion is simple.

Proof Sketch. This is also quite straightforward, noting that good_1 implies S1, $\mathsf{good}_1 \wedge \mathsf{good}_2$ implies S2 and S3. Finally good_4 implies S4. □

Lemma 20 $\Pr(\bigvee_{i\in[4]} \mathsf{bad}_i) \leq \varepsilon$, *where* $\varepsilon = \mathcal{O}(q^8/2^n)$.

The proof is given in Supplementary Material B.4.

Chapter 7

Conclusion

The notion of Indifferentiability is a foundational concept in modern cryptography, and serves as a powerful framework to evaluate whether a given construction can securely emulate an ideal primitive, such as a random oracle or an ideal permutation. This thesis has undertaken a comprehensive study of the indifferentiability properties of various symmetric key cryptographic constructions, yielding both positive and negative results. For the positive results, the thesis describes Simulators which succeed against the best polynomially-bounded adversaries for each specific construction. For the negative results, the thesis demonstrates cryptographic attacks that succeed with overwhelming odds against any Simulator in the indifferentiability game. The investigations carried out herein contribute both novel insights and significant refinements to existing literature in this area. The study was centered around three key classes of constructions:

Confusion-Diffusion Networks: The thesis first addresses the indifferentiability of 2-round and 3-round Confusion-Diffusion Networks. It corrects existing errors in previously proposed attacks against the 2-round construction $\mathsf{CDN}_{2,2}$ by providing a more rigorous and general attack strategy, thereby conclusively demonstrating that $\mathsf{CDN}_{2,2}$ is not secure even under the weaker notion of sequential indifferentiability. Conversely, a positive result is established for the 3-round construction $\mathsf{CDN}_{3,2}$ with linear diffusion layers, which is shown to be indifferentiable from an ideal permutation. This establishes a tight bound on the number of rounds required for achieving provable security in this construction paradigm.

Cascade Ciphers: The thesis presents a generalised attack against the 3-round Cascade Cipher construction $\mathsf{CC}^{\mathcal{E}^3}$, which works for a broad class of non-cryptographic key

scheduling functions. This extends prior negative results, which relied on more specific key schedules. The result underscores the limitations of $\mathsf{CC}^{\mathcal{E}^3}$ and supports the necessity of using at least 4 rounds for achieving indifferentiability, thus reaffirming and extending the existing consensus on the topic.

Feistel Networks: The final component of the thesis introduces a general proof framework for analyzing the indifferentiability of Feistel constructions. Within this framework, new indifferentiability proofs are provided for both 7-round and 8-round Feistel networks (Ψ_7 and Ψ_8, respectively). Notably, the result for Ψ_7 is the first of its kind, and the Ψ_8 proof simplifies prior approaches, making it more accessible and adaptable. The framework further shows promise for future work aimed at closing the gap for Ψ_6, the security proof of which remains an open problem in the field.

In summary, these results enhance our theoretical understanding of symmetric key primitives under the indifferentiability paradigm. The findings provide tighter bounds on the rounds required for security and extend the existing literature. While some of the security bounds achieved are not tight from a practical implementation standpoint, they provide a solid foundation for future optimizations and improvements.

Future Research Directions. The resolution of the 6-round Feistel Network remains an open research area, and is a natural extension of the work done on Feistel Networks in this thesis. Another area of exploration is the Cascade cipher with arbitrary number of rounds 'ℓ' and using an offline $(\ell-1)n$-bit to ℓn-bit key scheduling function.

Bibliography

An, J. H. and Bellare, M. (1999). Constructing vil-macsfrom fil-macs: Message authentication under weakened assumptions. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, *Lecture Notes in Computer Science*, vol. 1666, Springer, pp. 252–269.

Aoki, K. et al. (2001). Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis. In D. R. Stinson and S. Tavares, eds., *Selected Areas in Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 39–56.

Baignères, T. and Vaudenay, S. (2006). Proving the Security of AES Substitution-Permutation Network. In B. Preneel and S. Tavares, eds., *Selected Areas in Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 65–81.

Beaulieu, R. et al. (2015). The SIMON and SPECK lightweight block ciphers. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6.

Bellare, M., Canetti, R. and Krawczyk, H. (1996). Pseudorandom functions revisited: the cascade construction and its concrete security. In *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 514–523.

Bellare, M. and Rogaway, P. (1993a). Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, New York, NY, USA: Association for Computing Machinery, CCS '93, p. 62–73.

Bellare, M. and Rogaway, P. (1993b). Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu and V. Ashby, eds., *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, ACM, pp. 62–73.

Bellare, M. and Rogaway, P. (1997). Collision-resistant hashing: Towards making uowhfs practical. In B. S. Kaliski, ed., *Advances in Cryptology — CRYPTO '97*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 470–484.

Bhaumik, R., Nandi, M. and Raychaudhuri, A. (2021). Improved indifferentiability security proof for 3-round tweakable luby-rackoff. *Des Codes Cryptogr*, 89(10), pp. 2255–2281.

Biham, E., Anderson, R. and Knudsen, L. (1998). Serpent: A New Block Cipher Proposal. In S. Vaudenay, ed., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 222–238.

Biryukov, A. and Wagner, D. A. (1999). Slide attacks. In L. R. Knudsen, ed., *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, *Lecture Notes in Computer Science*, vol. 1636, Springer, pp. 245–259.

Blaze, M. and Schneier, B. (1995). The MacGuffin block cipher algorithm. In B. Preneel, ed., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 97–110.

Bogdanov, A. et al. (2007). PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, eds., *Cryptographic Hardware and Embedded Systems - CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 450–466.

Brachtl, B. O. et al. (1990). Data authentication using modification detection codes based on a public one way encryption function.

Canetti, R. (2000). Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Paper 2000/067, https://eprint.iacr.org/2000/067.

Canetti, R., Goldreich, O. and Halevi, S. (2004). The random oracle methodology, revisited. *J ACM*, 51(4), p. 557–594.

Chakraborty, D. and Sarkar, P. (2006). A New Mode of Encryption Providing a Tweakable Strong Pseudo-random Permutation. In M. Robshaw, ed., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 293–309.

Cogliati, B. et al. (2018). Provable Security of (Tweakable) Block Ciphers Based on Substitution-Permutation Networks. In H. Shacham and A. Boldyreva, eds., *Advances in Cryptology – CRYPTO 2018*, Cham: Springer International Publishing, pp. 722–753.

Coron, J., Dodis, Y., Malinaud, C. and Puniya, P. (2005a). Merkle-damgård revisited: How to construct a hash function. In V. Shoup, ed., *Advances in Cryptology - CRYPTO 2005: 25th*

Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, *Lecture Notes in Computer Science*, vol. 3621, Springer, pp. 430–448.

Coron, J.-S., Patarin, J. and Seurin, Y. (2008a). The random oracle model and the ideal cipher model are equivalent. In D. Wagner, ed., *Advances in Cryptology – CRYPTO 2008*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–20.

Coron, J.-S., Patarin, J. and Seurin, Y. (2008b). The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In D. Wagner, ed., *Advances in Cryptology – CRYPTO 2008*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–20.

Coron, J.-S., Dodis, Y., Malinaud, C. and Puniya, P. (2005b). Merkle-damgård revisited: How to construct a hash function. In V. Shoup, ed., *Advances in Cryptology – CRYPTO 2005*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 430–448.

Coron, J.-S., Dodis, Y., Malinaud, C. and Puniya, P. (2005c). Merkle-damgård revisited: How to construct a hash function. In V. Shoup, ed., *Advances in Cryptology – CRYPTO 2005*, p. unknown.

Coron, J.-S., Dodis, Y., Mandal, A. and Seurin, Y. (2010). A domain extender for the ideal cipher. In D. Micciancio, ed., *Theory of Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 273–289.

Coron, J.-S. et al. (2016). How to build an ideal cipher: The indifferentiability of the feistel construction. *Journal of Cryptology*, 29(1), pp. 61–114.

Da, Q., Xu, S. and Guo, C. (2021a). Sequential indifferentiability of confusion-diffusion networks. In A. Adhikari, R. Küsters and B. Preneel, eds., *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, *Lecture Notes in Computer Science*, vol. 13143, Springer, pp. 93–113.

Da, Q., Xu, S. and Guo, C. (2021b). Sequential Indifferentiability of Confusion-Diffusion Networks. In A. Adhikari, R. Küsters and B. Preneel, eds., *Progress in Cryptology – INDOCRYPT 2021*, Cham: Springer International Publishing, pp. 93–113.

Dachman-Soled, D., Katz, J. and Thiruvengadam, A. (2015). 10-round feistel is indifferentiable from an ideal cipher. Cryptology ePrint Archive, Paper 2015/876, https://eprint.iacr.org/2015/876.

Dachman-Soled, D., Katz, J. and Thiruvengadam, A. (2016). 10-round feistel is indifferentiable from an ideal cipher. In M. Fischlin and J. Coron, eds., *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, *Lecture Notes in Computer Science*, vol. 9666, Springer, pp. 649–678.

Daemen, J. and Rijmen, V. (2000). The Block Cipher Rijndael. In J.-J. Quisquater and B. Schneier, eds., *Smart Card Research and Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 277–284.

