

**THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ**

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Paul Frixons

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

**Cryptographie à clé secrète et attaquant quantique
dans le monde des télécommunications**

Soutenue publiquement le 25 novembre 2022

devant le jury composé de :

María NAYA-PLASENCIA	Inria de Paris	Directrice
Sébastien CANARD	Orange Labs	Co-directeur
Loïc FERREIRA	Orange Labs	Co-encadrant
Marine MINIER	LoRIA	Rapportrice
Gilles VAN ASSCHE	STMicroelectronics	Rapporteur
Charles BOUILLAGUET	LIP6	Examinateur
Céline CHEVALIER	Paris-Panthéon-Assas University	Examinatrice
Henri GILBERT	ANSSI	Examinateur
Xavier BONNETAIN	LoRIA	Examinateur

Résumé

Pour la cryptographie moderne, la sécurité d'un système est défini comme la somme des ressources nécessaires pour le briser. Avec la venue d'ordinateurs quantiques efficaces et les nouvelles possibilités algorithmiques que cela ouvre, ce montant de ressources est voué à changer.

Dans cette thèse, nous effectuons un pas en direction d'une meilleure compréhension de cette menace quantique. Après une introduction au calcul quantique et à la cryptographie, nous montrons des attaques quantiques contre la fonction pseudo-aléatoire de Legendre dans le cadre avec le moins de suppositions. Par la suite, nous exposons une manière générale de transposer les attaques boomerang en algorithmique quantique ainsi que quelques applications. Nous continuons sur une méthode de doublement de taille de blocs pour les chiffrements à blocs inspirée sur le schéma Encrypt-Mix-Encrypt et nous en montrons la sécurité. Nous finissons par la construction d'une version quantique du protocole d'authentification de la 3G/4G/5G UMTS-AKA avant d'en montrer la sécurité ainsi que celle des primitives sous-jacentes Milenage et TUAK.

Mots-clés : cryptographie à clé symétrique, cryptanalyse quantique, preuve de sécurité, protocole d'échange de clé authentifié.

Abstract

For modern cryptography, the security of a system is defined as the sum of the resources required to break it. With the advent of efficient quantum computers and the new algorithmic possibilities that this opens, this amount of resource is destined to change.

In this thesis, we take a step towards a better understanding of this quantum threat. After an introduction to quantum computation and cryptography, we show quantum attacks against the Legendre PRF in the setting with the less suppositions. Afterwards, we present a general way to transpose boomerang attacks into quantum attacks as well as some applications. We continue on a doubling method for block ciphers inspired by the Encrypt-Mix-Encrypt scheme and prove its security. We end by building a quantum version of the 3G/4G/5G UMTS-AKA authentication protocol before showing the security as well as the underlying primitives Milenage and TUAK.

Keywords: symmetric-key cryptography, quantum cryptanalysis, security proofs, authenticated key exchange.

Introduction

Cryptography is the science of transmitting information through an insecure channel. It accompanies communications from antiquity to our modern days. However it became a branch of mathematics only in the 20th century. It is a subset of complexity theory, the study of the hardness of computational problems.

Computational Security. Modern cryptography contains a certain set of tools, mathematical objects which are expected to hold security up to a computational bound. Indeed, these tools are designed to make information unreadable to any unauthorized entity (or adversary) until it spends a certain (unreachable) amount of resources.

There are many ways to ensure this kind of security. The first one is relying on the difficulty of a general problem. As an example, RSA (one of the most widely used cryptosystems) relies on the difficulty of the factorization problem. This way of building is powerful as many instances can be built from one problem. Then the main focus of verification is the underlying problem. However, it does not mean that the instantiation is left unchecked (many early versions of RSA were vulnerable). The second one is designing building blocks called “primitives” that are the focus for the security evaluation. Then, these can be used with modes of operations which allows those primitives to encrypt practical messages. These modes of operations are also proven to be secure to ensure the security of the whole process.

Quantum Threat. The concept of quantum computers was proposed in the early 1980s by visionary physicists. Simulating the evolution of complex quantum systems, like long molecules, is a hard problem for our computers. The concept of quantum computers is to turn this problem around, making quantum systems a new way to make computations. Physicists that worked in this direction quickly understood that it would lead to a quantum computer executing quantum algorithm with its own notion of complexity.

However, the first demonstration of the possibilities of quantum computing was made in 1992 with the Deutsch–Jozsa algorithm, which states whether a function $\{0, 1\}^n$ to $\{0, 1\}$ is constant or equilibrated ($|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$) in only one query, whereas a classical computer needs $2^{n-1} + 1$ queries

to achieve certainty. It was closely followed by Simon’s algorithm and Shor’s algorithm, both in 1994. The first one recovers hidden periods and the second one factorizes large numbers.

The factorization of large numbers happens to be the problem on which RSA relies on for its security.

The realization of such quantum computers began in 1998 with a working 2-qubit computer implementing Deutsch–Jozsa algorithm. It was followed by the first factorization using Shor’s algorithm in 2001 by IBM with a 7-qubit machine. Another important step has been the claim by Google to have achieved quantum supremacy (having a quantum computer solving a problem that would be “irrealizable” for any existing classical computer) in 2019 and IBM promising a 1121-qubit computer by 2023 and a million-qubit computer by 2030.

Quantum Cryptanalysis. We gain confidence in the security of the different cryptographic constructions by challenging their claimed security bounds. Cryptanalysis is the set of all techniques employed and developed in that manner.

The consideration of the quantum computer as a potential adversary led to many new attacks and techniques known as quantum cryptanalysis. While many systems and constructions were theoretically broken, the true potential of quantum computing is still unknown. Indeed, discovered quantum algorithm are few and even combining known quantum algorithms falls in a gray area.

Security Proofs. Provable security as a domain is making security proofs of a construction on the assumption of the security of the underlying blocks. While we are far from determining every construction this way, provable security has many tools at its disposal against classical adversaries.

However, against quantum adversaries many of those tools do not meet its quantum equivalent as many notions that appear trivial in the classical world are broken in the quantum world.

Quantum Protocols. As cryptography is about transmitting information in a secure way through an insecure channel, it involves caring about the information left in said channel. Cryptographic protocols can be defined as procedures involving two or more legitimate entities in the goal to transmit some information and verifying some security properties.

Like quantum computers were conceptualized because of the complexity of quantum systems, most quantum protocols use the properties of quantum mechanics to build unforgeable materials.

Post-Quantum Future of Cryptography. As security is defined as the amount of resources needed to break it, our communication can only be secure

for a certain duration determined by current (classical) analysis. However, as more than 90% of the worldwide communications use cryptosystems that are vulnerable to the quantum computers, the moment of their arrival is the end of the security of those communications. This does not concern only communications made past the arrival of quantum computers but also those who were guaranteed to remain safe at that time.

Then communications have to change to quantum-secure cryptosystems not only when quantum computers arrive but before their security duration are at threat to get brutally shortened.

Moreover, making a new standard is long, not including having it implemented by different companies (the NIST competition for post-quantum standards took five years to decide a first algorithm to be standardized and the competition is still on). Cryptography has to be one step ahead.

Organization

Chapter 1. We present the tools of quantum computing used in the following chapters. It starts with a quick history of the domain and continues with the elementary notions of quantum computing: qubits, superposition, entanglement, operators and quantum complexity.

Chapter 2. We expose the major algorithmic building blocks that pave our work: Grover's search and its generalization, the amplitude amplification algorithm, BHT collision search, Shor's algorithm, Simon's period finding algorithm and some of its intricacies and Kuperberg's algorithm.

Chapter 3. We present the notions of cryptography necessary to the next chapters. It starts with an overview of the domain including asymmetric cryptography, symmetric cryptography and its constructions. It continues with foundations of cryptanalysis and specific notions related to quantum cryptanalysis. It ends with an overview of communication protocols.

Chapter 4. We expose new quantum attacks against the Legendre PRF, from a joint work with André Schrottenloher, published at MathCrypt in 2021 [57].

We present two quantum algorithms for solving the Legendre hidden shift problem (SLS) when classical queries are given, and without quantum RAM. The first one (distinguished table-based collisions) allows to reach an advantage against Grover's algorithm when more data is given. The second one is the offline Kuperberg's algorithm.

Chapter 5. It is based on a joint work with María Naya-Plasencia and André Schrottenloher published at Selected Areas in Cryptography in 2021 [58].

We propose an efficient quantum boomerang attack, as well as a variant for mixing boomerang attacks. We propose several improvements for different cases, as reducing the quantum RAM need or making $Q2$ attacks work in $Q1$ under certain circumstances. In some cases, our attacks reach a quadratic speedup with respect to classical attacks. This shows that boomerang attacks are also performant cryptanalysis tools for the post-quantum world, that will be needed for correctly determining the best security margins.

Chapter 6. We present a joint work with María Naya-Plasencia, Ritam Bhaumik and André Chailloux that is currently under submission.

It focuses on the Encrypt-Mix-Encrypt construction. We provide the first proposal of a generic way for extending both the key and the state size, with quantum security arguments and significant classical proof. In addition we propose some new kind of superposition attacks on the Encrypt-Mix-Encrypt construction, an original distinguisher matching the bound of our construction, and a method considering simulations for supporting conjectures from proofs.

Chapter 7. It is based on a joint work with Sébastien Canard and Loïc Ferreira that is currently in a review process. We first define a security model that does not rely on unconditional security and allows honest parties and attackers to use superposition messages. In particular, assuming that we are in a quantum world, we provide a discussion on what can be put into superposition by the client, the server and the operator during a honest execution of such *quantum UMTS-AKA* protocol. More precisely, the quantum version of the UMTS-AKA that we describe assumes quantum computations as well as quantum communications (i.e., the messages that are exchanged between parties are in a superposition of quantum states).

We then formally prove that this quantum version of the UMTS-AKA is as secure in this quantum model as it is in the classical setting [4]. As a supplementary contribution, we also exhibit a new attack against the state confidentiality of a mobile user. This attack holds in the quantum but also in the classical setting.

We finally study the security of the underlying primitives that can instantiate the UMTS-AKA, Milenage and TUAK. We show an attack on Milenage as a qPRF, however the context of the UMTS-AKA makes this attack unrealistic and we further prove the security of Milenage based on the security of AES. We also show a reduction from the security of TUAK to the security of Keccak-f.

Publications and Pre-prints

- [1] Paul Frixons, María Naya-Plasencia, and André Schrottenloher. “Quantum Boomerang Attacks and Some Applications.” *International Conference on Selected Areas in Cryptography*. Springer, Cham, 2022.
- [2] Paul Frixons, and André Schrottenloher. “Quantum security of the legendre prf.” *Mathematical Cryptology* 1.2 (2021): 52-69.
- [3] Ritam Bhaumik, André Chailloux, Paul Frixons, and María Naya-Plasencia. “Safely Doubling your Block Ciphers for a Post-Quantum World.”
- [4] Sébastien Canard, Loïc Ferreira, and Paul Frixons. “Quantum Security of the UMTS-AKA Protocol and its Primitives, Milenage and TUAK.”

Contents

Introduction	5
Publications and Pre-prints	9
Contents	10
1 Introduction to Quantum Computing	17
1.1 History	17
1.2 Fundamentals of Quantum Computing	18
1.2.1 Qubits	18
1.2.2 Quantum Circuits	19
1.2.3 Notions for Quantum Algorithms	22
1.2.4 Quantum Complexity	25
2 Major Quantum Algorithms	27
2.1 Deutsch-Jozsa Algorithm	28
2.1.1 Premise	28
2.1.2 Description of Algorithm 2.1	28
2.2 Grover's Search	29
2.2.1 Premise	29
2.2.2 Classical Search	29
2.2.3 Description of Grover's algorithm	29
2.3 Amplitude Amplification	30
2.4 BHT Algorithm	31
2.4.1 Premise	31
2.4.2 Description	31
2.5 Shor's Algorithm	32
2.5.1 Reducing Factorization to Period Finding	32
2.5.2 Shor's Period Finding Algorithm	32
2.5.3 Reducing Discrete Logarithm to Period Finding	33
2.6 Simon's Algorithm	33
2.6.1 Simon's Problem	34
2.6.2 Description of the Algorithm	34
2.6.3 Weakening Simon's Premise	35

2.6.4	Simon’s algorithm as a quantum circuit	36
2.7	Kuperberg’s Algorithm	36
2.7.1	Kuperberg’s Premise	36
2.7.2	Description of the Algorithm	37
3	Introduction to Cryptography	41
3.1	Overview	42
3.1.1	Kerckhoffs’s Principles	42
3.1.2	Asymmetric Cryptography	42
3.1.3	Symmetric Cryptography	43
3.2	Elements of Symmetric Cryptography	44
3.2.1	Block Ciphers	44
3.2.2	Stream Ciphers	45
3.2.3	Hash Functions	46
3.2.4	MACs	46
3.2.5	AEAD	47
3.2.6	Permutation-Based Cryptography	47
3.3	Preliminaries of Cryptanalysis	47
3.3.1	(Quantum) Security Models	48
3.3.2	Differential Cryptanalysis	49
3.3.3	Linear Cryptanalysis	49
3.4	Some Quantum Attacks	49
3.4.1	Distinguisher on One-Time Pad	50
3.4.2	3-round Feistel Network	51
3.4.3	The Even-Mansour Cipher	52
3.4.4	FX Construction	52
3.4.5	Q1 attack on Even-Mansour cipher	53
3.4.6	Other Simon-based Attacks	54
3.5	Protocols	55
4	Quantum Security of the Legendre PRF	59
4.1	The Legendre PRF and Context	60
4.2	Table-based Attacks	62
4.2.1	Classical Algorithm	62
4.2.2	Quantum Search with Early Abort	63
4.2.3	Distinguished Collisions	64
4.3	A Reversible Version of Kuperberg’s Algorithm	66
4.3.1	The Decisional Hidden Shift Problem	66
4.3.2	Kuperberg’s Algorithm Made Reversible	66
4.3.3	Label States and Label Qubits	67
4.3.4	Combining Two Labels Reversibly	68
4.3.5	Combining All Labels	69
4.3.6	Behavior in the Non-shifted Case	71
4.3.7	Bounding the Errors	71

4.4	Quantum Search with an Approximate Test	72
4.5	The Offline-DAHS Algorithm	75
4.5.1	High-level Description	75
4.5.2	Approximate Promise	76
4.5.3	Application to the Legendre Symbol	77
4.6	Detailed Complexities	78
4.7	Conclusion	78
5	Quantum Boomerang Attacks and Some Applications	81
5.1	Classical Boomerang Attacks	82
5.1.1	The Classical Boomerang Attack	83
5.1.2	Mixing Boomerang Attacks	86
5.2	Quantum Boomerang Attacks	88
5.2.1	Quantum Boomerang Distinguisher	88
5.2.2	Quantum Boomerang Last-rounds Attack	88
5.2.3	Quantum Mixing Boomerang Distinguisher	91
5.3	Application to SAFER	92
5.3.1	Description of the Cipher	92
5.3.2	5-Round Classical Boomerang Attack	93
5.3.3	Quantum Boomerang Attack	96
5.4	Application to KASUMI	98
5.4.1	Description of the Cipher	98
5.4.2	Classical Attack	98
5.4.3	Quantum Attack	99
5.5	Application to Related-key AES	100
5.5.1	Description of AES	100
5.5.2	Classical Attack	102
5.5.3	Quantum Attack	103
5.6	Application to 5-Round AES	108
5.6.1	The boomerang	108
5.6.2	Friend Pairs	110
5.6.3	Value-difference Correspondence on AES S-box	111
5.6.4	Classical Attack	111
5.6.5	Quantum Attack	113
5.7	Conclusion	113
6	Safely Doubling your Block Ciphers for a Post-Quantum World	115
6.1	Introduction	115
6.2	General Related Concepts	117
6.2.1	Mirror Theory	117
6.2.2	H-Coefficient Technique	119
6.3	Applying Simon's Algorithm to Functions Restricted on a Subset	120

6.4	Constructions Based on Encrypt-Mix-Encrypt Paradigm: First Attempt and Attack	121
6.4.1	New Superposition Attack on the EME Construction	122
6.5	New Construction and Classical Security Proofs	125
6.5.1	Proof of Classical Security	126
6.5.2	IND-CPA Security Proof of n -bit Security using Mirror Theory	128
6.5.3	IND-CCA Security Proof of n -bit Security using Mirror Theory	131
6.5.4	IND-CPA Security Proof of $(2n/3)$ -bit Security	133
6.5.5	IND-CCA Security Proof of $(2n/3)$ -bit Security	135
6.6	Simulation of the Mirror Theory	138
6.7	An $O(2^n)$ -Distinguisher on our Scheme	141
6.8	Quantum Security	143
6.8.1	Hardness of Distinguishing a Random Permutation from a Random Function with Small Range	144
6.8.2	Quantum Security Statement	144
6.8.3	Discussion	147
6.9	Proposing a Concrete Instance: Double-AES	148
6.9.1	How to Extend the Keys?	148
6.9.2	Introducing a Domain in AES for Defining E_j	149
6.9.3	Double-AES Concrete Proposals	149
6.9.4	Security Claims	150
6.9.5	Discussion	150
6.10	Best Attacks Found of Double-AES	151
6.10.1	Attack on X-3-3 Without Last MC in Blocks E_1, E_2 and E	151
6.10.2	Attack on X-2-X Without Last MC in Blocks E_1, E_2 and E	153
6.10.3	Best known attacks on AES	154
6.11	Conclusion	154
7	Quantum Security of UMTS-AKA	157
7.1	The AKA Protocol	158
7.1.1	The Classic Version of AKA	158
7.1.2	Quantum AKA Protocol	162
7.2	Adversarial Model and Building Blocks	165
7.2.1	Oracles	166
7.2.2	Targeted Notions	168
7.2.3	Pseudo-random Function	169
7.2.4	Pseudo-random Key-derivation	170
7.3	Quantum Security of the UMTS-AKA Protocol	170
7.3.1	Session Keys Indistinguishability	170
7.3.2	Client-impersonation Resistance	174

7.3.3	Weak Server-impersonation Resistance	176
7.3.4	Soundness	179
7.3.5	State Confidentiality	181
7.4	Quantum Security of Milenage	184
7.4.1	Description of Milenage	184
7.4.2	Attack on Milenage as a Pseudo-Random Function . . .	184
7.4.3	Security Proof of Milenage in AKA	185
7.5	Quantum Security of TUAKE	190
7.5.1	Description of TUAKE	190
7.5.2	Security Arguments of TUAKE	191
7.6	Conclusion	191
8	Conclusion	195
8.1	Summary of Results	195
8.2	Open Problems	196
	Bibliography	199

Notations

Symbol	Description
\mathbb{Z}	set of integers
\mathbb{R}	set of real numbers
\mathbb{R}^+	set of non-negative real numbers
\mathbb{C}	set of complex numbers
\bar{z}	complex conjugate of z
\oplus	bit-wise xor
$+$	integer addition
\cdot	inner product of vectors
$ A $	cardinal of the set A
$[b]$	the boolean value of b (1 if b is true, 0 otherwise)
\lll	left shift
val_2	2-adic valuation
\parallel	concatenation
rev	reversion

Chapter 1

Introduction to Quantum Computing

This chapter presents the tools of quantum computing used in the following chapters. It starts with a quick history of the domain and continues with the elementary notions of quantum computing: qubits, superposition, entanglement, operators and quantum complexity.

Contents

1.1	History	17
1.2	Fundamentals of Quantum Computing	18
1.2.1	Qubits	18
1.2.2	Quantum Circuits	19
1.2.3	Notions for Quantum Algorithms	22
1.2.4	Quantum Complexity	25

1.1 History

Quantum computing, as the name suggests, is based on quantum physics. The original notion is attributed to Paul Benioff and Richard Feynman, the former for showing in 1980 that a computer could operate with quantum mechanics and proposing a model for a quantum computer and the latter for arguing about the difficulty for a classical computer to simulate quantum mechanics in 1981.

During the following years, quantum mechanics have been studied considering different applications, like unforgeable materials and quantum key distribution, applying its unique properties, namely the “no-cloning” theorem, stipulating that one cannot clone a qubit.

Interest for quantum computer grew when it was shown to outperform its classical counterpart in certain cases. It started with the Deutsch–Jozsa algorithm, published in 1992, which states whether a function $\{0, 1\}^n$ to $\{0, 1\}$ is constant or equilibrated ($|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$) in only one query, whereas a classical computer needs $2^{n-1} + 1$ to achieve certainty. However, a small number of queries are sufficient to be almost sure and thus the community was not completely convinced until Peter Shor developed in 1994 a quantum

algorithm for factorizing large numbers exponentially faster than the best known classical algorithms. In 1996, Lov Grover developed an efficient quantum database search, providing quantum computers a general advantage over its classical counterpart and further emphasizing the interest for such a technology.

The realization of such quantum computers began in 1998 with a working 2-qubit computer implementing Deutsch–Jozsa algorithm. It was followed by the first factorization using Shor’s algorithm in 2001 by IBM with a 7-qubit machine. Another important step has been the claim by Google to have achieved quantum supremacy (having a quantum computer solving a problem that would be “irrealizable” for any existing classical computer) in 2019 and IBM promising a 1121-qubit computer by 2023 and a million-qubit computer by 2030 [70, 69, 61].

1.2 Fundamentals of Quantum Computing

In this section, we present the fundamentals of quantum computing following the lecture notes of de Wolf [119].

1.2.1 Qubits

In order to define the qubits and operations on them, we first remind the notion of Hilbert spaces.

Definition 1.1 (Hilbert Spaces). *A Hilbert space \mathcal{H} is a vector space over \mathbb{C} with an inner product $\langle \cdot | \cdot \rangle$ i.e., a function from \mathcal{H}^2 to \mathbb{C} that verifies the following properties:*

- **Conjugate Symmetric.** *For all $x, y \in \mathcal{H}$, $\langle y | x \rangle = \overline{\langle x | y \rangle}$.*
- **Bilinearity.** *For all $x, y, z \in \mathcal{H}$, $\mu, \lambda \in \mathbb{C}$, $\langle \lambda x + \mu y | z \rangle = \lambda \langle x | z \rangle + \mu \langle y | z \rangle$.*
- **Definite Positivity.** *For all $x \in \mathcal{H}$, $\langle x | x \rangle \in \mathbb{R}^+$ and $\langle x | x \rangle = 0 \Leftrightarrow x = 0$.*

We also note $\|x\| = \sqrt{\langle x | x \rangle}$ the norm induced by the inner product.

Dirac Notation. For further use, the spaces we consider will be of the form \mathbb{C}^N . The elements are noted as $\sum_{i=1}^N x_i |i\rangle = (x_1, x_2, \dots, x_N)$. We use the canonical inner product $\left\langle \sum_{i=1}^N x_i |i\rangle \left| \sum_{i=1}^N y_i |i\rangle \right. \right\rangle = \sum_{i=1}^N x_i \bar{y}_i$ (the $|i\rangle$ form an orthonormal basis). x_i is called the amplitude associated to the state $|i\rangle$.

We can define the qubit, quantum equivalent of the classical bit.

Definition 1.2 (Qubit). *A qubit is an element of \mathbb{C}^2 with norm 1 i.e., an element of the form $a|0\rangle + b|1\rangle$ with $|a|^2 + |b|^2 = 1$.*


Concatenation and Entanglement of Qubits. A qubit takes greater complexity when considered as an ensemble. A system of n qubits is a element of \mathbb{C}^{2^n} with norm 1. A concatenation of systems makes every possible state of the two subsystems appear with its own amplitude:

$$\left(\sum_{i \in \{0,1\}^n} a_i |i\rangle \right) \otimes \left(\sum_{j \in \{0,1\}^n} b_j |j\rangle \right) = \sum_{i,j \in \{0,1\}^n} a_i b_j |i\rangle \otimes |j\rangle = \sum_{(i,j) \in \{0,1\}^{2n}} a_i b_j |i\rangle \otimes |j\rangle$$

There are many possible notations for a concatenation of classical states: $|i\rangle \otimes |j\rangle = |i\rangle |j\rangle = |i, j\rangle$.

A system that cannot be seen as two subsystems concatenated is said to be entangled. An example of entanglement is $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Definition 1.3 (Measuring a System of Qubits). *When measured, a superposition $\sum_{i \in \{0,1\}^n} a_i |i\rangle$ has a probability of $|a_i|^2$ of collapsing to $|i\rangle$. Contrary to a superposition, the result of a measurement can be read.*

In circuits, it will be symbolized by .

Definition 1.4 (Partial Measurement). *A superposition $\sum_{i,j \in \{0,1\}^n} a_{i,j} |i, j\rangle$, when measured only on the first group of qubits, has a probability of $\sum_{j \in \{0,1\}^n} |a_{i,j}|^2$ of collapsing to $|i\rangle$. The rest of the superposition becomes $\sum_{j \in \{0,1\}^n} \frac{a_{i,j}}{\sqrt{\sum_{j \in \{0,1\}^n} |a_{i,j}|^2}} |j\rangle$.*

Other Measurements There are more general notions of measurement we do not cover in this introduction, like positive operator-valued measures (POVMs). However they are equivalent up to the use of additional qubits.

1.2.2 Quantum Circuits

In this subsection, we expose the general operations on qubits and the most commonly used ones.

1.2.2.1 Definitions

Definition 1.5 (Unitary Operator). *A unitary operator of a Hilbert space is a linear application that preserves the norm.*

Definition 1.6 (Quantum Operator). *A quantum operator on n qubits is a unitary operator on \mathbb{C}^{2^n} .*

Remark. When using quantum objects, the operators considered must be linear (maintain the amplitude) and unitary (the input and the outputs are qubits)

Definition 1.7 (Inverse Operator). *As any quantum operator \mathcal{O} is unitary, it has an inverse noted \mathcal{O}^\dagger , applying this operator is called “uncomputing”.*

Matrix Notation. Quantum operators on n qubits are unitary operators on \mathbb{C}^{2^n} . They can be noted as matrices. With the Dirac notation, it corresponds to $f = \sum_{i,j \in \{0,1\}^n} a_{i,j} |i\rangle \langle j|$ with for all k in $\{0,1\}^n$, $(|i\rangle \langle j|) |k\rangle = |i\rangle$ if $k = j$ and 0 otherwise.

An “elementary” operator (often acts on a small number of qubits) is called a gate, in analogy to classical logic gates (NOT, AND, OR).

1.2.2.2 Common Gates

We describe the most common gates and some constructions. We start by the one-qubit operators.

Definition 1.8 (One-qubit Gates).

- X (or *NOT*) implements the classical “not”, it maps $|0\rangle$ to $|1\rangle$ and conversely.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \left| \quad |b\rangle \text{ — } \boxed{X} \text{ — } |b \oplus 1\rangle \right.$$

- Z flips the amplitude of $|1\rangle$ and leaves $|0\rangle$ intact.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \left| \quad |b\rangle \text{ — } \boxed{Z} \text{ — } (-1)^b |b\rangle \right.$$

- H maps $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. It is called the Hadamard gate.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \left| \quad |b\rangle \text{ — } \boxed{H} \text{ — } \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b |1\rangle) \right.$$

- R_θ leaves $|0\rangle$ intact and rotates the amplitude of $|1\rangle$. $R_{\pi/4}$ is named T .

$$R_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \quad \left| \quad |b\rangle \text{ — } \boxed{R_\theta} \text{ — } e^{ib\theta} |b\rangle \right.$$

We now describe some useful operators on more qubits.

Definition 1.9 (Composition of operators). *Let U_1 and U_2 be two operators acting on n qubits, $U_2 \circ U_1$ acts on n qubits and $(U_2 \circ U_1) |x\rangle = U_2(U_1 |x\rangle)$.*

$$\underbrace{U_2 \circ U_1}_{\text{as operators}} = \underbrace{U_2 \times U_1}_{\text{as matrices}} \quad \left| \quad |x\rangle \text{ — } \boxed{U_1} \text{ — } \boxed{U_2} \text{ — } U_2 \circ U_1 |x\rangle \right.$$

We also define I the identity operator, i.e. the “do-nothing” operator.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \left| \quad |x\rangle \text{ — } \text{—————} |x\rangle \right.$$

Definition 1.10 (Tensor product of operators). Let U_1 and U_2 be two operators acting on n and m qubits respectively, $U_1 \otimes U_2$ acts on $n + m$ qubits and $(U_1 \otimes U_2)(|x\rangle \otimes |y\rangle) = (U_1 |x\rangle) \otimes (U_2 |y\rangle)$. This definition can be extended to any state by linearity.

We also define $U^{\otimes n} = \underbrace{U \otimes U \otimes \dots \otimes U}_{n \text{ times}}$.

Definition 1.11 (Controlled operator). Let U be an operator acting on n qubits, the controlled operator CU acts on $n + 1$ qubits and

$$CU(|b\rangle \otimes |x\rangle) = \begin{cases} |0\rangle \otimes |x\rangle & \text{if } b=0 \\ |1\rangle \otimes (U|x\rangle) & \text{if } b=1 \end{cases}$$

$$CU = \left[\begin{array}{c|c} Id & 0 \\ \hline 0 & U \end{array} \right] \left| \begin{array}{c} |b\rangle \text{---} \bullet \text{---} |b\rangle \\ |x\rangle \text{---} \boxed{U} \text{---} U^b|x\rangle \end{array} \right.$$

The most common ones are $CNOT$, implementing the xor, and $CCNOT$, also known as Toffoli gate.

$$CNOT = \left[\begin{array}{c|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array} \right] \left| \begin{array}{c} |x\rangle \text{---} \bullet \text{---} |x\rangle \\ |y\rangle \text{---} \oplus \text{---} |x \oplus y\rangle \end{array} \right.$$

$$CCNOT = \left[\begin{array}{c|c} I_4 & 0 \\ \hline 0 & CNOT \end{array} \right] \left| \begin{array}{c} |x\rangle \text{---} \bullet \text{---} |x\rangle \\ |y\rangle \text{---} \bullet \text{---} |y\rangle \\ |z\rangle \text{---} \oplus \text{---} |z \oplus (x \wedge y)\rangle \end{array} \right.$$

Definition 1.12 (Exchange gate). A last common gate (which can be built from $CNOT$) is $SWAP$, exchanging two qubits.

$$SWAP = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right] \left| \begin{array}{c} |x\rangle \text{---} \times \text{---} |y\rangle \\ |y\rangle \text{---} \times \text{---} |x\rangle \end{array} \right.$$

Definition 1.13 (Quantum Fourier Transform). The Quantum Fourier Transform (QFT) acts on n qubits and maps $|x\rangle$ to $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \chi_{2^n}(xy) |y\rangle$ where $\chi_{2^n}(x) = e^{\frac{2ix\pi}{2^n}}$.

$$QFT = \left[\begin{array}{cccc} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{2^n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^n-1} & \dots & \omega^{(2^n-1)^2} \end{array} \right] \left| \begin{array}{c} |x\rangle \text{---} \boxed{QFT} \text{---} \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \chi_{2^n}(xy) |y\rangle \end{array} \right.$$

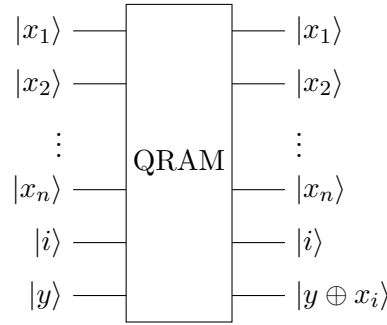


Figure 1.1: QRAM gate

The implementation of such circuit can be done efficiently by using the following formulation :

$$\begin{aligned}
 QFT |x\rangle &= \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \chi_{2^n}(xy) |y\rangle \\
 &= \frac{1}{2^{n/2}} \sum_{y_1, \dots, y_n \in \{0,1\}} \chi_{2^n}(x(\sum_{j=1}^n y_j 2^{n-j})) |y_1, \dots, y_n\rangle \\
 &= \frac{1}{2^{n/2}} \sum_{y_1, \dots, y_n \in \{0,1\}} \prod_{j=1}^n e^{2\pi i x y_j / 2^j} |y_1, \dots, y_n\rangle \\
 &= \bigotimes_{j=1}^n \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i x / 2^j} |1\rangle)
 \end{aligned}$$

There also exists QFT for other integers than 2^n .

1.2.3 Notions for Quantum Algorithms

In this subsection we detail some generalities of quantum computing.

1.2.3.1 Ancilla Qubits

This is the equivalent of the classical memory: to compute a function, there is often a need for extra room for operations. For operators U_1 and U_2 acting on n and $n + m$ qubits respectively, we say that U_1 and U_2 encode the same operator if and only if for all n -qubit state $|x\rangle$, $U_2(|x\rangle \otimes |0\rangle) = (U_1 |x\rangle) \otimes |0\rangle$. The extra $|0\rangle$ qubits are called ancilla qubits.

1.2.3.2 QRAM (Quantum Random Access Memory)

Speaking of quantum memory, there is also a need for a quantum equivalent of a table access. This QRAM gate allows to manage a quantum database by allowing to store elements on quantum memory and to quickly access them and even to make superposition of them (see [5]). Depending on the nature of the $|x_i\rangle$ (classical or superposition), the estimated cost of the QRAM changes. Thus, there is a distinction (as shown in Figure 1.2) between QRACM (Quantum Random Access to Classical Memory) and QRAQM (Quantum Random Access to Quantum Memory).

However an efficient implementation of QRAM is still unknown, making it an assumption that can be discarded in regards to practicality.

	index i	classical	superposition
elements $ x_i\rangle$			
classical		classical RAM	QRACM
superposition		quantum circuits	QRAQM

Figure 1.2: Quantum memory types

1.2.3.3 Equivalence of Oracles

With classical computing, the output of a function is always in a set of bits, whether it is a new set of bits (out-of-place computation) or rewriting on the same set of bits (in-place computation). With quantum computing, there is also the possibility to write on the amplitudes of the system.

For a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, the standard oracle O_f and the phase oracle PO_f act on $n + m$ qubits with $O_f |x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ and $PO_f |x, y\rangle \mapsto (-1)^{f(x) \cdot y} |x, y\rangle$ respectively.

For a permutation, $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, the in-place oracle IO_f acts on n qubits with $IO_f : |x\rangle \mapsto |f(x)\rangle$. (This formulation implies the access to $IO_{f^{-1}} = IO_f^\dagger$ and is not common in the literature.)

These function formulations are equivalent as $PO_f = (I \otimes H^{\otimes m}) \circ O_f \circ (I \otimes H^{\otimes m})$. (This can be seen as a generalization of the identity $H \circ X \circ H = Z$.)

For permutations, we suppose a formulation for f and f^{-1} . Then $O_{f^{-1}} \circ SWAP \circ O_f$ encodes the same operator as IO_f and $O_f = (IO_{f^{-1}} \otimes I) \circ CNOT \circ (IO_f \otimes I)$.

1.2.3.4 No Cloning Theorem

We now present some limitations to the power of quantum computing.

Theorem 1.1 (No Cloning Theorem [120]). *There exist no operator U and state $|\gamma\rangle$ such that for every one-qubit state $|\phi\rangle$, $U(|\phi\rangle |0\rangle |\gamma\rangle)$ is of the form $|\phi\rangle |\phi\rangle |\gamma_\phi\rangle$.*

Proof. Let U be a cloning operator and $|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$ (with $|\alpha|^2 + |\beta|^2 = 1$). Then by linearity,

$$\begin{aligned}
U(|\phi\rangle |0\rangle |\gamma\rangle) &= \alpha U(|0\rangle |0\rangle |\gamma\rangle) + \beta U(|1\rangle |0\rangle |\gamma\rangle) \\
|\phi\rangle |\phi\rangle |\gamma_\phi\rangle &= \alpha |0\rangle |0\rangle |\gamma_0\rangle + \beta |1\rangle |1\rangle |\gamma_1\rangle \\
\alpha^2 |0\rangle |0\rangle |\gamma_\phi\rangle + \alpha\beta |0\rangle |1\rangle |\gamma_\phi\rangle \\
+ \alpha\beta |1\rangle |0\rangle |\gamma_\phi\rangle + \beta^2 |1\rangle |1\rangle |\gamma_\phi\rangle &= \alpha |0\rangle |0\rangle |\gamma_0\rangle + \beta |1\rangle |1\rangle |\gamma_1\rangle
\end{aligned}$$

This implies that $\alpha\beta = 0$ as the right term contains no element starting with $|0\rangle |1\rangle$ or $|1\rangle |0\rangle$. It means that if U can clone the classical states $|0\rangle$ and $|1\rangle$, it can only clone the classical states. \square

Remark.

- While this statement only concerns one-qubit states, cloning states on any fixed size implies cloning one-qubit states.
- While a cloning operator does not exist, it mostly means that we cannot clone unknown qubits. Yet, we can use an operator on multiple sets of qubits to get many “clones” of the same state.
- There also exists approximate cloning methods that uses the unknown qubit and produces two qubits closer to the original one than random.

1.2.3.5 No Deleting Theorem

Theorem 1.2 (No Deleting Theorem [82]). *There exist no operator U and state $|\gamma\rangle$ such that for every one-qubit state $|\phi\rangle$, $U(|\phi\rangle|\phi\rangle|\gamma\rangle)$ is of the form $|\phi\rangle|0\rangle|\gamma_\phi\rangle$.*

Proof. Let U be a deleting operator and $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ (with $|\alpha|^2 + |\beta|^2 = 1$). Then by linearity,

$$\begin{aligned} U(|\phi\rangle|\phi\rangle|\gamma\rangle) &= \alpha^2 U(|0\rangle|0\rangle|\gamma\rangle) + \alpha\beta U(|0\rangle|1\rangle|\gamma\rangle) \\ &\quad + \alpha\beta U(|1\rangle|0\rangle|\gamma\rangle) + \beta^2 U(|1\rangle|1\rangle|\gamma\rangle) \\ \alpha|0\rangle|0\rangle|\gamma_\phi\rangle + \beta|1\rangle|0\rangle|\gamma_\phi\rangle &= \alpha^2 U(|0\rangle|0\rangle|\gamma\rangle) + \beta^2 U(|1\rangle|1\rangle|\gamma\rangle) \\ &\quad + \underbrace{\sqrt{2}\alpha\beta \left(\frac{1}{\sqrt{2}} U(|0\rangle|1\rangle|\gamma\rangle) + \frac{1}{\sqrt{2}} U(|1\rangle|0\rangle|\gamma\rangle) \right)}_{|\Phi\rangle} \end{aligned}$$

Then by projecting on $|0\rangle|0\rangle|\gamma_0\rangle$ and $|1\rangle|0\rangle|\gamma_1\rangle$, there is $|\epsilon\rangle$ orthogonal to $|\gamma_0\rangle$ and $|\gamma_1\rangle$ such that:

$$\begin{aligned} |\gamma_\phi\rangle &= \alpha|\gamma_0\rangle + \beta|\gamma_1\rangle + |\epsilon\rangle \\ |\Phi\rangle &= \frac{1}{\sqrt{2}}(|0\rangle|0\rangle|\gamma_1\rangle + |1\rangle|0\rangle|\gamma_0\rangle) + \alpha|0\rangle|0\rangle|\epsilon\rangle + \beta|1\rangle|0\rangle|\epsilon\rangle \end{aligned}$$

Then by considering the norm on the second equation, we get that $|\epsilon\rangle = 0$. Then the right term of the first equation has always norm 1, meaning that $|\gamma_0\rangle$ and $|\gamma_1\rangle$ constitute a base. This implies that the information of $|\phi\rangle$ is still in $|\gamma_\phi\rangle$. \square

Remark.