Dai, Y. and Steinberger, J. (2015). Indifferentiability of 10-round feistel networks. Cryptology ePrint Archive, Paper 2015/874, https://eprint.iacr.org/2015/874.

Dai, Y. and Steinberger, J. (2016a). Indifferentiability of 8-round feistel networks. In M. Robshaw and J. Katz, eds., *Advances in Cryptology – CRYPTO 2016*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 95–120.

Dai, Y. and Steinberger, J. P. (2016b). Indifferentiability of 8-round feistel networks. In M. Robshaw and J. Katz, eds., *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, *Lecture Notes in Computer Science*, vol. 9814, Springer, pp. 95–120.

Damgård, I. B. (1990). A design principle for hash functions. In G. Brassard, ed., *Advances in Cryptology — CRYPTO' 89 Proceedings*, p. unknown.

Dodis, Y., Stam, M., Steinberger, J. P. and Liu, T. (2016a). Indifferentiability of confusion-diffusion networks. In M. Fischlin and J. Coron, eds., *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, *Lecture Notes in Computer Science*, vol. 9666, Springer, pp. 679–704.

Dodis, Y., Stam, M., Steinberger, J. and Liu, T. (2016b). Indifferentiability of confusion-diffusion networks. In M. Fischlin and J.-S. Coron, eds., *Advances in Cryptology – EUROCRYPT 2016*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 679–704.

Dodis, Y., Stam, M., Steinberger, J. and Liu, T. (2016c). Indifferentiability of Confusion-Diffusion Networks. In M. Fischlin and J.-S. Coron, eds., *Advances in Cryptology – EUROCRYPT 2016*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 679–704.

Dodis, Y., Katz, J., Steinberger, J., Thiruvengadam, A. and Zhang, Z. (2017). Provable Security of Substitution-Permutation Networks. Cryptology ePrint Archive, Paper 2017/016.

Even, S. and Goldreich, O. (1985). On the power of cascade ciphers. *ACM Trans Comput Syst*, 3(2), pp. 108–116.

Feistel, H. (1970). Cryptographic Coding for Data-Bank Privacy. Tech. rep., Watson Research Center, Yorktown Heights, New York.

Feistel, H. (1973). Cryptography and computer privacy. *Scientific American*, 228(5), pp. 15–23.

Gao, Y., Guo, C., Wang, M., Wang, W. and Wen, J. (2020). Beyond-Birthday-Bound Security for 4-round Linear Substitution-Permutation Networks. *IACR Transactions on Symmetric Cryptology*, 2020, p. 305–326.

Guo, C., Lin, D. and Liu, M. (2016). Revisiting cascade ciphers in indifferentiability setting. Cryptology ePrint Archive, Paper 2016/825.

Hirose, S. (2006). Some plausible constructions of double-block-length hash functions. In M. Robshaw, ed., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 210–225.

Hoang, V. T. and Rogaway, P. (2010). On generalized feistel networks. In T. Rabin, ed., *Advances in Cryptology – CRYPTO 2010*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 613–630.

Holenstein, T., Künzler, R. and Tessaro, S. (2010). Equivalence of the random oracle model and the ideal cipher model, revisited. *CoRR*, abs/1011.1264, 1011.1264.

Holenstein, T., Künzler, R. and Tessaro, S. (2011a). The equivalence of the random oracle model and the ideal cipher model, revisited. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, New York, NY, USA: ACM, STOC '11, pp. 89–98.

Holenstein, T., Künzler, R. and Tessaro, S. (2011b). The equivalence of the random oracle model and the ideal cipher model, revisited. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, New York, NY, USA: Association for Computing Machinery, STOC '11, p. 89–98.

Iwata, T. and Kurosawa, K. (2001). On the Pseudorandomness of the AES Finalists - RC6 and Serpent. In G. Goos, J. Hartmanis, J. van Leeuwen and B. Schneier, eds., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 231–243.

Jean, J. (2016). TikZ for Cryptographers. https://www.iacr.org/authors/tikz/.

Junod, P. and Macchetti, M. (2009). Revisiting the IDEA Philosophy. In O. Dunkelman, ed., *Fast Software Encryption*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 277–295.

Junod, P. and Vaudenay, S. (2005). FOX : A New Family of Block Ciphers. In H. Handschuh and M. A. Hasan, eds., *Selected Areas in Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 114–129.

Keliher, L., Meijer, H. and Tavares, S. (2000). Provable security of substitution-permutation encryption networks against linear cryptanalysis. In *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492)*, vol. 1, pp. 37–42 vol.1.

Lai, X. and Massey, J. L. (1991). A Proposal for a New Block Encryption Standard. In I. B. Damgård, ed., *Advances in Cryptology — EUROCRYPT '90*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 389–404.

Lai, X. and Massey, J. L. (1993). Hash functions based on block ciphers. In R. A. Rueppel, ed., *Advances in Cryptology — EUROCRYPT' 92*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 55–70.

Lampe, R. and Seurin, Y. (2013). How to construct an ideal cipher from a small set of public permutations. In K. Sako and P. Sarkar, eds., *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*, *Lecture Notes in Computer Science*, vol. 8269, Springer, pp. 444–463.

Liu, T., Pelecanos, A., Tessaro, S. and Vaikuntanathan, V. (2023). Layout Graphs, Random Walks and the t-Wise Independence of SPN Block Ciphers. In H. Handschuh and A. Lysyanskaya, eds., *Advances in Cryptology – CRYPTO 2023*, Cham: Springer Nature Switzerland, pp. 694–726.

Luby, M. and Rackoff, C. (1988). How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2), pp. 373–386, https://doi.org/10.1137/0217022.

Mandal, A., Patarin, J. and Seurin, Y. (2012a). On the public indifferentiability and correlation intractability of the 6-round feistel construction. In R. Cramer, ed., *Theory of Cryptography*

- *9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, *Lecture Notes in Computer Science*, vol. 7194, Springer, pp. 285–302.

Mandal, A., Patarin, J. and Seurin, Y. (2012b). On the public indifferentiability and correlation intractability of the 6-round feistel construction. In *Proceedings of the 9th International Conference on Theory of Cryptography*, Berlin, Heidelberg: Springer-Verlag, TCC'12, p. 285–302.

Maurer, U., Renner, R. and Holenstein, C. (2004a). Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In M. Naor, ed., *Theory of Cryptography Conference — TCC 2004*, *Lecture Notes in Computer Science*, vol. 2951, Springer-Verlag, pp. 21–39.

Maurer, U., Renner, R. and Holenstein, C. (2004b). Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, ed., *Theory of Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 21–39.

Maurer, U. and Sjödin, J. (2005). Single-key AIL-MACs from any FIL-MAC. In L. Caires, ed., *Automata, Languages and Programming — ICALP 2005*, *Lecture Notes in Computer Science*, vol. 3580, Springer-Verlag, pp. 472–484.

Maurer, U. M. and Pietrzak, K. (2003). The security of many-round luby-rackoff pseudo-random permutations. In E. Biham, ed., *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, *Lecture Notes in Computer Science*, vol. 2656, Springer, pp. 544–561.

Merkle, R. C. (1990). One way hash functions and des. In G. Brassard, ed., *Advances in Cryptology — CRYPTO' 89 Proceedings*, p. unknown.

Merkle, R. C. (1991). Fast Software Encryption Functions. In A. J. Menezes and S. A. Vanstone, eds., *Advances in Cryptology-CRYPTO' 90*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 477–501.

Miles, E. and Viola, E. (2015). Substitution-Permutation Networks, Pseudorandom Functions, and Natural Proofs. *J ACM*, 62(6).

Nakahara, J., Rijmen, V., Preneel, B. and Vandewalle, J. (2004). The MESH Block Ciphers. In K.-J. Chae and M. Yung, eds., *Information Security Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 458–473.

Nandi, M. (2010). The characterization of luby-rackoff and its optimum single-key variants. In G. Gong and K. C. Gupta, eds., *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, *Lecture Notes in Computer Science*, vol. 6498, Springer, pp. 82–97.

Nandi, M., Paul, S. and Saha, A. (2023). Indifferentiability of the Confusion-Diffusion Network and the Cascade Block Cipher. In S. El Hajji, S. Mesnager and E. M. Souidi, eds., *Codes, Cryptology and Information Security*, Cham: Springer Nature Switzerland, pp. 178–195.

Naor, M. and Reingold, O. (1999). On the Construction of Pseudorandom Permutations: Luby—Rackoff Revisited. In *Journal of Cryptology*, Springer, pp. 29–66.

National Institute of Standards and Technology (1977). Data Encryption Standard. Tech. Rep. Federal Information Processing Standards Publications (FIPS PUBS) 46, January 15, 1977, U.S. Department of Commerce, Washington, D.C.

National Institute of Standards and Technology (1998). SKIPJACK and KEA algorithm specification. Tech. rep.

National Soviet Bureau of Standards (1989). GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms. Tech. rep.

National Soviet Bureau of Standards (2015). GOST R 34.12-2015: Block Cipher "Kuznyechik". Tech. rep.

Oliynykov, R. et al. (2015). A New Encryption Standard of Ukraine: The Kalyna Block Cipher. Cryptology ePrint Archive, Paper 2015/650.

Patarin, J. (2003). Luby-rackoff: 7 rounds are enough for $2^{n(1-\epsilon)}$ security. In D. Boneh, ed., *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, *Lecture Notes in Computer Science*, vol. 2729, Springer, pp. 513–529.

Patarin, J. (2009). The "coefficients h" technique. In R. M. Avanzi, L. Keliher and F. Sica, eds., *Selected Areas in Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 328–345.

Pfitzmann, B. and Waidner, M. (2004). A model for asynchronous reactive systems and its application to secure message transmission. *IEEE Symposium on Security and Privacy.*

Preneel, B., Govaerts, R. and Vandewalle, J. (1994). Hash functions based on block ciphers: a synthetic approach. In D. R. Stinson, ed., *Advances in Cryptology — CRYPTO' 93*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 368–378.

Seurin, Y. (2009). *Primitives et protocoles cryptographiques à sécurité prouvée*. Ph.D. thesis, Versailles-St Quentin en Yvelines.

Seurin, Y. (2015). A note on the indifferentiability of the 10-round feistel construction. *IACR Cryptol ePrint Arch*, p. 903.

Shannon, C. E. (1949a). Communication theory of secrecy systems. *Bell Syst Tech J*, 28(4), pp. 656–715.

Shannon, C. E. (1949b). Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4), pp. 656–715.

Shimizu, A. and Miyaguchi, S. (1988). FEAL — Fast Data Encipherment Algorithm. *Systems and Computers in Japan*, 19(7), pp. 20–34.

Sorkin, A. (1984). Lucifer, a cryptographic algorithm. *Cryptologia*, 8(1), pp. 22–42.

Vaudenay, S. (1999). On the Lai-Massey Scheme. In K.-Y. Lam, E. Okamoto and C. Xing, eds., *Advances in Cryptology - ASIACRYPT'99*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 8–19.

Winternitz, R. S. (1984). A secure one-way hash function built from des. In *1984 IEEE Symposium on Security and Privacy*, pp. 88–88.

Zhang, W. et al. (2015). RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. In *Science China Information Sciences*, Springer, pp. 1–15.

Appendix A

Supplementary Material for $\mathsf{CDN}_{3,2}$

A.1 Proof of Lemma 8: Probability bound of bad events

Proof 9 *First we claim the following bound on $|\mathcal{P}|$ which will lead to the probability bounds.*

Lemma 21 *For any adversary $\mathcal{A}$ making at most q queries,*

- *$|\mathcal{P}^{\mathcal{A}|\mathcal{S}}|$ is bounded by $4q^2 + 12q$;*
- *$|\mathcal{P}|$ is bounded by $(4q^2 + 12q)^2$;*
- *$|\mathsf{C}^{\mathcal{S}}|$ is bounded by $3(4q^2 + 12q)^2 + q$.*

- *Since $|\mathcal{P}| \leq (4q^2 + 12q)^2$, we get*

$$\Pr(\textit{Injectivity Loss in } \mathcal{P}) \leq (4q^2 + 12q)^4/2^n$$

- *Using the bound for $|\mathcal{P}|$, we get*

$$\Pr(\textit{Accidental Equality in Diffusion Layer}) \leq (4q^2 + 12q)^3/2^n$$

(Note that this probability is dominated by the accidental formation of a $\mathcal{P}^3$-chain)

- *Using the bound for $|\mathcal{P}|$ once again, we get*

$$\begin{aligned}\Pr(\textit{Hit in } \mathcal{G}^{\mathsf{Int}}) &\leq q\big[|V(\mathcal{G}^{\mathsf{Int}})| \times |\mathcal{P}| \times |\textit{New Prim Pairs}|\big]/2^n \\ &\leq q\big[(4q^2+12q)^4 \times (4q^2+12q)^2 \times (4q^2+12q)\big] \\ &\leq q(4q^2+12q)^7/2^n\end{aligned}$$

- *Since $|\mathsf{C}^{\mathcal{A}} \cup \mathsf{C}^{\mathcal{S}}| \leq 3(4q^2+12q)^2+q$ and using the bound for $|\mathcal{P}|$, we get*

$$\Pr(\textit{Hitting in } \Pi) \leq \Big(3(4q^2+12q)^2+q\Big)\Big(4(4q^2+12q)^2+q\Big)/2^n$$

Combining the above probabilities, we get the required result. □

Proof of Lemma 21. *$\mathcal{P}^{\mathcal{A}|\mathcal{S}}$ is composed of primitive queries made directly by the adversary, pairs that* Sim *obtains through revealed adversary construction queries, and lastly, additional pairs that* Sim *may obtain in order to keep $(\mathcal{P}^{\mathcal{A}|\mathcal{S}}, \mathsf{C}^{\mathcal{S}})$ consistently saturated. The first two cases may contribute at most $6q$ pairs each, whereas the third case contributes at most $4q^2$ pairs ($2q$ choices for each of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_5$ and $\mathcal{P}_6$) This gives us the required bound.*

Next, note that the growth of $|\mathcal{P}|$ is dominated by the the fact that Sim *maintains complete chains in its transcript for every pair (of primitive pairs) in $(\mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_3) \times (\mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_4)$. From the bound of $|\mathcal{P}^{\mathcal{A}|\mathcal{S}}|$ given above, we obtain the required bound for $|\mathcal{P}|$.*

Finally, recall that Sim *makes a construction query for each pair (of primitive pairs) in $(\mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_3) \times (\mathcal{P}^{\mathcal{A}|\mathcal{S}} \cap \mathcal{P}_4)$. This gives us the first term of the bound. The second term is due to the fact that the adversary is forced to complete all their construction queries by making appropriate primitive queries after the q queries afforded to the adversary are exhausted.* □

A.2 Illustration of a Substitution-Permutation Network

Fig. A.2.1 shows an illustration of an r-round SPN with 8 4-bit S-boxes in each substitution layer, and bit-permutations as P-boxes. KS represents a key-schedule which takes a master key and outputs the round keys.

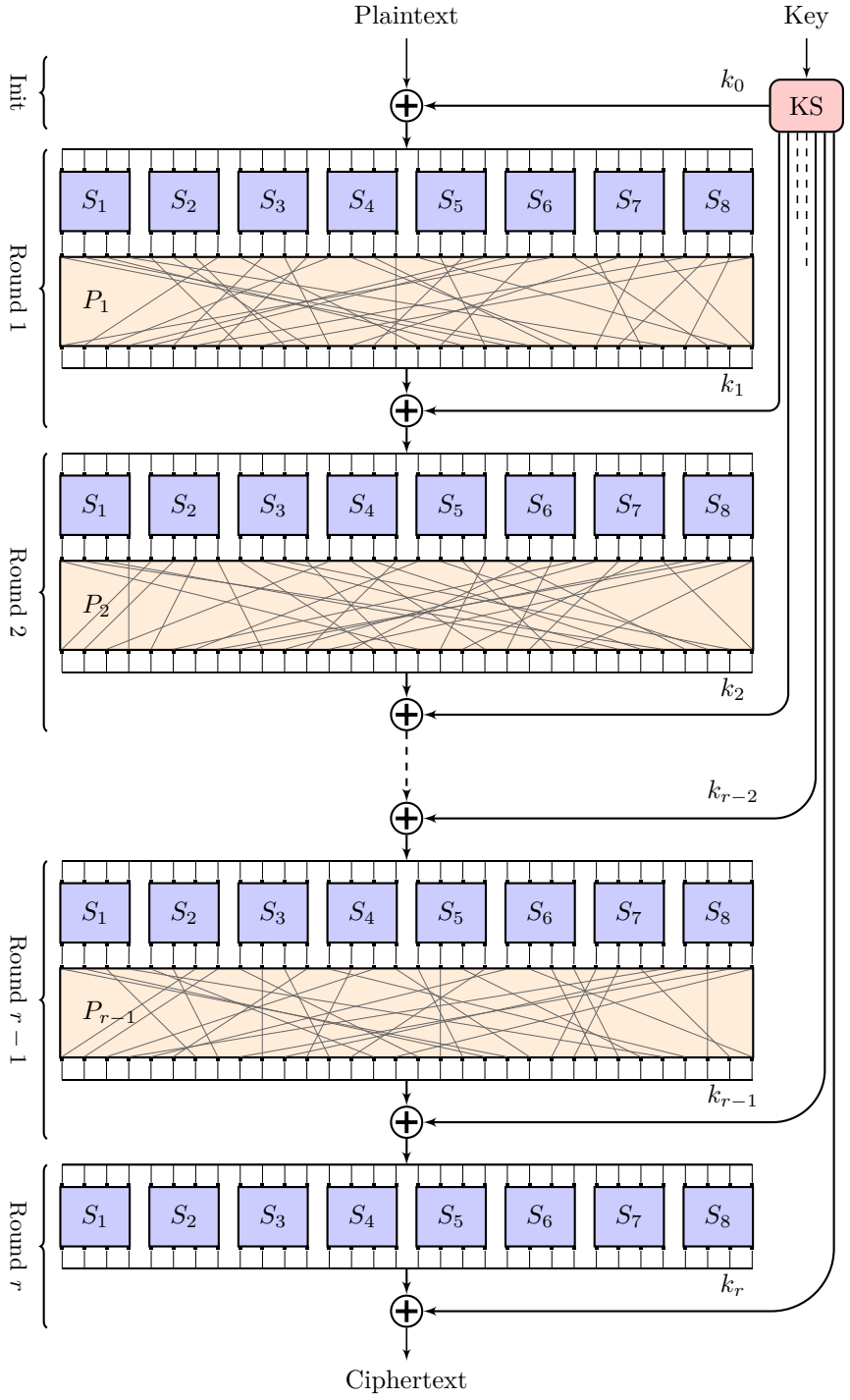

Figure A.2.1: Example of a Substitution-Permutation Network over 32 bits. (Adapted from an example in Jean (2016).)