- Like the cloning theorem, deleting larger states means being able to do so on one-qubits states.
- In the same vein, it mostly means that we cannot delete unknown qubits. Yet, the inverse of operators can delete its corresponding state.

1.2.4 Quantum Complexity

Like in many domains, the conception of algorithms in cryptography has a heavy emphasis on efficiency. While the metrics of classical algorithms are well-known (time, data and memory complexity), there are many formulations for quantum circuits.

Here are the main ones:

Size. We count the number of elementary gates used (for presented circuits X , Z , H , R_θ , $CNOT$, Toffoli). This is our main concern in the rest of this thesis as it is equivalent to classical time complexity.

Width. We count the number of qubits used (input, output and ancillas). This is the equivalent to memory complexity. It seconds the size measurement.

Depth. This is the highest number of gates involving a single fixed qubit. This is an important measure for implementations of quantum circuits, however we will not consider it in this thesis as implementation is not our primary goal.

Query complexity. When using an external function, we count the number of O_f gates as query complexity.

With those notions in mind, we can expose the quantum algorithms that are fundamental to this thesis: Grover's search and its generalization, the amplitude amplification, BHT collision search, Shor's algorithm, Simon's period finding and Kuperberg's algorithm.

Chapter 2

Major Quantum Algorithms

We expose the major algorithmic building blocks that pave our work: Grover's search and its generalization, the amplitude amplification algorithm, BHT collision search, Shor's algorithm, Simon's period finding algorithm and some of its intricacies and Kuperberg's algorithm for which we provide some deepening in Chapter 4.

Contents

2.1	Deutsch-Jozsa Algorithm	28
2.1.1	Premise	28
2.1.2	Description of Algorithm 2.1	28
2.2	Grover's Search	29
2.2.1	Premise	29
2.2.2	Classical Search	29
2.2.3	Description of Grover's algorithm	29
2.3	Amplitude Amplification	30
2.4	BHT Algorithm	31
2.4.1	Premise	31
2.4.2	Description	31
2.5	Shor's Algorithm	32
2.5.1	Reducing Factorization to Period Finding	32
2.5.2	Shor's Period Finding Algorithm	32
2.5.3	Reducing Discrete Logarithm to Period Finding	33
2.6	Simon's Algorithm	33
2.6.1	Simon's Problem	34
2.6.2	Description of the Algorithm	34
2.6.3	Weakening Simon's Premise	35
2.6.4	Simon's algorithm as a quantum circuit	36
2.7	Kuperberg's Algorithm	36
2.7.1	Kuperberg's Premise	36
2.7.2	Description of the Algorithm	37

2.1 Deutsch-Jozsa Algorithm

We start the algorithm presentation by the oldest one, the Deutsch-Jozsa algorithm [45]. It was first described in 1992. While it did not change the view of the community on quantum computing, it is the first quantum algorithm that solves a problem faster than its classical counterpart.

2.1.1 Premise

We introduce the problem solved by the Deutsch-Jozsa algorithm.

Problem 2.1. *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a function that is either constant ($f = 0$ or $f = 1$) or balanced ($|f^{-1}(0)| = |f^{-1}(1)|$). Distinguish between cases.*

2.1.2 Description of Algorithm 2.1

Algorithm 2.1 Description of the Deutsch-Jozsa algorithm

Input: *superposition* oracle access to f , either constant or balanced

Output: “Constant” or “Balanced”

- 1: Start with the state $|0^n\rangle |1\rangle$
 - 2: Apply the Hadamard gate on all qubits, obtaining the state $\frac{1}{2^{(n+1)/2}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes (|0\rangle - |1\rangle)$
 - 3: Apply the oracle $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$ to the state, obtaining $\sum_{x \in \{0,1\}^n} \frac{1}{2^{(n+1)/2}} |x\rangle \otimes (|f(x)\rangle - |f(x) \oplus 1\rangle)$
 - 4: Rewrite it as $\frac{1}{2^{(n+1)/2}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes (|0\rangle - |1\rangle)$
 - 5: Apply the Hadamard gate on all qubits, obtaining the state $\frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} |y\rangle (-1)^{f(x)+x \cdot y} \otimes |1\rangle$
if f is constant, **then** it simplifies to $|0^n\rangle \otimes |1\rangle$.
if f is balanced, **then** the amplitude of the state $|0^n\rangle \otimes |1\rangle$ is 0.
 - 6: Measure and check if the result is $0^n || 1$
 - 7: **if** it is **return** “Constant” **Else return** “Balanced”
-

Remarks. The fact that the Deutsch-Jozsa algorithm needs only one query to the function f has two effects:

- as we will see with other quantum algorithms, a superposition query can do more than any classical query. A classical computer needs at least two queries to differentiate between a constant function and a non-constant one;
- needing only one query means that a property can be found even if the function is never used again. In cryptography, as presented in Chapter 3, many constructions use always-changing values to prevent an adversary to use a trivial property that would be exposed otherwise.

2.2 Grover's Search

We present the Grover's search [65], introduced in 1996 by Lov Grover, which is the quantum equivalent of the exhaustive search that exists in classical algorithms.

2.2.1 Premise

Problem 2.2. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a function. We call elements x such that $f(x) = 1$ "good" and others "bad". We note Y the set of "good" elements and X the search space, here $\{0, 1\}^n$. Our goal is to find a "good" element.

2.2.2 Classical Search

With classical means, the best we can do (without further specification of the function f) is to take an element x at random and check if x is "good" and retry with a different element if it is not. The probability of success is $\frac{|Y|}{|X|}$ and we expect to try about $\frac{|X|}{|Y|}$ times.

2.2.3 Description of Grover's algorithm

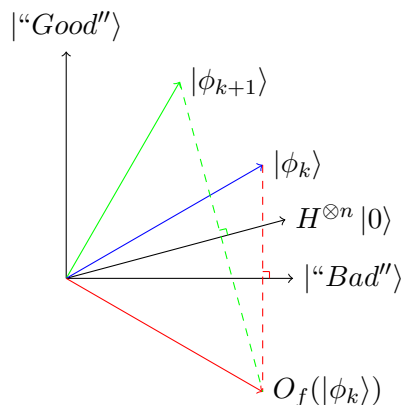


Figure 2.1: Representation of a round of Grover's search routine.

Grover's search is the composition of rounds of a routine applied on the uniform superposition $\sum_{x \in X} \sqrt{\frac{1}{|X|}} |x\rangle$.

The routine is composed of two parts:

- The application of the phase change $\mathcal{O}_f : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$. It can be seen as the symmetry around the state $|'Bad''\rangle = \sum_{x \in X \setminus Y} \sqrt{\frac{1}{|X| - |Y|}} |x\rangle$. This corresponds to the red symmetry in Figure 2.1.

- The application of $H^{\otimes n} \circ \mathcal{O}_0 \circ H^{\otimes n}$ ($\mathcal{O}_0 : |x\rangle \mapsto (-1)^{[x=0]} |x\rangle$). As \mathcal{O}_0 is the symmetry around the state $|0\rangle$ and $H^{\otimes n}$ a base change, $H^{\otimes n} \circ \mathcal{O}_0 \circ H^{\otimes n}$ is the symmetry around the state $H^{\otimes n} |0\rangle = \sum_{x \in X} \sqrt{\frac{1}{|X|}} |x\rangle$. This is represented by the green symmetry in Figure 2.1.

The situation can be analyzed in the plane formed by $|“Good”\rangle = \sqrt{\frac{1}{|X|}} \sum_{x \in Y} |x\rangle$ and $|“Bad”\rangle$. Note that those two states are orthogonal.

Then these two operations form the rotation of angle 2θ where θ is the angle between $|“Bad”\rangle$ and $H^{\otimes n} |0^n\rangle$.

We want to get the closest vector to $|“Good”\rangle$ as our goal is to measure an element in $|“Good”\rangle$. This translates to $2\alpha\theta + 1 \simeq \frac{\pi}{2}$, where α is the number of rounds. This simplifies to $t = \lfloor \frac{\pi}{4\theta} \rfloor$.

We can write $H^{\otimes n} |0^n\rangle$ as $\sqrt{\frac{|X|-|Y|}{|X|}} |“Bad”\rangle + \sqrt{\frac{|Y|}{|X|}} |“Good”\rangle$.

We deduce that $\sin(\theta) = \sqrt{\frac{|Y|}{|X|}}$ and then $\theta = \arcsin(\sqrt{\frac{|Y|}{|X|}})$.

We conclude that the number of rounds (and thus the complexity of the algorithm) is $t = \left\lfloor \frac{\pi}{4 \arcsin(\sqrt{\frac{|Y|}{|X|}})} \right\rfloor \simeq \frac{\pi}{4} \sqrt{\frac{|X|}{|Y|}}$. We add, for further considerations, that the probability of success, i.e. the probability of measuring an element of Y at the end of Grover’s algorithm is at least $\max(1 - |Y|/|X|, |Y|/|X|)$.

Optimization. This algorithm replaces the exhaustive search that is optimal for classical computation and has been proven to be also optimal for quantum computations [124].

2.3 Amplitude Amplification

Grover’s algorithm requires that the search space is of the form $\{0, 1\}^n$. While it is enough to get importance as it is a relatively common search space, it prevents some more advanced techniques from using it.

The amplitude amplification, introduced by Brassard, Hoyer, Mosca and Tapp [31], extends Grover’s algorithm to any search space by replacing $H^{\otimes n}$ by a (reversible) procedure C that generates a uniform superposition of the search space from $|0^n\rangle$.

Technically, the starting point becomes $C |0^n\rangle$ and the routine $C \circ \mathcal{O}_0 \circ C^\dagger \circ \mathcal{O}_f$.

Then the calculations made for Grover’s search also hold for the amplitude amplification.

As mentioned earlier, this algorithm is a building block for the following chapters and it will appear as $\text{Amplify}(\mathcal{C}, f)$.

2.4 BHT Algorithm

Many cryptographic constructions build their security upon the difficulty to get a collision. The main quantum algorithms to find collisions are Ambainis' algorithm [5] and BHT algorithm, which we detail below.

2.4.1 Premise

Problem 2.3. *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a function ($m < 2n$ to have a good probability that a collision exists). Our goal is to search for $x, y \in \{0, 1\}^n$ such that $x \neq y$ and $f(x) = f(y)$.*

Remark. There is also the problem to find a claw $f(x) = g(y)$ for f and g two functions from $\{0, 1\}^n$ to $\{0, 1\}^m$. However, it can be reduced (with good probability) to the above problem by the function $h : (b, x) \mapsto f(x) \oplus b(f(x) \oplus g(x))$.

Classical algorithm. For classical computation, and an output set of size n , we need $\mathcal{O}(\sqrt{n})$ queries (the birthday paradox bound) for a collision to appear. The known way to do so is the Pollard's rho algorithm [101].

2.4.2 Description

Brassard et al. proposed in 1998 [32] an approach to use Grover's search with the birthday paradox:

1. start by querying the function f $2^{m/3}$ times and list the obtained values (and their inputs);
2. sort those values and build a quantum circuit that searches by dichotomy if a new element is already in this list;
3. use Grover's algorithm to search an element such that the output by f is in the list.

The searched elements have a probability $2^{-2m/3}$ to pass the test and then the Grover's search takes roughly $2^{m/3}$ computations to return a right element. The total complexity of the BHT algorithm is about $\mathcal{O}(2^{m/3})$ computations and needs a QRAM of the same size.

Full quantum circuit. It is possible to make BHT a fully quantum algorithm by using a quantum list of superposed elements instead of a classical one (sorting the list can be done with ancillary qubits remembering the permutations). This has the advantage to return the uniform superposition of the collisions instead of a simple collision pair.

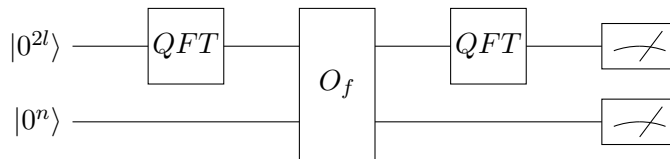


Figure 2.2: Shor’s period finding algorithm

Chailloux, Naya-Plasencia and Schrottenloher [34] managed to build a version of this algorithm without the need for QRAM and negligible memory use with time complexity $\mathcal{O}\left(2^{2m/5}\right)$.

2.5 Shor’s Algorithm

We present the (in)famous Shor’s algorithm [110] that efficiently computes factoring (breaking the RSA scheme) and discrete logarithm (breaking the DH, ECDH, ECDSA schemes). A realization of this algorithm for the concerned sizes (256-bit for elliptic curve variants and 4096-bit for finite field variants) would mean the end of the security properties for almost 90% of communications over the Internet [107, 114].

2.5.1 Reducing Factorization to Period Finding

First, we fix the number to factorize N and a “seed” x . Then, we consider the function $n \mapsto x^n \bmod N$. This function has a smallest period $r < N$. Considering that N is odd and not a power of a prime (which cases are easy to reduce), then with probability greater than $\frac{1}{2}$, r is even and $(x^{r/2} + 1)(x^{r/2} - 1) \equiv 0 \pmod N$ is not a trivial zero divisor. It follows that $\gcd(x^{r/2} + 1, N)$ and $\gcd(x^{r/2} - 1, N)$ are two non-trivial factors of N .

2.5.2 Shor’s Period Finding Algorithm

We detail the core of Shor’s algorithm (Figure 2.2).

2.5.2.1 Premise for Shor’s algorithm

Shor’s algorithm solves the following problem.

Problem 2.4. *Let $f : \mathbb{Z} \rightarrow \{0, 1\}^n$ be a periodic function with a period $r < 2^l$. Our goal is to recover r .*

2.5.2.2 Description of the Algorithm

We consider a quantum implementation of f , O_f acting on $2l + n$ qubits with $O_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$.

The algorithm starts with the state

$$|0^{2l}\rangle |0^n\rangle.$$

We apply a *QFT* gate on the first register to get the state

$$\frac{1}{2^l} \sum_{x=0}^{2^{2l}-1} |x\rangle |0^n\rangle.$$

Then we apply O_f , leading to the state

$$\frac{1}{2^l} \sum_{x=0}^{2^{2l}-1} |x\rangle |f(x)\rangle.$$

We measure the second register and get a value c . (The distribution of this value is uniform on the image of f but it does not matter for our concern.) The state becomes

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + x_0\rangle$$

with x_0 the first value such that $f(x_0) = c$ and $m = \min\{j | jr + x_0 \geq 2^{2l}\}$.

We apply another QFT and get the state

$$\frac{1}{2^l \sqrt{m}} \sum_{y=0}^{2^{2l}-1} e^{2i\pi \frac{yx_0}{2^{2l}}} \frac{1 - e^{2i\pi \frac{ymr}{2^{2l}}}}{1 - e^{2i\pi \frac{yr}{2^{2l}}}} |y\rangle.$$

Then, when we measure the state, with high probability (see [110]), we get a value y such that there exists an integer c such that $\left| \frac{yr}{2^{2l}} - c \right| < \frac{1}{2m}$.

Then $\left| \frac{y}{2^{2l}} - \frac{c}{r} \right| < \frac{1}{r^2}$, which makes $\frac{c}{r}$ one of the continued fraction development of $\frac{y}{2^{2l}}$.

2.5.3 Reducing Discrete Logarithm to Period Finding

For discrete logarithm, given x and y in a group G , we search r such that $y = x^r$ (supposing it exists). We consider the function $f : (a, b) \mapsto x^a y^b$. This function admits a period $(r, -1)$. While we use a function with two entries instead of one, we just use *QFT* on both entries and the calculations turn out to be similar.

2.6 Simon's Algorithm

We present Simon's hidden subgroup algorithm. While Shor's algorithm breaks current public key cryptosystem, Simon's algorithm [111] impacts secret key cryptography, as we will see along this thesis.

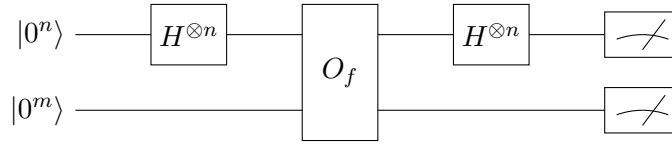


Figure 2.3: Simon's routine

2.6.1 Simon's Problem

We present the problem solved by Simon's algorithm.

Problem 2.5. Let $s \in \{0, 1\}^n$ and $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ a function such that for all x, y in $\{0, 1\}^n$, $f(x) = f(y) \Leftrightarrow x = y$ or $x = y \oplus s$. Given f , our goal is to find s .

Usually we better take interest in the following variation of the problem.

Problem 2.6. Let $s \in \{0, 1\}^n$ and $f, g : \{0, 1\}^n \rightarrow \{0, 1\}^m$ two injective functions such that for all x in $\{0, 1\}^n$, $f(x) = g(x \oplus s)$. Given f and g , our goal is to find s .

This problem is equivalent to the first one by taking

$$F : (b, x) \mapsto \begin{cases} f(x) & \text{if } b = 0 \\ g(x) & \text{if } b = 1 \end{cases}.$$

2.6.2 Description of the Algorithm

We first describe Simon's routine (Figure 2.3).

We start with the state

$$|0^n\rangle |0^m\rangle.$$

We apply the Hadamard gate on all qubits of the first register, leading to the state

$$\frac{1}{2^{n/2}} \sum_{x \in \{0, 1\}^n} |x\rangle |0^m\rangle.$$

We apply the oracle O_f , thus the state becomes

$$|\phi_f\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0, 1\}^n} |x\rangle |f(x)\rangle.$$

We measure the second register and get a value $c = f(x_0)$ for a unknown x_0 . By the premise, we get the state

$$\frac{1}{\sqrt{2}} (|x_0\rangle + |x_0 \oplus s\rangle).$$

We apply the Hadamard gate on all qubits and we get the state

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} \left((-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus s) \cdot y} \right) |y\rangle.$$

This simplifies to

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{x_0 \cdot y} \underbrace{(1 + (-1)^{s \cdot y})}_{0 \text{ if } y \cdot s = 1} |y\rangle.$$

Then when we measure the state, we get a uniformly random y such that $y \cdot s = 0$.

Simon's algorithm consists in applying this routine $n+l$ times, thus getting (y_1, \dots, y_{n+l}) and solving the linear system with unknown s

$$\begin{cases} y_1 \cdot s = 0 \\ \vdots \\ y_{n+l} \cdot s = 0 \end{cases}$$

Because the y_i are taken uniformly in the subspace $\{y, y \cdot s = 0\}$ (of dimension $n-1$), this system has rank $n-1$ with probability $\prod_{i=0}^{n-2} (1 - 2^{n+l-i}) \leq 1 - 2^{-l-1}$ (by equality of row and column rank, the first row only has to be non-zero and the next ones have to be outside the space generated by the previous ones).

2.6.3 Weakening Simon's Premise

Most of the time, the functions on which we will use Simon's algorithm will not fit the above premise exactly. This one only considers collisions of the form $f(x) = f(x \oplus s)$ but our functions will have other (random) collisions. While the outputs of Simon's routine still follows $y \cdot s = 0$, it is no longer uniform in that subspace and may lead to a system with smaller rank. By taking more outputs of Simon's routine, we still get the desired result.

Theorem 2.1 ([76]). *After $n+l$ uses of Simon's routine, Simon's algorithm returns s with probability greater than $1 - 2^n \left(\frac{1+\epsilon(f,s)}{2}\right)^{n+l}$ where $\epsilon(f,s) = \max_{t \in \{0,1\}^n \setminus \{0,s\}} (\mathbb{P}(f(x) = f(x \oplus t)))$.*

Proof. While the outputs of Simon's routine still follows $y \cdot s = 0$, the algorithm fails if the rank of the system is lower than expected. For each $t \in \{0,1\}^n \setminus \{0,s\}$, we have $\mathbb{P}(y_i \cdot t = 0) = \frac{1+\epsilon(f,s)}{2}$. Then the probability of t being solution of the system is $\left(\frac{1+\epsilon(f,s)}{2}\right)^{n+l}$. By taking the union of all possible t , we get $\mathbb{P}(\text{the algorithm fails}) \leq 2^n \left(\frac{1+\epsilon(f,s)}{2}\right)^{n+l}$. \square

Remark. A high ϵ value on a cryptographic function is considered a weakness and leads to a differential attack (see Subsection 3.3.2).

2.6.4 Simon's algorithm as a quantum circuit

As presented, Simon's algorithm is a quantum procedure implying a quantum machine running the routine and applying the measurement, and another computer (most likely classical) registering the outputs and solving the linear system. However, in many applications, it is useful to detect if there is a period as a subroutine of a bigger computation. We present some insight showing how it is possible to make a quantum circuit *QSimon* computing a variant of Simon's algorithm (Figure 2.4).

A first remark is to see that measurements can be postponed: the value $f(x_0)$ does not matter to the computation, it only serves to separate the states $|x_0\rangle$ and $|x_0 \oplus s\rangle$ from the rest. To the same effect, the value y can be in superposition for the calculations as all remaining states $|y\rangle$ share the same statement $y \cdot s = 0$.

A second remark is that the routine needs only a superposition

$$|\phi_f\rangle = O_f \circ (H^{\otimes n} \otimes I) |0^{n+m}\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

(second step of the routine), the rest consists in applying $H^{\otimes n} \otimes I$ on the state (we just observed that the measurement is not necessary), meaning that we can keep several copies of $|\phi_f\rangle$ as a database.

A third insight is making the treatment of the y_i a quantum circuit. The computation of whether the rank is maximal or not can be done by a classical reversible circuit.

2.7 Kuperberg's Algorithm

We present Kuperberg's hidden subgroup algorithm [83]. It is an equivalent of Simon's algorithm when the bit-wise xor is replaced by a modular addition.

2.7.1 Kuperberg's Premise

We present the problem solved by Kuperberg's algorithm.

Problem 2.7. Let $s \in \mathbb{Z}/2^n\mathbb{Z}$ and $f, g : \mathbb{Z}/2^n\mathbb{Z} \rightarrow \{0, 1\}^m$ be two injective functions such that for all x in $\{0, 1\}^n$, $f(x) = g(x + s)$. Given f and g , our goal is to find s .

For clarification, we call

$$F : (a, x) \mapsto \begin{cases} f(x) & \text{if } a = 0 \\ g(x) & \text{if } a = 1 \end{cases}$$

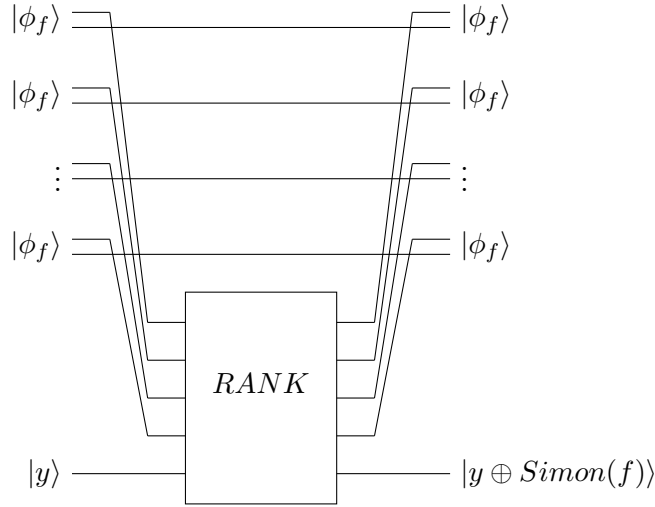


Figure 2.4: $QSimon$ gate

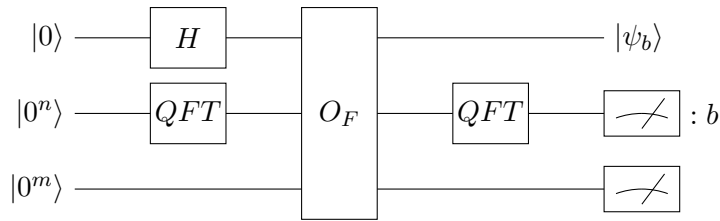


Figure 2.5: Kuperberg's routine

2.7.2 Description of the Algorithm

2.7.2.1 Kuperberg's routine

We start by describing Kuperberg's routine (Figure 2.5).

We start with the state

$$|0\rangle |0^n\rangle |0^m\rangle.$$

We apply the Hadamard gate on the first qubit and QFT on the second register, leading to the state

$$\frac{1}{2^{(n+1)/2}} \sum_{\substack{a \in \{0, 1\} \\ x \in \mathbb{Z}/2^n\mathbb{Z}}} |a\rangle |x\rangle |0^m\rangle.$$

We apply the oracle O_F , thus the state becomes

$$|\phi_{f,g}\rangle = \frac{1}{2^{(n+1)/2}} \sum_{x \in \mathbb{Z}/2^n\mathbb{Z}} |0\rangle |x\rangle |f(x)\rangle + |1\rangle |x\rangle |g(x)\rangle.$$

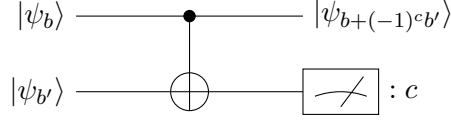


Figure 2.6: Combining outputs

We measure the second register and get a value $c = f(x_0)$ for a unknown x_0 . By the premise, we get the state

$$\frac{1}{\sqrt{2}} (|0\rangle |x_0\rangle + |1\rangle |x_0 - s\rangle).$$

We apply the *QFT* gate on the second register and we get the state

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{b \in \mathbb{Z}/2^n \mathbb{Z}} (\chi_{2^n}(x_0 b) |0\rangle + \chi_{2^n}((x_0 - s)b) |1\rangle) |b\rangle$$

This simplifies to

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{b \in \mathbb{Z}/2^n \mathbb{Z}} \chi_{2^n}(x_0 b) (|0\rangle + \chi_{2^n}(-sb) |1\rangle) |b\rangle.$$

Then we measure the state and we get a uniformly random b . The state becomes

$$|\psi_b\rangle = \frac{1}{\sqrt{2}} (|0\rangle + \chi_{2^n}(-sb) |1\rangle).$$

Exploiting the Result. At this point, exploiting $|\psi_b\rangle$ directly seems too costly. We want values of b that makes the computation easier. The simplest is $b = 2^{n-1}$, for which $|\psi_{2^{n-1}}\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^s |1\rangle) = H |s \bmod 2\rangle$.

2.7.2.2 Combining Outputs

We expose a way to combine the different states $|\psi_b\rangle$ to make more interesting ones (Figure 2.6).

We start with the state

$$|\psi_b\rangle \otimes |\psi_{b'}\rangle = \frac{1}{2} (|0, 0\rangle + \chi_{2^n}(-sb) |1, 0\rangle + \chi_{2^n}(-sb') |0, 1\rangle + \chi_{2^n}(-s(b+b')) |1, 1\rangle).$$

We apply the *CNOT* gate and we get the state

$$\frac{1}{2} (|0, 0\rangle + \chi_{2^n}(-s(b+b')) |1, 0\rangle + \chi_{2^n}(-sb') |0, 1\rangle + \chi_{2^n}(-sb) |1, 1\rangle).$$

which simplifies to

$$\frac{1}{2} (|0, 0\rangle + \chi_{2^n}(-s(b+b')) |1, 0\rangle + \chi_{2^n}(-sb') (|0, 1\rangle + \chi_{2^n}(-s(b-b')) |1, 1\rangle)).$$

We measure the second qubit and get the value c . The remaining state becomes

$$\frac{1}{\sqrt{2}} (|0\rangle + \chi_{2^n}(-s(b + (-1)^c b')) |1\rangle) = |\psi_{b+(-1)^c b'}\rangle.$$

Algorithm 2.2 Description of the Kuperberg's algorithm

Input: *superposition* oracle access to F
Output: $s \bmod 2$

- 1: Compute $\tilde{\mathcal{O}}\left(2\sqrt{2\log_2(3)n}\right)$ elements $|\psi_b\rangle$ and store them in a list L
- 2: $k \leftarrow \sqrt{2\log_2(3)n}$
- 3: $e \leftarrow \sqrt{\frac{2n}{\log_2(3)}}$
- 4: **for** $i = 0$ to $\lceil e \rceil$ **do**
- 5: $L' \leftarrow \emptyset$
- 6: **while** L contains two elements $|\psi_b\rangle$ and $|\psi_{b'}\rangle$ such that $2^{\lceil k \rceil} | (b - b') |$ **do**
- 7: Pop $|\psi_b\rangle$ and $|\psi_{b'}\rangle$ out of L
- 8: Combine $|\psi_b\rangle$ and $|\psi_{b'}\rangle$ into $|\psi_B\rangle$
- 9: **if** $B = b - b'$ **then**
- 10: Put $|\psi_B\rangle$ into L'
- 11: **else**
- 12: Put $|\psi_B\rangle$ back into L
- 13: $L \leftarrow L'$
- 14: $k \leftarrow k + \sqrt{2\log_2(3)n} - i \log_2(3)$
- 15: **if** $|\psi_{2^{n-1}}\rangle$ is found **then**
- 16: **return** $s \bmod 2$
- 17: **else**
- 18: **return** "Failure"

2.7.2.3 Recovering $|\psi_{2^{n-1}}\rangle$

We finally show how to use the different elements we have computed to recover s .

Proposition 2.1. *In algorithm 2.2, if, at the beginning of the turn, we start with $a2^k$ elements, we end the turn with at least $(a - 1)2^k/3$ elements.*

Proof. At first, we pair as many elements as possible and combine them. There are at least $(a - 1)2^k/2$ pairs as at most 2^k elements cannot get paired. Then, we get $(a - 1)2^k/4$ good combinations and $(a - 1)2^k/2$ bad combinations. Then from those bad combinations and the unpaired elements we make new pairs and so on. At the end, at most 2^k elements could not be paired and we got $(a - 1)2^k/3$ good combinations. \square

Recovering s . There are many ways to conclude here. The first one is to remark that $f' : x \mapsto f(2x)$ and $g : x \mapsto g(2x - (s \bmod 2))$ follows Kuperberg's premise with $s' = \frac{s - (s \bmod 2)}{2}$. A second one is keeping the $|\psi_{2^k}\rangle$ for $k < n$ and observing that once we recovered $s \bmod 2$, we can recover $s \bmod 4$ from $|\psi_{2^{n-2}}\rangle$ and so on.

Chapter 3

Introduction to Cryptography

This chapter presents the notions of cryptography necessary to the next chapters. It starts with an overview of the domain including asymmetric cryptography, symmetric cryptography and its constructions. It continues with foundations of cryptanalysis and specific notions related to quantum symmetric cryptanalysis. It ends with an overview of communication protocols.

Contents

3.1	Overview	42
3.1.1	Kerckhoffs's Principles	42
3.1.2	Asymmetric Cryptography	42
3.1.3	Symmetric Cryptography	43
3.2	Elements of Symmetric Cryptography	44
3.2.1	Block Ciphers	44
3.2.2	Stream Ciphers	45
3.2.3	Hash Functions	46
3.2.4	MACs	46
3.2.5	AEAD	47
3.2.6	Permutation-Based Cryptography	47
3.3	Preliminaries of Cryptanalysis	47
3.3.1	(Quantum) Security Models	48
3.3.2	Differential Cryptanalysis	49
3.3.3	Linear Cryptanalysis	49
3.4	Some Quantum Attacks	49
3.4.1	Distinguisher on One-Time Pad	50
3.4.2	3-round Feistel Network	51
3.4.3	The Even-Mansour Cipher	52
3.4.4	FX Construction	52
3.4.5	Q1 attack on Even-Mansour cipher	53
3.4.6	Other Simon-based Attacks	54
3.5	Protocols	55

3.1 Overview

Cryptography as a whole is the science of protecting information transmitted through an insecure channel. This protection is usually declined into three main properties.

Confidentiality. A third party should not be able to recover any information from exchanged messages.

Integrity. A third party should not be able to modify the exchanged information without making it invalid.

Authenticity. A third party should not be able to impersonate a rightful entity.

3.1.1 Kerckhoffs's Principles

Auguste Kerckhoff stated in 1883 [78, 79] six rules that still apply nowadays to guide the design of cryptographic systems.

- The first Kerckhoffs's principle states that “the system should be computationally, if not mathematically, indecipherable”. Each cipher or construction must be evaluated to verify its security. We will discuss how in the next sections.
- According to the second principle, “the system should not require secrecy, and it should not be a problem if it falls into enemy hands”. It separates cryptography from steganography, which is the science of dissimulating a message: even if the adversary reads the exchange and knows the method of encryption, it can not recover the information without the secret key.
- The rest of the principles are concerns about practicability. While our means are not the same as the ones Auguste Kerckhoff knew, encryption is still conceived with constraints in mind: cryptographers compete to get the most secure system but also for the fastest encryption/decryption and the smallest size.

The world of cryptography is divided in two families: symmetric cryptography, which assumes sharing a prior secret and asymmetric cryptography that does not (and is mainly used to agree on the secret).

3.1.2 Asymmetric Cryptography

First introduced by Diffie and Hellman in 1976 [46], asymmetric (or public-key) cryptography allows two parties to exchange secure messages without prior agreement on a common secret by using a private-key public-key pair.

The security of such systems relies on *hard problems*. While many of them are not proved to be as hard as it could be (or more formally, NP-complete problems[59]), like factorization and discrete logarithm (hard problems behind most currently used systems in public-key cryptography), the best algorithms solving them are carefully evaluated.

Because Shor’s algorithm (Section 2.5) solves factorization and discrete logarithm with unprecedented speed, we will need to adapt the public-key crypto-systems. That is the idea of the NIST Post-Quantum Cryptography Standardization process started in 2017 [102], which selected systems to be standardized in July 2022 with an additional round for supplementary systems.

3.1.3 Symmetric Cryptography

Symmetric cryptography dates back to antiquity with the well-known Caesar’s cipher, where each letter of the alphabet is shifted by a fixed secret number of positions, or the scytal, which changes the positions of letters in a linear pattern (some date it back to around 1500 BC with the first encrypted document). While the standards of cryptography have been greatly improved, the intent remains the same: using a shared secret to transfer a message through an insecure channel, whether it is a Gallic horseman or the Internet. In symmetric cryptography, the key used for encryption and the one for decryption are the same.

Primitives and Constructions. The world of symmetric cryptography is built from the ground up. It begins with “elementary” functions, called “primitives” (block ciphers, permutations, . . .) meant to achieve specific security properties, which we detail in the next section. Then, we combine them with operating modes, to build practical functions for different uses. For example, a block cipher takes only an input of predetermined length and we use a mode of operation for encrypting a message of any length in a safe way.

Security of Primitives. The security of a primitive is defined by how difficult is it to attack it, i.e. to break one of its security property. For example, as we will see later, considering a keyed primitive, it must be difficult to recover the key. We will see later what difficult means.

Generic Attacks. For each security property, there is a limit to what is achievable independently of the design of the primitive, called a generic bound. The attack matching the bound is called a generic attack. To complete our last example, one can always try every key and find the right one through the process, which is called exhaustive search.

Dedicated Attacks. However, the security evaluation of a specific primitive implies dedicated attacks. If a dedicated attack performs better than the best

generic attack, the primitive is usually considered to be broken. While some dedicated attacks can seem impractical, they reveal a weakness in the design that can only get worse, and it would be irresponsible to accept a broken primitive when unbroken ones exist without a proper justification.

Security Margin. Usually, when broken, a primitive is depreciated. Most of primitives are designed as a composition of smaller functions called “rounds” and before a primitive is broken, many versions of the primitive involving less rounds, called “round reduced” versions, are broken. Looking at the evolution of the number of rounds broken, we can make effective provisions of security of the full primitive. For example, the best known attacks against AES cover 7 rounds out of 10 since 2008 [88] (Section 6.10.3).

3.2 Elements of Symmetric Cryptography

In this section, we present the main elements of symmetric cryptography, their security goals and the associated generic attacks.

3.2.1 Block Ciphers

A block cipher is a family of permutations on a message space, usually $\{0, 1\}^n$ indexed by a key K in a key space, usually $\{0, 1\}^k$:

$$\begin{aligned} E &: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \\ D &: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \\ \forall k, x, D_k(E_k(x)) &= x \end{aligned}$$

Security Goal. A block cipher is meant to be used as a secret pseudo-random permutation and is evaluated as such. Without the key, the block cipher should be indistinguishable from a random permutation.

We can define this property in terms of experiments involving a challenger defining the frame and an attacker trying to win by getting some information (This idea of security game is further detailed in Section 3.5). Here is an indistinguishability (IND) game for the security of a block cipher E :

1. The challenger generates a random key K and a random permutation Π . The challenger gives the attacker either the random permutation Π or the keyed block cipher E_K as a black box.
2. The attacker uses the given black box.
3. The attacker guesses if the given black box is an implementation of the block cipher or permutation, and wins if it is right.

The block cipher E is (t, q, ϵ) -indistinguishable if any adversary \mathcal{A} using at most t computations and q queries to the black box has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2} \leq \epsilon$.

Depending on the context, we can distinguish \mathbf{Adv}_{prp} (when the attacker has access to E_k as a black box, corresponding to the IND-CPA game, prp meaning pseudo-random permutation, compared to prf meaning pseudo-random function), \mathbf{Adv}_{sprp} (when the attacker has access to E_k and E_K^{-1} as black boxes, corresponding to the IND-CCA game, sprp meaning strong pseudo-random permutation) and \mathbf{Adv}_{qprp} (when the attacker has access to E_k and E_K^{-1} as quantum black boxes, qprp meaning quantum pseudo-random permutation).

There is a weaker version where the attacker only wins if it recovers the key.

Generic Attacks. The generic attack against a block cipher is an exhaustive search for the key, verifying which key is the right one. It can be done using few plaintext-ciphertext pairs by simply checking if the encryption of the plaintext with the tested key outputs the expected ciphertext. This means there is always a classical attack using $\mathcal{O}(2^k)$ computations. The quantum equivalent is searching the right key with Grover's search (Section 2.2) instead of classic exhaustive search. It uses $\mathcal{O}(2^{k/2})$ computations.

Another generic attack consists in querying all possible inputs and making a table of all possible plaintext-ciphertext pairs, also known as the “codebook”. This means there is always an attack using $\mathcal{O}(2^n)$ computations and memory (for classical and quantum computers alike). While this does not recover the key or explicitly distinguish the block cipher from a true random permutation, an attacker with the whole codebook can easily reverse any message (if the same key is still used after querying the whole codebook).

3.2.2 Stream Ciphers

A stream cipher encrypts messages of any size by generating a sequence called “keystream” and XORs it to the message to produce the ciphertext. It uses a key K in a key space, usually $\{0, 1\}^k$, a nonce N in a nonce space, usually $\{0, 1\}^n$. The nonce is a public value that should not be repeated to avoid re-encryption:

$$\begin{aligned} E &: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^* \rightarrow \{0, 1\}^* \\ D &: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^* \rightarrow \{0, 1\}^* \\ \forall k, x, N \quad D_k(E_k(x, N), N) &= x \end{aligned}$$

A way of building a stream cipher is applying a certain mode of operation (like the counter mode) to a block cipher.

3.2.3 Hash Functions

A hash function is a (public) function that takes as input messages of any size and produces a tag of fixed size t :

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^t.$$

It is evaluated mainly along three security requirements:

Preimage Resistance: the hash function is hard to invert, i.e. for a tag T , finding a message m such that $h(m) = T$. The classical generic attack is a search on the input until T is generated, which takes $\mathcal{O}(2^t)$ computations. The quantum generic attack is a Grover's search (Section 2.2) on the input, which takes $\mathcal{O}(2^{t/2})$ computations.

Second Preimage Resistance: it is hard to get a second preimage, i.e. for a message m , finding a message $m' \neq m$ such that $h(m') = h(m)$. The classical generic attack is a search on the input, which takes $\mathcal{O}(2^t)$ computations. The quantum generic attack is a Grover's search on the input, which takes $\mathcal{O}(2^{t/2})$ computations.

Collision Resistance: it is hard to get a collision on the hash function, i.e. finding two different messages m and m' such that $h(m) = h(m')$. The classical generic attack is again a collision search on the input, which takes $\mathcal{O}(2^{t/2})$ computations. The quantum generic attack is applying the BHT algorithm (Section 2.4) on the function, which takes $\mathcal{O}(2^{t/3})$ computations, and QRAM.

3.2.4 MACs

A Message Authentication Code (or MAC) is a family of functions indexed by a key K , usually in $\{0, 1\}^k$:

$$MAC : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^t.$$

A MAC is meant to protect the integrity of a message with the receiver checking if the tag is correct (if an attacker tries to change the message, it will not be able to compute a correct tag as it does not have the key). There are two security notions for MACs.

Unforgeability: An attacker is unable to make a forgery, i.e. a valid message-tag pair which has not been given to the attacker. While this definition does not transpose well to a quantum attacker (what does it mean if the attacker makes a superposition query of all inputs?), we have another formulation: an attack is outputting more valid message-tag pair than queried.

The classical generic attack consists in either searching the key (which takes $\mathcal{O}(2^k)$ computations) or trying every tag (which takes $\mathcal{O}(2^t)$ computations). The quantum generic attack consists in either searching the key with Grover's algorithm (which takes $\mathcal{O}(2^{k/2})$ computations) or trying every tag (which takes $\mathcal{O}(2^t)$ computations).

Undistinguishability: An attacker is unable to distinguish the MAC from a random function of the same shape. This is evaluated the same way than block ciphers (replacing the block cipher by the MAC and the random permutation by a random function).

The generic attack consists in searching the key (which takes $\mathcal{O}(2^k)$ computations for classical computers and $\mathcal{O}(2^{k/2})$ computations for quantum computers).

3.2.5 AEAD

An Authenticated Encryption with Associated Data (or AEAD) encrypts messages of any length and produce a tag T usually in $\{0, 1\}^t$. The decryption decrypts the message if the tag is valid and returns an error \perp otherwise. It uses a key K in a key space, usually $\{0, 1\}^k$, a nonce N in a nonce space, usually $\{0, 1\}^n$ and an associated data AD of any length. Like stream ciphers, the nonce is a public value that should not be repeated to avoid re-encryption:

$$\begin{aligned} E &: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^t \\ D &: \{0, 1\}^k \times \{0, 1\}^n \times \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^t \rightarrow \{0, 1\}^* \cup \{\perp\} \\ \forall k, x, N, AD & D_k(E_k(x, N, AD), N, AD) = x \end{aligned}$$

3.2.6 Permutation-Based Cryptography

Permutation-based cryptography is an approach to building primitives. It consists in using public strong permutation with modes of operations (like the sponge constructions) applied to it. It was popularized by Bertoni, Daemen, Peeters and Van Assche with the proposal of Keccak [15] in 2008, which became the SHA3 standard, and later Farfalle [16]. This design became a popular trend as many finalists [47, 6, 72, 11, 48, 121, 38] of the NIST lightweight cryptography competition are designed this way.

3.3 Preliminaries of Cryptanalysis

In this section, we present some basis of cryptanalysis of symmetric constructions.

3.3.1 (Quantum) Security Models

We describe the different settings usually considered for cryptanalysis.

Q0. This is the classical setting: no quantum computing is involved. This is the usual classification (ordered by increasing security) for a function E .

- Known-ciphertext (KC): The attacker has only access to some ciphertexts $E(m_i)$.
- Known-plaintext (KP): The attacker has access to some plaintext-ciphertexts pairs $(m_i, E(m_i))$.
- Chosen-plaintext (CP): The attacker gets to choose the m_i and gets the pair $(m_i, E(m_i))$ back. If the attacker chooses the m_i all at once, it is non-adaptative, it is adaptative (CPA) if the chosen messages depend on previous outcomes.
- Chosen-ciphertext (CC) (only for revertible primitives): The attacker get to choose the m_i and a “mode” (either “encryption” or “decryption”) for each query and gets the pair $(m_i, E^{\pm 1}(m_i))$ back. If the attacker chooses the m_i all at once, it is non-adaptative, it is adaptative (CCA) if the chosen messages depend on previous outcomes.

Q1. Also known as the post-quantum setting, legitimate parties still use classical computers but consider an adversary has access to a powerful quantum computer. More formally, the adversary can query primitives with classic values but can treat those offline with a quantum machine.

Q2. Also known as the full-quantum setting, this allows an adversary to use a quantum computer but also superposition queries, effectively using the oracles O_E described in Section 1.2.3.3.

Discussion on the *Q1* and *Q2* Models [kaplan2015quantum, 29, 126, 71]. *Q1* is the most immediate quantum setting as it becomes reality as soon as there exists a quantum computer efficient enough. This setting also englobes the most immediate threat of an adversary recording communications hoping to break it with a quantum computer when it will be powerful enough. This makes it the most meaningful setting for quantum attacks.

Q2 is less immediate as it requires a machine running a quantum encryption circuit for legitimate parties and an attacker having access to a quantum encryption circuit without its key. While *Q2* setting implies *Q1*, some scenarios make the *Q2* model relevant nearly at the same time as the arrival of efficient quantum computers, especially white-box implementation. This setting has led to the most successful quantum attacks and inspired some of the *Q1* attacks while leaving many primitives unaffected. This makes *Q2* the most meaningful setting for security as it also include many intermediate scenarios.

3.3.2 Differential Cryptanalysis

Differential cryptanalysis is the most well-known family of attacks. It was first developed by Biham and Shamir [20] against the DES [103].

The basic method consists in considering pairs of plaintexts with a fixed difference and study the outputs difference, hoping for a statistical property between them. For a function f and a pair of differences (α, β) , we say that the differential $\alpha \xrightarrow{f} \beta$ has probability

$$p = \Pr(\alpha \xrightarrow{f} \beta) = \mathbb{P}_X(f(X \oplus \alpha) \oplus f(X) = \beta).$$

If p is higher than what would be expected from a random function, we have a distinguisher on the function. This distinguisher can be extended to a key-recovery attack by adding some rounds and guessing the corresponding keys until we find the good one i.e., the one that reveals the distinguisher property.

Such basic method got many variations including:

- truncated differentials, where instead of a unique difference, a family of differences is considered;
- higher order differential, where instead of at a pair, a set of plaintexts with fixed differences is used;
- boomerang attacks which we will detail further in Chapter 5.

3.3.3 Linear Cryptanalysis

Linear cryptanalysis is a general cryptanalysis method developed by Matsui against the cipher FEAL [94] and later against DES [93].

The general method is to find a linear approximation of the form $\bigoplus_{j=1}^m P_{i_j} \oplus \bigoplus_{j=1}^{m'} C_{i'_j} = \bigoplus_{j=1}^{m''} K_{i''_j}$ that holds with high probability where P is the plaintext, C is the ciphertext and K is the key.

With an ideal cipher, any equality of this kind happens with probability $\frac{1}{2}$. An attack consists in finding an equality that occurs with a probability far enough from $\frac{1}{2}$ and to detect it with a reasonable amount of data, producing a distinguisher. A key-recovery attack can also be built by adding some rounds to the distinguisher.

3.4 Some Quantum Attacks

We now present some fundamental quantum attacks that motivated further research on the subject.

3.4.1 Distinguisher on One-Time Pad

A distinguisher against the One-Time Pad, while not explicitly useful, extends our comprehension of the possibilities offered by quantum computing. It essentially breaks the usual notions of security based on the attacker only accessing limited values of the function.

3.4.1.1 Description of the One-Time Pad

The One-Time Pad was first discovered by Miller in 1882 [95] and patented by Vernam in 1919 [117]. From a n -bit message and a n -bit key, the ciphertext is the bit-wise xor between the message and the key ($E_k(m) = m \oplus k$). Since the key is supposed to be used only once (hence the name), it was proved to achieve information-theoretically security by Shannon in 1949 [109].

The One-Time Pad is still a staple of cryptography, at the base of stream ciphers. Those ciphers essentially take a seed and generate a long pseudo-random sequence that will be used as keys for executions of One-Time Pad.

3.4.1.2 Quantum Attack (in $Q2$)

The attack consists in considering the function $x \mapsto E_k(x) \oplus x$, which is constant (equal to k), and applying a variant of Deutsch-Jozsa algorithm, as shown in Algorithm 3.1.

Algorithm 3.1 Description of the distinguisher on One-Time Pad

Input: (single) oracle access to E , either a One-Time Pad or a random permutation

Output: “One-Time Pad” or “Random”

- 1: Start with the state $|0^n\rangle |0^n\rangle$
 - 2: Apply the Hadamard gate on all qubits of the first register, obtaining the state $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |0^n\rangle$
 - 3: Apply the oracle $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus E(x)\rangle$ to the state, obtaining $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |E(x)\rangle$
 - 4: Apply the CNOTs from the first register to the second one $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus x\rangle$, obtaining $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |E(x) \oplus x\rangle$
 - 5: Apply the Hadamard gate on first register, obtaining the state $\frac{1}{\sqrt{2^n}} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} |y, E(x) \oplus x\rangle$
If E is a One-Time Pad, **then** it simplifies to $|0^n\rangle \otimes |k\rangle$
If E is a random permutation, **then** the probability of measuring the state $|0^n\rangle$ in the first register is less than $\frac{n}{2^n}$.
 - 6: Measure the first register and check if the result is 0^n
 - 7: **If** it is **return** “One-Time Pad” **else return** “Random”
-

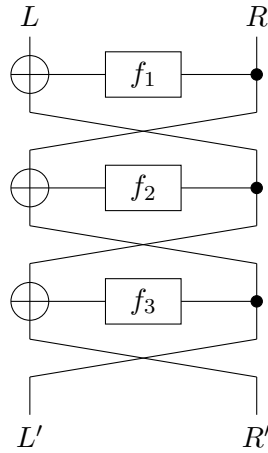


Figure 3.1: 3-round Feistel network

3.4.2 3-round Feistel Network

We then present a quantum attack that provided the first example of a $Q2$ attack using Simon’s algorithm.

3.4.2.1 Description of Feistel Networks

Feistel networks are a way of designing block ciphers first used for the Lucifer cipher, direct precursor of the DES. This design uses pseudo-random functions to build a block cipher on a doubled state. It is an iterated cipher built from a round that separates the input into a “left” part and a “right” part, applies a function parameterized by a secret key to the “left” part, xor the result to the “right” part and swaps the “left” part and “right” part.

Luby and Rackoff [89] proved in 1988 the security of 3-round Feistel (Figure 3.1) against chosen plaintext attacks and 4-round Feistel against chosen ciphertext attacks (up to $\mathcal{O}(2^{n/4})$ classical queries where n is the size of the input).

3.4.2.2 Quantum Attack (in $Q2$)

We present the attack published by Kuwakado and Morii in 2010 [84].

It exploits the fact that $L' = R \oplus f_2(L \oplus f_1(R))$. Then by fixing two distinct elements α_0 and α_1 in $\{0, 1\}^n$, we observe that $F : (b, x) \mapsto \alpha_b \oplus L'(x || \alpha_b)$ admits the period $(1, f_1(\alpha_0) \oplus f_1(\alpha_1))$. This property is exactly what Simon’s algorithm is made for and we can recover $f_1(\alpha_0) \oplus f_1(\alpha_1)$ in $\mathcal{O}(n)$ superposition queries and $\mathcal{O}(n^3)$ computations.

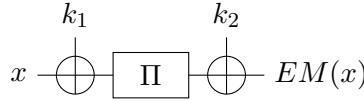


Figure 3.2: Even-Mansour Cipher

3.4.3 The Even-Mansour Cipher

The Even-Mansour cipher is a classic construction that is a staple in provable security. The quantum attacks we detail below changed the perception of what the Q1 setting allows for.

3.4.3.1 Description of the Even-Mansour Cipher

The Even-Mansour construction [53] (Figure 3.2) is a simple method to build a block cipher from a permutation. It takes a public pseudo-random n -bit permutation Π and two n -bit keys k_1 and k_2 and applies it the following way: for all $x \in \{0, 1\}^n$, $EM(x) = \Pi(x \oplus k_1) \oplus k_2$.

It is proven secure up to $\mathcal{O}(2^{n/2})$ classical queries and computations.

3.4.3.2 Quantum Attack in Q2

We present the attack published by Kuwakado and Morii in 2012 [85]. It recovers the key k_1 in $\mathcal{O}(n)$ superposition queries and $\mathcal{O}(n^3)$ computations (then the key k_2 can be recovered by using $k_2 = EM(x) \oplus \Pi(x \oplus k_1)$).

We define the function $F : x \mapsto EM(x) \oplus \Pi(x) = k_2 \oplus \Pi(x \oplus k_1) \oplus \Pi(x)$ and remark it admits k_1 as a period. Then because Π is a pseudo-random function, Simon's algorithm can recover k_1 using $\mathcal{O}(n)$ superposition queries and $\mathcal{O}(n^3)$ computations.

3.4.4 FX Construction

3.4.4.1 Description of the FX Construction

The FX construction is a simple method to increase the key length of a block cipher. It takes a n -bit block cipher E_K instantiated with a m -bit key K and two n -bit keys k_1 and k_2 and combine them as shown in Figure 3.3: for all $x \in \{0, 1\}^n$, $FX(x) = E_K(x \oplus k_1) \oplus k_2$.

It is proven secure up to $\mathcal{O}(2^{\frac{n+m}{2}})$ classical queries and computations [81].

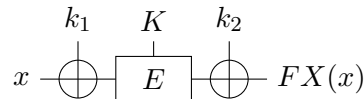


Figure 3.3: FX Construction

3.4.4.2 Grover-meets-Simon Attack

We present the attack published by Leander and May in 2017 [86]. It is the first attack combining different quantum algorithms.

The idea is to guess the key K , then the system comes down to an Even-Mansour cipher with $\Pi = E_K$ and the attack described above can be applied.

For all $k \in \{0, 1\}^m$, we define the function $F_k : x \mapsto FX(x) \oplus E_k(x)$. It admits k_1 as a period if $k = K$ and does not admits a period otherwise. Then we can use the quantum circuit implementing Simon's algorithm we discussed in Section 2.6.4 to make a circuit that evaluates whether $k = K$ or not. This circuit can be used as the evaluating function f of a Grover's search, thus making a complete circuit that recovers K in $\mathcal{O}(n)$ superposition queries and $\mathcal{O}(n^3 2^{m/2})$ computations.

Remark. The circuit implementing Simon's algorithm is approximate, i.e. has a probability of error. However we can reduce it as much as we want with little increase of time and query complexity (a constant as long as $m = \mathcal{O}(n)$).

As shown in [24], it is also possible to use only $\mathcal{O}(n)$ superposition queries by remembering that the quantum circuit only needs databases (notation defined in Section 2.6.4)

$$|\phi_{F_k}\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle |FX(x) \oplus E_k(x)\rangle.$$

Furthermore, the databases can be built on the fly as

$$|\phi_{F_k}\rangle = \underbrace{O_{E_k}}_{\substack{\text{does not need} \\ \text{a query}}} \underbrace{|\phi_{FX}\rangle}_{\substack{\text{queried} \\ \text{once}}}.$$

3.4.5 Q1 attack on Even-Mansour cipher

We finally present the different attacks on the Even-Mansour cipher in the $Q1$ setting.

3.4.5.1 Collision Attack

As the $Q1$ setting is classical queries with quantum algorithms, a first approach is the conversion of the best classical attack. This was studied by Kuwakado and Morii in [85] It consists in finding a claw between $x \mapsto \Pi(x) \oplus \Pi(x \oplus 1)$ and $y \mapsto EM(y) \oplus EM(y \oplus 1)$. Then, with good probability ($1/2$ as Π is a pseudo-random permutation), $x \oplus y \in \{k_1, k_1 \oplus 1\}$.

The $Q1$ attack then consists in finding the claw using the BHT algorithm instead of classical collision search. The $Q1$ attack has time, query and QRAM complexity $\mathcal{O}(2^{n/3})$.

An other version using the collision search without QRAM (Section 2.4.2) was proposed by Hosoyamada and Sasaki in [68] with complexity $\mathcal{O}\left(2^{3n/7}\right)$ computations, classical queries, $\mathcal{O}\left(2^{n/7}\right)$ classical memory and negligible quantum memory.

3.4.5.2 Offline Simon

Another approach inspired by the attacks in the $Q2$ setting. Bonnetain, Hosoyamada, Naya-Plasencia, Sasaki and Schrottenloher proposed it in [24]. As we cannot make the full superposition query anymore (unless we make all the classical queries and build the superposition, which defeats the purpose of this approach), we guess a part of k_1 and make the queries to apply the $Q2$ attack on the rest of the key.

More formally, we separate the key $k_1 = \underbrace{k_L}_{n-i \text{ bits}} \parallel \underbrace{k_R}_{i \text{ bits}}$.

We query the whole function $x \mapsto EM(0^{n-i} \parallel x)$ (2^i queries).

We search k_L (with Grover's algorithm, Figure 3.4) by checking if $F_k : x \mapsto EM(0^{n-i} \parallel x) \oplus \Pi(k \parallel x)$ is periodic (if $k = k_L$, k_R is a period).

If we had to make $|\phi_{F_k}\rangle$ at each iteration of the Grover's search, this procedure would lose its interest as we would need QRAM to make it effective. However, we can use the remark on the quantum attack against the FX construction to reuse the database.

$$|\phi_{F_k}\rangle = \underbrace{O_{\Pi(k \parallel \cdot)}}_{\substack{\text{does not need} \\ \text{a query}}} \underbrace{|\phi_{EM(0^{n-i} \parallel \cdot)}\rangle}_{\substack{\text{computed} \\ \text{once per} \\ \text{database}}}.$$

This attack uses 2^i queries and $\mathcal{O}\left(2^i + \sqrt{2^{n-i}}\right)$ computations, which optimizes for $i \simeq n/3$ to $\mathcal{O}\left(2^{n/3}\right)$ queries and computations.

3.4.6 Other Simon-based Attacks

Kaplan, Leurent, Leverrier and Naya-Plasencia developed in [76] a series of attacks which consist in recovering unexpected periods using Simon's algorithm. Their technique breaks many schemes including the LRW construction for tweakable block ciphers, the CBC-MAC, PMAC, GMAC building scheme for MACs and the GCM and OCB constructions for authenticated encryption.

Bonnetain, Leurent, Naya-Plasencia and Schrottenloher showed in [25] a new type of attacks called linearization attacks. It consists in using Deutsch-Jozsa algorithm (Section 2.1) to recover a linearity property even if the linear function changes for every query.

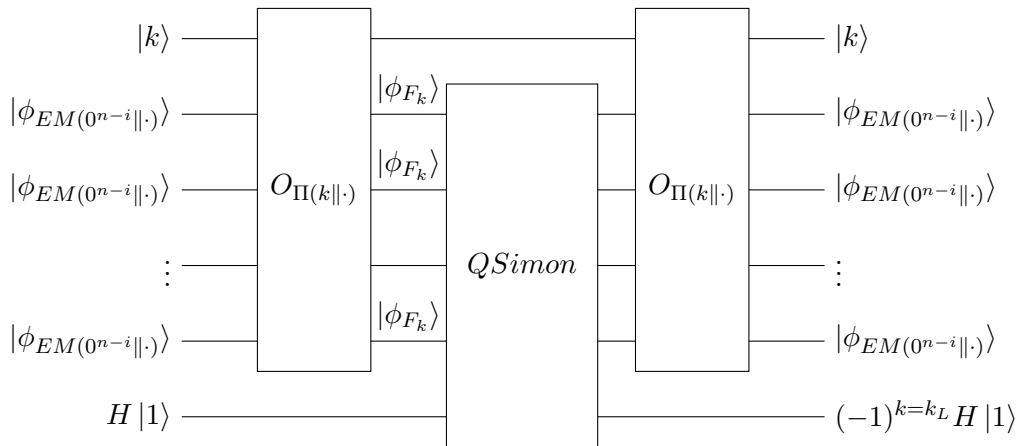


Figure 3.4: Evaluation function of the Grover's search

3.5 Protocols

Cryptographic protocols are procedures involving two or more legitimate entities in the goal to transmit some information and verifying some security properties.

While many other kinds of protocols exist, like multi-party computation, the focus in this thesis, speaking of protocols, is on Authenticated Key Exchange protocols.

AKE Protocols. Authenticated Key Exchange (AKE) protocols are aimed to turn an insecure channel into a secure one by authenticating each party and producing a shared secret key, called “session key”.

Bellare, Pointcheval and Rogaway [12, 13] modeled AKE protocols and their security requirements (specific AKE protocols may involve more security properties like privacy or internal state protection). Some preliminary notions are defined beforehand.

Security is defined in terms of experiments, formally called “games” (which are defined below). A challenger simulates executions of the protocol between some simulated legitimate parties (usually servers and clients) and an adversary is given a goal to achieve with the different tools, called “oracles”, it has access to.

For further definitions, an instance that accepts the authentication is said to be in “accepting” state (often explicit in the protocol description).

Partnering and Freshness. As a proof-making method, partnering of instances is defined by the sharing of a *sid* value (defined for each protocol, but a correct execution of the protocol must lead to partnership).

Then an instance is said to be fresh if it is not partnered to a revealed instance or one whose parent party is corrupted. An attack will only be considered valid if the target instance is still fresh at the end. A variant of freshness (to prove forward secrecy, i.e. a session key is not vulnerable to future corruption) consists in considering an instance fresh even if its parent party or the parent of a partnered instance is corrupted after the *Test* query.

Oracles. General oracles are defined. They are the ways the challenger makes the frames for further security games and the attacker interacts with the simulation.

- *Initialization* : initiates entities;
- *NewInstance*(P) : creates a new instance for party P ;
- *Execute*(P_1, P_2) : takes instances P_1 and P_2 and runs the protocol between them, returns the transcript (allows the attacker to make passive attacks);
- *Send*(P_1, m) : sends the message m to the instance P_1 and returns the adequate message (allows the attacker to make active attacks);
- *Corrupt*(P) : corrupts the entity P ;
- *Reveal*(P_1) : checks if P_1 is in accepting state and if it is, reveals the session key of P_1 and otherwise, fails;
- *Test*(P_1) : checks if P_1 is in accepting state and aborts if not. Otherwise, it initializes a secret bit b , if $b = 1$ it returns the session key of P_1 and a random value drawn from the same set otherwise.

Security. The security of such an AKE protocol is then decomposed in two properties:

Authenticity. An adversary is unable to make an instance end in accepting state without another instance registering a matching conversation, i.e. the messages sent by one is received by the other and vice versa. More formally, all polynomial-time adversaries have a negligible advantage to win the following game:

1. The challenger initializes parties.
2. The attacker fixes a target instance.
3. The attacker has access to the different oracles.
4. The attacker wins if it makes the target accept, still fresh and the instance has not a unique intended partner.

Session Key Protection. An adversary is unable to distinguish the session key of a fresh instance from random. More formally, all polynomial-time adversaries have a negligible advantage to win the following game:

1. The challenger initializes parties.
2. The attacker fixes a target instance.
3. The attacker has access to the different oracles.
4. The attacker makes a unique *Test* query on the target.
5. The attacker has access to the different oracles (second phase).
6. The attacker guesses the secret bit of the *Test* query. The attacker wins if the guess is right and the target instance is still fresh.

Chapter 4

Quantum Security of the Legendre PRF

In this chapter, we perform the first analysis of quantum algorithms for the Shifted Legendre Symbol problem (defined below) without QRAM. We give two techniques and we discuss their applicability:

1. a *quantum table-based collision search* that builds over the multi-target preimage search of [34];
2. an *offline decisional abelian hidden shift algorithm*, or *offline-DAHS* for short, which is an offline adaptation of Kuperberg’s algorithm (Section 2.7), similar to the *offline Simon’s algorithm* (Section 3.4.5.2).

These results are a joint work with André Schrottenloher, published at MathCrypt in 2021 [57].

Contents

4.1	The Legendre PRF and Context	60
4.2	Table-based Attacks	62
4.2.1	Classical Algorithm	62
4.2.2	Quantum Search with Early Abort	63
4.2.3	Distinguished Collisions	64
4.3	A Reversible Version of Kuperberg’s Algorithm	66
4.3.1	The Decisional Hidden Shift Problem	66
4.3.2	Kuperberg’s Algorithm Made Reversible	66
4.3.3	Label States and Label Qubits	67
4.3.4	Combining Two Labels Reversibly	68
4.3.5	Combining All Labels	69
4.3.6	Behavior in the Non-shifted Case	71
4.3.7	Bounding the Errors	71
4.4	Quantum Search with an Approximate Test	72
4.5	The Offline-DAHS Algorithm	75
4.5.1	High-level Description	75
4.5.2	Approximate Promise	76
4.5.3	Application to the Legendre Symbol	77

4.6	Detailed Complexities	78
4.7	Conclusion	78

4.1 The Legendre PRF and Context

Let P be a prime number and $a \in \mathbb{F}_P$. The *Legendre symbol* of a modulo P is defined as $\left(\frac{a}{P}\right) = 1$ if a is a square modulo P and -1 otherwise (by convention, $\left(\frac{0}{P}\right) = 1$). Its use in cryptography was first proposed by Damgård in [43], who conjectured the hardness of the following problem:

Problem 4.1 (Legendre sequence randomness). *Given a sequence of consecutive Legendre symbols starting at some given value $a \in \mathbb{F}_P$, of some length $m \in \text{poly}(\log_2 P)$:*

$$\left(\frac{a}{P}\right), \left(\frac{a+1}{P}\right), \dots, \left(\frac{a+m-1}{P}\right), \text{ then find } \left(\frac{a+m}{P}\right).$$

That is, consecutive Legendre symbols form a pseudo-random sequence of bits. A similar problem can be defined for the *Jacobi symbol*, which is a generalization of the Legendre symbol to a composite basis $N = P_1 \times \dots \times P_r$: $\left(\frac{a}{N}\right) := \prod_i \left(\frac{a}{P_i}\right)$.

The Legendre PRF. The conjecture of Damgård naturally leads to the definition of a pseudo-random function based on the *shifted Legendre symbol*, as in [42]:

$$\begin{cases} F_{\text{Leg}} : \mathbb{F}_P \times \mathbb{F}_P \rightarrow \{-1, 1\} \\ (s, x) \mapsto \left(\frac{s+x}{P}\right) \end{cases}$$

or as in [64], by remapping $\{-1, 1\}$ on $\{0, 1\}$. For a given secret s , distinguishing $F_{\text{Leg}}(s, x)$ from random values is the *decisional shifted Legendre symbol problem* (DSLS), the decisional version of the *shifted Legendre symbol problem* (SLS), which asks for the recovery of s . At the moment, no separation between the SLS and the DSLS is known. In this chapter, as in previous works, we will focus on solving the SLS.

Problem 4.2 (Shifted Legendre Symbol). *Let P be a prime number. Given query access to the function $F_{\text{Leg},s} : x \mapsto \left(\frac{x+s}{P}\right)$ for some secret $s \in \mathbb{F}_P$, find s .*

The Legendre symbol PRF has recently regained significant interest with the proposal of Grassi, Rechberger, Rotaru and Scholl [64] to use it in a multi-party computation scenario. They showed precisely that there existed a simple MPC protocol to compute the PRF on secret-shared data, using the malleability of the Legendre symbol.

Since then, the PRF has been considered for use in the Ethereum blockchain, and the Ethereum foundation has proposed several challenges to encourage

cryptanalysis research [54]. While the Ethereum challenges are not explicitly concerned with quantum adversaries, this is the case of the LegRoast signature scheme proposed in [18], which adapts the Picnic construction [35] with the Legendre PRF. Solving the SLS problem allows to break these different schemes. However, in both cases the adversary cannot make chosen-plaintext queries to the PRF. In the Ethereum challenges he only knows a (large) sequence of successive Legendre symbols, a condition that can be satisfied by all the algorithms studied in this chapter. In LegRoast, he only learns the evaluation of the PRF on a few random, uncontrolled inputs.

Here, we focus on the SLS problem, and so, the security of the Legendre PRF, against a quantum adversary. The algorithms considered derive their advantage from making a large number of queries, and so, they do not impact all applications of the PRF in the same way (e.g., LegRoast does not seem to be affected).

Previous Results. The SLS problem is a particular case of the *shifted character problem*, where the Legendre symbol is replaced by any multiplicative character of a finite field. This problem was studied by van Dam *et al.* [42] in the quantum setting. When *superposition query access* to the shifted character is given, they showed that the problem could be solved in polynomial time, using a single query.

However, when only *classical* queries are available, the SLS problem is believed to remain intractable for a quantum attacker, and was conjectured so by van Dam *et al.* [42] and by Grassi *et al.* [64].

In the classical setting, several authors have studied and improved the Legendre PRF key-recovery attacks [80, 74, 17]. We give the detailed complexities in Table 4.1. The most advanced results were obtained by Beullens, Beyne, Udovenko and Vitto *et al.* [17] and concurrently by Kaluderović, Kleinjung and Kostic *et al.* [74, 75]. Given a sequence of M Legendre symbols, they recover the secret in about $\tilde{O}\left(\frac{P}{M^2}\right)$ operations. Their technique is a *table-based collision search*, whose principle will be reviewed in Section 4.2.

Even more recently, Seres, Horváth and Burcsi [108] showed that the problem of recovering the key of a Legendre PRF is equivalent to solving some multivariate quadratic system of equations. So far this approach did not yield better attacks than those of [74, 17], as dedicated algebraic attacks seem inefficient against this system.

On the Use of QRACM. In the quantum setting, it was proposed [74] to use a quantum version of the table-based collision search (some remarks were also made in [17]). However, in the same way that the classical attack uses a large table, the quantum attack will use a table of similar size. This table requires the model of *classical memory with quantum random-access* (QRACM). Although this is a powerful memory model, it is required by many

quantum algorithms, e.g., for BHT quantum collision search [32], which makes it theoretically worth studying.

No practical, scalable implementation of QRACM exists at the moment, and near-term quantum architectures are expected to consist only of error-corrected quantum circuits of small width. Several authors envision an advantage coming from parallel circuits rather than memory usage [10, 73]. In this context, QRACM can be seen as a conservative assumption, that cannot be always accurate. This is why, more recently, quantum cryptanalytic algorithms have been developed that aim for an advantage over classical algorithms, while using standard computing qubits only.

4.2 Table-based Attacks

In this section, we first recall the classical *table-based collision search* of [17, 74] (Section 4.2.1), and the idea of *early abort* in a Grover search (Section 4.2.2), which is a folklore technique that we will use to save logarithmic factors. Next, we introduce our new *quantum table-based collision search*. It combines the ideas of the classical attack and the quantum collision search algorithm of [34]. The notations follow those of Beullens *et al.* [17].

As all the attacks studied in this paper, the table-based collision attacks use chosen, but non-adaptive queries to the Legendre PRF. More precisely, we assume that we can query $M < \sqrt{P}$ consecutive values of the Legendre PRF, of the form $\left(\frac{x+s}{P}\right)$ where $0 \leq x \leq M - 1$ and s is the secret. This is actually the setting of the Ethereum challenges [54].

4.2.1 Classical Algorithm

We set $m = 3 \lceil \log_2 P \rceil$ and $L \leq (\log_2(P))^2$ the time to compute a Legendre symbol. The attack parses the M sequential Legendre symbols to create *L-sequences*: words of m bits that allow to discriminate a guess of s . The definition of *L-sequences* is from [17] and we slightly simplify it. We define:

$$L_a = \left(\left(\frac{a}{P} \right), \left(\frac{a+1}{P} \right), \dots, \left(\frac{a+m-1}{P} \right) \right).$$

Assuming that the Legendre PRF is “sufficiently random”, and that m is big enough, a collision of *L-sequences* implies the equality of their parameters: $L_a = L_b \implies a = b$. With our choice of m , we will assume that *no random collisions occur at all*. This is a stronger heuristic than the one commonly used in the classical cryptanalysis of the Legendre PRF (see [17], Assumption 1). It will simplify our analysis of quantum algorithms.

Heuristic 4.1. For all $a, b \in \mathbb{Z}_P^*$, $L_a = L_b \implies a = b$.

In order to recover the secret s , one then looks for a collision between an *L-sequence* of unknown parameter (depending on s) and an *L-sequence* of

known parameter. A basic attack could use the M queries to obtain $M - m$ L -sequences, and then, evaluate random L -sequences in search for a collision. This would give a complexity $\mathcal{O}\left(M + \frac{P}{M}\right)$ (as two L -sequences can only collide with probability $\frac{1}{P}$).

The attacks of [17, 74, 75] improve the trade-off between M and the time complexity, by extracting more L -sequences from the available queries. This uses the multiplicativity of the Legendre symbol. The attacks run as follows:

1. From the M consecutive Legendre symbols, extract $\frac{M^2}{m}$ L -sequences of the form:

$$\begin{aligned} & \left(\left(\frac{a+s}{P} \right), \left(\frac{a+b+s}{P} \right), \dots, \left(\frac{a+b(m-1)+s}{P} \right) \right) \\ &= \left(\frac{b}{P} \right) \left(\left(\frac{a/b+s/b}{P} \right), \left(\frac{a/b+1+s/b}{P} \right), \dots, \left(\frac{a/b+(m-1)+s/b}{P} \right) \right) \\ &= \left(\frac{b}{P} \right) L_{(s/b+a/b) \bmod P}. \end{aligned}$$

Since we can use $b \leq \lfloor \frac{M}{m} \rfloor$ and $a < M - bm + 1$, the number of sequences is increased quadratically with respect to the naive extraction.

2. Store all the extracted sequences $(L_{(s/b+a/b) \bmod P}, a, b)$ in a table.
3. Sample c at random until (L_c, a, b) appears in the table for some a, b . Such a collision yields a candidate key s such that: $s/b + a/b = c \implies s = cb - a \bmod P$. We can then test if this candidate is the good one with a few more computations. With Heuristic 4.1, there are no false positives, and s is the right key.

This classical attack requires M^2 storage for the table, and expectedly mP/M^2 samples must be tested in Step 3 before a collision occurs. Thus Step 3 requires $\mathcal{O}(m^2P/M^2)$ Legendre symbol computations. Further optimizations allow to reduce the memory to M^2/m and to amortize the cost of computing Legendre symbols in an iteration of the loop.

Quantum Version with QRACM. This procedure yields a quantum attack as proposed in [74]. The precomputation stage (Step 1) is unchanged, but now Step 3 is a quantum search. Instead of running $\mathcal{O}(mP/M^2)$ classical iterations, we need only $\mathcal{O}(\sqrt{mP/M^2})$ iterations, each of which performs $\mathcal{O}(m)$ Legendre symbol computations and a memory access. This can only be efficient if we use classical memory with quantum random-access (QRACM).

4.2.2 Quantum Search with Early Abort

Using the same M classical sequential queries, but without QRACM, the first quantum attack available is a direct quantum search of the secret s . We

query the L -sequence L_s (thus using only m data) and search for $x \in \mathbb{Z}_P^*$ such that $L_x = L_s$. We simply apply amplitude amplification (Section 2.3), where the amplified algorithm $S_{\mathbb{Z}_P^*}$ consists in sampling an element $x \in \mathbb{Z}_P^*$ at random: $S_{\mathbb{Z}_P^*}|0\rangle = \sum_{x \in \mathbb{Z}_P^*} |x\rangle$, and the test f consists in checking if $L_x = L_s$. By Heuristic 4.1, there is only one such solution. Thus the algorithm requires $\mathcal{O}(m\sqrt{P})$ (quantum) Legendre symbol computations, in total $\mathcal{O}(m^3\sqrt{P})$ quantum gates.

Early-aborting. In a classical search for a sequence matching L_s , we can stop the computation of L_x at the first bit that does not match. This reduces the average number of Legendre symbols computed from m to a constant. In the quantum setting, this folklore idea allows to amortize the factor m down to $\log_2 m$.

We select a constant $i \leq m$ and define a subset of \mathbb{Z}_P^* :

$$X_i = \{x \in \mathbb{Z}_P^*, L_x \text{ and } L_s \text{ match on the first } i \text{ bits}\}.$$

We know that s belongs to X_i , and by the pseudorandomness of the Legendre PRF, that X_i is roughly of size $\frac{P}{2^i}$. Furthermore, testing if a given x is in X_i requires to compute only i Legendre symbols, not m . This allows to filter out most of the incorrect values.

We first use amplitude amplification to create an “inner” search: an algorithm S_{X_i} that samples from X_i . This algorithm runs a quantum search over $x \in \mathbb{Z}_P^*$ with $\mathcal{O}(\sqrt{2^i})$ iterations, computing i Legendre symbols each. Since we do not know exactly the size of X_i , we must run the search with a fixed number of iterations, and the output state is not exactly the uniform superposition over X_i . However, we only need to sample from X_i with constant probability \mathfrak{p} . Afterwards, the output of S_{X_i} is “good” for us (equal to s) with probability $\mathfrak{p} \frac{2^i}{P}$, where \mathfrak{p} is a constant.

We can thus use amplitude amplification again. We amplify S_{X_i} , using $\mathcal{O}(\sqrt{\frac{P}{2^i}})$ iterations in total. Each iteration contains a computation of S_{X_i} and $m - i$ Legendre symbols. This gives a total complexity:

$$\mathcal{O}\left(\sqrt{\frac{P}{2^i}} \left(\sqrt{2^i}(iL) + (m - i)L\right)\right).$$

Taking $i = 2 \log_2 m$ and simplifying gives: $\mathcal{O}(\log_2 m \sqrt{PL})$.

4.2.3 Distinguished Collisions

Since we do not assume QRACM, we have to modify the table-based collision strategy if we are to beat the square-root complexity given by Grover search. We will use the strategy of [34] for multi-target preimage search, and adapt the algorithm of Section 4.2.1 as follows:

1. From the M Legendre symbols, extract $\frac{M^2}{m}$ L -sequences.
2. Store only the $\frac{M^2}{m2^t}$ “distinguished” sequences that start with t zeroes (an arbitrary choice).
3. Sample x such that L_x is distinguished, until it matches one of the stored sequences.

We use amplitude amplification again. First, we build a quantum algorithm S_D that samples x such that L_x is distinguished. We can do that in time $\mathcal{O}(\log_2 t \sqrt{2^t L})$ using an early-aborted quantum search. Again, S_D is not exact, and if we measure its output, we get a distinguished L_x with constant probability only. But this is enough.

We next estimate the probability that S_D returns a “good” output i.e., an x such that L_x collides with one of the stored sequences. There are on average $P/2^t$ distinguished sequences, and the probability to collide on the table is roughly $\frac{M^2/(m2^t)}{P/2^t}$. Depending on the exact number of distinguished sequences, this probability also deviates from the expectation, but not more than by a constant.