Appendix B

Supplementary Material for Feistel

B.1 Proof of Lemma 14

Take any F-undefined point u belonging to the shore i. We show that $\mathsf{Sim}_{findL}(u)$ is a well approximation of $\mathsf{L}(u)$.

$i = 1$: *For a revealed point u, let $\ell := \ell(u)$, then $\ell_7, \ell_8 \in \mathsf{DF}$ and $((\ell_8, \ell_9), (\ell_0, \ell_1)) \in P \subseteq \Pi$. Thus from definition of $\mathsf{Sim}^{\Pi}_{find}$ we have that $\ell(u) \in \mathsf{Sim}^{\Pi}_{find}(u|F)$. Now there cannot be any other line ℓ' in $\mathsf{Sim}^{\Pi}_{find}(u|F)$, since then we will have $\Pi_2^{-1}(\ell_8, \ell_9) = \Pi_2^{-1}(\ell'_8, \ell'_9)$, for $\ell'_{(*..9)} \notin \mathscr{L}_{\mathsf{dir\_BDRY}}$, which violates the boundary goodness of Π.*

$i = 2$: *Let $\ell_{(8..2)} \in \mathsf{V}_1(u)$. Then ℓ_8 is a revealed boundary point, which implies there exists a Gen-2+ wrap-around line ℓ' with $\ell \nearrow_8 \ell'$. Thus we have $\ell_1, \ell'_1, \ell'_2 \in \mathsf{DF}$ and $((\ell_0, \ell_1), (\ell_8, \ell_9))$, $((\ell'_0, \ell'_1), (\ell_8, \ell'_9)) \in P \subseteq \Pi$, and hence by definition of $\mathsf{Sim}^{\Pi}_{find}$ it follows that $\ell \in \mathsf{Sim}^{\Pi}_{find}(u|F)$.*

Now, suppose $\ell_{(8..2)} \in \mathsf{Sim}^{\Pi}_{find}(u|F) \setminus \mathsf{V}_2(u)$. Then there exists $\ell'_{(0..*)}$ such that $\Pi_1(\ell_0, \ell_1) = \Pi_1(\ell'_0, \ell'_1)$. Π is bad with respect to (P, F') if either $(\ell_0, \ell_1) \notin \mathrm{Dom}(P)$ or $(\ell'_0, \ell'_1) \notin \mathrm{Dom}(P)$. If both $(\ell_0, \ell_1), (\ell_0, \ell_1) \in \mathrm{Dom}(P)$, then $\ell \in \mathsf{V}_1(u)$, a contradiction.

Let $\ell_{(7..2)} \in \mathsf{V}_2(u)$. Then we have $\ell_1, \ell_8 \in \mathsf{DF}$ with $((\ell_0, \ell_1), (\ell_8, \ell_9)) \in P \subseteq \Pi$, implying that $\ell \in \mathsf{Sim}^{\Pi}_{find}(u|F)$. Now suppose $\ell_{(7..2)} \in \mathsf{Sim}^{\Pi}_{findL}(u|F) \setminus \mathsf{L}_2(u)$. Then $(\ell_0, \ell_1) \notin P$, implying that $\ell_{(0..*)}$ is a F'-line in $\mathscr{L}_{\mathsf{dir\_BDRY}}(\tau') \setminus \mathscr{L}_{\mathsf{dir\_BDRY}*}(\tau')$ for which $\Pi_1(\ell_0, \ell_1) \in \mathsf{DF}$, implying that Π was bad with respect to τ', a contradiction.

$i = 4$**:** *It follows from the definition of* $\mathsf{Sim}^{\Pi}_{find}$ *that* $\mathsf{L}(u) \setminus \mathsf{L}(u)_{(4..9)} \subseteq \mathsf{Sim}^{\Pi}_{find}(u|F)$*. Also for any* $(4..9)$ *line* ℓ*, the simulator will find a* $(4..1)$ *line* ℓ' *such that* $\ell = d(\ell')$*, which will also be a leaf line if* Π *is not boundary bad. Any line in* $\mathsf{Sim}^{\Pi}_{find}(u|F) \setminus \mathsf{L}(u)$ *cannot be direct by definition. If* $\ell_{(4..1)} \in \mathsf{Sim}^{\Pi}_{findL}(u|F) \setminus \mathsf{L}(u)$*, then we must have* $\ell_5, \ell_6, \ell_7, \ell_8 \in \mathsf{DF}$ *but* $(\ell_8, \ell_9) \notin \mathrm{Ran}(P)$*, which would imply* $d(\ell)$ *is a* τ*-boundary direct line.*

So we have proved the case for $i = 1, 2$*, shored points* v, v' *and shore 4 point* w*,*

$$\mathsf{Sim}^{\Pi}_{findL}(v|F) = \mathsf{L}_{\tau}(v),\ \mathsf{Sim}^{\Pi}_{findL}(v, v'|F) = \mathsf{L}_{\tau}(v, v'),$$

$$\mathsf{Sim}^{\Pi}_{findV}(v, v'|F) = \mathsf{V}_{\tau}(v, v'), \mathsf{Sim}^{\Pi}_{findV}(w|F) \supseteq \mathsf{V}_{\tau}(w).$$

Moreover, every $v \in \mathsf{Sim}^{\Pi}_{findV}(w|F) \setminus \mathsf{L}_{\tau}(w)$ *is a* $\mathscr{L}_{\tau}(u)$*-fresh point. Similar results hold for the rest of the shores that will follow symmetrically. We can show in a similar fashion that* $\mathsf{Sim}^{\Pi}_{findL}(v, v'|F) = \mathsf{L}_{\tau}(v, v'), \mathsf{Sim}^{\Pi}_{findV}(v, v'|F) = \mathsf{V}_{\tau}(v, v')$*. Hence, the result follows.*

Probability Bound for Boundary Bad Π**.** *We derive the probability bounds for cases 1 and 3. Cases 2 and 4 are analogous. Let* bad^{Π}_1 *be the event* $v_{r-1} \in \mathrm{Dom}', v_r \in \mathsf{DF}, (v_r, v_{r+1} := v_{r-1} \oplus F(v_r)) \notin \mathrm{Ran}(P)\ :\ \Pi_2^{-1}(v_r, v_{r+1}) \in \textsc{val}(\tau) \cup \{u\}$*, and* bad^{Π}_3 *be the event* $u_1, v_1 \in \mathsf{DF}, u_2, v_2 \in \mathrm{Dom}', (u_0 := F(u_1) \oplus u_2, u_1) \notin \mathrm{Dom}(P)$*:* $\Pi_1(u_0, u_1) = \Pi_1(v_0, v_1)$*. Define* $\mathsf{bad}^{\Pi}_2, \mathsf{bad}^{\Pi}_4$ *similarly. Note that* $|\textsc{val}(\tau)|$ *for shore* 2 *(resp.,* $r - 1$*) is of the order* q^2 *instead of* q^4*. This is because* $|\mathsf{DF}_3|, |\mathsf{DF}_4| \leq 2q$ *(resp.,* $|\mathsf{DF}_{r-3}|, |\mathsf{DF}_{r-2}| \leq 2q$*). We refer to the value points at shore* $2/r - 1$ *as* $\textsc{val}_{2,r-1}(\tau)$

$$\begin{aligned}\Pr(\mathsf{bad}^{\Pi}_1) &= [|\textsc{val}_{2,r-1}(\tau)|^2 \times |\mathsf{DF}_{1/r}|]/2^n \\ &\leq \mathcal{O}(q^6)/2^n\end{aligned}$$

$$\begin{aligned}\Pr(\mathsf{bad}^{\Pi}_3) &= [|\textsc{val}_{2,r-1}(\tau)| \times |\mathsf{DF}_{1/r}|]^2/2^n \\ &\leq \mathcal{O}(q^8/2^n)\end{aligned}$$

Hence, we get $\Pr(\bigvee_{i \in [4]} \mathsf{bad}^{\Pi}_i) \leq \mathcal{O}(q^8)/2^n$ □