Next, to test if a distinguished L_x matches one of the stored sequences, we use a sequential circuit containing quantum gates *controlled by classical values*. A single L_a can be compared to the current L_x with a comparator using $\mathcal{O}(m)$ gates; we repeat this for all distinguished sequences of our table.

Assuming that we use all the data, this yields an algorithm of complexity:

$$\mathcal{O}\left(M^2 + \sqrt{\frac{P}{2^t \times \frac{M^2}{m2^t}}} \left(\log_2 t \sqrt{2^t L} + mL + \frac{M^2}{m2^t} m\right)\right).$$

Note that the outer number of iterations has been reduced, because two distinguished sequences have a higher probability to collide than two random sequences. By taking $t = \frac{4}{3} \log_2(M/m)$ we get a complexity:

$$\begin{aligned} \mathcal{O}\left(M^2 + \sqrt{\frac{Pm}{M^2}} \left(\frac{M}{m}\right)^{2/3} (\log_2 \log_2 M) m^2\right) \\ = \mathcal{O}\left(M^2 + \frac{\sqrt{P}}{M^{1/3}} m^{11/6} \log_2 \log_2 M\right). \end{aligned}$$

Note that a memory of size M is required during Step 1 (extraction), and $M^{2/3}m$ during Step 3 (search). Both are only classical. Also, the memory of Step 3 is accessed only once per iteration ($\sqrt{\frac{Pm}{M^2}}$ in total) and in a sequential way. The complexity of the classical table-based collision decreases with the available data M , from $\tilde{\mathcal{O}}(P)$ down to $\tilde{\mathcal{O}}(P^{1/2})$ when $M = P^{1/4}$. This quantum variant decreases from $\tilde{\mathcal{O}}(P^{1/2})$ to $\tilde{\mathcal{O}}(P^{3/7})$, reaching this minimum when $M = P^{3/14}$ data is available.

4.3 A Reversible Version of Kuperberg's Algorithm

In this section, our goal is to present a quantum circuit that applies Kuperberg's algorithm. Indeed, Section 2.7 shows a procedure implying many intermediate measurements and classical procedures.

4.3.1 The Decisional Hidden Shift Problem

Quantum algorithms for *hidden period* or *hidden shift* problems have seen numerous applications in quantum cryptanalysis (Section 3.4).

As it was already remarked in [42], the SLS problem is an instance of Problem 2.7, where: $g(x) = L_{s+x}$, obtained by m queries to the PRF $F_{\text{Leg},s}$, and $f(x) = L_x$, obtained by m computations of Legendre symbols modulo P . In this chapter, we will not use Kuperberg's algorithm to solve directly the SLS, but a *decisional* version of Problem 2.7.

Problem 4.3 (Decisional abelian hidden shift). *Let $f, g : \mathbb{Z}/2^n\mathbb{Z} \rightarrow X$ be two injective functions such that either: $\exists s, \forall x, f(x+s) = g(x)$, or $\text{Im}(f) \cap \text{Im}(g) = \emptyset$. Decide which is the case.*

4.3.2 Kuperberg's Algorithm Made Reversible

As we focus on Problem 4.3, we will work with an abelian group of the form $G = \mathbb{Z}_{2^n}$ for some n . We note $M = 2^n$ the cardinality of the group. Note that the generalization to an arbitrary abelian group would be technical, but without significant incidence on the time complexity [28]. The assumption that the functions are injective is also helpful for our study, but not strictly necessary.

Sample Database. We define a *sample state* as:

$$|\phi_{f,g}\rangle = \sum_{0 \leq x \leq M-1} |0\rangle |x\rangle |f(x)\rangle + |1\rangle |x\rangle |g(x)\rangle,$$

omitting the common amplitude factor for ease of notation. One creates it with a single oracle query to O_f and O_g . Kuperberg's algorithm starts by producing $t = \tilde{O}(2^{\sqrt{\alpha n}})$ such states, where $\alpha = 2 \log_2 3$ is obtained from the complexity analysis. We name $|\phi_{f,g}\rangle^{\otimes t}$ the *sample database* of (f, g) : it contains all the information on f and g that we need to solve Problem 4.3.

Our goal is to process this state to decide whether f and g are shifted, *without destroying the database*. Measurements can always be removed from a quantum computation. Thus, we know that we can follow the standard operations of Kuperberg's algorithm, but without performing any measurement, and obtain a quantum circuit DAHS that yields the same result.

However, the standard procedure handles a lot of classical data, and performs non-trivial memory operations. This may increase significantly the

time complexity in a fully reversible variant, especially since we do not want to rely on quantum RAM. We will show that reversibility costs only at most a polynomial factor in time. Given the *sample database* $|\phi_{f,g}\rangle^{\otimes t}$, the circuit DAHS finds whether f and g are shifted with constant probability of success. This probability can then be boosted by taking a polynomial number of copies of the circuit. All in all, we prove the following theorem.

Theorem 4.1. *There exists a quantum circuit DAHS that maps:*

$$|\phi_{f,g}\rangle^{\otimes t} |b\rangle \mapsto |\phi_{f,g}\rangle^{\otimes t} |b \oplus \text{DAHS}(f, g)\rangle + |\delta_{f,g}\rangle,$$

where $\text{DAHS}(f, g) = 1$ if and only if f and g are a shifted pair, and $\|\delta_{f,g}\|$ is bounded by a constant for all f, g satisfying the conditions of Problem 4.3. With $t = \tilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$, it contains $\tilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$ quantum gates and ancilla qubits. With a factor r in time and memory complexity, we can reduce the bound on $\|\delta_{f,g}\|$ to $\mathcal{O}(2^{-r/2})$.

We devote the rest of this section to the details of this procedure. For now, we will assume that f and g are shifted, and we will go back to the other case at the end of this section.

4.3.3 Label States and Label Qubits

The first step in Kuperberg's algorithm is to transform all the independent sample states into *label states*, by measuring a value a in the third register and applying an M -dimensional QFT on the second register. In the reversible variant, we do not measure this a . We do not write it either for simplicity (after the QFT, it does not intervene further in the algorithm).

Since the functions are injective, there exists a single x_0 that maps to a through f and, by assumption, $x_0 - s$ maps to a through g . We obtain:

$$\begin{aligned} |\psi_{f,g}\rangle &= \text{QFT}_M(|0\rangle |x_0\rangle + |1\rangle |x_0 - s\rangle) \\ &= \sum_{0 \leq y \leq M-1} \left(\chi_M(x_0 y) |0\rangle + \chi_M((x_0 - s)y) |1\rangle \right) |y\rangle \\ &= \sum_{0 \leq y \leq M-1} \chi_M(x_0 y) \left(|0\rangle + \chi_M(-sy) |1\rangle \right) |y\rangle. \end{aligned}$$

Next, we would have measured the register $|y\rangle$ to obtain a random value y and a *label qubit*: $|\psi_y\rangle = |0\rangle + \chi_M(-sy) |1\rangle$ (up to a global phase factor). Again, we do not measure y . So we simply write label states as $|\psi_y\rangle |y\rangle$, keeping in mind that there is a superposition over y .

A label qubit contains some information about s , but it is not immediately exploitable, unless we can obtain specific values of y . For example, if $y = 2^{n-1}$, then the qubit is either $|0\rangle + |1\rangle$ or $|0\rangle - |1\rangle$ depending on the least significant bit of s . Recall that we are interested in *deciding* whether there is a shift or not (Problem 4.3). We can do that from many independent copies of $|\psi_{2^{n-1}}\rangle$.

Classical Combinations. The key step in the algorithm is when we create these wanted labels from the random initial ones. In the standard procedure, the values of the labels are known. From two label qubits $|\psi_{y_1}\rangle$ and $|\psi_{y_2}\rangle$, we can obtain $|\psi_{y_1 \pm y_2}\rangle$ with a CNOT and a measurement. This erases both qubits and returns either $y_1 + y_2$, or $y_1 - y_2$, with probability $\frac{1}{2}$. The procedure then consists in combining pairs of labels (y_1, y_2) that maximize the expected valuation of $y_1 \pm y_2$ modulo 2. We let $\text{val}_2(z) = \max\{i, 2^i | z\}$ denote this valuation.

Complexity. The complexity analysis in [83] gives that $\tilde{O}(2^{\sqrt{an}})$ initial label qubits are enough. Simulations in [26] showed that $2^{\sqrt{an}}$ were essentially enough to solve Problem 2.7 with constant probability, for groups of the form \mathbb{Z}_{2^n} . Interestingly, the algorithm is *pseudoclassical*: the combination step can be easily simulated by sampling labels at random. This allows to compute precisely how many labels will be needed for a given instance of the problem, and to obtain the corresponding success probability.

4.3.4 Combining Two Labels Reversibly

When combining two labels, we start from the joint state:

$$\begin{aligned} |\psi_{y_1}\rangle |\psi_{y_2}\rangle |y_1\rangle |y_2\rangle &= \left(|0\rangle + \chi_M(-y_1 s) |1\rangle \right) \left(|0\rangle + \chi_M(-y_2 s) |1\rangle \right) |y_1\rangle |y_2\rangle = \\ &= \left(|00\rangle + \chi_M(-y_1 s) |10\rangle + \chi_M(-y_2 s) |01\rangle + \chi_M(-(y_1 + y_2) s) |11\rangle \right) |y_1 y_2\rangle, \end{aligned}$$

and we CNOT the first qubit into the second one, mapping $|10\rangle$ to $|11\rangle$ and $|11\rangle$ to $|10\rangle$. We obtain:

$$\begin{aligned} &\left(|0\rangle + \chi_M(-(y_1 + y_2) s) |1\rangle \right) |0\rangle |y_1\rangle |y_2\rangle \\ &\quad + \chi_M(-y_2 s) \left(|0\rangle + \chi_M(-(y_1 - y_2) s) |1\rangle \right) |1\rangle |y_1\rangle |y_2\rangle. \end{aligned}$$

In order to mimic the classical recomputation of labels, we perform a controlled addition or subtraction of y_2 in place, on the register that contains y_1 . We obtain the state:

$$\left(|\psi_{y_1+y_2}\rangle |0\rangle |y_1 + y_2\rangle + \chi_M(-(y_1 - y_2) s) |\psi_{y_1-y_2}\rangle |1\rangle |y_1 - y_2\rangle \right) |y_2\rangle.$$

Once this operation has been performed, we get a qubit $|0\rangle$ or $|1\rangle$ that is used to indicate whether we obtained the sum $y_1 + y_2$ or the difference $y_1 - y_2$. Here, it is just kept along for reversibility. The register that contains y_2 has become entangled with the other and cannot be used for further combinations.

In our DAHS circuit, we will define a “combination circuit” Comb_v (Figure 4.1). We flag all the label states with additional qubits that inform

us whether the label can be used for further combination or not. The circuit Comb_v then first tests that the two labels y_1, y_2 have valuation v and that their flags b_1, b_2 are equal to 1 (Flag). If this is true, then it performs an addition or subtraction in place (Sub, Add), then NOTs the flag qubit b_2 (X), since y_2 cannot be used for combination anymore.

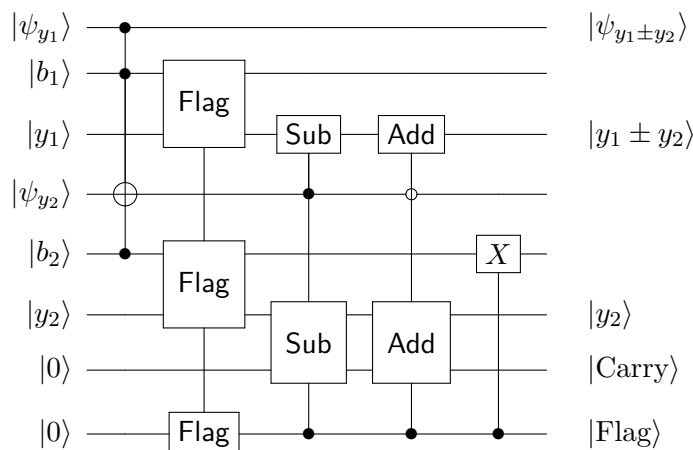


Figure 4.1: Combination circuit Combine_v . Note the “control on 0” which applies the addition only if the qubit $|\psi_{y_2}\rangle$ is 0.

Choosing which Labels to Combine. Among the pairs of labels y_1, y_2 having the same valuation modulo 2, we want to select those which maximize the expected valuation of $y_1 \pm y_2$. We check whether the second to last bit of $y/2^{\text{val}_2(y)}$ is 0 or 1. If it is 0, then our hope is to add y to another label that maximizes the overlap of least significant bits. If it is 1, then our hope is to subtract y to another label that overlaps with $2^n - y$ on as many least significant bits as possible. Thus we can define a function F on labels¹:

$$F(y) = \begin{cases} (1, 0) & \text{if } y = 0 \text{ or } y \text{ cannot be combined anymore} \\ (-\text{val}_2(y), \text{rev}(2^n - y, n)) & \text{if the second to last bit of } y/(2^{\text{val}_2(y)}) \text{ is 1} \\ (-\text{val}_2(y), \text{rev}(y, n)) & \text{otherwise} \end{cases}$$

where $\text{rev}(y, n)$ reverses the bits in y (for a total of n bits). By sorting the labels according to F , we ensure that the best pairs, such that $y_1 \pm y_2$ has the best expected valuation, are put together.

4.3.5 Combining All Labels

We start from a list of $t = 2^\ell$ label states for some integer ℓ . For $v = 0, \dots, n-2$:

¹Although we did not find its explicit definition in previous works, it appears in the simulation code of [26].

- we perform a reversible sorting network for F : we compute F in ancillas, perform a sorting network, and then uncompute F . We consider that the computation of F can be neglected. The sorting network is a series of comparators and swaps (controlled on the results of the comparators). The labels are moved in place. For reversibility, the outcome of each comparator must be written in a new qubit;
- we apply the combination circuit Comb_v on each pair of labels at positions $(2i, 2i + 1)$ for $i \leq 2^{\ell-1}$. It writes 2ℓ new qubits that contain carries and inform whether the combinations occurred.

In practice, the combination layer at step v consumes all labels of valuation v and creates labels of higher valuation. The main difference with the classical process is that, although the labels are sorted to ensure a maximal number of zeroes after combination, since we take them 2 by 2 on arbitrary positions, we might create a few suboptimal pairs. This does not change the asymptotic complexity of the procedure.

Note that all labels equal to $0 \bmod 2^n$, and the labels that cannot be combined anymore, are moved to the bottom of the list by sorting. The labels equal to 2^{n-1} will be moved to the top. Thus, we know where to look for them.

Sorting Network. We use the *odd-even mergesort* of Batcher [9]. On input a list of 2^ℓ n -bit strings, it uses a total of $S(2^\ell) = 2^{\ell-1} \frac{\ell(\ell-1)}{2} + 2^\ell - 1 = \mathcal{O}(\ell^2 2^\ell)$ comparators and controlled SWAPs. In order to be made reversible, it also needs to write $\mathcal{O}(\ell^2 2^\ell)$ new qubits.

The Full Circuit. The full combination circuit contains $n - 1$ layers of sorting, followed by layers of combination circuits: at layer i (starting from 0), we combine only the labels having valuation i . The complexity mainly depends on the sorting steps: there are in total $\mathcal{O}(n) \times \mathcal{O}(n) \times S(2^\ell) = \mathcal{O}(n^2 \ell^2 2^\ell)$ quantum gates used, mainly for comparators and SWAPs. The circuit writes $nS(2^\ell) + (n - 1)2^\ell$ ancillas. They are uncomputed afterwards.

After the combination, we look at the r first label registers, for some value r to choose later. We expect them to contain copies of 2^{n-1} . As we have seen, in the shifted case, the corresponding qubits contain identical copies of $|0\rangle + |1\rangle$, or $|0\rangle - |1\rangle$ depending on the parity of s . Thus, we perform a Hadamard transform on them and test if the result is all-zero or all-one.

Note that if the first label registers do not contain the expected copies of 2^{n-1} , then *we know that the circuit has failed*. The fact that failures are detectable allows to reduce easily the probability of failure with multiple copies of the circuit: we simply take the result that has not failed.

4.3.6 Behavior in the Non-shifted Case

In the non-shifted case, by our assumptions, there is no overlap between functions f and g . Thus, sample states are the sums of two independent parts:

$$|\phi_{f,g}\rangle = \underbrace{\left(\sum_x |0\rangle |x\rangle |f(x)\rangle \right)}_{|\phi_f\rangle} + \underbrace{\left(\sum_x |1\rangle |x\rangle |g(x)\rangle \right)}_{|\phi_g\rangle},$$

and the whole sample database $|\phi_{f,g}\rangle^{\otimes t}$ can be rewritten as a sum of the 2^t states of the form $|\phi_{h_1}\rangle \otimes \cdots \otimes |\phi_{h_t}\rangle$ where $h_t \in \{f, g\}$. By linearity, we can focus on the output of DAHS on one of these states only.

The QFT applied to $|\phi_h\rangle$ yields a state $|b\rangle \sum_y |y\rangle (\sum_x \chi_M(xy) |h(x)\rangle)$ where b depends only on h . Therefore, after performing the combination step, and after obtaining labels equal to 2^{n-1} , the corresponding qubits are not $|0\rangle \pm |1\rangle$, but either $|0\rangle$, or $|1\rangle$, depending on the exact sequence of combinations performed.

Recall that in the shifted case, the r qubits on which we apply the final Hadamard transform contain copies of $|0\rangle \pm |1\rangle$. If the algorithm succeeds, these copies are obtained in all cases, so they are disentangled from the rest of the state. In contrast, in the non-shifted case, we obtain a single r -dimensional basis state depending on the sequence of combinations. Thus after applying the Hadamard transform, the total amplitude on $|0^r\rangle$ (resp. $|1^r\rangle$) is equal to $2^{-r/2}$.

4.3.7 Bounding the Errors

The circuit DAHS is not exact. In the positive (shifted) case, it maps:

$$|\phi_{f,g}^t\rangle |b\rangle \xrightarrow{\text{DAHS}} |\phi_{f,g}^t\rangle |b \oplus 1\rangle + |\delta_{\text{FN}}^{f,g}\rangle |b\rangle$$

where $\|\delta_{\text{FN}}^{f,g}\| \leq \epsilon_{\text{FN}}$, and ϵ_{FN}^2 is the (small) *probability of false negative*; and in the negative (non shifted) case, it maps:

$$|\phi_{f,g}^t\rangle |b\rangle \xrightarrow{\text{DAHS}} |\phi_{f,g}^t\rangle |b\rangle + |\delta_{\text{FP}}^{f,g}\rangle |b \oplus 1\rangle$$

where $\|\delta_{\text{FP}}^{f,g}\| \leq \epsilon_{\text{FP}}$ and ϵ_{FP}^2 is the (small) *probability of false positive*. We now bound both ϵ_{FN} and ϵ_{FP} separately, and independently of f and g .

False Negatives. False negatives come from all the sequences of labels $Y = y_1, \dots, y_t$ on which the combination cannot produce enough labels 2^{n-1} . As unitary operators preserve the euclidean norm, we can bound $\|\delta_{\text{FN}}\|$ before the combination step. Let us consider the state:

$$|\psi_{f,g}\rangle^{\otimes t} = \sum_X \sum_Y \chi_M(X \cdot Y) \left(\bigotimes_{1 \leq i \leq t} (|0\rangle + \chi_M(-y_i) |1\rangle) |y_i\rangle \right) |f(X)\rangle, \quad (4.1)$$

where $X = x_1, \dots, x_t$, $\chi_M(X \cdot Y) = \prod \chi_M(x_i y_i)$ and $f(X) = f(x_1), \dots, f(x_t)$. All the “bad” sequences Y contribute to the probability of false negatives, so we bound:

$$\left\| \sum_X \sum_{Y \text{ bad}} \chi_M(X \cdot Y) \left(\bigotimes_i (|0\rangle + \chi_M(-y_i s) |1\rangle) |y_i\rangle \right) |f(X)\rangle \right\| \leq \sqrt{\frac{|\text{bad } Ys|}{|\text{all } Ys|}}.$$

Thus, if both f and g are injective, we have $\epsilon_{FN} = \sqrt{\frac{|\text{bad } Ys|}{|\text{all } Ys|}}$ (and this does not depend from f and g). The classical analysis of Kuperberg’s algorithm gives a constant probability of finding a good label if we start from $\mathcal{O}(2^{\sqrt{cn}})$ of them. Thus, if we take copies of the combination circuit, we can obtain r good labels with probability $1 - \epsilon$ if we start from $\mathcal{O}((- \log_2 \epsilon) r 2^{\sqrt{cn}})$ labels.

False Positives. We have seen above that when the functions are not shifted, we have still some probability to fall on $|0^r\rangle$ or $|1^r\rangle$ after the final Hadamard transform. This probability is also independent of f and g . The amplitude on the state $|0^r\rangle$ or $|1^r\rangle$ is exactly $\frac{1}{\sqrt{2^r}}$, and for all f : $\|\delta_{FP}\|^2 = \frac{2}{2^r} \implies \epsilon_{FP} = 2^{-(r-1)/2}$.

4.4 Quantum Search with an Approximate Test

Next, we use another result on Grover search using an *approximate test oracle*. This is an important object in the context of *offline* search algorithms such as *offline-Simon* [24] or *offline-DAHS* below using the DAHS circuit above.

We consider a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and an oracle O_f that uses an *ancillary state* $|\psi\rangle$. This ancillary state is required by the oracle to perform its computations, and it must be preserved. So one would expect to map: $|x\rangle |\psi\rangle \xrightarrow{O_f} (-1)^{f(x)} |x\rangle |\psi\rangle$. However, the exact O_f cannot be implemented, only an approximation O'_f .

Definition 4.1. An approximate test oracle for f , denoted O'_f , is a unitary operator that maps:

$$\forall x \in \{0, 1\}^n, |x\rangle |\psi\rangle \xrightarrow{O'_f} (-1)^{f(x)} |x\rangle |\psi\rangle + |\delta_x\rangle$$

where $|\psi\rangle$ is the predefined ancillary state which must be preserved by the oracle, and $\max_x \|\delta_x\| \leq \epsilon$.

That is, on each input vector, the oracle incurs a *uniformly bounded* error of amplitude ϵ (the same for all basis states). If this error is small enough, an amplitude amplification using O'_f instead of O_f will not “see” the difference. In our case, contrary to [24], we will also start from an *approximate initial state*. We summarize these needs with the following.

Theorem 4.2 (Quantum search with approximate test). *Let $\mathcal{A}, X, Y, \theta = \arcsin(\sqrt{\frac{|Y|}{|X|}})$, $t = \lfloor \frac{\pi}{4\theta} \rfloor$ be defined as in Section 2.2. Let f be a boolean function that tests if an output of \mathcal{A} is “good”, and let O'_f be an approximate oracle for f with error ϵ , using an ancillary state $|\psi\rangle$, as per Definition 4.1. Let $|\psi'\rangle$ be a quantum state such that $\| |\psi\rangle - |\psi'\rangle \| \leq \nu$. We run the following algorithm:*

$$\left((I \otimes \mathcal{A})(I \otimes O_0)(I \otimes \mathcal{A}^\dagger)O'_f \right)^t |\psi'\rangle,$$

that is, t iterations of “approximate” quantum search using O'_f , where I is the identity operator applied to the ancillary state $|\psi\rangle$. Then measuring the output yields a good result with probability greater than $(1 - t\epsilon - \nu)^2 \max(1 - |Y|/|X|, |Y|/|X|)$.

Proof. The proof uses a “hybrid argument” as in [14] or [5, Lemma 5]. We will consider the “perfect” run of the algorithm, that starts with the initial $|\psi\rangle$ and applies the perfect test O_f , and compare it with the “imperfect” one, that starts with $|\psi'\rangle$ and applies the imperfect test O'_f .

Let $|\psi'_k\rangle$ be the state after k iterations of the imperfect search, and $|\psi_k\rangle$ after the perfect search. Our goal is to bound $\| |\psi'_k\rangle - |\psi_k\rangle \|$. Let $U = (I \otimes \mathcal{A})O_0(I \otimes \mathcal{A}^\dagger)$, then the quantum search iterates are respectively UO'_f and UO_f . Before the first iteration, we have:

$$\| |\psi'_0\rangle - |\psi_0\rangle \| = \nu,$$

by definition of $|\psi'\rangle$. Next, for each $k \geq 0$:

$$\begin{aligned} \| |\psi'_{k+1}\rangle - |\psi_{k+1}\rangle \| &= \| UO'_f |\psi'_k\rangle - UO_f |\psi_k\rangle \| \\ &= \| O'_f |\psi'_k\rangle - O_f |\psi_k\rangle \| \\ &= \| O'_f (|\psi'_k\rangle - |\psi_k\rangle) + O'_f |\psi_k\rangle - O_f |\psi_k\rangle \|, \end{aligned}$$

and using the triangle inequality:

$$\begin{aligned} \| |\psi'_{k+1}\rangle - |\psi_{k+1}\rangle \| &\leq \| O'_f (|\psi'_k\rangle - |\psi_k\rangle) \| + \| O'_f |\psi_k\rangle - O_f |\psi_k\rangle \| \\ &\leq \| |\psi'_k\rangle - |\psi_k\rangle \| + \| O'_f |\psi_k\rangle - O_f |\psi_k\rangle \|. \end{aligned}$$

In order to bound the second term, we use the fact that O'_f induces a *uniformly bounded* error ϵ . More specifically, if $|\psi_k\rangle = \sum_x \alpha_x |x\rangle$, we have: $O'_f |\psi_k\rangle - O_f |\psi_k\rangle = \sum_x \alpha_x |\delta_x\rangle$ where $|\delta_x\rangle$ is the error induced by O'_f on the input state x . Then we have

$$\| O'_f |\psi_k\rangle - O_f |\psi_k\rangle \|^2 = \sum_x \alpha_x^2 \| |\delta_x\rangle \|^2 \leq \epsilon^2 \implies \| O'_f |\psi_k\rangle - O_f |\psi_k\rangle \| \leq \epsilon,$$

by definition of ϵ . Thus, each iteration adds an error ϵ .

After t iterations, we have $|\psi'_t\rangle = |\psi_t\rangle + |\psi_{\text{err}}\rangle$ where $\| |\psi_{\text{err}}\rangle \| \leq \nu + t\epsilon$. By the Cauchy-Schwarz inequality, we have:

$$|\langle \psi_t | \psi_{\text{err}} \rangle| \leq \| |\psi_t\rangle \| \| |\psi_{\text{err}}\rangle \| \leq \nu + t\epsilon.$$

Measuring $|\psi'_t\rangle$, we project on $|\psi_t\rangle$ with a probability greater than:

$$(1 - |\langle \psi_t | \psi_{\text{err}} \rangle|)^2 \geq (1 - \nu - t\epsilon)^2$$

and then, by amplitude amplification, we have a probability greater than $\max(1 - |Y|/|X|, |Y|/|X|)$ to measure a “good” element. \square

It can seem surprising to require a *uniformly bounded* error ϵ . Indeed, in a classical exhaustive search with a single solution, we can afford a constant probability of false negative (not recognizing the solution), and still obtain a constant probability of success, as we expect to look at the solution only once.

Our classical intuition then dictates that the same should be true of a quantum search: we could afford a much higher probability of false negatives than of false positives. We found that this was not the case, due to the stateful nature of the search. Indeed, taking different ϵ_{FP} and ϵ_{FN} changes the error term ϵ added at iteration k to a term:

$$\sqrt{\sin^2((2k+1)\theta)\epsilon_{FN}^2 + \cos^2((2k+1)\theta)\epsilon_{FP}^2}.$$

In order to have a constant probability of success in the end, we must measure a state in which a constant proportion of the amplitude is on the solution. That is, $(2k+1)\theta$ must be sufficiently close to $\frac{\pi}{2}$ to have $\sin^2((2k+1)\theta)$ constant. This means that in the last iterations, the error term is close to ϵ_{FN} . (Although in the first iterations, it was close to ϵ_{FP} .) Over all the iterations, the solution and the bad elements both capture roughly (up to a constant) the same amount of amplitude, and so both terms ϵ_{FN} and ϵ_{FP} will have roughly the same effect.

Corollary 4.1. *If $\nu \leq \frac{1}{4}$ and $\epsilon \leq \frac{1}{4t}$, then the procedure of Theorem 4.2 succeeds with probability $\geq \frac{1}{8}$.*

Proof. We use simply that $\sin^2((2t+1)\theta) \geq \frac{1}{2}$ and $1 - t \max(\epsilon_{FN}, \epsilon_{FP}) - \epsilon \geq 1 - \frac{1}{4} - \frac{1}{4} = \frac{1}{2}$. \square

We stress that Theorem 4.2 and Corollary 4.1 contain two important features, that are both crucial for solving the SLS problem:

- the test oracle can be imperfect, as soon as the error is *uniformly bounded* and *exponentially small* (when t is exponential);
- the ancillary state used by the approximate test can also be approximated, and the corresponding error ν can be a constant.

4.5 The Offline-DAHS Algorithm

In this section, we describe an offline-DAHS algorithm that will help us solve the SLS with classical queries, based on the reversible circuit DAHS.

The algorithm looks for a pair of shifted functions $g(\cdot) = f(\cdot + s)$ over an abelian group, when g is fixed and f goes through a family $(f_i)_{i \in I}$. We will consider a simple version of this problem, in which the group is \mathbb{Z}_{2^n} , all functions are injective and admit distinct image sets.

Problem 4.4 (Finding a shifted pair, injective case). *Let $g : \mathbb{Z}_{2^n} \rightarrow X$ be a function, and $f_i : \mathbb{Z}_{2^n} \rightarrow X$ be a family of functions indexed by I , such that:*

- g and all f_i are injective;
- there exists a single $i_0 \in I$ and a shift s such that $\forall x \in \mathbb{Z}_{2^n}, g(x) = f_{i_0}(x + s)$;
- $\bigcup_{i \neq i_0} \text{Im}(f_i)$ and $\text{Im}(g)$ are disjoint.

Then find i_0 .

4.5.1 High-level Description

First of all, we describe offline-DAHS in a generic way, following the layout of the offline-Simon algorithm by Bonnetain, Hosyamada, Naya-Plasencia, Sasaki and Schrottenloher [24]. Note that [24] already proposed to combine their algorithm with Kuperberg's, but did not analyze the resulting algorithm nor its time complexity. Originally, the offline-Simon algorithm, which itself follows Grover-meet-Simon [86], combines a quantum search and a quantum circuit for Simon's algorithm. Roughly speaking, offline-DAHS is obtained by replacing Simon's algorithm by Kuperberg's (which is why we needed to define a quantum circuit for it).

Description. We use a quantum search for the right index $i_0 \in I$. Testing a given i means finding whether f_i is a shift of g . For this, we use the circuit DAHS. Recall that DAHS takes in input the sample database of $(f_i, g): |\psi_{f_i, g}\rangle^{\otimes t}$ and writes a single output bit. Thus, the naive Grover search would reconstruct the database at each iteration, then compute DAHS and return the database to $|0\rangle$.

However, due to the asymmetric nature of the problem, the function g in the database remains the same from one iteration to the next. Let us introduce the sample database of $(0, g)$:

$$|\psi_{0, g}\rangle^{\otimes t} = \left(\sum_x |0\rangle |x\rangle |0\rangle + |1\rangle |x\rangle |g(x)\rangle \right)^{\otimes t}.$$

It contains all the data on g that we need for the successive iterations of quantum search. The algorithm has then two steps.

- Precomputation step: construct $|\psi_{0,g}\rangle^{\otimes t}$.
- Search step: run the quantum search for i_0 . At each search iterate, compute f_i inside the database to obtain the state $|\psi_{f_i,g}\rangle^{\otimes t}$, run the circuit DAHS, then compute f_i again to return to the state $|\psi_{0,g}\rangle^{\otimes t}$. Due to the *approximate* nature of DAHS, this is a quantum search with an approximate test (Theorem 4.2).

We have bounded the error of DAHS and given its complexity in Theorem 4.1. As long as the initial state $|\psi_{0,g}\rangle^{\otimes t}$ can be constructed exactly, this becomes an easy instance of Theorem 4.2, and we deduce the following result.

Proposition 4.1. *Let $\alpha = 2\log_2 3$. Problem 4.4 can be solved within a time $\tilde{O}(2^{\sqrt{\alpha n}}\sqrt{I})$ using $\tilde{O}(2^{\sqrt{\alpha n}})$ qubits, with constant probability. There are $\tilde{O}(2^{\sqrt{\alpha n}})$ queries to g and $\tilde{O}(2^{\sqrt{\alpha n}}\sqrt{I})$ queries to f (in superposition for both).*

The fact that the queries to O_g are now performed only in the precomputation step is the reason for the *offline* denomination. Note that the polynomial factors in the \tilde{O} in Proposition 4.1 are not negligible, and depend on $\log_2 I$ and n together.

4.5.2 Approximate Promise

We now go into technical details specific to offline-DAHS and not discussed in the previous literature.

So far, our analysis has assumed that we could start from an *exact* sample database $|\psi_{0,g}\rangle^{\otimes t}$ as defined above. But it is not the case in the SLS problem. Here, g will be the secretly shifted Legendre symbol $(\frac{s+x}{P})$. This function is defined on \mathbb{Z}_P , but it is queried on an interval of length M .

In particular, in order to run the algorithm as expected from Proposition 4.1, we would need sample states of the form:

$$|\psi_{0,g}^{\text{exact}}\rangle = \sum_{0 \leq x \leq M-1} |0\rangle |x\rangle |0\rangle + \sum_{-s \leq x \leq M-s-1} |1\rangle |x\rangle |g(x)\rangle,$$

which would allow for a total interference between the matching values of $f(x)$ and $g(x)$, when querying the good f .

However, since we do not know the value of s , such states cannot be created. Instead, we rely on *approximate* sample states:

$$|\psi_{0,g}^{\text{approx}}\rangle = \sum_{0 \leq x \leq M-1} |0\rangle |x\rangle |0\rangle + \sum_{0 \leq x \leq M-1} |1\rangle |x\rangle |g(x)\rangle.$$

There is an error vector $|\psi_{0,g}^{\text{err}}\rangle$ such that $|\psi_{0,g}^{\text{approx}}\rangle = |\psi_{0,g}^{\text{exact}}\rangle + |\psi_{0,g}^{\text{err}}\rangle$:

$$\| |\psi_{0,g}^{\text{err}}\rangle \| = \left\| \sum_{-s \leq x \leq -1} |1\rangle |x\rangle |g(x)\rangle - \sum_{M-s \leq x \leq M-1} |1\rangle |x\rangle |g(x)\rangle \right\| \leq \sqrt{\frac{2s}{4M}},$$

and we can bound the distance between the “exact” database of $(0, g)$ and the “approximate” one:

$$\left\| \left(|\psi_{0,g}^{\text{approx}}\rangle \right)^{\otimes t} - \left(|\psi_{0,g}^{\text{exact}}\rangle \right)^{\otimes t} \right\|^2 \leq t \| |\psi_{0,g}^{\text{err}}\rangle \|^2 = \frac{ts}{2M}.$$

This gives a total “starting error” $\nu = \sqrt{\frac{ts}{2M}}$. As we have seen in Theorem 4.2, we need this error to be constant, thus s needs to be subexponentially smaller than M for the algorithm to work.

4.5.3 Application to the Legendre Symbol

Given a sequence of $M = 2^n$ successive outputs $(\frac{s+x}{P})$ of the Legendre PRF, we define a function $g(x) = L_{s+x}$ on \mathbb{Z}_M . Next, we choose an integer n' such that $2^{n'} < M$ and we write: $s = s_1 + 2^{n'} s_2$, where $s_1 < 2^{n'}$. For a given s_2 , we define $f(x) = L_{x+2^{n'} s_2}$. Then there exists a single s_2 such that $g(x) = f(x + s_1)$.

Note that we need a subexponential gap between $2^{n'}$, where n' is the number of bits of the secret handled by the (offline) DAHS subroutine, and $M = 2^n$, the amount of data given. It is due to the approximation discussed in Section 4.5.2. The DAHS subroutine still runs with labels of $n = \log_2 M$ bits.

By taking L -sequences of length $\geq 3 \log_2 P$, our Heuristic 4.1 ensures that the functions are injective, and that they have distinct image sets: two L -sequences cannot collide randomly. Thus, we have an instance of Problem 4.4 with an approximate promise, and we can apply Theorem 4.2.

We use the M classical queries to build the approximate sample states $|\psi_{0,g}^{\text{approx}}\rangle$, and then, we run a Grover search over the remaining secret s_2 . Building the sample database from the classical queries is costly ($\mathcal{O}(Mm)$ quantum gates for each sample), but done only once in the precomputation step.

Let $\alpha = 2 \log_2 3$. In order to make the starting error ν smaller than $\frac{1}{4}$, we must take t labels where $\sqrt{t 2^{n'} / (2M)} \leq \frac{1}{4} \implies t \leq M / 2^{n'+3}$. But since $t = \tilde{\mathcal{O}}(2^{\sqrt{\alpha n}})$, this means the circuit DAHS can only recover n' bits of s , where $n' + \sqrt{\alpha n} = \log_2 M = n$. Thus $n' = n - \sqrt{\alpha n}$. The remaining $(\log_2 P - n')$ bits must be searched with Grover’s algorithm.

Theorem 4.3. *Given a sequence of M outputs of the Legendre PRF, we can solve the SLS problem using $\tilde{\mathcal{O}}\left(2^{\sqrt{\alpha \log_2 M}}\right)$ qubits, in quantum time:*

$$\tilde{\mathcal{O}}\left(M 2^{\sqrt{\alpha \log_2 M}}\right) + \tilde{\mathcal{O}}\left(2^{\frac{3}{2} \sqrt{\alpha \log_2 M}} \sqrt{\frac{P}{M}}\right).$$

Proof. We use Corollary 4.1 and Theorem 4.1. We run a quantum search with an approximate test (the circuit DAHS) and an approximate starting state (the approximate sample states).

The analysis of DAHS assumes an exact ancillary state $|\psi_{0,g}\rangle^{\otimes t}$. Thus, Theorem 4.2 is crucial here to ensure that the approximate starting state does not disrupt the algorithm. \square

The minimum occurs when the two terms are equal, which constrains: $\sqrt{\alpha \log_2 M} + \log_2 M = \frac{3}{2}\sqrt{\alpha \log_2 M} + \frac{1}{2}\log_2 P - \frac{1}{2}\log_2 M \implies \log_2 M = \frac{1}{3}\sqrt{\alpha \log_2 M} + \frac{1}{3}\log_2 P$, up to the polynomial factors. We get $\sqrt{\log_2 M} = \sqrt{\frac{1}{3}\log_2 P + \frac{\alpha}{36}} + \frac{\sqrt{\alpha}}{6} = \sqrt{\frac{1}{3}\log_2 P} + \mathcal{O}(1)$ and $\log_2 M = \frac{1}{3}\log_2 P + \frac{\sqrt{\alpha}}{3\sqrt{3}}\sqrt{\log_2 P} + \mathcal{O}(1)$. This gives a time complexity of order: $P^{1/3}2^{\frac{4}{3}}\sqrt{\alpha}\sqrt{\log_2 P}^{1/3}$.

4.6 Detailed Complexities

We give in Table 4.1 a recap of the different algorithms.