B.2 Proof of Lemma 17

Let τ be a good view. We start by establishing the size bounds for $\textsc{val}(\tau)$ and $\mathscr{L}$. We know that $|\mathsf{DF}_4, \mathsf{DF}_5| \leq 2q$. Further, due to (4,5)-completion, we have $|\mathsf{DF}_i| \leq 4q^2,\ i \in \{1,2,3,6,7,8\}$ This implies that the number of direct lines in $\mathscr{L}$ is bounded by $16q^4$. Thus, $|\mathscr{L}(\tau)| \leq q + 16q^4$. Taking into consideration the adversary query bound q, we have

$$|\textsc{val}(\tau)| \leq |\mathsf{DF}| + 2|\mathscr{L}_{\textsc{wa}}(\tau)| + 2|\mathscr{L}_{\mathsf{dir}}(\tau)| \leq (24q^2 + 4q) + 2q + 32q^4 \leq 32q^4 + kq^2,\ k \in \mathbb{N}.$$

Next, note that $|\mathscr{L}_{\mathsf{dir}}| \leq 4q^2$, using $|\mathsf{DF}_4|, |\mathsf{DF}_5| \leq 2q$ (since any direct line in $\mathscr{L}$ passes through points in $\mathsf{DF}_4, \mathsf{DF}_5$). Hence, we have $|\mathscr{L}| \leq |\mathscr{L}_{\textsc{wa}}| + |\mathscr{L}_{\mathsf{dir}}| \leq q + 4q^2$. Hence, $|\textsc{samp}(\mathscr{L})| \leq 4|\mathscr{L}|$, and $|\textsc{samp}(\mathscr{L}) \setminus \mathit{Bd}(\mathscr{L})| \leq 3|\mathscr{L}|$. We are now equipped to find the desired probability bounds.

$$\begin{aligned}
\Pr(\mathsf{bad}_1) &\leq \Pr\Big(\ell_k = \ell'_m,\ (\ell,k) \neq (\ell',m) \in \textsc{samp}(\mathscr{L}) \setminus \mathit{Bd}(\mathscr{L})\Big) + \\
&\quad \Pr\Big(\ell_k = v,\ (\ell,k) \in \textsc{samp}(\mathscr{L}) \setminus \mathit{Bd}(\mathscr{L}), v \in \textsc{val}(\tau)\Big) \\
&\leq \big(9|\mathscr{L}(\tau)|^2 + 3|\mathscr{L}(\tau)| \times |\textsc{val}(\tau)\big)|/2^n \\
&\leq 9(q + 4q^2)^2/2^n + 3(q + 4q^2)(32q^4 + kq^2)/2^n \leq 384q^6/2^n + k_1 q^5/2^n
\end{aligned}$$

Where $k_1 \in \mathbb{N}$. Next, observe that $|\textsc{set}(\mathscr{L})| \leq 2|\mathscr{L}|$, and $|\textsc{set}_1(\mathscr{L})| \leq 2|\mathscr{L}(\tau)|$, which gives us $|\textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})| \leq 2|\mathscr{L}| + 2|\mathscr{L}(\tau)| \leq 2(q + 4q^2 + q + 16q^4) \leq 32q^4 + k_2 q^2,\ k_2 \in \mathbb{N}$. Hence,

$$\begin{aligned}
\Pr(\mathsf{bad}_2) &\leq \Pr\Big(\ell_k = \ell'_m,\ (\ell,k) \neq (\ell',m) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})\Big) + \\
&\quad \Pr\Big(\ell_k = v,\ (\ell,k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L}), v \in \textsc{val}(\tau)\Big) \\
&\leq \big(4(|\mathscr{L}| + |\mathscr{L}(\tau)|)^2 + 2(|\mathscr{L}| + |\mathscr{L}(\tau)|) \times |\textsc{val}(\tau)|\big)/2^n \\
&\leq (32q^4 + k_2 q^2)^2/2^n + (32q^4 + k_2 q^2)(32q^4 + kq^2)/2^n \\
&\leq 1024q^8/2^n + k_3 q^6/2^n\ \ (k_3 \in \mathbb{N})
\end{aligned}$$

$$\begin{aligned}
\Pr(\mathsf{bad}_3) &\leq \Pr\Big(\ell_k = \ell'_m,\ (\ell,k) \in \textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L}), (\ell',m) \in \textsc{samp}(\mathscr{L})\Big) \\
&\leq \big(|\textsc{set}(\mathscr{L}) \cup \textsc{set}_1(\mathscr{L})| \times |\textsc{samp}(\mathscr{L})|\big)/2^n \\
&\leq \big(2(|\mathscr{L}| + |\mathscr{L}(\tau)|) \times 4|\mathscr{L}|\big)/2^n
\end{aligned}$$

$$\leq 4(32q^4 + k2q^2)(q + 4q^2)/2^n \leq 512q^6/2^n + k_4q^5/2^n \ (k_4 \in \mathbb{N})$$

Finally, observe that $|\mathsf{DF}'| \leq |\mathsf{DF}| + 6|\mathscr{L}| \leq 24q^2 + 4q + 6(q + 4q^2) \leq 48q^2 + 10q$. *Define* $\mathscr{L}_4 := \{\ell_4 \mid \ell \in \mathscr{L}\}$, *and similarly,* $\mathscr{L}_5$

$$\begin{aligned}
\Pr(\mathsf{bad}_4) &= \Pr(\ell_3 \textit{ or } \ell_6 \in \mathsf{DF}',\ \ell_{(3..6)} \textit{ is a new segment}) \\
&= \Pr(\ell_3 \textit{ or } \ell_6 \in \mathsf{DF}',\ \textit{at least one of } \ell_4, \ell_5 \in \mathsf{DF}' \setminus \mathsf{DF}) \\
&\leq 2 \times \Pr(\ell_3 \in \mathsf{DF}',\ \ell_4 \in \mathsf{DF}' \setminus \mathsf{DF}, \ell_5 \in \mathsf{DF}) + \\
&\quad 2 \times \Pr(\ell_3 \in \mathsf{DF}',\ \ell_4, \ell_5 \in \mathsf{DF}' \setminus \mathsf{DF}) \\
&\leq 2\big(|\mathsf{DF}'| \times |\mathscr{L}_4| \times |\mathsf{DF}|\big)/2^n + 2\big(|\mathsf{DF}'| \times |\mathscr{L}_4| \times |\mathscr{L}_5|\big)/2^n \\
&\leq 2\big((48q^2 + 10q)(q)(24q^2 + 4q) + (48q^2 + 10q)q^2\big)/2^n \\
&\leq 1152q^5 + k_5q^4 \ (k_5 \in \mathbb{N})
\end{aligned}$$

Combining the above, we have $\Pr(\vee_i\ \mathsf{bad}_i) \leq 1024q^8/2^n + k_6q^6/2^n$, *where* $k_6 \in \mathbb{N}$.

B.3 Proof of Lemma 18

Similar to 8 rounds, we will split the proof for a single input u *from shore* i, *and a pair of inputs* u, u' *from shores* i, j *respectively. With respect to the single input case, identification of wrap-around lines (for* $i = 1, 2, 6, 7$*) and identification of direct lines (for* $i = 3, 5$*) are identical to 8 rounds, the details for which are omitted. We direct our arguments towards proving the lemma for a pair of inputs* u, u'.

$i = 1, j = 6$. *From the definition of* $\mathsf{Sim}^{\Pi}_{findL}(u, u')$, *the simulator finds all size 4 sets* $(\ell_{(6..1)}, k, k', \ell'_{(6/7..1)})$ *such that* $\ell_1 = u, \ell_6 = u'$ *using the condition* $k_0 \oplus k'_0 \oplus \ell_0 \oplus \ell'_0 = k_4 \oplus k'_4$, *and leveraging the fact that the simulator is readily able to identify the (3..1) lines from* DF. *If* Π *is not boundary bad, elements from* $\mathscr{L}_{\mathsf{FI}}$ *are exactly identified, and the direct boundary lines in* L *are accounted for in* $\mathsf{Sim}^{\Pi}_{findL})(u, u')$ *as wrap-around boundary leaf lines.*

$i = 2, j = 7$. *Analogous to the above case.*

$i = 2, j = 3$. *This case refers to the identification of* $\mathscr{L}_{\mathsf{MFI}}$, *the arguments for which are similar to the case for* $\mathscr{L}_{\mathsf{FI}}$ *described above.*

$i = j = 2$. *This case refers to the identification of size 8 sets in* $\mathscr{L}_{\mathsf{FI}}$. *This, too, is very similar*

to the first case. For a size 8 set $(\ell_{(7..2)}, \ell'_{(7..5)}, \ell''_{(7..5)}, k_{(7..1)}, k'_{(7..1)}, \ell^{\dagger}_{(7..5)}, \ell^{\ddagger}_{(7..5)}, \ell'''_{(7..2)}$*, the simulator can readily compute lines* $\ell', \ell'', \ell^{\dagger}, \ell^{\ddagger}$*, use the linear condition for forced intersection to obtain lines* k, k' *such that* $k_1 = k'_1$*. If* Π *is not bad, then the size 8 sets are exactly identified, with the only exception being the direct boundary lines in* L *which are represented as wrap-around boundary lines in* $\mathsf{Sim}^{\Pi}_{findL})(u, u')$

Hence, $(\mathsf{Sim}^{\Pi}_{findV}(u, u'), \mathsf{Sim}^{\Pi}_{findL}(u, u'))$ *is a well-approximation of* (V, L). □

B.4 Proof of Lemma 20

We begin by establishing some preliminary size bounds for $\mathscr{L} := \mathscr{L}(\tau)$*. We know that the adversary queries are bounded by* q*. In order to obtain a probability bound for the above described bad events, we need to find size bounds for the following:*

(i) $\textsc{val}(\tau)$*, and*

(ii) Any simulation closed component $\mathscr{L} \subseteq \mathscr{L}(\tau)$

Note that primitive pairs in $F_i, i \in [3,5]$ *originate either directly from adversary primitive queries on shore* i*, or from completion of adversary* Π *queries. Let* τ *be a good transcript. Since number of wrap-around lines in* $\mathscr{L}\tau$ *is bounded by* q*, we have* $|F_i| \leq 2q$*,* $i \in [3,4,5]$*. This means that, at any stage of the interaction,* $|\mathscr{L}_{(2..5)}(\tau) \cup \mathscr{L}_{(3..6)}(\tau)| \leq 2q^2$*. Hence,* $|F_i| \leq 2q + 2q^2$*,* $i \in [1,2,6,7]$*, which in turn implies that* $|F| \leq 6q + 6q + 6q^2 \leq 12q + 6q^2$*. Further,* $|\mathscr{L}_{\mathsf{dir}}(\tau)| \leq (2q + 2q^2)^2$ *and* $|\mathscr{L}(\tau)| = |\mathscr{L}_{\textsc{wa}}(\tau)| + |\mathscr{L}_{\mathsf{dir}}(\tau)| \leq q + (2q + 2q^2)^2$*. This gives us our bound for* $|\textsc{val}(\tau)| = |F| + 2|\mathscr{L}_{\textsc{wa}}(\tau)| + 2|\mathscr{L}_{\mathsf{dir}}(\tau)| \leq 12q + 6q^2 + 2q + 2(2q + 2q^2)^2 \leq 8q^4 + kq3, k \in \mathbb{N}$*. Also,* $|\mathscr{L}| \leq |\mathscr{L}_{\textsc{wa}}(\tau)| + |\mathscr{L}_{(2..5)} \cup \mathscr{L}_{(3..6)}| \leq q + 8q^2$*, which implies* $\textsc{samp}(\mathscr{L}), \textsc{set}(\mathscr{L}) \leq q + 8q^2$*, and* $\textsc{set}_1(\mathscr{L}) \leq |\mathscr{L}(\tau)| \leq q + (2q + 2q^2)^2$*.* $\textsc{setsamp}(\mathscr{L})$ *is trivially bounded by twice the number of wrap-around lines (*$2q$*). Now, we bound our bad events.*

$$\begin{aligned}\Pr(\mathsf{bad}_1) &= \big[|\textsc{samp}(\mathscr{L})| \times (|\textsc{val}(\tau)| + |\textsc{setsamp}(\mathscr{L})|)\big]/2^n \\ &\leq \mathcal{O}(q^6/2^n)\end{aligned}$$

$$\begin{aligned}\Pr(\mathsf{bad}_2) = \\ &\big[(|\textsc{set}(\mathscr{L})| + |\textsc{set}_1(\mathscr{L})|) \times (|\textsc{val}(\tau)| + |\textsc{samp}(\mathscr{L})| + |\textsc{setsamp}(\mathscr{L})|)\big]/2^n \\ &\leq \mathcal{O}(q^8/2^n)\end{aligned}$$

$$\Pr(\mathsf{bad}_3) = |\mathit{SETSAMP}(\mathscr{L})|^2/2^n \le q^2/2^n$$

$$\begin{aligned}
\Pr(\mathsf{bad}_4) &= 2\Pr(\ell_{(2..5)} \in \mathscr{L}(\tau), \ell_5, \ell_6 := \ell_4 \oplus \hat{\ell_5} \in \mathsf{DF}')+ \\
&\quad 2\Pr(u_3 \in \mathsf{DF}, u_4, u_5, u_6 := u_4 \oplus \hat{u_5} \in \mathsf{DF}', u_3 = \hat{u_4} \oplus u_5)+ \\
&\quad \Pr(u_3, u_4, u_5 \in \mathsf{DF}', u_3 = \hat{u_4} \oplus u_5) \\
&\quad \le [2|\mathscr{L}_{\mathsf{dir}}(\tau)| \times |\mathit{SAMP}(\mathscr{L}) \cup \mathit{SET}(\mathscr{L})| + 2|\mathsf{DF}_3| \times |\mathit{SAMP}(\mathscr{L}) \cup \mathit{SET}(\mathscr{L})|^2 \\
&\quad + |\mathit{SAMP}(\mathscr{L}) \cup \mathit{SET}(\mathscr{L})|^3]/2^n \\
&\quad \le \mathcal{O}(q^6/2^n)
\end{aligned}$$

□

B.5 Details of 7-round simulator

Wrap-Around Gen-4 lines. *For the sub-collection of Gen-4 lines $\mathscr{L}_{(3..1)}$ (resp., $\mathscr{L}_{(7..5)}$), define*

$$\begin{aligned}
\mathit{SAMP}(\mathscr{L}_{(3..1)}) &= \{(\ell, 3) \ : \ \ell_{(3..1)} \in \mathscr{L}_{(3..1)}\}, \\
\mathit{SET}(\mathscr{L}_{(3..1)}) &= \{(\ell, 2) \ : \ \ell_{(3..1)} \in \mathscr{L}_{(3..1)}\}, \\
\mathit{SAMP}(\mathscr{L}_{(7..5)}) &= \{(\ell, 5) \ : \ \ell_{(7..5)} \in \mathscr{L}_{(7..5)}\}, \\
\mathit{SET}(\mathscr{L}_{(7..5)}) &= \{(\ell, 6) \ : \ \ell_{(7..5)} \in \mathscr{L}_{(7..5)}\}.
\end{aligned}$$

Wrap-Around Boundary Gen-2/3 lines. *For the sub-collections of Gen-2/3 boundary lines $\mathscr{L}_{\mathit{WA\_BDRY},\mathsf{Gen\text{-}2}}, \mathscr{L}_{\mathit{WA\_BDRY},\mathsf{Gen\text{-}3}}$, define*

$$\begin{aligned}
\mathit{SAMP}(\mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}3}}) &= \{(\ell, k) \ : \ \ell_{(i..j)} \in \mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}3}}, k \in \{j, i\}\}, \\
\mathit{SET}(\mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}3}}) &= \{(\ell, k) \ : \ \ell_{(i..j)} \in \mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}3}}, k \in \{j+1, i-1\}\}, \\
\mathit{SAMP}(\mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}2}}) &= \{(\ell, k) \ : \ \ell_{(i..j)} \in \mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}3}}, k \in \{j, j+2, i\}\}, \\
\mathit{SET}(\mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}2}}) &= \{(\ell, k) \ : \ \ell_{(i..j)} \in \mathscr{L}_{\mathit{WA_BDRY},\mathsf{Gen\text{-}2}}, k \in \{j+1, i-1\}\}.
\end{aligned}$$

(6..2) lines. *For the sub-collection of Gen-2 lines $\mathscr{L}_{(6..2)}$, define*

$$SAMP(\mathscr{L}_{(6..2)}) = \{(\ell, k) \ : \ \ell_{(6..2)} \in \mathscr{L}_{(6..2)}, k \in \{2,4,6\}\},$$
$$SET(\mathscr{L}_{(6..2)}) = \{(\ell, k) \ : \ \ell_{(6..2)} \in \mathscr{L}_{(6..2)}, k \in \{3,5\}\}.$$

Forced-Intersection and Mixed Forced-Intersection Gen-0/1 lines. *Let $\mathfrak{L}_{FI}$, $\mathscr{L}_{FI}$, $\mathfrak{L}_{MFI}$, $\mathscr{L}_{MFI}$ be defined as before. Let $\mathscr{L}_{FI,(7..2)}, \mathscr{L}_{MFI,(7..2)}$ (resp., $\mathscr{L}_{FI,(6..1)}$, $\mathscr{L}_{MFI,(6..1)}$) be the set of all (7..2) (resp., (6..1)) lines in $\mathscr{L}_{FI}, \mathscr{L}_{MFI}$, respectively. Define*

$$SAMP(\mathscr{L}_{MFI,(7..2)}) = \{(\ell, k) \ : \ \ell_{(7..2)} \in \mathscr{L}_{MFI,(7..2)}, k \in \{2,4\}\},$$
$$SET(\mathscr{L}_{MFI,(7..2)}) = \{(\ell, k) \ : \ \ell_{(7..2)} \in \mathscr{L}_{MFI,(7..2)}, k \in \{3,5,6\}\},$$
$$SAMP(\mathscr{L}_{MFI,(6..1)}) = \{(\ell, k) \ : \ \ell_{(6..1)} \in \mathscr{L}_{MFI,(6..1)}, k \in \{4,6\}\},$$
$$SET(\mathscr{L}_{MFI,(6..1)}) = \{(\ell, k) \ : \ \ell_{(6..1)} \in \mathscr{L}_{MFI,(6..1)}, k \in \{2,3,5\}\}.$$