Table 4.1: Comparison of classical and quantum algorithms to solve the SLS problem, including previous works and our new results. We note $m = 3 \lceil \log_2 P \rceil$, $\alpha = 2 \log_2 3$, $L \leq (\log_2 P)^2$ the time to compute a Legendre symbol, and we omit constant factors.

Method	Queries	Time	Memory	Source
Classical algorithms				
Pollard's rho	$\sqrt{P}m$	$L\sqrt{P}m$	m	[80]
Table	M	$M^2 + Pm^2/M^2$	M^2/m	[17]
Table	M	$M^2 + Pm \log_2 m/M^2$	M^2	[74]
Quantum algorithms				
Sup. queries	2	$\text{poly}(m)$	$\text{poly}(m)$	[42]
Table (QRACM)	M	$M^2 + m^2\sqrt{P}/M^2L$	M^2 QRACM	[74]
Grover search	m	$\log_2 m\sqrt{PL}$	m qubits	4.2.2
Distinguished points	M	$M^2 + \frac{\sqrt{P}}{M^{1/3}}m^{\frac{11}{6}}\log_2 \log_2 M$	M classical + m qubits	4.2.3
Offline-DAHS	M	$\tilde{O}\left(\frac{M2\sqrt{\alpha \log_2 M} +}{2^{\frac{3}{2}}\sqrt{\alpha \log_2 M}}\sqrt{\frac{P}{M}}\right)$	$\tilde{O}(2\sqrt{\alpha \log_2 M})$ qubits	4.5.3

4.7 Conclusion

In this chapter, we presented two quantum algorithms for solving the Legendre hidden shift problem (SLS) when classical queries are given, and without

quantum RAM. The first one (distinguished table-based collisions) allows to reach an advantage against Grover's algorithm when more data is given. The second one is the *offline Kuperberg's algorithm*. It is an interesting method, notably the only one reaching a time-memory product below $\mathcal{O}(\sqrt{P})$.

Although a very efficient algorithm exists when superposition queries are allowed [42], it does not seem amenable to an *offline* version, which requires to define reduced instances of the problem (e.g., guessing part of the secret and finding the remaining bits by searching for a shift). Nevertheless, the algebraic properties of the Legendre symbol might still find a use in this context, and we leave this as an open question.

Chapter 5

Quantum Boomerang Attacks and Some Applications

In this chapter, we consider boomerang attacks, which form a particular type of differential attacks introduced by Wagner in [118]. We study how to build an efficient quantum version of boomerang attacks and of mixing boomerang attacks [49] recently introduced in the context of AES. We propose, for the first time, efficient *quantum boomerang attacks*, and we apply them to several reduced-round versions of well known ciphers.

The quantum attacks studied in this chapter are also based on quantum search. One should note that, since quantum search always provides a quadratic speedup, our procedures admit a quadratic speedup at best. By comparing with the quadratic speedup of exhaustive key search, it follows that we will not be able to attack more rounds than in the classical setting: any quantum attack in our framework can be converted back into a valid classical attack. Though this result can seem rather negative at first sight, our new attacks, like previous works with similar conclusions [27, 77], tend to show that block ciphers should be assumed to retain half of their bits of classical security against quantum adversaries, even when this security has been reduced with respect to the generic key search.

It is based on a joint work with María Naya-Plasencia and André Schrottenloher published at Selected Areas in Cryptography in 2021 [58].

Contents

5.1	Classical Boomerang Attacks	82
5.1.1	The Classical Boomerang Attack	83
5.1.2	Mixing Boomerang Attacks	86
5.2	Quantum Boomerang Attacks	88
5.2.1	Quantum Boomerang Distinguisher	88
5.2.2	Quantum Boomerang Last-rounds Attack	88
5.2.3	Quantum Mixing Boomerang Distinguisher	91
5.3	Application to SAFER	92
5.3.1	Description of the Cipher	92
5.3.2	5-Round Classical Boomerang Attack	93
5.3.3	Quantum Boomerang Attack	96

5.4	Application to KASUMI	98
5.4.1	Description of the Cipher	98
5.4.2	Classical Attack	98
5.4.3	Quantum Attack	99
5.5	Application to Related-key AES	100
5.5.1	Description of AES	100
5.5.2	Classical Attack	102
5.5.3	Quantum Attack	103
5.6	Application to 5-Round AES	108
5.6.1	The boomerang	108
5.6.2	Friend Pairs	110
5.6.3	Value-difference Correspondence on AES S-box . . .	111
5.6.4	Classical Attack	111
5.6.5	Quantum Attack	113
5.7	Conclusion	113

5.1 Classical Boomerang Attacks

In this section, we introduce the classical Boomerang attack from [118] and the Mixing Boomerang attack from [49]. We give generic formulas for their time complexity depending on the parameters of the cipher attacked.

Throughout this chapter, we consider an n -bit block cipher E , with unknown key k . Standard block ciphers are built by iterating a round function, and we will sometimes decompose E into subciphers, e.g., $E = E_2 \circ E_1$ where E_1 forms the r_1 first rounds and E_2 the r_2 last rounds.

As specified in Section 3.3.2, boomerang cryptanalysis is a subset of differential cryptanalysis, which studies the propagation of differences in a cipher. We recall that for a cipher (or subcipher) E , and a pair of differences (α, β) , we say that $\alpha \rightarrow \beta$ is a *differential* for E of probability:

$$\Pr(\alpha \xrightarrow{E} \beta) = \mathbb{P}_X(E(X \oplus \alpha) \oplus E(X) = \beta).$$

Since E is a keyed function, the probability of a differential depends on the choice of key. In the analysis, it is usually computed on average over the key. When running an attack, we consider a black box with a fixed given key. We then assume that the differential probability is equal to the analyzed average (even if there is a small variation in practice, we may run the attack again with another estimate – this is valid for classical as well as quantum attacks).

5.1.1 The Classical Boomerang Attack

We briefly describe the *boomerang distinguisher* introduced in [118] and the last-round key-recovery attack that can be based on it. The notation that we introduce here (E, p, q, α, \dots) will be kept throughout the chapter.

Boomerang Distinguisher. As above, let E be a block cipher of block size n and key size k , that can be decomposed into: $E = E_2 \circ E_1$. We assume that each part has a high-probability differential: $\alpha \rightarrow \beta$ for E_1 and $\delta \rightarrow \gamma$ for E_2^{-1} .

$$\Pr(\alpha \xrightarrow{E_1} \beta) = p_{\downarrow}, \quad \Pr(\beta \xrightarrow{E_1^{-1}} \alpha) = p_{\uparrow}, \quad \Pr(\delta \xrightarrow{E_2^{-1}} \gamma) = q .$$

The distinguisher is given in Algorithm 5.1. It uses $\frac{4}{p_{\downarrow}p_{\uparrow}q^2}$ encryptions, decryptions, and negligible memory. We can check its correctness as follows (see also Figure 5.1 for the notations):

Step 3: using the differential on E_1 ; with probability p_{\downarrow} , $E_1(P_1) \oplus E_1(P_2) = \beta = E_2^{-1}(C_1) \oplus E_2^{-1}(C_2)$

Step 4: using the differential on E_2^{-1} ; with probability q^2 , $E_2^{-1}(C_3) \oplus E_2^{-1}(C_4) = \gamma$ and $E_2^{-1}(C_4) \oplus E_2^{-1}(C_2) = \gamma$. Thus, by summing these equations: $E_2^{-1}(C_3) \oplus E_2^{-1}(C_4) = E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$.

Step 5: since we have established $E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$ with probability $p_{\downarrow}q^2$, it remains to satisfy the differential on E_1^{-1} , and we obtain $P_3 \oplus P_4 = \alpha$ with probability $p_{\uparrow} \times p_{\downarrow}q^2$.

Then, the full path is satisfied with probability $(p_{\downarrow}p_{\uparrow}q)^2$ instead of 2^{-n} for a random permutation, and making $1/(p_{\downarrow}p_{\uparrow}q)^2$ trials is enough to determine the case. Usually, we also have $p_{\downarrow} = p_{\uparrow} = p$ and this formula simplifies into $(pq)^2$.

Algorithm 5.1 Boomerang Distinguisher

Input: oracle access to $E = E_2 \circ E_1$, and its inverse (or a random permutation)

Output: “ E ” or “Random”

- 1: **Repeat** $(p_{\uparrow}p_{\downarrow}q)^{-2}$ **times** \triangleright (probability of success of $\frac{1}{2}$)
 - 2: Select P_1 at random and set $P_2 = P_1 \oplus \alpha$
 - 3: Encrypt: $C_1 = E(P_1)$ and $C_2 = E(P_2)$
 - 4: Compute: $C_3 = C_1 \oplus \delta$ and $C_4 = C_2 \oplus \delta$
 - 5: Decrypt: $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$
 - 6: **if** $P_4 = P_3 \oplus \alpha$ **then return** “ E ”
 - 7: **EndRepeat**
 - 8: **return** “Random”
-

Key Recovery on the Last Round. We now explain how to build a key-recovery attack using this boomerang distinguisher.

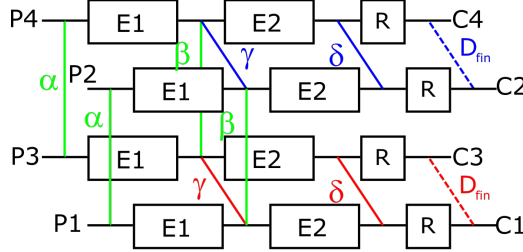


Figure 5.1: Last-round key recovery

We append one (or more) additional rounds R to the cipher, which is now $E = R \circ E_2 \circ E_1$ (if the additional round is at the beginning, consider E^{-1}). Let D_{fin} denote the set of differences that can be obtained from δ after the additional rounds, p_{out} the probability that we get δ back from an element in D_{fin} by computing the last rounds backwards, and k_{out} the number of key bits involved in these additional rounds. We need D_{fin} to be a vector space. This is usually the case as most of the time, non-zero components of δ become unknown after a passage through an S-Box. This property is used to make data *structures* of the form $\{X \oplus \Delta / \Delta \in D_{\text{fin}}\}$, from which we can extract quadratically many pairs with differences in D_{fin} : $\forall \Delta_1, \Delta_2, (X \oplus \Delta_1) \oplus (X \oplus \Delta_2) = \Delta_1 \oplus \Delta_2 \in D_{\text{fin}}$.

To simplify for now, we assume that $\frac{1}{\sqrt{p_{\text{out}}pq}} < |D_{\text{fin}}|$, in which case a single structure is needed. Otherwise when $\frac{1}{\sqrt{p_{\text{out}}pq}} \geq |D_{\text{fin}}|$, we will actually generate multiple structures with $|D_{\text{fin}}|$ ciphertexts each, so that they form a total of $\frac{1}{p_{\text{out}}(pq)^2}$ pairs.

Algorithm 5.2 detects a good guess of the k_{out} key bits using the boomerang distinguisher. We will now explain its correctness and compute its average time complexity (up to a constant factor).

We start by generating a structure S of $\frac{1}{\sqrt{p_{\text{out}}pq}}$ ciphertexts with differences in D_{fin} . This first step costs a time and memory: $\frac{1}{\sqrt{p_{\text{out}}pq}}$.

Now we partially decrypt these ciphertexts and obtain pairs with difference δ . Among the $\frac{1}{p_{\text{out}}(pq)^2}$ pairs (C_i, C_j) in S , we expect $\frac{1}{(pq)^2}$ of them to be such that $R^{-1}(C_i) \oplus R^{-1}(C_j) = \delta$, where R involves the actual partial key K_{out} used in the last rounds. Then we define $C'_i = E(E^{-1}(C_i) \oplus \alpha)$ and $C'_j = E(E^{-1}(C_j) \oplus \alpha)$. By the boomerang property, with probability $(pq)^2$, we will have $R^{-1}(C'_i) \oplus R^{-1}(C'_j) = \delta$. This difference gets then mapped to D_{fin} with probability 1. Obtaining the list L_{pairs} costs $\frac{2}{\sqrt{p_{\text{out}}pq}}$ data and memory and $\frac{1}{\sqrt{p_{\text{out}}pq}} \log\left(\frac{1}{\sqrt{p_{\text{out}}pq}}\right)$ computations for sorting the C'_i (since D_{fin} is a vector space, we can sort according to its cosets). This sorting allows to extract efficiently the pairs C'_i, C'_j such that $C'_i \oplus C'_j \in D_{\text{fin}}$ at Step 4 of Algorithm 5.2.

Algorithm 5.2 Boomerang last-round attack

- Input:** oracle access to $E = R \circ E_2 \circ E_1$
- Output:** guess of partial key K_{out} involved in the round(s) R
- 1: Generate a *structure* S of $\frac{1}{\sqrt{p_{\text{out}}pq}}$ ciphertexts of the form: $S \subseteq \{X \oplus \Delta, \Delta \in D_{\text{fin}}\}$
 - 2: $L_{\text{pairs}} \leftarrow \emptyset$
 - 3: Compute all the $C' = E(E^{-1}(C) \oplus \alpha)$ for all $C \in S$, sort S by values of C'
 - 4: **ForEach** pair C'_i, C'_j such that $C'_i \oplus C'_j \in D_{\text{fin}}$
 - 5: $L_{\text{pairs}} \leftarrow L_{\text{pairs}} \cup \{(C_i, C_j, C'_i, C'_j)\}$
 - 6: **EndFor** ▷ Here $|L_{\text{pairs}}| = \frac{|D_{\text{fin}}|}{2^n} \frac{1}{p_{\text{out}}(pq)^2}$
 ▷ Note that S being sorted, these pairs are computed efficiently
 - 7: $L_{\text{triplets}} \leftarrow \emptyset$
 - 8: **ForEach** pair $C_i, C_j, C'_i, C'_j \in L_{\text{pairs}}$
 - 9: **ForEach** possible value K_{out} of the k_{out} bits of key
 ▷ $2^{k_{\text{out}}}$ trials, $2^{k_{\text{out}}} p_{\text{out}}^2$ solutions
 - 10: If both (C_i, C_j) and (C'_i, C'_j) lead to a difference δ through R^{-1} under K_{out}
 - 11: $L_{\text{triplets}} \leftarrow L_{\text{triplets}} \cup \{(C_i, C_j, K_{\text{out}})\}$
 - 12: **EndFor**
 - 13: **EndFor** ▷ Here $|L_{\text{triplets}}| = \frac{|D_{\text{fin}}|}{2^n} \frac{1}{(pq)^2} 2^{k_{\text{out}}} p_{\text{out}}$
 - 14: **return** all guesses of K_{out} in the triplet list
-

There are enough quartets (C_i, C_j, C'_i, C'_j) in L_{pairs} to ensure that, for the good key guess of K_{out} , one of them is an actual boomerang quartet. Let C_{out} be the time complexity to obtain all valid keys K_{out} for a given pair (C_i, C_j) (Step 9 in Algorithm 5.2 is the naive way to do so, but there is usually a better algorithm as we will see in the applications). Since we expect on average $2^{k_{\text{out}}} p_{\text{out}}^2$ such keys, we have $C_{\text{out}} \geq 2^{k_{\text{out}}} p_{\text{out}}^2$.

After having obtained the list L_{pairs} of size $\frac{|D_{\text{fin}}|}{2^n} \frac{1}{p_{\text{out}}(pq)^2}$, we find all possible key guesses for each of these pairs i.e., all possible K_{out} such that (C_i, C_j) and (C'_i, C'_j) lead to a difference δ through R^{-1} (in which case they form a boomerang quartet). This costs a time $\frac{|D_{\text{fin}}|}{2^n} \frac{1}{p_{\text{out}}(pq)^2} C_{\text{out}}$. The list of triplets obtained is of size: $\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2}$, and we know that the actual key K_{out} occurs in one of these triplets, because one of the pairs went through the boomerang. Thus, if we have: $\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2} \leq 2^{k_{\text{out}}}$, we have reduced the number of possibilities for K_{out} . We perform an exhaustive search over the remaining $k - k_{\text{out}}$ bits of key. We obtain a key-recovery attack of total time complexity:

$$\frac{1}{\sqrt{p_{\text{out}}pq}} \log \left(\frac{1}{\sqrt{p_{\text{out}}pq}} \right) + \frac{|D_{\text{fin}}|}{2^n} \frac{1}{p_{\text{out}}(pq)^2} \left(C_{\text{out}} + p_{\text{out}}^2 2^k \right), \quad (5.1)$$

in number of queries to E and E^{-1} or evaluations of the cipher. Note that C_{out} is counted relatively to the cost of an evaluation of E . The procedure used $\max\left(\frac{2}{\sqrt{p_{\text{out}}pq}}, \frac{|D_{\text{fin}}| 2^{k_{\text{out}}} p_{\text{out}}}{2^n (pq)^2}\right)$ memory and $\frac{2}{\sqrt{p_{\text{out}}pq}}$ data.

Remark. Another situation commonly encountered in boomerang attacks is when two (or more) boomerang pairs occur within the trials. In that case, it is possible to recognize immediately the good guess of K_{out} , as the only one that appears in two (or more) triples in the list L_{triplets} .

5.1.2 Mixing Boomerang Attacks

We now give some details on a variant of boomerang attacks [49], which is related to mixture distinguishers.

Distinguisher. As in Section 5.1.1, the cipher is decomposed as $E = E_2 \circ E_1$ and we assume that a good (truncated) differential $\alpha \rightarrow \beta$ exists on E_1 with probability p_{\downarrow} forwards and with probability p_{\uparrow} backwards. We further assume that E_2 can be divided into a “left” and a “right” part $E_2 = (E_{2,L}, E_{2,R})$ (see the notations on Figure 5.2).

The distinguisher relies on the independence of $E_{2,L}$ and $E_{2,R}$. It requires $\frac{4}{p_{\downarrow}p_{\uparrow}}$ encryptions-decryptions, operations and negligible memory. We repeat:

1. select P_1 at random, set $P_2 = P_1 \oplus \alpha$ and encrypt $C_1 = E(P_1)$ and $C_2 = E(P_2)$;
2. compute $C_3 = (C_{1,L}, C_{2,R})$ and $C_4 = (C_{2,L}, C_{1,R})$;
3. decrypt: $P_3 = E^{-1}(C_3)$ and $P_4 = E^{-1}(C_4)$.

Indeed, in Step 1 we have $E_1(P_1) \oplus E_1(P_2) = \beta$ with probability p_{\downarrow} . By definition of E , $E_1(P_i) = E_2^{-1}(C_i)$, so in Step 2, we know that $E_2^{-1}(C_1) \oplus E_2^{-1}(C_2) = \beta$. If we separate $\beta = \beta_L || \beta_R$, this rewrites:

$$\begin{cases} E_{2,L}^{-1}(C_{1,L}) \oplus E_{2,L}^{-1}(C_{2,L}) = \beta_L \\ E_{2,R}^{-1}(C_{1,R}) \oplus E_{2,R}^{-1}(C_{2,R}) = \beta_R \end{cases} \quad (5.2)$$

Thus, by definition of C_3, C_4 , we also have $E_2^{-1}(C_3) \oplus E_2^{-1}(C_4) = \beta$. Finally in Step 3, with probability p_{\uparrow} , β gets mapped to α . After $(p_{\downarrow}p_{\uparrow})^{-1}$ iterations, we obtain a valid quartet with probability $1/2$.

Attack on First Round. We append one or more additional rounds R before those covered by the distinguisher. We write $E = E_2 \circ E_1 \circ R$. We let D_{start} denote the set of differentials that can lead to α for the additional rounds R , p_{in} the probability to get α from an element of D_{start} and k_{in} the number of bits of the key involved in the additional rounds.

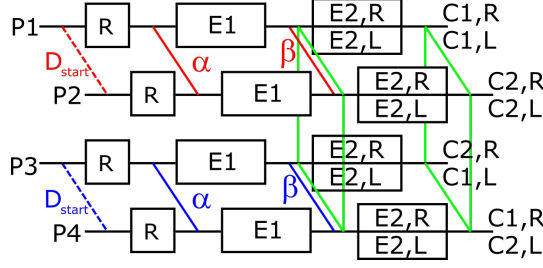


Figure 5.2: First-round mixing boomerang attack.

Step 1: Generating Pairs. We generate a list of $\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ pairs of plaintexts (P_1, P_2) such that their difference is in D_{start} . Again, there can be one or multiple structures, but as the term $\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ (the number of pairs) will appear in the complexity, there is no difference between these two cases.

For each pair (P_1, P_2) we compute:

$$(C_1 = E(P_1), C_2 = E(P_2)), \quad (C_3 = (C_{1,L}, C_{2,R}), C_4 = (C_{2,L}, C_{1,R})), \\ (P_3 = E^{-1}(C_3), P_4 = E^{-1}(C_4)). \quad (5.3)$$

For each pair, we have $R(P_1) \oplus R(P_2) = \alpha$ with probability p_{in} . Then, $(E_2 \circ E_1)^{-1}(C_3) \oplus (E_2 \circ E_1)^{-1}(C_4) = \alpha$ with probability $p_{\downarrow}p_{\uparrow}$. Thus, we expect that one pair satisfies the boomerang distinguisher. By decrypting through R we fall back on D_{start} .

Thus, we keep only the pairs (P_1, P_2) such that $P_3 \oplus P_4 \in D_{\text{start}}$, among which the pair that went through the boomerang must remain. We get a list of $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ pairs at a cost of $\frac{4}{p_{\text{in}}p_{\downarrow}p_{\uparrow}}$ computations and data.

Step 2: Sieving Key Bits. For each kept pairs (P_1, P_2) , we determine the values of K_{in} (the k_{in} bits of key that intervene in R) such that (P_1, P_2) and (P_3, P_4) both lead to a difference α through R . The average expected number of possible values for K_{in} for each pair is thus $2^{k_{\text{in}}}(p_{\text{in}})^2$. We let C_{in} denote the cost of finding all these possible values.

At a cost of $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}} C_{\text{in}}$ computations, we get a list of $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\downarrow}p_{\uparrow}} 2^{k_{\text{in}}} p_{\text{in}}$ elements $(P_1, P_2, K_{\text{in}})$. If

$$\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\downarrow}p_{\uparrow}} 2^{k_{\text{in}}} p_{\text{in}} < 2^{k_{\text{in}}}, \quad (5.4)$$

then we have reduced the number of possible values for K_{in} and we do an exhaustive search of the remaining $k - k_{\text{in}}$ bits of key. The attack has a total time complexity:

$$\frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}} + \frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\text{in}}p_{\downarrow}p_{\uparrow}} (C_{\text{in}} + p_{\text{in}}^2 2^k), \quad (5.5)$$

Algorithm 5.3 Quantum Boomerang Distinguisher

Input: *superposition* oracle access to $E = E_2 \circ E_1$, or a random permutation
Output: “ E ” or “Random”
1: **Repeat** $\frac{\pi}{4}(pq)^{-1}$ **times**
2: Select P_1 at random
3: **If** $E^{-1}(E(P_1) \oplus \delta) = E^{-1}(E(P_1 \oplus \alpha) \oplus \delta) \oplus \alpha$ **then**
4: the state (P_1) is “good” **Else** the state (P_1) is “bad”
5: **EndRepeat**
6: Measure and check if a solution was found
7: **If** it was found **return** “ E ” **Else Return** “Random”

in number of queries to E and E^{-1} or evaluations of the cipher. Note that C_{in} is counted relatively to the cost of an evaluation of E .

5.2 Quantum Boomerang Attacks

In this section, we adapt the attacks presented in Section 5.1 to the quantum setting. This form a framework for applications.

5.2.1 Quantum Boomerang Distinguisher

Similarly to the differential attacks from [77], the boomerang distinguisher of Algorithm 5.1 can be accelerated *in the Q2 setting* (Algorithm 5.3). The **Repeat** loop becomes a quantum search with $\frac{\pi}{4}(pq)^{-1}$ iterations looking for an element P_1 such that $P_1, P_1 \oplus \alpha$ forms a boomerang pair. If E satisfies the expected property, such a P_1 will be found, otherwise measuring the result will give us an invalid pair. We make in total $\frac{\pi}{4}(pq)^{-1} \times 8$ queries to E and E^{-1} (each query is made twice for reversibility), which are in superposition since the quantum search space is on the value P_1 .

5.2.2 Quantum Boomerang Last-rounds Attack

Recall that the last-rounds attack of Section 5.1.1 is in two phases: first, we use the boomerang property to sieve the k_{out} bits that intervene in the last rounds R . Second, we perform an exhaustive search on the $k - k_{\text{out}}$ remaining key bits.

For now, let us put aside the small constant factors. We count all quantum times in evaluations of E or E^{-1} and consider a Q2 query to E or E^{-1} to cost the same. The generic key-recovery is Grover’s exhaustive search, with a time $2^{k/2}$ and a negligible number of Q1 queries. Our first idea is to go below that using the sieve on the k_{out} bits of key that the boomerang property gives. If we can produce the superposition of valid K_{out} with a time complexity below

$2^{(k-k_{\text{out}})/2}$, and if there are strictly less than $2^{k_{\text{out}}}$ elements in this superposition, then we can do better than Grover search: we use amplitude amplification on an algorithm that first samples a K_{out} , then tries to complete the key in time $2^{(k-k_{\text{out}})/2}$.

In order to produce a possible value of K_{out} , we look for a pair C_1, C_2 such that:

- $C_1 \oplus C_2 \in D_{\text{fin}}$ and $C'_1 \oplus C'_2 \in D_{\text{fin}}$, where $C'_i = E(E^{-1}(C_i) \oplus \alpha)$;
- there exists a subkey K_{out} such that (C_1, C_2) and (C'_1, C'_2) decrypt to the difference δ through R^{-1} under K_{out} .

Finding such a valid pair C_1, C_2 is a collision search problem, solved with BHT algorithm. Let us assume that $|D_{\text{fin}}| \geq 2^{n/3}$, which corresponds to the case where a single structure is enough. A structure contains $|D_{\text{fin}}|^2$ pairs, and the constraint $C'_1 \oplus C'_2 \in D_{\text{fin}}$ selects a proportion $\frac{|D_{\text{fin}}|}{2^n}$ of them. Thus there is at least one solution.

Algorithm 5.4 Quantum Last-rounds Key-recovery

Input: *superposition* oracle access to E and E^{-1}
Output: the full key

- 1: **Amplify** $\left(\frac{|D_{\text{fin}}| 2^{k_{\text{out}} p_{\text{out}}}}{2^n (pq)^2}\right)$ **times**
- 2: Sample a subkey guess K_{out} using:
- 3: **Amplify** $\left(\frac{1}{2^{k_{\text{out}} p_{\text{out}}^2}}\right)$ **times**
- 4: Use BHT algorithm to find a valid tuple (C_1, C_2, C'_1, C'_2)
- 5: **Amplify** $2^{k_{\text{out}}}$ **times**
- 6: Sample a guess of K_{out}
- 7: **If** $R^{-1}(C_1) \oplus R^{-1}(C_2) = \delta = R^{-1}(C'_1) \oplus R^{-1}(C'_2)$ then K_{out} is “good”
- 8: **EndAmplify**
- 9: **If** we obtained a valid K_{out} then (C_1, C_2, C'_1, C'_2) is “good”
- 10: **EndAmplify**
- 11: Given this subkey guess K_{out} :
- 12: **Amplify** $2^{(k-k_{\text{out}})}$ **times**
- 13: Sample a choice of the $k - k_{\text{out}}$ other key bits, check if the full K matches
- 14: **EndAmplify**
- 15: **If** we obtained a valid full key K then K_{out} is “good”
- 16: **EndAmplify**
- 17: Measure K_{out} , recompute K and return it

We now check the number of iterations in the loops of Algorithm 5.4.

Step 1: the number of iterations to perform depends on the probability that a subkey guess K_{out} , obtained with the procedure in the loop, is the

good one. In the classical analysis of the procedure, we obtained a list of $\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2}$ tuples $(C_1, C_2, C'_1, C'_2, K_{\text{out}})$, and we took the key K_{out} from one of these tuples. We know that the good one is in one of these tuples. The computation inside the **Repeat** loop is the same, so the probability of success (the full key is found) is $\frac{2^n}{|D_{\text{fin}}|} \frac{(pq)^2}{2^{k_{\text{out}}} p_{\text{out}}}$ and the number of iterations follows.

Step 3: here, we are iterating on tuples (C_1, C_2, C'_1, C'_2) until we find a key candidate. The probability that a tuple yields a key candidate is $2^{k_{\text{out}}} p_{\text{out}}^2$, thus there are $\max(1, \frac{1}{\sqrt{2^{k_{\text{out}}} p_{\text{out}}^2}})$ iterations. We will assume $2^{k_{\text{out}}} p_{\text{out}}^2 \leq 1$.

Note that the key candidate (actually the superposition of all possible candidates for the given tuple) is obtained by exhaustive search, but a more involved procedure could likely be applied in order to reduce the time complexity.

Step 12: having obtained a candidate for K_{out} , it remains to try it: for this, we perform a simple Grover search over the remaining $k - k_{\text{out}}$ bits.

By changing the loops into nested quantum searches, the *quantum* time complexity of Algorithm 5.4 is:

$$\begin{aligned} & \sqrt{\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2}} \left(\frac{1}{\sqrt{2^{k_{\text{out}}} p_{\text{out}}^2}} \left(\left(\frac{2^n}{|D_{\text{fin}}|} \right)^{1/3} + 2^{k_{\text{out}}/2} \right) + 2^{(k-k_{\text{out}})/2} \right) \\ = & \underbrace{\sqrt{\frac{|D_{\text{fin}}|}{2^n} \frac{2^{k_{\text{out}}} p_{\text{out}}}{(pq)^2}}}_{< 2^{k_{\text{out}}/2} \text{ by the classical analysis}} \left(\frac{2^{n/3}}{2^{k_{\text{out}}/2} p_{\text{out}} |D_{\text{fin}}|^{1/3}} + \frac{1}{p_{\text{out}}} + \underbrace{2^{(k-k_{\text{out}})/2}}_{\text{Exhaustive search}} \right). \end{aligned}$$

The algorithm requires Q2 queries to E and E^{-1} , and uses $(2^n/|D_{\text{fin}}|)^{1/3}$ qRAM due to the use of BHT (or Ambainis' algorithm) to find valid pairs. However, if we can replace the factor $(2^n/|D_{\text{fin}}|)^{1/3}$ by $(2^n/|D_{\text{fin}}|)^{1/2}$, then we can use a Grover search instead of BHT, and the algorithm will now require a small number of qubits only.

Making the Attack Q1. Recall that the attack requires only a structure of $\frac{1}{\sqrt{p_{\text{out}}(pq)}}$ ciphertexts. For each of these, we will have to compute $E(E^{-1}(C) \oplus \alpha) := f(C)$. Thus, all the queries made to E and E^{-1} asked during the attack fall actually in a set of size $\frac{1}{\sqrt{p_{\text{out}}(pq)}}$. If we have $\frac{1}{\sqrt{p_{\text{out}}(pq)}} < 2^{k/2}$, then we can *make all the queries classically beforehand* and store the results in a quantum RAM of size $\frac{1}{\sqrt{p_{\text{out}}(pq)}}$. Inside Algorithm 5.4, we replace the on-the-fly computation of f by a memory lookup.

Algorithm 5.5 Quantum First-rounds Key-recovery

Input: *superposition* oracle access to E and E^{-1}
Output: the full key

- 1: **Amplify** $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\uparrow} p_{\downarrow}} 2^{k_{\text{in}}} p_{\text{in}}$ **times**
- 2: Sample a subkey guess K_{in} using:
- 3: **Amplify** $1/(2^{k_{\text{in}}} p_{\text{in}}^2)$ **times**
- 4: Sample a valid (P_1, P_2) using:
- 5: **Amplify** $\frac{2^n}{|D_{\text{start}}|}$ **times**
- 6: Sample a pair $P_1, P_2 = P_1 \oplus \alpha$, compute P_3, P_4 as in the distinguisher
- 7: **If** $P_3 \oplus P_4 = \alpha$, then (P_1, P_2) is “good”
- 8: **EndAmplify**
- 9: **Amplify** $2^{k_{\text{in}}}$ **times**
- 10: Sample a guess of K_{in}
- 11: **If** $R(P_1) \oplus R(P_2) = \alpha = R(P_3) \oplus R(P_4)$, then K_{in} is “good”
- 12: **EndAmplify**
- 13: **If** there is a valid K_{in} , then $(P_1, P_2, K_{\text{in}})$ is “good”
- 14: **EndAmplify**
- 15: Given this subkey guess K_{in} :
- 16: **Amplify** $2^{k-k_{\text{in}}}$ **times**
- 17: Sample a choice of the $k - k_{\text{in}}$ other key bits, check if the full K matches
- 18: **EndAmplify**
- 19: If the valid full key K was obtained, K_{in} is good
- 20: **EndAmplify**
- 21: Measure K_{in} , recompute the full K and return it

5.2.3 Quantum Mixing Boomerang Distinguisher

We now study the mixing boomerang attacks presented in Section 5.1.2. Since the distinguisher is a single loop similar to Algorithm 5.1, it can be replaced by a quantum search of a valid pair $(P_1, P_1 \oplus \alpha)$ with about $1/\sqrt{p_{\downarrow} p_{\uparrow}}$ iterations, with Q2 queries and negligible memory.

Quantum First-round Mixing Attack. The attack will work similarly as the quantum last-round attack of Algorithm 5.4. The main difference is that we do not need a collision search algorithm to filter out the valid pairs (P_1, P_2) such that $P_3 \oplus P_4 \in D_{\text{start}}$.

We analyze the time complexity of Algorithm 5.5, when translated into nested amplitude amplifications.

Step 1: here, the number of iterations depends on the number of possible keys K_{in} obtained in the sieving step. From the classical analysis, we know

that there will be $\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\uparrow} p_{\downarrow}} 2^{k_{\text{in}}} p_{\text{in}} < 2^{k_{\text{in}}}$ tuples $(P_1, P_2, P_3, P_4, K_{\text{in}})$ obtained, among which the good subkey K_{in} appears (at least once). Thus there are at most $(\frac{|D_{\text{start}}|}{2^n} \frac{1}{p_{\uparrow} p_{\downarrow}} 2^{k_{\text{in}}} p_{\text{in}})^{1/2}$ iterations.

Step 3: from a valid quadruple (P_1, P_2, P_3, P_4) , i.e., one that satisfies $P_1 \oplus P_2 \in D_{\text{start}}$ and $P_3 \oplus P_4 \in D_{\text{start}}$, the probability that we obtain a subkey guess is $2^{k_{\text{in}}} p_{\text{in}}^2$. This gives the number of iterations.

Step 5: the probability for a given pair to be valid is simply $\frac{|D_{\text{start}}|}{2^n}$.

Step 16: similarly to Algorithm 5.4, we complete the quantum search for the key.

The total quantum time complexity of Algorithm 5.5 is given by:

$$\begin{aligned} & \sqrt{\frac{|D_{\text{start}}|}{2^n} \frac{2^{k_{\text{in}}} p_{\text{in}}}{p_{\uparrow} p_{\downarrow}}} \left(\frac{1}{\sqrt{2^{k_{\text{in}}} p_{\text{in}}^2}} \left(\sqrt{\frac{2^n}{|D_{\text{start}}|}} + 2^{k_{\text{in}}/2} \right) + 2^{(k-k_{\text{in}})/2} \right) \\ &= \sqrt{\frac{|D_{\text{start}}|}{2^n} \frac{2^{k_{\text{in}}} p_{\text{in}}}{p_{\uparrow} p_{\downarrow}}} \left(\frac{2^{n/2}}{2^{k_{\text{in}}/2} p_{\text{in}} |D_{\text{start}}|^{1/2}} + \frac{1}{p_{\text{in}}} + 2^{(k-k_{\text{in}})/2} \right). \quad (5.6) \end{aligned}$$

The baseline algorithm does not use quantum RAM and requires only a small number of qubits, but it also relies on Q2 queries. Depending on the amount of data required, it may be possible to make it Q1: for this, we query all the $\frac{1}{p_{\text{in}} p_{\uparrow} p_{\downarrow}}$ pairs required by the classical attack, and store the tuples (P_1, P_2, P_3, P_4) in a quantum RAM. This can only work if $\frac{1}{p_{\text{in}} p_{\uparrow} p_{\downarrow}} < 2^{k/2}$.

5.3 Application to SAFER

SAFER is a family of block ciphers (Substitution-Permutation Networks) that dates back to the SAFER K-64 proposal of [90]. SAFER-K-64 was a block cipher of 64 bits. A new version SAFER+, with 128-bit blocks, was a candidate of the AES competition organized by the NIST [91]. Finally, another variant, SAFER++, was submitted to the NESSIE project [92]. Here, we focus on SAFER++ and more precisely, on the boomerang attack presented in [21]. It reaches 5.5 rounds out of 7 total rounds.

5.3.1 Description of the Cipher

We use \boxplus and \boxminus to denote addition and subtraction modulo 2^8 . We consider the version of SAFER++ with a 128-bit key (and 128-bit blocks), which has 7 rounds. The round function (Figure 1 in [21]) consists of key additions, a nonlinear layer and a linear layer. The state and the round keys are represented as 16 bytes numbered from 0 to 15, partitioned into $S_1 = \{0, 3, 4, 7, 8, 11, 12, 15\}$ and $S_2 = \{1, 2, 5, 6, 9, 10, 13, 14\}$. A single round performs the following steps:

Upper Key Addition: bytes in S_1 of the 16-byte round key are XOR-ed to the corresponding bytes of the block and the others are added modulo 2^8 .

Nonlinear Layer: it uses two functions X and L :

$$X(a) = (45^a \bmod 257) \bmod 256, L(a) = \log_{45}(a) \bmod 257 \text{ and } L(0) = 128, \quad (5.7)$$

where L is the inverse of X . Bytes in S_1 go through X and bytes in S_2 go through L .

Lower Key Addition: another 16-byte round key is added to the state, using modular addition in S_1 and XOR in S_2 .

Linear Layer: it repeats twice the following: a permutation of the bytes, followed by a Pseudo Hadamard Transform to groups of 4 bytes. This linear layer contains a total of 48 modular additions of individual bytes and can be described by the matrix given in [21].

5.3.2 5-Round Classical Boomerang Attack

We present the attack from [21]. It relies on a boomerang distinguisher on 4.5 rounds of SAFER++. If A denotes the linear layer and S the nonlinear layer (X and L), the distinguisher works against a sequence $A_0 S_1 A_1 - S_2 - A_2 S_3 A_3 S_4 A_4$. It would work against 4 rounds if any S-Boxes were used, but it reaches an additional 0.5 round thanks to the following property of the inverse-based S-Boxes in SAFER++:

$$\forall a, X(a) \boxplus X(a \boxplus 128) = 1 \bmod 256 . \quad (5.8)$$

The top part of the boomerang starts with pairs of plaintexts having difference $\alpha = (0, x, 0, 0, x, x, 0, 0, 0, 0, 0, 0, -4x, 0, 0, x, -x)$. This difference is such that it maps to a single active byte after A_0 , and up to 16 active bytes after $A_1 \circ S_1$.

The difference added to ciphertexts is:

$$\delta = (0, 0, 128, 0, 128, 128, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0) .$$

After $S_4^{-1} \circ A_4^{-1}$, it maps to a difference $(0, 0, 0, 0, 0, x, -x, 0, \dots)$ with probability 2^{-7} (not 2^{-8} because the difference 128 maps to odd differences through X). Then after A_3^{-1} , bytes 3, 9, 11 and 14 are active.

In order to traverse the middle S-Box, the authors observe that if the byte-differences coming both from the top and the bottom part of the boomerang are 128, then the boomerang traverses this S-Box layer for free. Indeed, if we encrypt $P_1, P_2 = P \boxplus 128$ through X , we obtain $X(P), 1 \boxplus X(P)$, i.e., two values that sum to 1. If the difference coming from the bottom part is 128,

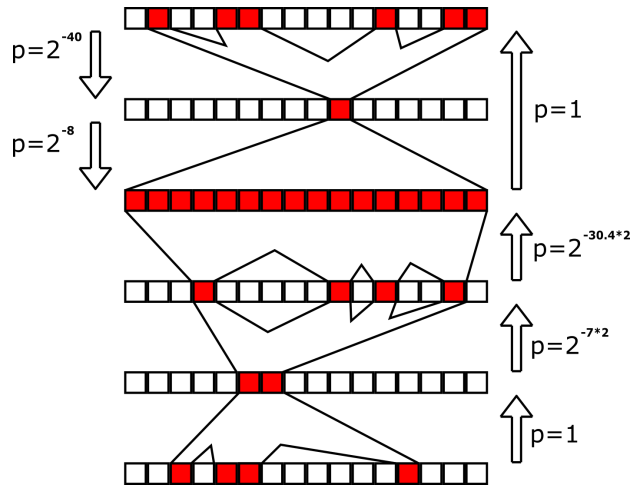


Figure 5.3: Classical boomerang attack on SAFER++ (reproduction of Fig. 4 in [21])

then the second pair of ciphertexts $C_3, C_4 = C_1 \boxplus 128, C_2 \boxplus 128$ still sums to 1. Thus, by going through X^{-1} , they get mapped to a difference 128 with probability 1.

Boomerang Attack. The full differential path of the classical 5-round boomerang attack of [21] is reproduced in Figure 5.3. We use $S_0A_0S_1A_1 - S_2 - A_2S_3A_3S_4A_4$ to denote the whole construction. The total probability for the boomerang to occur is: $(2^{-7-30.4})2^{-40-8} = 2^{-122.8}$.

Top Boomerang: At the top, the difference is active in 6 bytes in positions 1, 4, 5, 11, 14, 15. This difference propagates to a single active byte after one round ($A_0 \circ S_0$) with probability 2^{-40} , as there are 5-byte conditions to meet. This difference is then mapped to a single active byte of difference 128 after the next nonlinear layer S_1 , with probability 2^{-8} , and with probability 1, we obtain a difference 128 in bytes 0, 1, 2, 3, 8, 9, 11, 13, 14, 15 after A_1 . All other bytes are inactive.

Bottom Boomerang: In the lower part of the boomerang, we start with a difference 128 (changing one bit) in the 4 bytes at positions 2, 4, 5, 12. By decrypting through S_4A_4 , this yields a difference $x, -x$ in bytes 5 and 6 with probability 1. By decrypting through S_3A_3 , this yields a difference active in bytes 3, 9, 11, 14 with probability 2^{-7} . Then, by decrypting through A_2 , we can obtain a difference 128 in all bytes except 2, 4, 9, 12 with probability $2^{-30.4}$.

Middle: In the middle, the S-Box trick allows to traverse the layer S_2 with probability 1. When going backwards from this middle to the top, everything propagates with probability 1.

We recall the attack procedure from [21] in Algorithm 5.6. It requires 2^{78} encryptions.

Algorithm 5.6 Classical attack on 5-round SAFER++ (adapted from [21], Section 6.3).

Input: access to E and E^{-1}

- 1: Prepare 2^{29} structures of 2^{48} plaintexts P_i that take all values in bytes 1, 4, 5, 11, 14 and 15 and are constant in the others
 - 2: For each structure, obtain the ciphertexts $C_i = E(P_i)$
 - 3: For each structure, add the difference δ (one bit in bytes 2, 4, 5, 12) to the C_i and decrypt: obtain $Q_i = E^{-1}(C_i \oplus \delta)$
 - 4: For each structure, sort the Q_i on the values of the bytes that are constant in the P_i and keep the pairs Q_i, Q_j that have zero difference on these ten bytes (all bytes except 1, 4, 5, 11, 14 and 15)
 - ▷ We search for a difference after the S-Box of the form $(0, x, 0, 0, x, x, 0, 0, 0, 0, -4x, 0, 0, x, -x)$
 - 5: For each possible quartet (P_j, P_k, Q_j, Q_k) , search for the upper key bytes $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$ such that after the first S-Box layer, we have a difference $(x, -4x, x, -x)$ (x odd) between P_j and P_k and between Q_j and Q_k . The quartets are kept with their partial key and the observed x .
 - 6: For each quartet, search for the key bytes K_0^1 and K_0^2 (upper and lower key addition in byte 0) such that the difference after the first S-Box is the right one. Repeat this for K_0^5 and K_0^6 .
 - 7: Find a suggestion of the 8 key bytes that appears twice, and run exhaustive search on the remaining bytes.
-

From a pool of 2^{48} plaintexts, we get approximately 2^{95} pairs, which have to satisfy an 80-bit condition, so we get 2^{15} pairs in Step 4. The probability that the structure contains a boomerang is approximately $2^{-27.8}$, so by using 2^{29} structures, we can expect two boomerang quartets to occur among the 2^{44} filtered quartets.

Next, we guess 4 bytes of the first upper round key $(K_0^4, K_0^{11}, K_0^{14}, K_0^{15})$ to remove wrong quartets. For each guess, we have a 25-bit condition on both pairs. Thus, 50 bits of restriction on the quartets. At this point, we obtain $2^{44} \times 2^{32} \times 2^{-50} = 2^{26}$ valid quartets and key guesses. Note that this already yields a valid key-recovery attack, as we have been able to reduce the number of possible $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$ from 2^{32} to 2^{26} .

For the next 4 key bytes guessed at Step 6, there is approximately one choice for each valid quartet currently kept. Thus, at this point, we have 2^{26} guesses of 8 bytes of the key, among which we expect the good one to occur. If we ran immediately an exhaustive search for the remaining 8 bytes, we would obtain a complexity $2^{26} \times 2^{8 \times 8} = 2^{90}$.

However, there are two valid boomerang pairs: one of the key guesses occurs twice. As the key guesses are 8 bytes and there are 2^{26} of them, we

can expect that random collisions are not likely to occur, and the only key guess occurring twice is the good one. Thus exhaustive search is ran only once. The bottleneck of the complexity lies in encrypting and decrypting the 2^{29} structures, for a total of 2^{78} time and queries.

Extension. Another half round can be added at the end, by making 30 bits of additional key guess and running Algorithm 5.6 2^{30} times.

5.3.3 Quantum Boomerang Attack

Our new quantum attack (Algorithm 5.7) on 5-round SAFER++ uses the quantum framework of Algorithm 5.4, adapted to the setting of the classical boomerang attack (Algorithm 5.6). We first explain its time complexity, and show how it goes below Grover's exhaustive search in time 2^{64} . Recall that the time complexity is counted in number of evaluations of SAFER++.

Algorithm 5.7 Attack on 5-round SAFER++.

Input: superposition access to E and E^{-1}
Output: the full key

- 1: **Amplify** 2^{26} times
- 2: Sample a guess for $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}, K_0^1, K_0^2, K_0^5, K_0^6$ using:
- 3: **Amplify** 2^{18} times ▷ a valid quartet yields a key guess with probability 2^{-18}
- 4: Find a valid quartet
 ▷ Either using Ambainis' algorithm, or quantum search
 ▷ We need the same number of quartets as classically: 2^{78} plaintexts in total
- 5: **Amplify** 2^{32} times ▷ at most one solution
- 6: Pick a key guess $K_0^4, K_0^{11}, K_0^{14}, K_0^{15}$
- 7: Check if this key guess yields the good difference $(x, -4x, x, -x)$ for both pairs of the quartet
- 8: **EndAmplify**
- 9: If a solution was found, return it (and the quartet)
- 10: **EndAmplify**
- 11: Search exhaustively for K_0^1, K_0^2 and K_0^6 (keep at most one value)
 ▷ there is on average one value; sometimes there can be more; but we only need this to work for the actual boomerang quartet
- 12: Given the current guess of 8 bytes of key, complete the key by Grover search
- 13: **EndAmplify**
- 14: Measure and return the full key

Assuming that we use Grover search at Step 4 (instead of collision search),

the attack *would* run in approximately:

$$\sqrt{2^{26}} \left(\sqrt{2^{18}} \left(\underbrace{\sqrt{2^{80}}}_{\text{Step 4}} + \sqrt{2^{32}} + \underbrace{2 \cdot 2^{16}}_{\text{Step 11}} \right) + \underbrace{\sqrt{2^{64}}}_{\text{Step 12}} \right) \quad (5.9)$$

calls to SAFER++ and its inverse. The dominating term comes from the search for quartets satisfying the partial collision condition on 10 bytes. Indeed, the classical attack can sample these quartets very easily using structures. If we use classical search in Algorithm 5.7, we can obtain an attack running with polynomial memory in little less time than 2^{128} . However, in the quantum setting, additional factors from amplitude amplification will prevent that, so we must resort to quantum collision search in Step 4.

Since a structure is of size 2^{48} and we need 80 bits of collision, we can use Ambainis' algorithm to produce a superposition of colliding pairs (thus valid quartets) in about $(\pi/2)^2 2^{80/3} \simeq 2^{28.3}$ calls to E and E^{-1} in superposition.

Assuming that the numbers of iterations are exact, but putting the additional factors of quantum search, we obtain:

$$\begin{aligned} & 2 \left\lfloor \frac{\pi}{4} 2^{13} \right\rfloor \left(2 \left\lfloor \frac{\pi}{4} 2^9 \right\rfloor \left(\underbrace{2^{28.3} \times 4}_{\text{Step 4}} + 2 \left\lfloor \frac{\pi}{4} 2^{16} \right\rfloor + 4 \cdot 2^{16} \right) + \underbrace{\left\lfloor \frac{\pi}{4} 2^{32} \right\rfloor \times 4}_{\text{Step 12}} \right) \\ & \simeq 2^{13.65} \left(2^{9.65} (2^{30.3} + 2^{16.65} + 2^{19}) + 2^{33.65} \right) \simeq 2^{53.6} < 2^{64} \quad (5.10) \end{aligned}$$

where the time is counted in computations of SAFER++ and its inverse, and we assume that a black-box (quantum) query costs the same. In the last step (quantum exhaustive search over 8 key bytes), we match against two given plaintext-ciphertext pairs.

In Step 12, we know that there is exactly one solution (or none) in a search space of size 2^{64} . Thus, we can run *exact* amplitude amplification with $\lfloor \frac{\pi}{4} 2^{32} \rfloor$ iterations and succeed with probability 1 (either we find the solution, or prove that there is none). For the other loops, we can ensure a high probability of success by making a few assumptions on the *classical* attack: • the number of valid quartets for each structure is close to the average (this ensures that Ambainis' algorithm works as intended; if necessary we make multiple copies of it);

- the actual boomerang pair yields only a single value at Step 11, and our algorithm does not lose it;
- the probability, for a given guess of 8 key bytes sampled at Step 2, to be the good one, is close to 2^{-26} .

The attack requires 2^{27} registers of 256 qubits to store the values of P_i and $E^{-1}(E(P_i) \oplus \delta)$ for each P_i , and $2^{53.6}$ computations and superposition queries to E and E^{-1} (faster than Grover's search, which use 2^{64} computations).

5.4 Application to KASUMI

KASUMI is a block cipher used in the confidentiality and the integrity algorithms of the 3GPP mobile communications [2]. In 2005, a related-key attack was found by Eli Biham, Orr Dunkelman and Nathan Keller [19] on KASUMI which made the security of the algorithms using KASUMI unprovable (no security proof but not necessarily an attack). This attack was further refined by Orr Dunkelman, Nathan Keller and Adi Shamir [51] and runs in practical time.

5.4.1 Description of the Cipher

KASUMI is a block cipher with a 128-bit key and a 64-bit block. It is an 8-round Feistel network using smaller functions FO and FL , FL then FO on even rounds and FO then FL on odd rounds. For now we note i the number of the round of the global Feistel network.

FL is a 2-round Feistel network with for the first round function a left rotation of one bit then a bit-wise AND with $KL_{i,1}$ and for the second round function, a bit-wise OR with $KL_{i,2}$ then a left rotation of one bit.

FO is a 3-round Misty Feistel network with a bit-wise XOR with $KO_{i,j}$ then application of $FI_{i,j}$ as j -th round of FO .

FI is a 4-round Misty Feistel unbalanced network with 9 bits on one side and 7 on the other. On odd round it applies $S9$ on the 9-bit part and xor the 7 first bits with the 7bit part, on second round, apply $S7$ on the 7-bit branch and xor it with the first 7 bits of the 9-bit branch and xor the whole state with $KI_{i,j}$, on last round apply $S7$ on the 7-bit branch and xor it with the first 7 bits of the 9-bit branch.

For the key schedule, we have $K = K_1||K_2||K_3||K_4||K_5||K_6||K_7||K_8$ and:

$$K' = K \oplus 0x0123456789ABCDEF FEDCBA9876543210$$

$$KL_{i,1} = K_i \lll 1$$

$$KL_{i,2} = K'_{i+2}$$

$$KO_{i,1} = K_{i+1} \lll 5$$

$$KO_{i,2} = K_{i+5} \lll 8$$

$$KO_{i,3} = K_{i+6} \lll 13$$

$$KI_{i,1} = K'_{i+4}$$

$$KI_{i,2} = K'_{i+3}$$

$$KI_{i,3} = K'_{i+7}$$

We refer to [51] for more details on the cipher.

5.4.2 Classical Attack

The attack of [51] relies on a boomerang distinguisher on 7 rounds of KASUMI with probability 2^{-14} .

Top Boomerang: in FL , each bit of key has only one half probability to express itself which make the following related key differential happen with $\Delta K = (0, 0, 0x8000, 0, 0, 0, 0, 0)$ and probability $\frac{1}{4}$:

Bottom Boomerang: the same differential happens on rounds 5-7 with $\Delta K = (0, 0, 0, 0, 0, 0, 0x8000, 0)$ and probability $\frac{1}{4}$.

Middle: we want two top differentials and a bottom differential to produce another bottom differential. In this case, the right part of the quartet is of the form $(X, X \oplus 00100000, X \oplus 00100000, X)$ and the goal is to transform it through FO_4 to a balanced quartet, knowing that FL_4 is linear.

The last round attack uses the Feistel structure to determine the elements that follow the path of the boomerang: once the structure of element is chosen, the difference of the left parts of $C_a^L \oplus C_c^L = FL_8(FO_8(C_a^R)) \oplus FL_8(FO_8(C_c^R))$ is the same among the elements that follow the path of the boomerang.

Once we have a correct quartet, we can recover $KL_{8,2}, KO_{8,1}, KI_{8,1}$ by considering the differences on the second round of FL_8 (we know the differences just after the round because it is the difference of the left part and the difference just before the round only depends on $KO_{8,1}, KI_{8,1}$) and knowing the values that go into the bitwise OR, we can eliminate wrong guesses for $KO_{8,1}, KI_{8,1}$ when we come to a contradiction.

Three quartets give four right pairs (C_a^R, C_c^R and C_a^L, C_c^L are the same for every quartets) on which the key guesses have to not get a contradiction. Wrong keys have a probability to be kept less than $\left(\frac{1+1}{16}\right)^{16} = 2^{-48}$ ($\left(\frac{1}{2}\right)^4$ is the probability that all four pairs agree with the i -th bit of $KL_{8,2}$ being a 0 (same difference before the OR as after) and same probability for agreeing on a 1 (always no difference after the OR)). Remark that in the process, $KL_{8,2}$ is also found.

The same analysis holds when we recover $KL_{8,1}, KO_{8,3}, KI_{8,3}$ by considering the differences on the first round of FL_8 . This result in the following attack:

We start with 2^{24} elements which make 2^{48} pairs. With the first condition we only keep 2^{16} quartets in which we expect to have $2^{16} \times 2^{-14} = 4$ right quartets. The probability of a 32-bit value $C_1^L \oplus C_3^L$ appearing more than three times from 2^{16} random values is less than $\binom{2^{16}}{3} 2^{3 \times (-32)} 2^{32} \leq 2^{-18}$. Then we are almost sure that the corresponding quartets are right. The total attack needs $2^{33.8}$ computations.

5.4.3 Quantum Attack

Our quantum attack on KASUMI uses the quantum framework of Algorithm 5.4, adapted to the setting of the classical attack (Algorithm 5.8).

Algorithm 5.8 Classical attack on KASUMI (adapted from [51], Sec. 5)

Input: access to E and E^{-1}

- 1: Choose a value A prepare a structures of 2^{24} plaintexts $C_a = (X_a, A)$
 - 2: For each element, obtain the plaintexts $P_a = E_{K_a}^{-1}(C_a)$
 - 3: For each element, compute $P_b = (0, 00100000) \oplus P_a$ and decrypt: obtain $C_b = E_{K_b}^{-1}(P_b)$
 - 4: Prepare a structure of 2^{24} plaintexts $C_c = (X_c, A \oplus 00100000)$, encrypt and get $P_c = E_{K_c}^{-1}(C_c)$ compute $P_d = (0, 00100000) \oplus P_c$ and decrypt: obtain $C_d = E_{K_d}^{-1}(P_d)$
 - 5: Sort the C_b and the C_d by their right parts and keep the quartets (C_a, C_b, C_c, C_d) such that $C_b^R = C_d^R \oplus 00100000$.
 - 6: Sort the quartets by their $C_a^L \oplus C_c^L$ and keep the pairs C_a, C_c that correspond to the most common value (we expect to get a group of 3 or more quartets).
 - ▷ We expect to get a group of 3 or more quartets then we are almost sure that the remaining quartets are valid one
 - 7: Recover the key parts $KL_{8,2}, KO_{8,1}, KI_{8,1}$ as described earlier
 - 8: Recover the key parts $KL_{8,1}, KO_{8,3}, KI_{8,3}$ as described earlier
 - 9: Recover the key parts $KO_{8,2}, KI_{8,2}$ by brute force
-

The full attack needs $2^{22.7}$ computations using Ambainis algorithm or 2^{24} computation without QRAM (then we use the classic part for quartet determination and Grover search for the key search). Both version are faster than Grover's search, which uses 2^{32} computations.

5.5 Application to Related-key AES

In this section, we present the boomerang related-key attack on AES-256 from [22] and study a corresponding quantum attack. We will see that the classical attack path is actually not adapted to the quantum setting. The quantum attack represents a relatively minor improvement on the classical one, and it can only be considered valid under a restriction on the number of related keys.

Note that the attack uses a tuple of classically related keys (the same as in the classical attack). In this situation, there is no trivial quantum attack such as the one of [106] (which needs a quantum superposition of related keys).

5.5.1 Description of AES

The AES [96], designed by Daemen and Rijmen [39], was the winner of the AES competition organized by the NIST. It is arguably one of the most analyzed ciphers to date. Similarly to SAFER++, it has blocks of 128 bits, divided in

Algorithm 5.9 Quantum attack on KASUMI

- Input:** superposition access to E and E^{-1}
Output: the full key
- 1: Find three correct quartets
 - 2: **Amplify** 2^{16} **times**
▷ $((2^{32})^{3/8}$ iterations using Ambainis)
 - 3: Find a superposition of quartets that satisfy the first criterion (Step 5 before)
▷ Either using Ambainis' algorithm, or quantum search
 - 4: **Amplify** $\sqrt{2^{32}}$ **times**
▷ $((2^{32})^{1/3}$ iterations using Ambainis)
 - 5: Pick a C_a, C_c , compute C_b, C_d
 - 6: Check if $C_d = C_b \oplus 00100000$
 - 7: **EndAmplify**
 - 8: Check that the three quartets share the same value for $C_a^L \oplus C_c^L$
 - 9: **EndAmplify**
▷ The obtained quartets are correct
 - 10: Search for $KO_{8,1}$ and $KI_{8,1}$:
 - 11: **Amplify** $\sqrt{2^{32}}$ **times**
 - 12: $KO_{8,1}$ and $KI_{8,1}$ have to let a possibility for $KL_{8,2}$
 - 13: **EndAmplify**
 - 14: Search for $KO_{8,3}$ and $KI_{8,3}$:
 - 15: **Amplify** $\sqrt{2^{32}}$ **times**
 - 16: $KO_{8,3}$ and $KI_{8,3}$ have to let a possibility for $KL_{8,1}$
 - 17: **EndAmplify**
 - 18: Search for $KO_{8,2}$ and $KI_{8,2}$:
 - 19: **Amplify** $\sqrt{2^{32}}$ **times**
 - 20: Verify the full key on a simple encryption
 - 21: **EndAmplify**
-

16 bytes numbered from 0 to 15. The state of AES is represented as a square and we will follow the byte numbering of [22]. Thus, in a plaintext P or a subkey K , the byte $P_{i,j}$ or $K_{i,j}$ is located at row i and column j .

AES State. The state of is composed of elements of \mathbb{F}_{256} organized in a 4×4 matrix :

$$\begin{bmatrix} \alpha_0 & \alpha_4 & \alpha_8 & \alpha_{12} \\ \alpha_1 & \alpha_5 & \alpha_9 & \alpha_{13} \\ \alpha_2 & \alpha_6 & \alpha_{10} & \alpha_{14} \\ \alpha_3 & \alpha_7 & \alpha_{11} & \alpha_{15} \end{bmatrix}$$

Composition of a Round. AES-128 is composed of 10 rounds which are composed of :

- AddKey xors the state with the round key (see Key Schedule);
- SubBytes which applies the AES Sbox on all individual elements α_i ;
- ShiftRows which shifts the i -th row by i position;
- MixColumns which multiplies each column by a fixed matrix.

The last round omits the Mixcolumns operation and applies one extra AddKey.

Key Schedule. The round key expansion from the 128-bit master key $K = k_0 || k_1 || k_2 || k_3$ for the subkey of round i , K_i for E is as follows:

$$K_0 = k_0 || k_1 || k_2 || k_3 \text{ and } K_{i+1} = (k_{4i+4} || k_{4i+5} || k_{4i+6} || k_{4i+7}) \text{ for } i \text{ from } 0 \text{ to } 9$$

$$\begin{aligned} k_{4i+4} &= \text{SubWord}(\text{RotWord}(k_{4i+3})) \oplus k_{4i} \oplus rc_i \\ k_{4i+5} &= k_{4i+4} \oplus k_{4i+1} \\ k_{4i+6} &= k_{4i+5} \oplus k_{4i+2} \\ k_{4i+7} &= k_{4i+6} \oplus k_{4i+3} \end{aligned}$$

$$\text{with } rc_i = \begin{pmatrix} X^i \bmod X^8 + X^4 + X^3 + X + 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

5.5.2 Classical Attack

We consider here the related-key attack of [22] on full-round AES-256 (14 rounds in total, numbered from 1 to 14). This key-recovery has a data and time complexity of $2^{99.5}$.

There are 4 related keys K_A, K_B, K_C, K_D in the attack of [22], where K_B, K_C, K_D are computed from K_A . These related keys are used in the four branches of the quartet. We refer to [22] for the definition of their relations. The path of the boomerang is represented on Figure 5.4 and Figure 5.5. The key-schedule of AES-256 has an internal state of 256 bits in which the full key is first loaded, then updated every two rounds (the first half is used for even rounds, the second half for odd rounds). We let K_0, K_1, \dots denote these successive 256-bit round keys, where $K_0 = K$ is the master key.

The main requirement for this attack are the differences $1f$ in the output of the active S-Boxes. With a difference 01 in the input, the output difference is $1f$ with probability 2^{-6} . Thus, assuming that we can start from a pair that satisfies the path of Figure 5.4 at Round 1, it passes the top differential to the beginning of round 9 with probability $2^{-6 \times 5} = 2^{-30}$. Then we use the *ladder switch* technique, which means that we consider the S-Boxes on the

bytes $(0, 0), (0, 1), (0, 2), (0, 3)$ to belong to the top part of the boomerang and the other ones to belong to the bottom part of the boomerang, this way there is no differential to pay in either parts of the boomerang for this round. In the bottom part of the boomerang, there remains 3 S-Boxes to pass, with a probability 2^{-18} . Thus the total probability is $2^{-30 \times 2 - 18 \times 2} = 2^{-96}$.

We focus now on the key-recovery attack of [22]. We start from $2^{25.5}$ structures of plaintexts that take all possible values in the 9 bytes $(0, 0), (1, 0), (2, 0), (3, 0), (1, 1), (1, 2), (1, 3), (2, 2), (3, 3)$ (that includes the first column, the second row and the first diagonal) and leave the other constant. For each structure:

1. the plaintexts are encrypted under K_A and K_B and the resulting ciphertexts stored in sets S_A and S_B ;
2. a (carefully chosen) difference Δ_C is XOR-ed to S_A and the resulting ciphertexts are decrypted under K_C , yielding a set S_C of plaintexts;
3. the same Δ_C is XOR-ed to S_B and the resulting ciphertexts are decrypted under K_D , yielding a set S_D of plaintexts;
4. S_C and S_D are then filtered for valid quartets: we find all the pairs $P_C \in S_C, P_D \in S_D$ such that $P_C \oplus P_D$ is 0 in the 7 inactive bytes of the structure $((0, 1), (0, 2), (0, 3), (2, 1), (3, 1), (3, 2), (2, 3))$;
5. two filtering steps for these quartets remain. We check whether $P_{A,i,0} \oplus P_{B,i,0} = P_{C,i,0} \oplus P_{D,i,0} (= \Delta K_{i,0}^0), i > 1$ because $\Delta K_{i,0}^0$ is equal for the key pairs (K_A, K_B) and (K_C, K_D) ;
6. another filtering allows to reduce the number of quartets by a factor 2^6 .

There were 2^{72} plaintexts in each structure, for a total of $2^{97.5}$ plaintexts. There are 2^{144} pairs per structure and a 9-byte condition is required to pass the first round. Thus, we can expect $2^{144 - 8 \times 9 - 96 + 25.5} \simeq 3$ boomerang pairs among them. At the first filtering step, we filter on 7-byte conditions, in the second, there are 2-byte conditions, then a 6-bit condition. Thus there are $2^{144 - 8 \times 9 - 6 + 25.5} = 2^{91.5}$ quartets at this point.

We can then begin to sieve some subkey bytes. It is shown in [22] that by performing a few trials, one can reduce the number of quartets by a factor 2^{19} and, for each of the remaining quartets, find 2^6 candidates for a total of 15 key bytes. Notably, 85 bits of K_A are found. This makes a total of $2^{78.5}$ choices for these key bytes. Since there are three boomerang quartets, one of these choices occurs three times (other collisions are unlikely).

5.5.3 Quantum Attack

The attack of [22] runs in time $2^{99.5}$, makes the same number of queries to E and E^{-1} , and uses a memory of size $2^{77.5}$ to recover the good guess for some

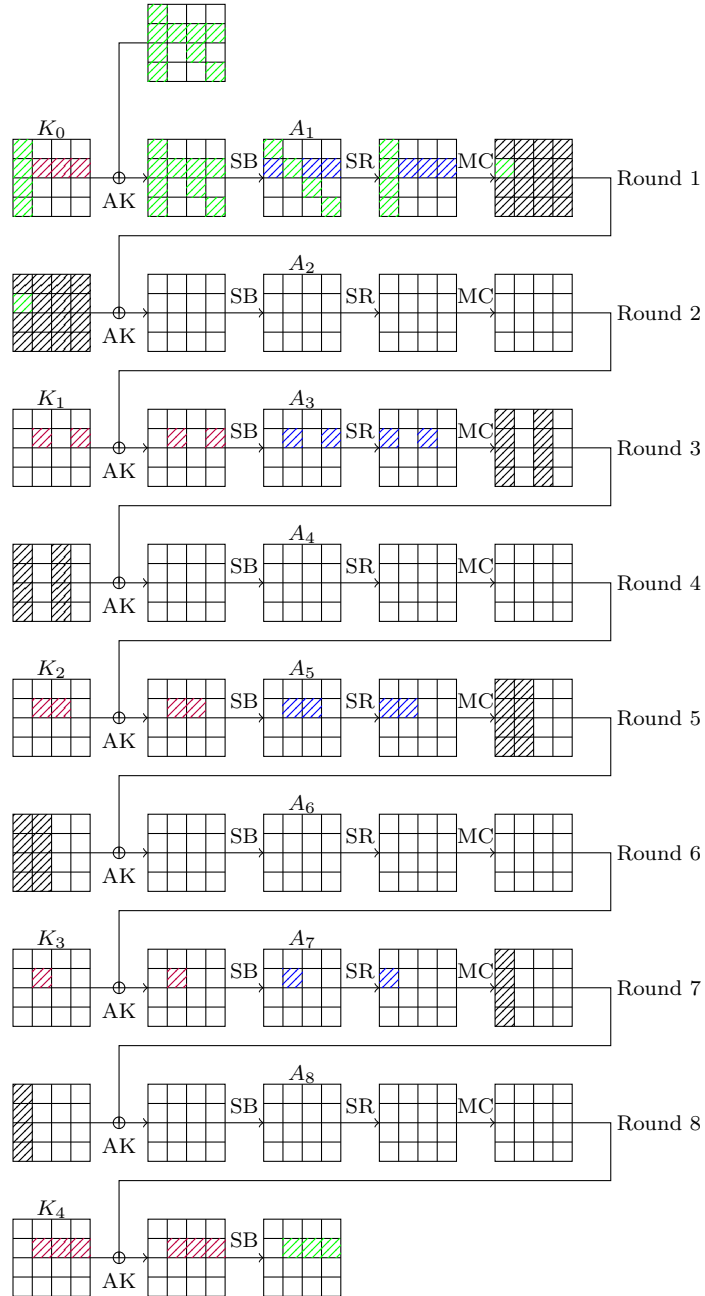


Figure 5.4: Top part of the boomerang (see [22], Fig. 7). We use purple for the difference 01, blue for the difference $1f$, green for unknown differences and black for differences that collide by chance. The state after the SubBytes layer of round i is denoted A_i .

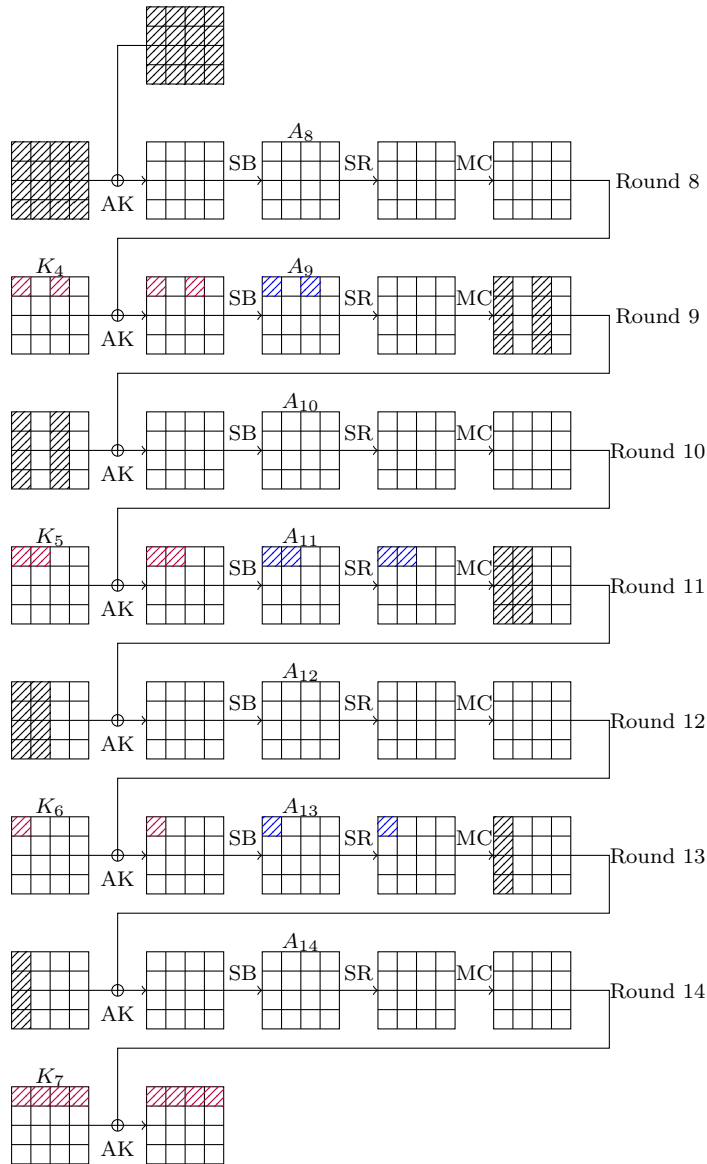


Figure 5.5: Bottom part of the boomerang (see [22], Fig. 7). We use purple for the difference 01, blue for the difference 1f, green for unknown differences and black for differences that collide by chance.

of the key bytes. It falls already below the complexity of Grover search for a single secret-key AES-256.

Generic Attack. In the related-key setting, and with an unbounded number of related keys, it is possible to reduce this to $2^{256/3} = 2^{85.3}$ queries and time using the same number of related keys ($2^{85.3}$). This is the quantum counterpart of the classical collision attack using 2^{128} related keys, which would simply encrypt a small set of plaintexts P_i under all the keys $K, K \oplus 1, \dots, K \oplus 2^{128}$ and find another key K' such that $E_{K'}(P_i)$ collides with one of the $E_{K \oplus j}(P_i)$. The quantum attack uses a smaller set of related keys and uses a Grover search in the second step; the complexities of both steps are balanced at roughly $2^{85.3}$.

More precisely, the quantum search step needs two plaintexts (hence four encryptions per iteration). The first step performs $2 \cdot 2^t$ queries and the second $\frac{\pi}{4} 2^{(256-t)/2}$ iterations, for a total of $\frac{\pi}{4} 2^{(256-t)/2+2} = 2^{128-t/2+2.65}$ encryptions. This is minimized for $t + 1 = 128 - t/2 + 2.65$, i.e., $t = 86.4$ and the total is $2^{87.4}$ encryptions.

Thus, we can choose to compare our attack complexity either with the generic bound of $2^{87.4}$, if we assume an unbounded number of classically related keys, *or* with a bound of 2^{127} if we assume that only 4 related keys are available (in which case the best procedure is the same as above, but limited by the number of keys that we have). In the former case, the attack will not be valid. That is, the complexity will go above $2^{87.4}$. It will only be valid for the second definition.

Attack Procedure. We now detail the quantum version of the previous attack. We use the same differential path and the same key relations. Contrary to the attack against SAFER++, we will not combine the boomerang with a complete recovery of the key. Since only 85 bits of the master key are obtained, it may take actually too much time to continue a Grover search on the remaining key bits; we would then need a more involved procedure to get as many key bits as possible within the allowed time. Instead, we will define a function that produces deterministically a quartet and a 85-bit key guess. Roughly speaking, this function starts from a set of plaintext values (some substructure), performs a search of valid quartets on this space, sieves them to obtain a single (expected) quartet leading to 2^6 guesses of key bits, and chooses one of these guesses. It will output $2^{78.5}$ key guesses of 85 bits. We expect one of these guesses to appear more often, due to the boomerang pairs. Thus we can use Ambainis' element distinctness algorithm [5]. However, we must recognize the only guess that appears three times, not two, so we must use the *3-distinctness* algorithm which runs in generic complexity $\mathcal{O}(N^{3/4})$ for N elements (also optimal). The constant in the \mathcal{O} is the same as in the 2-distinctness algorithm.

Algorithm 5.10 Related-key attack on AES-256 (as a classical combination of **Repeat** loops).

Input: access in superposition to encryptions and decryptions under K_A, K_B, K_C, K_D

- 1: **function** $F(i, b)$ $\triangleright i$ is a 72.5-bit value, b is a 6-bit value
- 2: Depending on i , choose a set of 2^{19} subsets of $2^{72/3}$ plaintexts P_A
- 3: **Amplify** 2^{19} **times**
- 4: Choose one of the 2^{19} subsets for P_A
- 5: Create a superposition of quartets passing the filter of Step 4 (56 bits), as follows:
- 6: Create the list of $2^{72/3}$ plaintexts P_A
- 7: By encrypting under K_A and decrypting under K_C , create the set S_C
- 8: **Amplify** $2^{2 \times 72/3}$ **times**
- 9: Create a single plaintext P_B
 $\triangleright P_B$ belongs to a space of size $2^{7 \times 9} = 2^{72}$, by the choice of P_A
- 10: Encrypt under K_B , decrypt under K_D , obtain P_D
- 11: Check if there is $P_C \in S_C$ such that: $P_C \oplus P_D$ is 0 in the 7 inactive bytes wanted, and the condition at Step 5 is also satisfied (2 bytes)
- 12: If there is, return P_D
- 13: **EndAmplify**
- 14: Find P_D
- 15: Check if the quartet P_A, P_B, P_C, P_D yields key guesses \triangleright
probability 2^{-19}
- 16: If it does, return it
- 17: **EndAmplify**
- 18: At this point, we have a (single) quartet returning key guesses: choose one of them depending on b
- 19: Apply Ambainis' 3-distinctness algorithm to F
- 20: Measure and return the single 3-collision among the key outputs of F

We compute the time complexity of Algorithm 5.10 as follows. It is dominated by the $2^{3 \times 78.5/4} \simeq 2^{58.9}$ calls to the function F that we defined. Calling F requires two loops, one of which corresponds actually to a quantum collision search for quartets matching the 9-byte condition. Note that the computations required to find if there are key guesses, and to select one of them, are negligible with respect to the other factors.

$$2 \cdot 2^{58.9} \left(2 \left\lfloor \frac{\pi}{4} \sqrt{2^{19}} \right\rfloor \left(2 \cdot 2^{72/3} + 4 \left\lfloor \frac{\pi}{4} 2^{72/3} \right\rfloor + \text{negl.} \right) + \text{negl.} \right) \simeq 2^{59.9} \left(2^{10.15} (2^{25} + 2^{25.65}) \right) \simeq 2^{96.4} . \quad (5.11)$$

The complexity that we obtain is slightly below the classical attack

complexity, but not significantly. It is also above the quantum bound of $2^{87.4}$ encryptions. We also use the same number of superposition queries, and about $2^{58.9}$ quantum random-access memory. Our main issue with this method is that the quantum time complexity of 3-distinctness is different than that of 2-distinctness, which is not the case classically. If the good key was the only one appearing twice, instead of three times, we could replace the term $2^{58.9}$ by $2^{52.3}$ and go more significantly below the classical complexity.

Furthermore, this method (using an element distinctness routine on top of the search of quartets) seems intrinsically less adapted than the direct combination with exhaustive search that we used against SAFER++. Thus, it seems necessary to modify the path of the attack in order to find more key bytes with a valid quartet, and to finish faster the recovery of the key.

5.6 Application to 5-Round AES

In this section, we present the mixing boomerang distinguisher on 5-round AES of [49], which is the fastest distinguisher known to date on this cipher, and leads to the fastest key-recovery attack on 5-round AES. We show that this attack admits a quantum version. Although the classical attack is already practical, we believe that its quantum variant is still of interest, as it might occur as a building block of more involved quantum attacks on reduced-round variants of the AES.