Since a line $\ell_{WA,\mathsf{Gen\text{-}1}}$ may belong to $\mathscr{L}_{FI,\mathsf{Gen\text{-}1}} \cap \mathscr{L}_{MFI,\mathsf{Gen\text{-}1}}$, we consider the wrap-around, Gen-1 lines that are exclusively a part of $\mathscr{L}_{FI}$. Let $\mathscr{L}'_{FI,(7..2)} = \mathscr{L}_{FI,(7..2)} \setminus \mathscr{L}_{MFI,(7..2)}$ (resp., $\mathscr{L}'_{FI,(6..1)} = \mathscr{L}_{FI,(6..1)} \setminus \mathscr{L}_{MFI,(6..1)}$).

$$SETSAMP(\mathscr{L}'_{FI,(7..2)}) = \{(\ell, 6) \ : \ \ell_{(7..2)} \in \mathscr{L}'_{FI,(7..2)}\},$$
$$SAMP(\mathscr{L}'_{FI,(7..2)}) = \{(\ell, k) \ : \ \ell_{(7..2)} \in \mathscr{L}'_{FI,(7..2)}, \ k \in \{2,4\}\},$$
$$SET(\mathscr{L}'_{FI,(7..2)}) = \{(\ell, k) \ : \ \ell_{(7..2)} \in \mathscr{L}'_{FI,(7..2)}, \ k \in \{3,5\}\},$$
$$SETSAMP(\mathscr{L}'_{FI,(6..1)}) = \{(\ell, 2) \ : \ \ell_{(6..1)} \in \mathscr{L}'_{FI,(6..1)}\},$$
$$SAMP(\mathscr{L}'_{FI,(6..1)}) = \{(\ell, k) \ : \ \ell_{(6..1)} \in \mathscr{L}'_{FI,(6..1)}, \ k \in \{4,6\}\},$$
$$SET(\mathscr{L}'_{FI,(6..1)}) = \{(\ell, k) \ : \ \ell_{(6..1)} \in \mathscr{L}'_{FI,(6..1)}, \ k \in \{3,5\}\}.$$

Let $\mathscr{L}_{FI,(7..1)}, \mathscr{L}_{MFI,(7..1)}$ be the set of all (7..1) lines in $\mathscr{L}_{FI}, \mathscr{L}_{MFI}$, respectively. For any $\ell_{(7..1)} \in \mathscr{L}_{FI,(7..1)} \cup \mathscr{L}_{MFI,(7..1)}$, we must have $\ell_1, \ell_7 \in \mathsf{Rev}(\mathscr{L})$. As before, since a (7..1) line ℓ may be included in $\mathscr{L}_{FI,(7..1)} \cap \mathscr{L}_{MFI,(7..1)}$, we must first categorise the lines in $\mathscr{L}_{FI,(7..1)} \cup \mathscr{L}_{MFI,(7..1)}$ before defining the sampling/setting functions for them. We call a (7..1) line $\ell \in \mathscr{L}_{FI,(7..1)} \cup \mathscr{L}_{MFI,(7..1)}$ two-side forced *if there exists a forced-intersection/mixed forced-intersection structure at both shores 1 and 7. If not, ℓ is called* one-side forced *Next, consider the following sets:*

- $\mathscr{L}_{\mathsf{MFI}^2,(7..1)} = \{\ell_{(7..1)} \ : \ \ell \in \mathscr{L}_{\mathsf{MFI},(7..1)}, \ell \text{ is two-side forced}\}$
- $\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1)} = \{\ell_{(7..1)} \ : \ \ell \in \mathscr{L}_{\mathsf{MFI},(7..1)} \setminus \mathscr{L}_{\mathsf{MFI}^2,(7..1)}, \ell \text{ is two-side forced}\}$ *(We further divide this collection into* $\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),1}$ *and* $\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),7}$ *where lines in* $\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),i}$ *contain a mixed forced-intersection structure at shore* i, $i \in \{1,7\}$*)*
- $\mathscr{L}_{\mathsf{FI}^2,(7..1)} = \{\ell_{(7..1)} \ : \ \ell \in \mathscr{L}_{\mathsf{FI},(7..1)} \setminus [\mathscr{L}_{\mathsf{MFI}^2,(7..1)} \cup \mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1)}], \ell \text{ is two-side forced}\}$

It is clear that $\mathscr{L}^{TSF}_{(7..1)} := \mathscr{L}_{\mathsf{MFI}^2,(7..1)} \cup \mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1)} \cup \mathscr{L}_{\mathsf{FI}^2,(7..1)}$ *is the disjoint union of all two-side forced (7..1) lines in* $\mathscr{L}_{\mathsf{FI}} \cup \mathscr{L}_{\mathsf{MFI}}$*. Now, define*

$$\begin{aligned}
SAMP(\mathscr{L}_{\mathsf{MFI}^2,(7..1)}) &= \{(\ell,4) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}\},\\
SET(\mathscr{L}_{\mathsf{MFI}^2,(7..1)}) &= \{(\ell,k) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}, k \in [1,7] \setminus \{4\}\},\\
SETSAMP(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),1}) &= \{(\ell,6) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}\},\\
SAMP(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),1}) &= \{(\ell,4) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}\},\\
SET(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1)},1) &= \{(\ell,k) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}, k \in \{2,3,5\}\},\\
SETSAMP(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),7}) &= \{(\ell,2) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}\},\\
SAMP(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1),7}) &= \{(\ell,4) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}\},\\
SET(\mathscr{L}_{\mathsf{MFI},\mathsf{FI},(7..1)},7) &= \{(\ell,k) \ : \ \ell \in \mathscr{L}_{\mathsf{MFI}^2,(7..1)}, k \in \{3,5,6\}\}.
\end{aligned}$$

Note that $\mathscr{L}^{OSF}_{(7..1)} := [\mathscr{L}_{\mathsf{FI},(7..1)} \cup \mathscr{L}_{\mathsf{MFI},(7..1)}] \setminus \mathscr{L}^{TSF}_{(7..1)}$ *is composed of one-side forced lines only. Let* $\mathscr{L}_{\mathsf{MFI},(7..1),1} \subseteq \mathscr{L}^{OSF}_{(7..1)}$ *(resp.,* $\mathscr{L}_{\mathsf{MFI},(7..1),7}$*) be the set of one-side forced (7..1) lines* ℓ *such that* $(\ell, \ell'_{(3..1)}, \ell''_{3..j}) \in \mathfrak{L}_{\mathsf{MFI}}$ *(resp.,*
$(\ell, \ell'_{(7..5)}, \ell''_{i..5}) \in \mathfrak{L}_{\mathsf{MFI}}$*) for some* ℓ', ℓ''*. Similarly, let* $\mathscr{L}_{\mathsf{FI},(7..1),1} \subseteq \mathscr{L}^{OSF}_{(7..1)} \setminus \mathscr{L}_{\mathsf{MFI},(7..1),1}$ *(resp.,* $\mathscr{L}_{\mathsf{FI},(7..1),7}$*) be the set of one-side forced (7..1) lines* ℓ *such that* $(\ell, \ell'_{(3..1)}, \ell''_{(3..1)}, \ell''') \in \mathfrak{L}_{\mathsf{FI}}$ *(resp.,* $(\ell, \ell'_{(7..5)}, \ell''_{(7..5)}, \ell''') \in \mathfrak{L}_{\mathsf{FI}}$*) for some* ℓ', ℓ'', ℓ'''*, and* $\ell \notin \mathscr{L}_{\mathsf{MFI},(7..1),1}$ *(resp.,* $\ell \notin \mathscr{L}_{\mathsf{MFI},(7..1),7}$*). Define*

$$\begin{aligned}
SAMP(\mathscr{L}_{\mathsf{MFI},(7..1),1}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{MFI},(7..1),1}, k \in \{4,5,7\}\},\\
SET(\mathscr{L}_{\mathsf{MFI},(7..1),1}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{MFI},(7..1),1}, k \in \{2,3,6\}\},\\
SAMP(\mathscr{L}_{\mathsf{MFI},(7..1),7}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{MFI},(7..1),7}, k \in \{1,3,4\}\},\\
SET(\mathscr{L}_{\mathsf{MFI},(7..1),7}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{MFI},(7..1),7}, k \in \{2,5,6\}\}.
\end{aligned}$$

and

$$\begin{aligned}
\textsc{setsamp}(\mathscr{L}_{\mathsf{FI},(7..1),1}) &= \{(\ell,2) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{FI},(7..1),1}\},\\
\textsc{samp}(\mathscr{L}_{\mathsf{FI},(7..1),1}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{FI},(7..1),1}, k \in \{4,5,7\}\},\\
\textsc{set}(\mathscr{L}_{\mathsf{FI},(7..1),1}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{MFI},(7..1),1}, k \in \{3,6\}\},\\
\textsc{setsamp}(\mathscr{L}_{\mathsf{FI},(7..1),7}) &= \{(\ell,6) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{FI},(7..1),7}\},\\
\textsc{samp}(\mathscr{L}_{\mathsf{FI},(7..1),7}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{FI},(7..1),7}, k \in \{1,3,4\}\},\\
\textsc{set}(\mathscr{L}_{\mathsf{FI},(7..1),7}) &= \{(\ell,k) \ : \ \ell_{(7..1)} \in \mathscr{L}_{\mathsf{FI},(7..1),7}, k \in \{2,5\}\}.
\end{aligned}$$

Other Wrap-Around Gen-1 lines. *Let $\mathscr{L}'_{(6..1)}$ (resp., $\mathscr{L}'_{(7..2)}$) be the sub-collection of Gen-1 lines such that $\ell \in \mathscr{L}'_{(6..1)} \iff \ell \nwarrow_1 \ell'_{(3..1)}$ and $\ell \in \mathscr{L} \setminus [\mathscr{L}_{\mathsf{FI}} \cup \mathscr{L}_{\mathsf{MFI}}]$ (resp., $\ell \in \mathscr{L}'_{(7..2)} \iff \ell \nwarrow_7 \ell'_{(7..5)}$ and $\ell \in \mathscr{L} \setminus [\mathscr{L}_{\mathsf{FI}} \cup \mathscr{L}_{\mathsf{MFI}}]$). Define*

$$\begin{aligned}
\textsc{samp}(\mathscr{L}'_{(6..1)}) &= \{(\ell,k) \ : \ \ell_{(6..1)} \in \mathscr{L}'_{(6..1)}, k \in \{3,4,6\}\},\\
\textsc{set}(\mathscr{L}'_{(6..1)}) &= \{(\ell,k) \ : \ \ell_{(6..1)} \in \mathscr{L}'_{(6..1)}, k \in \{2,5\}\},\\
\textsc{samp}(\mathscr{L}'_{(7..2)}) &= \{(\ell,k) \ : \ \ell_{(7..2)} \in \mathscr{L}'_{(7..2)}, k \in \{2,4,5\}\},\\
\textsc{set}(\mathscr{L}'_{(7..2)}) &= \{(\ell,k) \ : \ \ell_{(7..2)} \in \mathscr{L}'_{(7..2)}, k \in \{3,6\}\}.
\end{aligned}$$

Next, for the sub-collection of Gen-1 lines $\mathscr{L}''_{(6..1)} := \mathscr{L}_{(6..1)} \setminus [\mathscr{L}_{\mathsf{FI}} \cup \mathscr{L}_{\mathsf{MFI}} \cup \mathscr{L}'_{(6..1)}]$ (resp., $\mathscr{L}''_{(7..2)} := \mathscr{L}_{(7..2)} \setminus [\mathscr{L}_{\mathsf{FI}} \cup \mathscr{L}_{\mathsf{MFI}} \cup \mathscr{L}'_{(7..2)}]$), define

$$\begin{aligned}
\textsc{samp}(\mathscr{L}''_{(6..1)}) &= \{(\ell,k) \ : \ \ell_{(6..1)} \in \mathscr{L}''_{(6..1)}, k \in \{1,3,4,6\}\},\\
\textsc{set}(\mathscr{L}''_{(6..1)}) &= \{(\ell,k) \ : \ \ell_{(6..1)} \in \mathscr{L}''_{(6..1)}, k \in \{2,5\}\},\\
\textsc{samp}(\mathscr{L}''_{(7..2)}) &= \{(\ell,k) \ : \ \ell_{(7..2)} \in \mathscr{L}''_{(7..2)}, k \in \{2,4,5,7\}\},\\
\textsc{set}(\mathscr{L}''_{(6..1)}) &= \{(\ell,k) \ : \ \ell_{(6..1)} \in \mathscr{L}''_{(6..1)}, k \in \{3,6\}\}.
\end{aligned}$$

Direct lines. *Recall that any direct line that may be contained in a simulation-closed line collection $\mathscr{L}$ can either be a $(3..j)$ line ($j \in [5,7]$) or an $(i..5)$ line ($i \in [1,3]$). Define for $j = 8$,*

$$\textsc{samp}(\mathscr{L}_{\mathsf{dir},(3..7)}) = \{(\ell, k) \ : \ \ell_{(3..7)} \in \mathscr{L}_{\mathsf{dir},(3..7)}, k \in \{0,1,3,7\}\},$$
$$\textsc{set}(\mathscr{L}_{\mathsf{dir},(3..7)}) = \{(\ell, k) \ : \ \ell_{(3..7)} \in \mathscr{L}_{\mathsf{dir},(3..7)}, k \in \{2,8\}\}.$$

and for $j \in \{5,6\}$,

$$\textsc{samp}(\mathscr{L}_{\mathsf{dir},(3..j)}) = \{(\ell, k) \ : \ \ell_{(3..j)} \in \mathscr{L}_{\mathsf{dir},(3..j)}, k \in \{0,1,3\} \cup \{j\} \cup [j+2,8]\},$$
$$\textsc{set}(\mathscr{L}_{\mathsf{dir},(3..j)}) = \{(\ell, k) \ : \ \ell_{(3..j)} \in \mathscr{L}_{\mathsf{dir},(3..j)}, k \in \{2, j+1\}\}.$$

Similarly define for $i = 1$,

$$\textsc{samp}(\mathscr{L}_{\mathsf{dir},(1..5)}) = \{(\ell, k) \ : \ \ell_{(1..5)} \in \mathscr{L}_{\mathsf{dir},(1..5)}, k \in \{1,5,7,8\}\},$$
$$\textsc{set}(\mathscr{L}_{\mathsf{dir},(1..5)}) = \{(\ell, k) \ : \ \ell_{(1..5)} \in \mathscr{L}_{\mathsf{dir},(1..5)}, k \in \{0,6\}\}.$$

and for $i = 2$,

$$\textsc{samp}(\mathscr{L}_{\mathsf{dir},(i..5)}) = \{(\ell, k) \ : \ \ell_{(i..5)} \in \mathscr{L}_{\mathsf{dir},(i..5)}, k \in [0, i-2] \cup \{i\} \cup \{5,7,8\}\},$$
$$\textsc{set}(\mathscr{L}_{\mathsf{dir},(i..5)}) = \{(\ell, k) \ : \ \ell_{(i..5)} \in \mathscr{L}_{\mathsf{dir},(i..5)}, k \in \{i-1, 6\}\}.$$

Extension of $\mathscr{L}_1$ *Lastly, we define functions* $\textsc{set}_{1+}, \textsc{set}_{1-}$ *for the set* $\mathscr{L}_1$. *We divide* $\mathscr{L}_1$ *into the following sub-collections:*

- $\mathscr{L}_{1,(7..1),\curlywedge_1}$:
  - $\exists\ \ell'_{(3..1)} \curlywedge_3 \ell''_{(3..1)} \curlywedge_1 \ell'''_{(6/7..1)} \in \mathscr{L}$ *such that* $\ell \curlywedge_1 \ell'$, *and* $\ell_0 \oplus \ell'_0 \oplus \ell''_0 \oplus \ell'''_0 = \ell'_4 \oplus \ell''_4$, *or*
  - $\exists\ \ell'_{(3..1)} \curlywedge_3 \ell''_{(3..j)} \in \mathscr{L},\ j \in [5,7] \cup \{1\}$ *such that* $\ell_0 \oplus \ell'_0 = \ell'_4 \oplus \ell''_4$
- $\mathscr{L}_{1,(7..1),\curlywedge_7}$: *Similarly, for opposite shores*

Also, let $\mathscr{L}'_1 = \mathscr{L}_1 \setminus [\mathscr{L}_{1,(7..1),\curlywedge_1} \cup \mathscr{L}_{1,(7..1),\curlywedge_7}]$. *Equipped with these sets, define*

$$\begin{aligned}
\mathit{SET}_{2+}(\mathscr{L}_{1,(7..1),\searrow_1}) &= \{(\ell,2),(\ell,3) \ : \ \ell \in \mathscr{L}_{1,(7..1),\searrow_1}\},\\
\mathit{SET}_{2-}(\mathscr{L}_{1,(7..1),\searrow_7}) &= \{(\ell,5),(\ell,6) \ : \ \ell \in \mathscr{L}_{1,(7..1),\searrow_7}\},\\
\mathit{SET}_{1+}(\mathscr{L}'_1) &= \{(\ell,j+1) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_j \in \mathit{Bd}(\mathscr{L})\},\\
\mathit{SET}_{1-}(\mathscr{L}'_1) &= \{(\ell,i-1) \ : \ \ell_{(i..j)} \in \mathscr{L}_1, \ell_i \in \mathit{Bd}(\mathscr{L})\}.
\end{aligned}$$

Finally, define

$$\mathit{SET}_{1/2}(\mathscr{L}_1) = \mathit{SET}_{1+}(\mathscr{L}_1) \cup \mathit{SET}_{1-}(\mathscr{L}_1) \cup \mathit{SET}_{2+}(\mathscr{L}_1) \cup \mathit{SET}_{2-}(\mathscr{L}_1)$$