5.6.1 The boomerang

We separate the cipher in three parts: E_0 is the AK which constitute the round -1, the rounds 0,1 and the steps SB and SR of the round 2, E_1 is the MC of the round 2 and E_2 is the AK of round 2 and the rounds 3 et 4.

Top Boomerang: E_0 admits a classical truncated differential: if we have a difference only on the bytes (0, 0), (1, 1), (2, 2), (3, 3) of the plaintext, then with probability 2^{-8} we do not have a difference in the bytes (0, 3), (1, 2), (2, 1), (3, 0) at the end of E_0 . Back-wise, if we do not have a difference on the bytes (0, 3), (1, 2), (2, 1), (3, 0) at the end of E_0 , the state just after the MC of the round 0 does not have a difference on the byte (1, 0) which only depends on the bytes (0, 0), (1, 1), (2, 2), (3, 3) of the plaintext.

Middle: E_1 is linear, which implies that if a list of states is balanced after E_1 , the list of states before E_1 is balanced too.

Bottom Boomerang: $MC^{-1} \circ E_2$ is separable in four super S-box, which means that the bytes (0, 0), (1, 3), (2, 2), (3, 1) of the end state only depends of the bytes (0, 0), (1, 1), (2, 2), (3, 3) of the entry state, and an analogue property holds for the other diagonals.

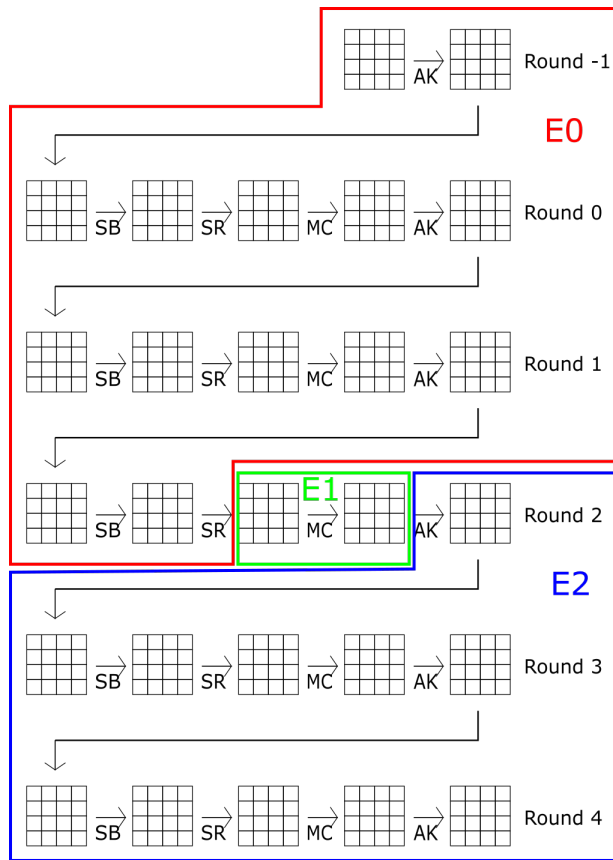


Figure 5.6: Decomposition of 5-round AES

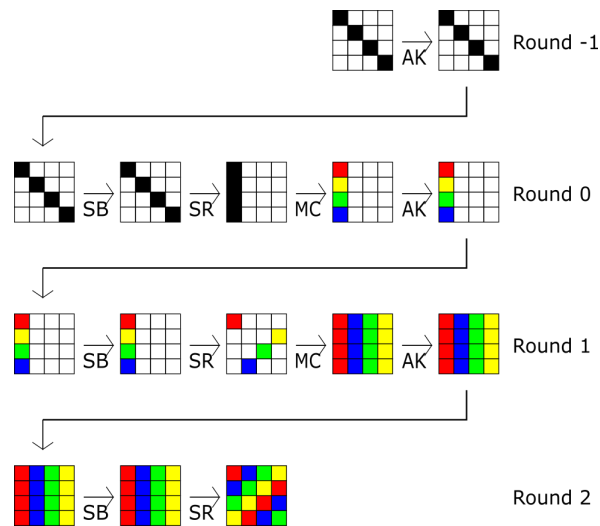


Figure 5.7: 2.5 round differential on AES

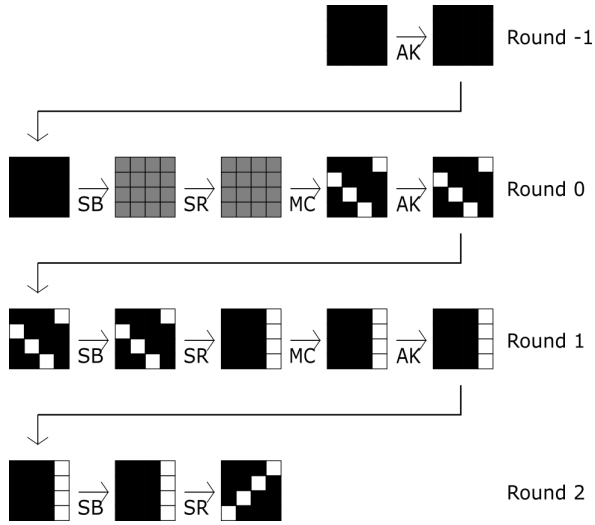


Figure 5.8: Rewinding the 2.5 round differential on AES

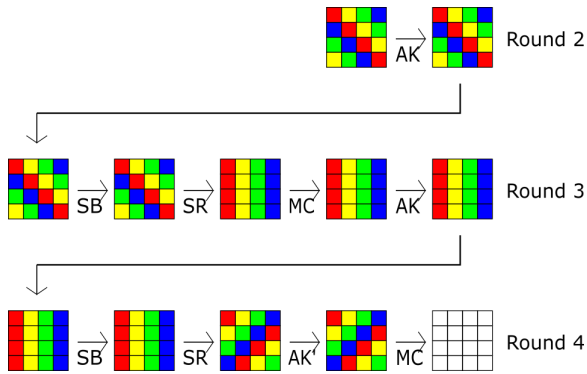


Figure 5.9: Separability of 1.5 round AES

5.6.2 Friend Pairs

If (P_1, P_2) is a pair of plaintexts we want to use in the boomerang, a “friend pair” (P'_1, P'_2) is a pair that correctly follows the boomerang if and only if the pair (P_1, P_2) does.

In our case, the byte $(1, 0)$ after the MC of round 0 is determined by

$$X_{(1,0)} = 01.S(P_{(0,0)} \oplus K_{(0,0)}) \oplus 02.S(P_{(1,1)} \oplus K_{(1,1)}) \oplus 03.S(P_{(2,2)} \oplus K_{(2,2)}) \oplus 01.S(P_{(3,3)} \oplus K_{(3,3)}) \quad (5.12)$$

Then we can explicitly build friend pairs by keeping the same differential and the same values on bytes $(0, 0)$, $(1, 1)$, $(2, 2)$, $(3, 3)$ than the original pair but changing the values on other bytes.

$$\left(\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} \right) \text{ and } \left(\begin{array}{|c|c|c|c|} \hline 0 & A & A & A \\ \hline A & 0 & A & A \\ \hline A & A & 0 & A \\ \hline A & A & A & 0 \\ \hline \end{array}, \begin{array}{|c|c|c|c|} \hline 1 & A & A & A \\ \hline A & 1 & A & A \\ \hline A & A & 1 & A \\ \hline A & A & A & 1 \\ \hline \end{array} \right)$$

Figure 5.10: Example of friend pairs

5.6.3 Value-difference Correspondence on AES S-box

The AES S-box can be decomposed into two steps:

1. an inversion on $GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ and 0 is mapped to 0;
2. the affine operation $f(x) = x \oplus (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 3) \oplus (x \lll 4) \oplus 0x63$.

Then with the identity for $y \neq x \neq 0$

$$\frac{y}{\frac{1}{x} \oplus \frac{1}{x \oplus y}} = \left(\frac{x}{y} \right)^2 \oplus \frac{x}{y}$$

computing x with y and $\frac{1}{x} \oplus \frac{1}{x \oplus y}$ is making a division, solving a linear system and multiplying by y which should be faster than an encryption. A quantum circuit for solving the AES S-Box differential equation was given in [27], that we can reuse.

5.6.4 Classical Attack

The attack consists in using the sharing of byte $X_{(1,0)}$ (Equation 5.12) with term simplifications thanks to the use of friend pairs.

The attack uses 2^{16} computations and 2^{15} encryptions.

Algorithm 5.11 Classical attack on 5-round AES (adapted from [49], Sec. 4.4)

Input: access to E and E^{-1}

- 1: Generate 2^8 plaintext pairs (P_1, P_2) such that they have a difference only on bytes $(0, 0)$ and $(1, 1)$.
- 2: For each pair, compute (P_3, P_4) for the pair and 2^7 friend pairs
- 3: For each pair, keep only the friend pair such that there is no difference in byte $(2, 2)$ or $(3, 3)$ in (P_3, P_4) and two other for verification.
- 4: For each pair, for each possible key byte $K_{(0,0)}$ compute the possible key byte $K_{(1,1)}$ that allow the boomerang to execute (equality on $X_{(1,0)}$ on the descending differential)(by correspondence between differential and value in the S-box)

▷ The equation is $(02)^{-1} \cdot (S(P_{1,(0,0)} \oplus K_{(0,0)}) \oplus S(P_{2,(0,0)} \oplus K_{(0,0)})) = S(P_{1,(1,1)} \oplus K_{(1,1)}) \oplus S(P_{2,(1,1)} \oplus K_{(1,1)})$ and is solved by correspondence between differential and value in the S-box

- 5: For each pair and key byte $K_{(0,0)}$, compute the possible key byte $K_{(3,3)}$ with the equality on byte $X_{(1,0)}$ (if we had no difference in the byte $(2, 2)$ on the friend pair, byte $(2, 2)$ otherwise)

▷

The equation is $S(P_{1,(0,0)} \oplus K_{(0,0)}) \oplus S(P_{2,(0,0)} \oplus K_{(0,0)}) \oplus 02 \cdot (S(P_{1,(1,1)} \oplus K_{(1,1)}) \oplus S(P_{2,(1,1)} \oplus K_{(1,1)})) = S(P_{1,(3,3)} \oplus K_{(3,3)}) \oplus S(P_{2,(3,3)} \oplus K_{(3,3)})$ and is solved by correspondence between differential and value in the S-box

- 6: For each pair and key byte $K_{(0,0)}$, compute the possible key byte $K_{(2,2)}$ (if we had no difference in the byte $(2, 2)$ on the friend pair, byte $(3, 3)$ otherwise) with the equality on byte $X_{(1,0)}$ on the initial pair.

▷ The

equation is $(03)^{-1} \cdot (S(P_{1,(0,0)} \oplus K_{(0,0)}) \oplus S(P_{2,(0,0)} \oplus K_{(0,0)}) \oplus 02 \cdot (S(P_{1,(1,1)} \oplus K_{(1,1)}) \oplus S(P_{2,(1,1)} \oplus K_{(1,1)})) \oplus S(P_{1,(3,3)} \oplus K_{(3,3)}) \oplus S(P_{2,(3,3)} \oplus K_{(3,3)}) = S(P_{1,(2,2)} \oplus K_{(2,2)}) \oplus S(P_{2,(2,2)} \oplus K_{(2,2)})$ and is solved by correspondence between differential and value in the S-box

- 7: Restart with other diagonals to get the full key
-

5.6.5 Quantum Attack

Algorithm 5.12 Quantum attack on 5-round AES

- Input:** access to E and E^{-1}
- 1: **Amplify** $\sqrt{2^8}$ **times**
 - 2: Sample a pair (P_1, P_2) such that they have a difference only on bytes $(0, 0)$ and $(1, 1)$.
 - 3: **Amplify** $\sqrt{2^7}$ **times**
 - 4: Sample a “friend” pair
 - 5: Verify that there is no difference in byte $(2, 2)$ or $(3, 3)$ in (P_3, P_4)
 - 6: **EndAmplify**
 - 7: Sample two other friend pairs for verification
 - 8: **Amplify** $\sqrt{2^8}$ **times**
 - 9: Sample a guess for $K_{(0,0)}$
 - 10: Compute the possible key bytes $K_{(1,1)}$ that allow the boomerang to execute (by correspondence between differential and value in the S-box)
 - 11: Compute the possible key bytes $(3, 3)$ with the equality on byte $(1, 0)$ (if we had no difference in the byte $(2, 2)$ on the friend pair, byte $(2, 2)$ otherwise)
 - 12: Compute the possible key bytes $(2, 2)$ (if we had no difference in the byte $(2, 2)$ on the friend pair, byte $(3, 3)$ otherwise) with the equality on byte $(1, 0)$ on the initial pair.
 - 13: **EndAmplify**
 - 14: **EndAmplify**
 - 15: Restart with other diagonals to get the full key
-

The attack uses $2^{8/2} \times (2^{7/2} + 2^{8/2}) = 2^8$ computations.

5.7 Conclusion

We proposed an efficient quantum boomerang attack, as well as a variant for mixing boomerang attacks. We proposed several improvements for different cases, as reducing the quantum RAM need or making $Q2$ attacks work in $Q1$ under certain circumstances. In some cases, our attacks reach a quadratic speedup with respect to classical attacks. This shows that boomerang attacks are also performant cryptanalysis tools for the post-quantum world, that will be needed for correctly determining the best security margins.

Chapter 6

Safely Doubling your Block Ciphers for a Post-Quantum World

6.1 Introduction

The aim of this chapter is to provide new useful information in order to advance towards building a generic and provable way of easily extending secure classical constructions into new ones with doubled key and doubled state-size, while their post-quantum security remains comparable to the original classical one. For this, we consider a new approach, based on the Encrypt-Mix-Encrypt paradigm with five block-cipher calls, that would allow to extend the internal state as well as the key size of any classically secure block-cipher.

The ECB-Mix-ECB or EME construction [66] was proposed as a highly parallelisable mode of operation to extend the domain of a block-cipher to arbitrary lengths. The ECB layers above and below make it a suitable candidate for resisting quantum attacks, since most Simon-like attacks rely on some part of the input passing through only one block-cipher call or being XOR-ed directly to the state, and the ECB layers ensure that every part of the input passes through at least two block-cipher calls during both the encryption and decryption routines.

Our Contributions. The main contributions of our work are:

1. An original quantum superposition attack on the EME construction. It introduces a new family of attacks that exhibits a periodic property found in collisions;
2. A new construction based on Encrypt-Mix-Encrypt that resists this attack, by replacing the XOR-then-Encrypt mixing layer with a tweakable permutation call, that we will call QuEME, and that is claimed to offer n bits of security in both the classical and the quantum setting;
3. Four classical proofs for this construction: an IND-CPA proof of security up to the width of the underlying block-cipher using mirror theory; an IND-CCA proof up to the same bound using a conjectured tighter version of mirror theory, for which we find numerical evidence through an original

approach using some simulations; and direct proofs of IND-CPA and IND-CPA security up to $2/3$ -rd the width of the block-cipher;

4. An original distinguisher that applies in particular to our construction that runs in time $O(2^n)$, which matches our security bound;
5. The first quantum security arguments for Encrypt-Mix-Encrypt constructions, to show that QuEME has, at least, $n/6$ bits of quantum security, which is in par for instance with the quantum security of LR4 [67], and show in particular that there is no collapse in the quantum security as can happen in LR3 for instance [84];
6. A concrete instantiation of the QuEME scheme with AES-128 for building a block-cipher with 256-bit state and key, with concrete (quantum) security claims; and some variants with fewer rounds than 10 for the building block that we also believe should be resistant.

It is based on a joint work with Ritam Bhaumik, André Chailloux and María Naya-Plasencia that is currently in a review process.

Contents

6.1	Introduction	115
6.2	General Related Concepts	117
	6.2.1 Mirror Theory	117
	6.2.2 H-Coefficient Technique	119
6.3	Applying Simon’s Algorithm to Functions Restricted on a Subset	120
6.4	Constructions Based on Encrypt-Mix-Encrypt Paradigm: First Attempt and Attack	121
	6.4.1 New Superposition Attack on the EME Construction	122
6.5	New Construction and Classical Security Proofs	125
	6.5.1 Proof of Classical Security	126
	6.5.2 IND-CPA Security Proof of n -bit Security using Mirror Theory	128
	6.5.3 IND-CCA Security Proof of n -bit Security using Mirror Theory	131
	6.5.4 IND-CPA Security Proof of $(2n/3)$ -bit Security	133
	6.5.5 IND-CCA Security Proof of $(2n/3)$ -bit Security	135
6.6	Simulation of the Mirror Theory	138
6.7	An $O(2^n)$ -Distinguisher on our Scheme	141
6.8	Quantum Security	143
	6.8.1 Hardness of Distinguishing a Random Permutation from a Random Function with Small Range	144
	6.8.2 Quantum Security Statement	144

6.8.3	Discussion	147
6.9	Proposing a Concrete Instance: Double-AES	148
6.9.1	How to Extend the Keys?	148
6.9.2	Introducing a Domain in AES for Defining E_j	149
6.9.3	Double-AES Concrete Proposals	149
6.9.4	Security Claims	150
6.9.5	Discussion	150
6.10	Best Attacks Found of Double-AES	151
6.10.1	Attack on X-3-3 Without Last MC in Blocks E_1 , E_2 and E	151
6.10.2	Attack on X-2-X Without Last MC in Blocks E_1 , E_2 and E	153
6.10.3	Best known attacks on AES	154
6.11	Conclusion	154

6.2 General Related Concepts

In this section, we introduce the Mirror theory and the H-coefficient technique.

6.2.1 Mirror Theory

We'll use some results from a line of research that focuses on finding tight approximations on the number of solutions to systems of bi-variate equations, which goes by the name Mirror Theory [97]. Some of these results have been proved [52, 37, 98], while others are still conjectural [100], but generally accepted in the community.

We'll use the Pochhammer falling factorial power notation

$$(a)_b := a(a-1)\dots(a-b+1).$$

Let N denote 2^n . Consider two sequences of n -bit variables Y_1, \dots, Y_{q_1} and Z_1, \dots, Z_{q_2} , with $q_1 < N, q_2 < N$. For some $q < q_1 + q_2$ suppose there are q bi-variate equations of the form

$$Y_i \oplus Z_j = \delta_{ij}.$$

Then the *Mirror Theorem* states the following.

Theorem 6.1 (Mirror Theorem (conjectural)). *The number of solutions to the system described above such that Y_i 's are all distinct and Z_i 's are all distinct is at least*

$$\frac{(N)_{q_1}(N)_{q_2}}{N^q} \cdot (1 - \epsilon),$$

where $\epsilon = O(q/N)$.

The intuition behind this is that the numerator in the above expression is the total number of solutions satisfying just the distinctness constraint, and any randomly chosen solution has a probability of about $1/N^q$ of satisfying all q bi-variate equations. However, the exact calculations needed to bound ϵ can be very complicated, and so far has only been completed for the special case with $q_1 = q_2 = q$ where each variable appears in exactly one equation. In spite of this, the result has been claimed earlier without complete proofs, first by Patarin and then by others citing him, and is generally accepted in the community. In this chapter, we'll use the mirror theorem as a black box.

Tighter Version. When many of the variables appear multiple times, the above bound can be made tighter. Consider the graph where a bi-variate equation involving Y_i and Z_j is represented by an edge between Y_i and Z_j . We assume none of the equations is redundant and the system of equations is consistent, so there are no cycles in this graph. Let $C^{(1)}, \dots, C^{(t)}$ be the connected components, where $t = q_1 + q_2 - q$. For each $j \in [1..t]$ let $q_1^{(j)}$ (resp. $q_2^{(j)}$) be the number of Y_i 's (resp. Z_i 's) that appear in $C^{(j)}$. Finally, define the cumulative sums

$$Q_b^{(j)} = \sum_{i=1}^{j-1} q_b^{(i)}$$

for each $b \in \{1, 2\}$ and each $j \in [1..t]$. Then the *Tight Mirror Theorem* states the following.

Theorem 6.2 (Tight Mirror Theorem (conjectural)). *The number of solutions to this system such that Y_i 's are all distinct and Z_i 's are all distinct is at least*

$$\frac{1}{N^q} \cdot \prod_{j=1}^t \left[\left(N - Q_1^{(j)} \right)^{q_1^{(j)}} \left(N - Q_2^{(j)} \right)^{q_2^{(j)}} \right] \cdot (1 - \epsilon),$$

where $\epsilon = O(q/N)$.

The intuition behind this is an extension of the intuition behind the Mirror Theorem. As before we randomly choose a valid solution for the $\{Y_i\}$ and the $\{Z_i\}$, and it satisfies the equations with (roughly) a probability $1/N^q$. However, in this case, the key additional observation is that when choosing the valid solution, instead of ensuring distinctness among all $\{Y_i\}$ and all $\{Z_i\}$, we just need to ensure that there are no collisions between components; since our system of equations is consistent, for any solution that satisfies the equations, within-component distinctness is automatically ensured. Thus when choosing the $q_1^{(j)}$ Y_i 's from the j -th component, we just choose them randomly from all the $N - Q_1^{(j)}$ unsampled values, and similarly for the Z_i 's.

When $q_1^{(j)} = q_2^{(j)} = 1$ for each j , this reduces to the Mirror Theorem bound, and is strictly tighter in other cases, the gap increasing as the component sizes

increase. We'll use this result as a black box too, to deal with cases where the component sizes are difficult to bound. In Section 6.6, we discuss some simulations we ran that suggest that this conjecture is not unreasonable.

6.2.2 H-Coefficient Technique

Suppose an adversary \mathcal{A} is playing a distinguishing game against two oracles, one representing an ideal cryptographic object f , and the other representing an actual cryptographic construction \mathcal{C} . We'll use the standard terminology where the oracle representing f is called the *ideal oracle*, denoted \mathcal{O}_0 , and the oracle representing \mathcal{C} is called the *real oracle*, denoted \mathcal{O}_1 . Formally, the challenger samples a secret bit b at the beginning, and gives \mathcal{A} the oracle \mathcal{O}_b ; \mathcal{A} makes q queries $\{u^i \mid i \in [1..q]\}$ and receives the corresponding responses $v^i = \mathcal{O}_b(u^i)$; at the end of the game, \mathcal{A} outputs a response bit b' , and wins if $b' = b$. The scenario where \mathcal{A} interacts with the ideal (resp. real) oracle will interchangeably be called the ideal (resp. real) world.

To bound the advantage of \mathcal{A} we use the *Coefficient H Technique*. Let τ be the *transcript* of a q -query game played by \mathcal{A} , i.e., $\tau = \{(u^i, v^i) \mid i \in [1..q]\}$. In addition, when \mathcal{A} interacts with \mathcal{C} , let τ^* denote the *internal transcript*, denoted $\tau^* = \{w^i \mid i \in [1..q]\}$; these are the intermediate variables computed when computing the responses to \mathcal{A} 's queries. Note that here u^i, v^i, w^i are of unspecified length—the first two will depend on the oracle interface and the last on the complexity of the internal computations.

Extended Transcripts. Let \mathcal{S} be a sampler that takes τ as input and simulates an internal transcript τ^* when \mathcal{A} interacts with f . We consider a modified game where at the end of the game τ^* is released to \mathcal{A} , which can be used to compute the final response bit; as this is extra information \mathcal{A} is free not to use, this can only increase the advantage of \mathcal{A} , so any upper bound we derive here must hold for the original game as well. (τ, τ^*) together will be called the *extended transcript*. (We'll simply call it a transcript when the context is unambiguous.)

Good Transcripts. We'll define one or several *bad events* in the ideal world, based on the internal random coins of f and \mathcal{S} . (Note that for \mathcal{A} we only consider deterministic adversaries.) In the first step of the proof, we'll need to show that for some suitably small ϵ_1 the probability that at least one bad event occurs is upper-bounded by ϵ_1 . We'll call a transcript (τ, τ^*) *good* if it can be obtained in the ideal world without encountering any of the bad events.

Interpolation Probabilities. Given a transcript (τ, τ^*) and an oracle \mathcal{O}_b , we'll examine the *interpolation probability* of (τ, τ^*) in \mathcal{O}_b , denoted $\Pr_b[(\tau, \tau^*)]$. This is the probability that the internal random coins of \mathcal{O}_b are *compatible* with (τ, τ^*) , i.e., (τ, τ^*) is obtained as the game transcript as long as \mathcal{A} queries u^1

first and for each $i \in [2..q]$, on observing v^1, \dots, v^{i-1} , next queries u^i . For an adversary who does not make these queries, the probability of obtaining (τ, τ^*) is trivially 0, and we ignore such adversaries when calculating the interpolation probability.

Ratio of Good Probabilities. For the second step of the proof, we'll need to find a suitably small ϵ_2 such that for any arbitrary good transcript (τ, τ^*) , the ratio of $\Pr_1[(\tau, \tau^*)]$ and $\Pr_0[(\tau, \tau^*)]$ is lower-bounded by $1 - \epsilon_2$. This ratio is often simply referred to as the *ratio of good probabilities*, whereas the probability of at least one bad event occurring (applicable only for the ideal world) is referred to as the *bad probability*.

Once we have the stated bounds on both the bad probability and the ratio of good probabilities, the main result of the H-Coefficient Technique [99] tells us that the distinguishing advantage of \mathcal{A} between \mathcal{O}_0 and \mathcal{O}_1 cannot exceed $\epsilon_1 + \epsilon_2$. Below we give the formal statement of the theorem we will use.

Theorem 6.3 (H-Coefficient Technique). *Suppose for an adversary \mathcal{A} playing a q -query distinguishing game between an ideal object f and a real construction \mathcal{C} , we can define bad events and find ϵ_1 and ϵ_2 such that the probability of a bad event in a game against f is at most ϵ_1 , and the ratio of good probabilities while interacting with \mathcal{C} and f for any fixed good transcript τ is at least $1 - \epsilon_2$. Then we have*

$$\text{Adv}_{\mathcal{C}}^{\mathcal{A}} \leq \epsilon_1 + \epsilon_2.$$

6.3 Applying Simon's Algorithm to Functions Restricted on a Subset

In this section, we explain how we proceed to use Simon's algorithm (Section 2.6.4) to functions defined on any set and extend the Offline-Simon algorithm (Section 3.4.5.2) in this context.

An important remark to Simon's algorithm is that we do not need g if we have access to cn superposition states $|\phi_g\rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle |g(x)\rangle$. Moreover, we do not need the superposition to include all x in $\{0,1\}^n$, it is possible to restrict g to a subset A as long as the subset admits s as a period i.e., $x \in A$ iff. $x \oplus s \in A$, and A does not make an artificial period appear (by restricting on elements such that $g(x \oplus t) = g(x)$ for a certain t). This can be taken to an extreme where $g = 0$ but A has the information of the period.

Theorem 6.4. *Suppose that $g : A \subseteq \{0,1\}^n \rightarrow X$ has a period s , i.e. $x \oplus s \in A$ $g(x \oplus s) = g(x)$ for all $x \in A$ and satisfies*

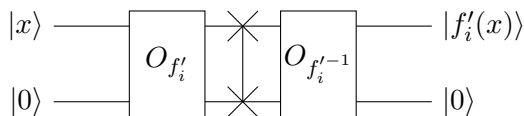
$$\max_{t \notin \{0,s\}} \mathbb{P}_{x \in A}(\tilde{g}(x \oplus t) = g(x)) \leq \frac{1}{2}$$

$$\text{where } \tilde{g}(x) = \begin{cases} g(x) & \text{if } x \in A \\ \perp & \text{otherwise} \end{cases}$$

When we apply Simon's algorithm to cn copies of $|\phi_g\rangle = \sum_{x \in A} \frac{1}{\sqrt{|A|}} |x\rangle |g(x)\rangle$, it returns s with a probability at least $1 - 2^n \cdot (3/4)^{cn}$. It is running in time cn^2 .

Then, because the properties of Simon's algorithm did not change because of the input restriction on g , we can apply the offline Simon's algorithm (Section 3.4.5.2) ideas.

This technique relies on the equality $|\phi_{f_i \oplus g}\rangle = O_{f_i} |\phi_g\rangle$ for preparing and recovering the databases $|\phi_g\rangle$. In our case, instead of $f_{i_0} \oplus g$ being periodic, we look for $f_{i_0} \oplus (g \circ f'_{i_0})$ being periodic with f'_i as public permutations. We build the operator $IN_{f'_i} : |x\rangle \mapsto |f'_i(x)\rangle$ using ancilla qubits and the following circuit:



We can compute $|\phi_{f_{i_0} \oplus g \circ f'_{i_0}}\rangle = O_{f_{i_0}} \circ (IN_{f'_{i_0}} \otimes I) |\phi_g\rangle$.

Theorem 6.5. Suppose that $m = O(n)$, $\{f_i\}_{i \in \{0,1\}^m}$ a family of public functions from $\{0,1\}^n \rightarrow \{0,1\}^l$, $\{f'_i\}_{i \in \{0,1\}^m}$ a family of public permutations from $\{0,1\}^n \rightarrow \{0,1\}^n$ and $g : A \subseteq \{0,1\}^n \rightarrow \{0,1\}^l$ on which we only get some databases $|\phi_g\rangle$ and there is a unique i_0 such that $f_{i_0} \oplus (g \circ f'_{i_0})$ has a period s and

$$\max_{(i,t) \notin \{0,1\}^m \times \{0\} \cup \{(i_0,s)\}} \mathbb{P}_{x \in f'^{-1}(A)} ((f_i \oplus (\tilde{g} \circ f'_i))(x \oplus t) = (f_i \oplus (g \circ f'_i))(x)) \leq \frac{1}{2}$$

With $O(n)$ databases $|\phi_g\rangle = \sum_{x \in A} \frac{1}{\sqrt{|A|}} |x\rangle |g(x)\rangle$, we can recover i_0 with a probability in $\Theta(1)$. The running time is $O(n^3 2^{m/2})$.

6.4 Constructions Based on Encrypt-Mix-Encrypt Paradigm: First Attempt and Attack

Our aim is to build a $2n$ -bit-to- $2n$ -bit block cipher using an n -bit block cipher, ideally with five or fewer calls to the blockcipher. Specifically, we studied the encrypt-mix-encrypt paradigm, where the plaintext blocks first pass through a (weak) encryption layer, then an invertible mixing layer with possibly non-linear components, and then another encryption layer. For the encryption layers we can begin with something very simple, like an ECB layer (with different keys). Then our generic encryption function becomes

$$(L, R) \mapsto (E_3(M(E_1(L), E_2(R))_\ell), E_4(M(E_1(L), E_2(R))_r)),$$

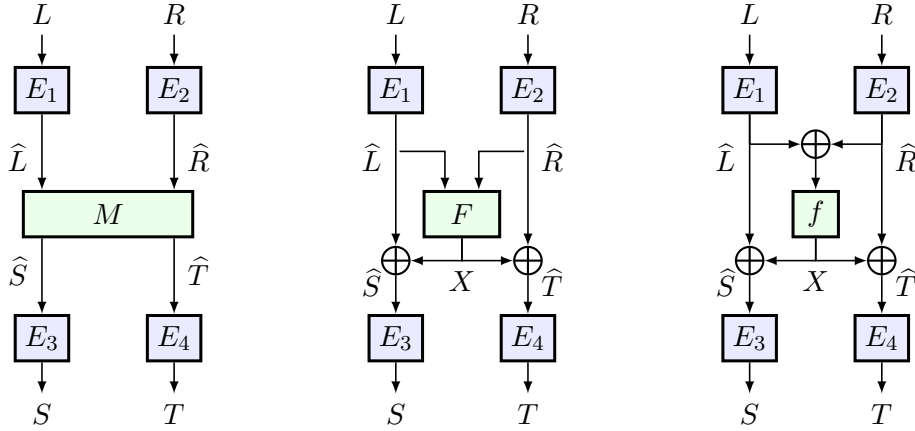


Figure 6.1: **Left:** The generic construction, with an invertible mixing layer $M : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$. **Centre:** With a specific mixing layer, with a compressing function $F : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$. **Right:** A more specific instantiation of F , with a prf $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

where E_1, \dots, E_4 are block ciphers with independent keys, and M is a $2n$ -bit-to- $2n$ -bit mixing layer with $M(\cdot, \cdot)_\ell$ and $M(\cdot, \cdot)_r$ indicating the left and right halves of its output respectively (Figure 6.1, left).

For a specific mixing function, we take a $2n$ -bit-to- n -bit compressing function F and use

$$M(x, y) := (x \oplus F(x, y), y \oplus F(x, y))$$

as our mixing layer (Figure 6.1, center). Since M needs to be invertible, we have the condition on F that (x, y) should be recoverable from $(x \oplus F(x, y), y \oplus F(x, y))$. One easy way to achieve this is to take an n -bit-to- n -bit function f and define

$$F(x, y) := f(x \oplus y).$$

This was the first specific construction we considered (Figure 6.1, right). We assumed f is a qprf, and E_1, \dots, E_4 are qprp's.

6.4.1 New Superposition Attack on the EME Construction

While analyzing the quantum security of this construction with $F(x, y) := f(x \oplus y)$, we discovered a new kind of superposition attack.

The original idea of the attack is to build in the beginning a superposition of states that partially collide in the left output, and next, performing an exhaustive search of the key we are able to build a function that will be periodic in the subset of colliding states. This procedure is presented in Algorithm 6.1, and takes $\tilde{O}(2^{k/2} + 2^{n/3})$ computations for recovering the key of E_2 . The same can be applied for recovering the keys of E_1 , E_3 or E_4 (as the inverse has the

same shape). This is a new kind of superposition attack, that combines for the first time BHT, Grover and Simon (introduced in Chapter 2), and where BHT is used to restraint the function to the interesting outputs that generate a partial collision, and that will next verify a certain property. We believe this attack might apply to more constructions, and it should be considered in further studies analyzing the security of symmetric primitives with respect to superposition attacks.

Algorithm 6.1 Attack on construction

- Input:** superposition oracle access to E
Output: the key k_2 of E_2
- 1: Choose two values L_0 and L_1
 - 2: **Repeat** $O(n)$ **times** (for confirmation)
 - 3: Search with BHT algorithm for claws on $F_b : R \mapsto S(L_b, R)$ (using $2^{n/3}$ turns)
 - ▷ We get a uniform superposition $\sum |R_0, R_1\rangle$ of all elements of the set $\{(R, R')/f(E_1(L_0) \oplus E_2(R_0)) \oplus f(E_1(L_1) \oplus E_2(R_1)) = E_1(L_0) \oplus E_1(L_1)\}$.
 - 4: **EndRepeat**
 - ▷ We get $O(n)$ superpositions that we use as a database for the following Grover search
 - 5: **Grover search** on k_2 with $2^{k/2}$ turns using the following oracle :
 - 6: **ForEach** superposition $\sum |R_0, R_1\rangle$
 - 7: Add an external qubit $|b\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
 - 8: Apply $(b, R_0, R_1) \mapsto (b, E_{2,k_2}(R_b), E_{2,k_2}(R_{1-b}))$
 - ▷ If we guessed right, we get a uniform superposition of the set $\{(b, R_0, R_1)/f(E_1(L_b) \oplus R_0) \oplus f(E_1(L_{1-b}) \oplus R_1) = E_1(L_0) \oplus E_1(L_1)\}$. This set admits the period $(1, E_1(L_0) \oplus E_1(L_1), E_1(L_0) \oplus E_1(L_1))$.
 - 9: **EndFor**
 - 10: Apply Simon's algorithm on the resulting superposition with the function $(b, R, R') \mapsto R' \oplus R$.
 - ▷ If we guessed right, the function on this set admits the period $(1, E_1(L_0) \oplus E_1(L_1), E_1(L_0) \oplus E_1(L_1))$. Detecting a period on a wrong guess would mean a weakness of f .
 - 11: Uncompute to get back the superpositions $\sum |R_0, R_1\rangle$
 - 12: **EndGrover**
 - 13: Return k_2
-

Description of the Attack. Let $S(L, R)$ be the function corresponding to this construction for inputs L, R (see rightmost construction of Figure 6.1) so

$$S(L, R) = E_3(E_1(L) \oplus f(E_1(L) \oplus E_2(R))).$$

We start by fixing two distinct values L_0 and L_1 for left entries. Then, we consider the uniform superposition of claws between $F_0 : R \mapsto S(L_0, R)$ and

$F_1 : R \mapsto S(L_1, R) :$

$$\frac{1}{\sqrt{|\{(R_0, R_1), S(L_0, R_0) = S(L_1, R_1)\}|}} \sum_{S(L_0, R_0) = S(L_1, R_1)} |R_0, R_1\rangle.$$

(We can obtain this with BHT algorithm with complexity $2^{n/3}$.)

We observe that $S(L_0, R_0) = S(L_1, R_1)$ is equivalent to

$$f(E_1(L_0) \oplus E_2(R_0)) \oplus f(E_1(L_1) \oplus E_2(R_1)) = E_1(L_0) \oplus E_1(L_1).$$

We add an external qubit $|b\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ to the state and apply the controlled exchange $(b, R_0, R_1) \mapsto (b, R_b, R_{1-b})$. The state becomes the uniform superposition of elements from

$$\{(b, R_0, R_1) / f(E_1(L_b) \oplus E_2(R_0)) \oplus f(E_1(L_{1-b}) \oplus E_2(R_1)) = E_1(L_0) \oplus E_1(L_1)\}.$$

This state is called $|\Phi\rangle$ for the rest of the attack.

Now, if we guess the key of E_2 right, we can apply $\mathcal{O}_{E_2}^{sup} : |x\rangle \rightarrow E_2(|x\rangle)$ on the 2 rightmost register of $|\Phi\rangle^1$ and get the superposition

$$\frac{1}{\sqrt{2|\{(R_0, R_1), S(L_0, R_0) = S(L_1, R_1)\}|}} \sum_{(b, R_0, R_1) \in A} |b, R_0, R_1\rangle$$

where $A = \{(b, R_0, R_1) / f(E_1(L_b) \oplus E_2(R_0)) \oplus f(E_1(L_{1-b}) \oplus E_2(R_1)) = E_1(L_0) \oplus E_1(L_1)\}$.

This set admits the period $(1, E_1(L_0) \oplus E_1(L_1), E_1(L_0) \oplus E_1(L_1))$, i.e. if (b, R_0, R_1) is in the set then $(b \oplus 1, R_0 \oplus E_1(L_0) \oplus E_1(L_1), R_1 \oplus E_1(L_0) \oplus E_1(L_1))$ is also in the set.

Then, we apply Simon's algorithm on $(b, R_0, R_1) \mapsto R_0 \oplus R_1$ to recover the existence of this period and uncompute the last steps to recover the states $|\Phi\rangle$.

This part is the execution of Theorem 6.5 with $g = 0$, $A = \{(b, R_0, R_1) | S(L_0, R_0) = S(L_1, R_1)\}$, $f'_i : (b, R_0, R_1) \mapsto (b, E_{2, k_2}(R_b), E_{2, k_2}(R_{1-b}))$, $f_i : (b, R, R') \mapsto R' \oplus R$ and $s = (1, E_1(L_0) \oplus E_1(L_1), E_1(L_0) \oplus E_1(L_1))$.

The attack has complexity $\tilde{O}(2^{k/2} + 2^{n/3})$.

Similar Mixing Layers. While the previous description of our attack targets the specific mixing layer from figure 6.1-right, our attack can be adapted and impact also the other mixing layers $M(x, y)$ the following way. We describe the mixing layer the following way:

$$M(x, y) = \Pi_2(f(\Pi_1(x, y)), x, y)$$

¹This is possible since from the key of E_2 , we can compute E_2 and E_2^{-1} efficiently. We then compute $|x\rangle|0\rangle \xrightarrow{O_{E_2}} |x\rangle|E_2(x)\rangle \xrightarrow{Swap} |E_2(x)\rangle|x\rangle \xrightarrow{O_{E_2}^{-1}} |E_2(x)\rangle|0\rangle$.

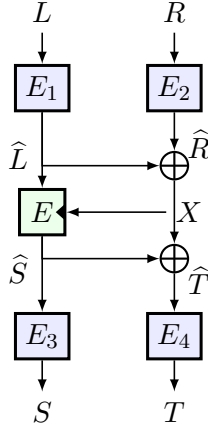


Figure 6.2: The new construction QuEME; the E in the middle layer takes its key as an input from the right.

By linearity, we write $\Pi_1(x, y) = \Pi_{1,L}(x) \oplus \Pi_{1,R}(y)$ and $\Pi_2(f, x, y) = f \oplus \Pi_{2,L}(x) \oplus \Pi_{2,R}(y)$ (even if it means rewriting the function f). The collision equation $S(L_0, R_0) = S(L_1, R_1)$ is then equivalent to

$$f(\Pi_{1,L} \circ E_1(L_0) \oplus \Pi_{1,R} \circ E_2(R_0)) \oplus f(\Pi_{1,L} \circ E_1(L_1) \oplus \Pi_{1,R} \circ E_2(R_1)) = \Pi_{2,L} \circ E_1(L_0) \oplus \Pi_{2,L} \circ E_1(L_1) \oplus \Pi_{2,R} \circ E_2(R_0) \oplus \Pi_{2,R} \circ E_2(R_1).$$

With a good guess and the controlled exchange, the equation of collisions becomes

$$f(\Pi_{1,L} \circ E_1(L_b) \oplus \Pi_{1,R}(R_0)) \oplus f(\Pi_{1,L} \circ E_1(L_{1-b}) \oplus \Pi_{1,R}(R_1)) = \Pi_{2,L} \circ E_1(L_0) \oplus \Pi_{2,L} \circ E_1(L_1) \oplus \Pi_{2,R}(R_0) \oplus \Pi_{2,R}(R_1).$$

Then, we discuss what happens whether $\Pi_{1,R}$ is reversible or not.

First case: $\Pi_{1,R}$ is not reversible. Then there exists $t \neq 0$ such that $\Pi_{1,R}(t) = 0$. and the set of collisions admits the period $s = (0, t, t)$.

Second case: $\Pi_{1,R}$ is reversible. Then we note $t = \Pi_{1,R}^{-1} \circ \Pi_{1,L}(E_1(L_0) \oplus E_1(L_1))$ and the set of collisions admits the period $s = (1, t, t)$.

Overall, the attack works the same way but the recovered period will be different.

6.5 New Construction and Classical Security Proofs

In this chapter, we propose the new construction QuEME shown in shown in Fig. 6.2. We define $\text{QuEME}^E : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ as

$$\text{QuEME}^E(L, R) := (S, T),$$

where

$$\begin{aligned}\widehat{L} &= E(K_1, L), & \widehat{R} &= E(K_2, R), \\ X &= \widehat{L} \oplus \widehat{R}, \\ \widehat{S} &= E(X, \widehat{L}), & \widehat{T} &= X \oplus \widehat{S}, \\ S &= E(K_3, \widehat{S}), & T &= E(K_4, \widehat{T}),\end{aligned}$$

for a block-cipher $E(\cdot, \cdot)$ and four keys K_1, \dots, K_4 . For each $i \in [1..4]$ we'll use the notation write $E_i := E(K_i, \cdot)$.

6.5.1 Proof of Classical Security

First we analyse the information-theoretic security of QuEME against a classical adversary. To move to the information-theoretic setting, we replace E_1, E_2, E_3, E_4 with independent random permutations $\pi_1, \pi_2, \pi_3, \pi_4$, and $E(\cdot, \cdot)$ with a tweakable random permutation $\tilde{\pi}$. We call this modified construction QuEME $^\pi$.

Since the inner call to E uses a dynamic key derived from the internal state, we consider the possibility that \mathcal{A} tries to guess the internal key with offline queries to E . However, we note that for a good block-cipher, the only way this can provide \mathcal{A} additional information about the game is if they can guess a key-message pair or a key-ciphertext pair. Since there are $O(q)$ internal input and output pairs for E , this gives a $O(qq'/2^{2n})$ probability of guessing one such pair correctly with q' offline queries to E . In addition, let $\alpha^E(q, q')$ be an upper-bound on the advantage of an adversary who has learnt q' key-plaintext-ciphertext triples for E and is trying to distinguish q other key-plaintext-ciphertext triples from random. In our security bounds, whenever we switch from QuEME $^\pi$ to QuEME E , we add the terms $O(qq'/2^{2n})$ and $\alpha^E(q, q')$ to \mathcal{A} 's advantage as the cost of replacing the inner call to E with $\tilde{\pi}$, which should be a $O(qq'/2^{2n})$ for a good block cipher.

Settings of the Game. For convenience, we first make a few standard stipulations and modifications to the sprp game. We do not allow \mathcal{A} to make any *redundant queries*, which can either be repetitions of earlier queries or the feeding back in the opposite direction of an earlier response. Since such queries can give \mathcal{A} no extra information, this constraint cannot decrease the advantage of \mathcal{A} .

As a final modification, we consider the extended transcript game defined in Section 6.2.2, with $\tau = \{(L^i, R^i, S^i, T^i) \mid i \in [1..q]\}$ and $\tau^* = \{(\widehat{L}^i, \widehat{R}^i, X^i, \widehat{S}^i, \widehat{T}^i) \mid i \in [1..q]\}$. Defining the sampler \mathcal{S} will be a critical part of the proof.

Transcript Graphs. As preparation for sampling the internal transcript τ^* , we first define two undirected bipartite graphs G and H on the external

transcript τ . The vertices of G are the q_1 distinct values L_1, \dots, L_{q_1} in the set $\{L^i \mid i \in [1..q]\}$ and the q_2 distinct values R_1, \dots, R_{q_2} in the set $\{R^i \mid i \in [1..q]\}$ (we'll soon specify how we pick these labels); we put an edge between L_j and R_k if they appear together in some query, i.e., there is a query $i \in [1..q]$ with $(L^i, R^i) = (L_j, R_k)$. H is defined identically except with the ciphertexts $\{(S^i, T^i) \mid i \in [1..q]\}$ replacing the plaintexts in the above definition of G .

Let α (resp. β) be the number of components in G (resp. H). We label these components $G^{(1)}, \dots, G^{(\alpha)}$ and $H^{(1)}, \dots, H^{(\beta)}$. For $t \in [1..\alpha]$ let $q_1^{(t)}$ (resp. $q_2^{(t)}$) be the number of L -nodes (resp. R -nodes) in $G^{(t)}$. Similarly, for $t \in [1..\beta]$ let $q_3^{(t)}$ (resp. $q_4^{(t)}$) be the number of S -nodes (resp. T -nodes) in $H^{(t)}$. Define the cumulative sums

$$Q_b^{(j)} = \sum_{i=1}^{j-1} q_b^{(i)}$$

for each $j \in [1..\alpha]$ when $b \in \{1, 2\}$ and each $j \in [1..\beta]$ when $b \in \{3, 4\}$. We assume the labelling of the L -nodes in G is such that the nodes $\{L_k \mid Q_1^{(j)} + 1 \leq k \leq Q_1^{(j+1)}\}$ are in $G^{(j)}$, and likewise for the R -nodes, S -nodes, and T -nodes.

6.5.1.1 Classical Security Claim.

We claim the following security bound for QuEME^π .

Theorem 6.6. *For any classical adversary \mathcal{A} playing a q -query IND-CCA game against QuEME^π , we have*

$$\mathbf{Adv}_{\text{sprp}}^{\text{QuEME}^\pi}(\mathcal{A}) = O\left(\frac{q}{2^n}\right).$$

For any classical adversary \mathcal{A}' playing a q -query IND-CCA game against QuEME^E with q' offline queries to E , we have

$$\mathbf{Adv}_{\text{sprp}}^{\text{QuEME}^E}(\mathcal{A}') \leq O\left(\frac{q}{2^n}\right) + O\left(\frac{qq'}{2^{2n}}\right) + \alpha^E(q, q').$$

The proof of this relies on the conjectured Tight Mirror Theorem (Theorem 6.2). A simulation supporting the conjecture following an innovative approach can be found in Section 6.6.

Four Proofs. In this section, we present four classical security proofs for QuEME^π . Section 6.5.2 describes a proof of IND-CPA security up to n bits using the Mirror Theorem; Section 6.5.3 gives a proof of IND-CCA security up to n bits using the Tight Mirror Theorem; Section 6.5.4 shows a proof of IND-CPA security up to $2n/3$ bits without any mirror theoretic assumptions; and finally, Section 6.5.5 shows a proof of IND-CCA security up to $2n/3$ bits

without any mirror theoretic assumptions. In each of the proofs we first describe the sampler for the internal transcripts and its bad events, then provide a bound for the bad events, and finally bound the ratio of good probabilities to apply the Coefficient-H Technique and derive the desired security bound.

6.5.2 IND-CPA Security Proof of n -bit Security using Mirror Theory

In this subsection we prove the following result.

Theorem 6.7. *For any classical adversary \mathcal{A} playing a q -query IND-CPA game against QuEME^π , under the assumption that the Mirror Theorem (Section 6.2.1) holds, we have*

$$\text{Adv}_{\text{prp}}^{\text{QuEME}^\pi}(\mathcal{A}) = O\left(\frac{q}{2^n}\right).$$

For any classical adversary \mathcal{A}' playing a q -query IND-CPA game against QuEME^E with q' offline queries to E , we have

$$\text{Adv}_{\text{prp}}^{\text{QuEME}^E}(\mathcal{A}') \leq O\left(\frac{q}{2^n}\right) + O\left(\frac{qq'}{2^{2n}}\right) + \alpha^E(q, q').$$

Sampler of Internal Transcripts. Here we'll define the sampler \mathcal{S} which takes $\tau = \{(L^i, R^i), (S^i, T^i) \mid i \in [1..q]\}$ as input and samples a $\tau^* = \{(\widehat{L}^i, \widehat{R}^i, \widehat{S}^i, \widehat{T}^i) \mid i \in [1..q]\}$. The sampling proceeds as follows:

1. \mathcal{S} first samples two independent n -bit random permutations π_1^* and π_2^* . Then it sets $\widehat{L}^i \leftarrow \pi_1^*(L^i)$, $\widehat{R}^i \leftarrow \pi_2^*(R^i)$, and $X^i \leftarrow \widehat{L}^i \oplus \widehat{R}^i$ for each $i \in [1..q]$.
2. Let Γ be the set of all $2q$ -sequences $(\widehat{S}^1, \widehat{T}^1, \dots, \widehat{S}^q, \widehat{T}^q)$ satisfying the following conditions:
 - $(\forall i, i' \in [1..q]) \widehat{S}^i = \widehat{S}^{i'} \iff S^i = S^{i'}$;
 - $(\forall i, i' \in [1..q]) \widehat{T}^i = \widehat{T}^{i'} \iff T^i = T^{i'}$;
 - $(\forall i \in [1..q]) \widehat{S}^i \oplus \widehat{T}^i = \widehat{L}^i \oplus \widehat{R}^i$.

Then \mathcal{S} samples $(\widehat{S}^1, \widehat{T}^1, \dots, \widehat{S}^q, \widehat{T}^q)$ uniformly at random from the set Γ .

Bad Events. We define the following bad events on the random coins of f and \mathcal{S} :

bad_0 : For some distinct $i, i' \in [1..q]$, $(S^i, T^i) = (S^{i'}, T^{i'})$;

bad_1 : For some distinct $i, i' \in [1..q]$

$$X^i = X^{i'} \wedge (S^i = S^{i'} \vee T^i = T^{i'}).$$

bad₂: For a path P of even length ≥ 4 in H ,

$$\bigoplus_{i \in P} X^i = 0;$$

bad₃: There is a cycle in H .

In bad₁, bad₂ and bad₃ we assume bad₀ has not happened.

6.5.2.1 Bad Probabilities.

We recall that the bad events (and hence bad probabilities) are only defined in the ideal world. bad₀ involves a random collision over $2n$ bits, with a choice of the two indices i and i' . Thus,

$$\Pr_0[\text{bad}_0] \leq \frac{\binom{q}{2}}{N^2} \leq \frac{q^2}{N^2}. \quad (6.1)$$

We now bound the probability of bad₁. Let's fix a pair of indices $i, i' \neq i$. Because we are in the ideal case, $S^i, S^{i'}, T^i, T^{i'}, X^i, X^{i'}$ are uniformly random strings in $[N]$. Therefore,

$$\Pr_0[\text{bad}_1] \leq \frac{q(q-1) \cdot 2}{N^2} \leq \frac{2q^2}{N^2}. \quad (6.2)$$

For a path P of even length $2m \geq 4$, we can argue similarly that the $2m$ values $X^i, i \in P$ are all mutually independent, and sum to 0 with a probability of $1/N$. The event involves $2m - 1$ further collisions (for forming the path P), choice of $2m$ indices, and a choice whether the path begins from an S -node or a T -node. Therefore,

$$\begin{aligned} \Pr_0[\text{bad}_2] &\leq \sum_{m \geq 2} \frac{q(q-1) \dots (q-2m+1) \cdot 2}{N^{2m}} \\ &\leq \sum_{m \geq 2} \frac{2q^{2m}}{N^{2m}} = \frac{2q^2}{N^2} \sum_{m \geq 2} \left(\frac{q^2}{N^2}\right)^{m-1} \leq \frac{2q^2}{N^2} \sum_{m \geq 2} \left(\frac{1}{2}\right)^{m-1} \leq \frac{2q^2}{N^2}. \end{aligned} \quad (6.3)$$

Finally, a cycle of length $2m$ (with $m \geq 2$, since a cycle of length 2 would imply bad₀) will need $2m$ collisions for the cycle and give a choice of $2m$ indices and a choice of whether the first node is an S or a T ; since the choice of this 'first node' is arbitrary, we divide the total count by m . This gives

$$\begin{aligned} \Pr_0[\text{bad}_3] &\leq \sum_{m \geq 2} \frac{q(q-1) \dots (q-2m+1) \cdot 2}{N^{2m} \cdot m} \\ &\leq \sum_{m \geq 2} \frac{2q^{2m}}{mN^{2m}} = \frac{q^2}{N^2} \sum_{m \geq 2} \left(\frac{q^2}{N^2}\right)^{m-1} \leq \frac{q^2}{N^2} \sum_{m \geq 2} \left(\frac{1}{2}\right)^{m-1} \leq \frac{q^2}{N^2}. \end{aligned} \quad (6.4)$$

From equations 6.1-6.4 it follows that the probability that at least one of the bad events occurs is upper-bounded by

$$\epsilon_1 = \frac{6q^2}{N^2}. \quad (6.5)$$

6.5.2.2 Ratio of Good Probabilities.

Next we turn to the second step of using the Coefficient H Technique: lower-bounding the ratio of good probabilities. Suppose (τ, τ^*) is a good transcript. Recall that q_1, q_2, q_3, q_4 are the number of distinct values respectively of L^i, R^i, S^i, T^i in τ . Further suppose that in τ^* , there are r distinct values of X^i , with the number of queries they appear in being t_1, \dots, t_r , where $t_1 + \dots + t_r = q$.

Real World. In the real world, for each $j \in [1..4]$, the probability that π_j is compatible with (τ, τ^*) is $1/(N)_{q_j}$, and the probability that $\tilde{\pi}$ is compatible with (τ, τ^*) is $1/[(N)_{t_1} \dots (N)_{t_r}]$. Thus,

$$\Pr_1[(\tau, \tau^*)] = \frac{1}{(N)_{q_1} \dots (N)_{q_4} (N)_{t_1} \dots (N)_{t_r}}. \quad (6.6)$$

Ideal World. In the ideal world, the probability that f is compatible with τ is $1/(N^2)^q$. The probabilities that π_1^* and π_2^* are compatible with τ^* are respectively $1/(N)_{q_1}$ and $1/(N)_{q_2}$. For the second step, the probability that the $(\widehat{S}^1, \widehat{T}^1, \dots, \widehat{S}^q, \widehat{T}^q)$ comes from Γ is $1/|\Gamma|$. Thus,

$$\Pr_0[(\tau, \tau^*)] = \frac{1}{(N^2)^q (N)_{q_1} (N)_{q_2} |\Gamma|}. \quad (6.7)$$

equations 6.6 and 6.7 give the ratio

$$\rho := \frac{\Pr_1[(\tau, \tau^*)]}{\Pr_0[(\tau, \tau^*)]} = \frac{(N^2)^q |\Gamma|}{(N)_{q_3} (N)_{q_4} (N)_{t_1} \dots (N)_{t_r}}. \quad (6.8)$$

Since $(N)_{t_j} < N^{t_j}$, and $t_1 + \dots + t_r = q$, we have $(N)_{t_1} \dots (N)_{t_r} < N^q$. Plugging this in equation. 6.8 gives

$$\rho \geq \frac{N^q |\Gamma|}{(N)_{q_3} (N)_{q_4}}. \quad (6.9)$$

Using Mirror Theory as a Black-box. We recall that in the ideal world, having chosen all \widehat{L}^i and \widehat{R}^i , we need to choose \widehat{S}^i and \widehat{T}^i such that, for each $i, j \in [1..q]$:

- $\widehat{S}^i = \widehat{S}^j \iff S^i = S^j$;

- $\widehat{T}^i = \widehat{T}^j \iff T^i = T^j$;

and for each $i \in [1..q]$, $\widehat{S}^i \oplus \widehat{T}^i = X^i$. Then we can formulate our problem as one of Mirror Theory as follows: we need to find $\widehat{S}_1, \dots, \widehat{S}_{q_3}$, all distinct, and $\widehat{T}_1, \dots, \widehat{T}_{q_4}$, all distinct, satisfying q bi-variate equations of the form $\widehat{S}_i + \widehat{T}_j = \delta_{ij}$. Since this is a good transcript, we know that none of the bad events happened, making this system of equations cycle-free and consistent. Then from the Mirror Theorem (Theorem 6.1) we see that

$$|\Gamma| \geq \frac{(N)_{q_3} (N)_{q_4}}{N^q} \cdot (1 - \epsilon_2), \quad (6.10)$$

where $\epsilon_2 = O(q/N)$.

Putting the bound of equation 6.10 in equation 6.9 gives the desired bound

$$\rho \geq 1 - \epsilon_2, \quad (6.11)$$

thus completing the proof.

6.5.3 IND-CCA Security Proof of n -bit Security using Mirror Theory

In this subsection we prove Theorem 6.6. For simplicity we assume there are no cycle queries (no cycle in G or H); the case when there are cycle queries can be similarly handled. Let $\widehat{L}_1, \dots, \widehat{L}_{q_1}, \widehat{R}_1, \dots, \widehat{R}_{q_2}, \widehat{S}_1, \dots, \widehat{S}_{q_3}$ and $\widehat{T}_1, \dots, \widehat{T}_{q_4}$ be the distinct values we need to choose for each permutation (\widehat{L}_i correspond to $E_1(L_i)$, \widehat{R}_i to $E_2(R_i)$, \widehat{S} to $E_3^{-1}(S_i)$ and \widehat{T}_i to $E_4^{-1}(T_i)$). Let H_τ be the graph analogous to G_τ , but for inputs, i.e., two vertices i and i' are adjacent in H_τ if $L^i = L^{i'}$ or $R^i = R^{i'}$; the edge (i, i') is blue for the first condition and red for the second. We change the sampling mechanism in the ideal world as follows:

1. \mathcal{S} first samples a X^1, \dots, X^q such that on any (non-empty) path P of even length in G or H ,

$$\bigoplus_{i \in P} X^i \neq 0.$$

Let Λ denote the set of all (X^1, \dots, X^q) satisfying this condition.

2. Next, \mathcal{S} samples $\widehat{L}_1, \dots, \widehat{L}_{q_1}$, all distinct, $\widehat{R}_1, \dots, \widehat{R}_{q_2}$, all distinct, subject to q bi-variate equations of the form $\widehat{L}_i \oplus \widehat{R}_j = \delta_{ij}^G$. Let Γ^G denote the set of all solutions to this system.
3. Finally \mathcal{S} samples $\widehat{S}_1, \dots, \widehat{S}_{q_3}$, all distinct, and $\widehat{T}_1, \dots, \widehat{T}_{q_4}$, all distinct, subject to q bi-variate equations of the form $\widehat{S}_i \oplus \widehat{T}_j = \delta_{ij}^H$. Let Γ^H denote the set of all solutions to this system.

Since we have assumed there are no cycles in G and H , we have $q_1 + q_2 - \alpha = q_3 + q_4 - \beta = q$.

Then the Tight Mirror Theorem (Theorem 6.2) tells us that

$$|\Gamma^G| \geq \prod_{j=1}^{\alpha} \left[\binom{N - Q_1^{(j)}}{q_1^{(j)}} \binom{N - Q_2^{(j)}}{q_2^{(j)}} \right] \cdot \frac{1}{N^q} \cdot (1 - \epsilon_2), \quad (6.12)$$

$$|\Gamma^H| \geq \prod_{j=1}^{\beta} \left[\binom{N - Q_3^{(j)}}{q_3^{(j)}} \binom{N - Q_4^{(j)}}{q_4^{(j)}} \right] \cdot \frac{1}{N^q} \cdot (1 - \epsilon_3), \quad (6.13)$$

where ϵ_1 and ϵ_2 are both $O(q/N)$.

Similarly, we can show that

$$|\Lambda| \geq N^q \left[\prod_{j=1}^{\alpha} \frac{\binom{N}{q_1^{(j)}} \binom{N}{q_2^{(j)}}}{N^{q_1^{(j)} + q_2^{(j)}} \right] \left[\prod_{j=1}^{\beta} \frac{\binom{N}{q_3^{(j)}} \binom{N}{q_4^{(j)}}}{N^{q_3^{(j)} + q_4^{(j)}} \right] (1 - \epsilon_3). \quad (6.14)$$

We observe that

$$\begin{aligned} \binom{N - Q_1^{(j)}}{q_1^{(j)}} \binom{N}{q_1^{(j)}} &= \prod_{k=0}^{q_1^{(j)} - 1} (N - Q_1^{(j)} - k) \\ &\geq \prod_{k=0}^{q_1^{(j)} - 1} (N - Q_1^{(j)} - k) N \\ &= \binom{N - Q_1^{(j)}}{q_1^{(j)}} N^{q_1^{(j)}}, \end{aligned} \quad (6.15)$$

so that

$$\prod_{j=1}^{\alpha} \binom{N - Q_1^{(j)}}{q_1^{(j)}} \binom{N}{q_1^{(j)}} \geq \binom{N}{q_1} N^{q_1}. \quad (6.16)$$

We can show a similar inequality for q_2 , q_3 , and q_4 .

Thus,

$$\frac{|\Gamma^G| |\Gamma^H| |\Lambda|}{\binom{N}{q_1} \binom{N}{q_2} \binom{N}{q_3} \binom{N}{q_4}} \geq \frac{1}{N^q} (1 - \epsilon_2 - \epsilon_3 - \epsilon_4). \quad (6.17)$$

Since

$$\Pr_0[(\tau, \tau^*)] = \frac{1}{(N^2)^q |\Gamma^G| |\Gamma^H| |\Lambda|}, \quad (6.18)$$

we have

$$\rho \geq 1 - \epsilon_2 - \epsilon_3 - \epsilon_4, \quad (6.19)$$

which completes the proof.

6.5.4 IND-CPA Security Proof of $(2n/3)$ -bit Security

In this subsection we prove the following result.

Theorem 6.8. *For any classical adversary \mathcal{A} playing a q -query IND-CPA game against QuEME^π , we have*

$$\text{Adv}_{\text{prp}}^{\text{QuEME}^\pi}(\mathcal{A}) = O\left(\frac{q^3}{2^{2n}}\right).$$

For any classical adversary \mathcal{A}' playing a q -query IND-CPA game against QuEME^E with q' offline queries to E , we have

$$\text{Adv}_{\text{prp}}^{\text{QuEME}^E}(\mathcal{A}') \leq O\left(\frac{q^3}{2^{2n}}\right) + O\left(\frac{qq'}{2^{2n}}\right) + \alpha^E(q, q').$$

For this proof we change the second step of the sampling procedure of Section 6.5.2. We observe that for each $j \in [1..\ell]$, once a value is assigned to \widehat{S}^{i_j} , it induces a value on $\widehat{S}^i, \widehat{T}^i$ for each $i \in C_j$, according to the following rules:

- if \widehat{S}^i is already set, $\widehat{T}^i \leftarrow \widehat{L}^i \oplus \widehat{R}^i \oplus \widehat{S}^i$;
- if \widehat{T}^i is already set, $\widehat{S}^i \leftarrow \widehat{L}^i \oplus \widehat{R}^i \oplus \widehat{T}^i$;
- if $S^i = S^{i'}$ and $\widehat{S}^{i'}$ is already set, $\widehat{S}^i \leftarrow \widehat{S}^{i'}$;
- if $T^i = T^{i'}$ and $\widehat{T}^{i'}$ is already set, $\widehat{T}^i \leftarrow \widehat{T}^{i'}$.

Since C_j is connected, all nodes in C_j are guaranteed to be covered in this manner. We refer to this as *extending \widehat{S}^{i_j} to all of C_j* .

\mathcal{S} uses two (initialised as empty) sets $D_{\widehat{S}}$ and $D_{\widehat{T}}$, to tabulate the sampled values of \widehat{S}^i and \widehat{T}^i respectively; every time a value is assigned to some \widehat{S}^i or \widehat{T}^i , it is added to the corresponding table. For a subgraph C of G_τ (which can be seen as a subset of the queries), $D_{\widehat{S}|C}$ and $D_{\widehat{T}|C}$ will respectively denote $D_{\widehat{S}}$ and $D_{\widehat{T}}$ restricted to C . The connected components will be considered listed by decreasing size.

Step 1 is the same as in Section 6.5.2. In Step 2 \mathcal{S} samples \widehat{S}^1 uniformly at random from $\{0, 1\}^n$ and extends \widehat{S}^1 to all of C^1 , and for each $j \in [2..\ell]$ \mathcal{S} samples over C_j as follows:

1. it computes the *banned set* B_j defined with the following condition: if and only if \widehat{S}^{i_j} is assigned a value in B_j and extended to all of C_j , at least one of the sets $D_{\widehat{S}|C_j} \cap D_{\widehat{S}}$ and $D_{\widehat{T}|C_j} \cap D_{\widehat{T}}$ will be non-empty;²

²To avoid more complicated notation we assume here that the newly assigned values are kept in $D_{\widehat{S}|C_j}$ and $D_{\widehat{T}|C_j}$ but not yet added to $D_{\widehat{S}}$ and $D_{\widehat{T}}$.

2. it samples \widehat{S}^{i_j} from $\{0, 1\}^n \setminus B_j$ and extends it to all of C_j .³

Once \mathcal{S} has completed Step 3 for C_ℓ , the sampling of τ^* is complete. We also include the additional bad event defined as follows:

bad_4 : There are is a path of length ≥ 3 in G_τ .

This happens with $O(q^3/N^2)$ probability. When bad_5 has not happened, G_τ has no component of size ≥ 2 . Thus, for each $j \in [2..\ell]$, $|B_j| \leq 2(j-1)$.

Good probability in ideal world. For each $j \in [1..\ell]$, the probability that the random sampling in the modified second step of \mathcal{S} correctly outputs the \widehat{S}^{i_j} is $1/(N - |B_j|)$. (We take B_1 to be the empty set.) Thus,

$$\Pr_0[(\tau, \tau^*)] = \frac{1}{(N^2)^q (N)_{q_1} (N)_{q_2} N(N - |B_2|) \dots (N - |B_\ell|)}. \quad (6.20)$$

equations 6.6 and 6.20 give the ratio

$$\rho = \frac{(N^2)^q N(N - |B_2|) \dots (N - |B_\ell|)}{(N)_{q_3} (N)_{q_4} (N)_{t_1} \dots (N)_{t_r}}. \quad (6.21)$$

Like in the derivation of equation 6.9, we plug in $(N)_{t_1} \dots (N)_{t_r} < N^q$ in equation 6.21 to get

$$\rho \geq \frac{N^q N(N - |B_2|) \dots (N - |B_\ell|)}{(N)_{q_3} (N)_{q_4}}. \quad (6.22)$$

³It is easy to see that $D_{\widehat{S}} \subseteq B_j$.

Since $q \geq \max(q_3, q_4)$, we have $\ell \leq \min(q_3, q_4)$. Thus, we can rewrite equation 6.22 as

$$\begin{aligned}
\rho &\geq \frac{N^q N(N-2) \dots (N-(2\ell-2))}{(N)_{q_3} (N)_{q_4}} \\
&\geq \frac{N^{q-\ell}}{(N-\ell)_{q_3-\ell} (N-\ell)_{q_4-\ell}} \cdot \frac{N^\ell N(N-2) \dots (N-(2\ell-2))}{(N)_\ell (N)_\ell} \\
&\geq \prod_{j=1}^{\ell} \frac{N(N-2(j-1))}{(N-(j-1))^2} \\
&= \prod_{j=1}^{\ell} \frac{N^2 - 2(N)(j-1)}{N^2 - 2(N)(j-1) + (j-1)^2} \\
&= \prod_{j=1}^{\ell} \left(1 - \frac{(j-1)^2}{N^2 - 2(N)(j-1) + (j-1)^2} \right) \\
&\geq \prod_{j=1}^{\ell} \left(1 - \frac{2(j-1)^2}{N^2} \right) \\
&\geq 1 - \sum_{j=1}^{\ell} \frac{2(j-1)^2}{N^2} \geq 1 - \frac{\ell^3}{N^2}, \tag{6.23}
\end{aligned}$$

which completes the proof with $\epsilon_2 = \ell^3/N^2$.

6.5.5 IND-CCA Security Proof of $(2n/3)$ -bit Security

In this subsection we instead prove the following weaker result without relying on any conjectured bound.

Theorem 6.9. *For any classical adversary \mathcal{A} playing a q -query IND-CCA game against QuEME^π , we have*

$$\text{Adv}_{\text{sprp}}^{\text{QuEME}^\pi}(\mathcal{A}) = O\left(\frac{q^3}{2^{2n}}\right).$$

For any classical adversary \mathcal{A}' playing a q -query IND-CCA game against QuEME^E with q' offline queries to E , we have

$$\text{Adv}_{\text{sprp}}^{\text{QuEME}^E}(\mathcal{A}') \leq O\left(\frac{q^3}{2^{2n}}\right) + O\left(\frac{qq'}{2^{2n}}\right) + \alpha^E(q, q').$$

When decryption queries are allowed, we partition the queries into two sets: let \mathcal{I}^* contain the queries where both output blocks are *fresh*, and \mathcal{I} contain the queries where one of the output blocks collides with an earlier block at the same position.

Sampler of Internal Transcripts. Here we'll define the sampler \mathcal{S} which takes $\tau = \{(L^i, R^i), (S^i, T^i) \mid i \in [1..q]\}$ with the direction of each query (encryption or decryption) as input and samples a $\tau^* = \{(\widehat{L}^i, \widehat{R}^i, \widehat{S}^i, \widehat{T}^i) \mid i \in [1..q]\}$. The sampling proceeds as follows:

1. \mathcal{S} initialises four tables $D_{\widehat{L}}$, $D_{\widehat{R}}$, $D_{\widehat{S}}$, and $D_{\widehat{T}}$ as empty.
2. For an encryption query i , \mathcal{S} first checks the tables $D_{\widehat{L}}$ and $D_{\widehat{R}}$ to see if \widehat{L}^i or \widehat{R}^i has already been sampled; whichever is not found in the table is freshly sampled from the set of unsampled values. X^i is set to be $\widehat{L}^i \oplus \widehat{R}^i$.
 - a) If $i \in \mathcal{I}$, one of \widehat{S}^i and \widehat{T}^i is already sampled, so the other one is set as the sum of the sampled one and X^i .
 - b) If $i \in \mathcal{I}^*$, both \widehat{S}^i and \widehat{T}^i are fresh, so \widehat{S}^i is sampled from outside $D_{\widehat{S}}$, such that $\widehat{T}^i = \widehat{S}^i \oplus X^i$ is also outside $D_{\widehat{T}}$.
3. For a decryption query i , \mathcal{S} first checks the tables $D_{\widehat{S}}$ and $D_{\widehat{T}}$ to see if \widehat{S}^i or \widehat{T}^i has already been sampled; whichever is not found in the table is freshly sampled from the set of unsampled values. X^i is set to be $\widehat{S}^i \oplus \widehat{T}^i$.
 - a) If $i \in \mathcal{I}$, one of \widehat{L}^i and \widehat{R}^i is already sampled, so the other one is set as the sum of the sampled one and X^i .
 - b) If $i \in \mathcal{I}$, both \widehat{L}^i and \widehat{R}^i are fresh, so \widehat{L}^i is sampled from outside $D_{\widehat{L}}$, such that $\widehat{R}^i = \widehat{L}^i \oplus X^i$ is also outside $D_{\widehat{R}}$.

Bad Events. We define the following bad events on the random coins of f and \mathcal{S} :

- bad₀:** For some $i, i', i'' \in [1..q]$ with $i > i'$ and $i > i''$, $S^i = S^{i'}$ and $T^i = T^{i''}$;
- bad₁:** In an encryption query $i \in \mathcal{I}$, a previously unsampled \widehat{S}^i or \widehat{T}^i is set to be equal to a previously sampled value at the same position;
- bad₂:** In a decryption query $i \in \mathcal{I}$, a previously unsampled \widehat{L}^i or \widehat{R}^i is set to be equal to a previously sampled value at the same position.

In **bad₁** and **bad₂** we assume **bad₀** has not happened.

Bad Probabilities. For **bad₀** the two collisions have a joint probability of $1/N^2$, with choice of the three indices i, i', i'' . Thus,

$$\Pr[\text{bad}_0] \leq \frac{q^3}{N^2}. \quad (6.24)$$

For bad_1 we need one collision with a previous i' for $i \in \mathcal{I}$, and one collision with a previously sampled value at some i'' . Again they have a joint probability of $1/N^2$, with choice of the three indices i, i', i'' . Thus,

$$\Pr[\text{bad}_1] \leq \frac{q^3}{N^2}. \quad (6.25)$$

Similarly we can show that

$$\Pr[\text{bad}_2] \leq \frac{q^3}{N^2}. \quad (6.26)$$

Ratio of Good Probabilities. As before, in the real world, we have

$$\Pr_1[(\tau, \tau^*)] = \frac{1}{(N)_{q_1} \dots (N)_{q_4} (N)_{t_1} \dots (N)_{t_r}}. \quad (6.27)$$

In the ideal world, we have a term $1/N^{2q}$ that comes from the sampling of the outputs in the online phase. As before, we use one N^q to cancel out $(N)_{t_1} \dots (N)_{t_r}$. If the j -th unique \widehat{L} or \widehat{R} first appears in an encryption query, it will contribute a term $1/(N-j-1)$ to the probability, and the resulting $(N-j-1)$ in the numerator will cancel out the same term in $(N)_{q_1}$ or $(N)_{q_2}$. Similarly, if the j -th unique \widehat{S} or \widehat{T} first appears in an encryption query, it will contribute a term $1/(N-j+1)$ to the probability, and the resulting $(N-j+1)$ in the numerator will cancel out the same term in $(N)_{q_3}$ or $(N)_{q_4}$. For each encryption query in \mathcal{I} , no sampling is done for \widehat{S} or \widehat{T} , and for each decryption query in \mathcal{I} , no sampling is done for \widehat{L} or \widehat{R} , so these do not contribute anything to the probability.

Finally, consider an encryption query in \mathcal{I}^* that contains the j -th unique \widehat{S} and j' -th unique \widehat{T} . Thus, when sampling \widehat{S} we need to avoid $j+j'-2$ values, so this contributes a term $1/(N-j-j'+2)$ to the probability. We combine this $(N-j-j'+2)$ in the numerator with one N -term in the numerator and the terms $(N-j+1)$ and $(N-j'+1)$ in the denominator (from $(N)_{q_3}$ or $(N)_{q_4}$ respectively), to get

$$\frac{N(N-j-j'+2)}{(N-j+1)(N-j'+1)} = 1 - \frac{(j-1)(j'-1)}{(N-j+1)(N-j'+1)} \geq 1 - \frac{2jj'}{N^2}, \quad (6.28)$$

where we use the inequalities $j, j' \leq N(1-1/\sqrt{2})$. This uses up $N^{|\mathcal{I}^*|}$, and the remaining $N^{|\mathcal{I}|}$ is used to cancel out the remaining terms in the denominator.

Thus we have

$$\rho \geq \prod_{j, j'} \left(1 - \frac{2jj'}{N^2}\right) \geq \prod_{j \in [1..q]} \left(1 - \frac{2j^2}{N^2}\right) \geq 1 - \frac{q^3}{N^2}, \quad (6.29)$$

since we can replace the smaller of j and j' with the bigger one without breaking the inequality. This completes the proof with $\epsilon_2 = q^3/N^2$.

6.6 Simulation of the Mirror Theory

In the proof of Theorem 6.6 we use a conjectured result from a line of research that focuses on finding tight approximations on the number of solutions to systems of bi-variate equations, which goes by the name Mirror Theory [97] (see Section 6.2.1). The particular conjecture we use can be summarised as follows: Consider two sequences of n -bit variables Y_1, \dots, Y_{q_1} and Z_1, \dots, Z_{q_2} , with $q_1 < N, q_2 < N$, where N denote 2^n . For some $q < q_1 + q_2$ suppose there are q bi-variate equations of the form

$$Y_i \oplus Z_j = \delta_{ij}.$$

Consider the graph where this equation is represented by an edge between Y_i and Z_j . We assume none of the equations is redundant and the system of equations is consistent, so there are no cycles in this graph. Let $C^{(1)}, \dots, C^{(t)}$ be the connected components, where $t = q_1 + q_2 - q$. For each $j \in [1..t]$ let $q_1^{(j)}$ (resp. $q_2^{(j)}$) be the number of Y_i 's (resp. Z_i 's) that appear in $C^{(j)}$. Finally, define the cumulative sums

$$Q_b^{(j)} = \sum_{i=1}^{j-1} q_b^{(i)}$$

for each $b \in \{1, 2\}$ and each $j \in [1..t]$. Then the conjectured Tight Mirror Theorem (Theorem 6.2) claims that the number of solutions to this system such that Y_i 's are all distinct and Z_i 's are all distinct is at least

$$\frac{1}{N^q} \cdot \prod_{j=1}^t \left[\left(N - Q_1^{(j)} \right)^{q_1^{(j)}} \left(N - Q_2^{(j)} \right)^{q_2^{(j)}} \right] \cdot (1 - \epsilon),$$

where $\epsilon = O(q/N)$.

In order to verify this conjecture, we have implemented a simulation that allow us to predict the number of possible internal transcripts. In this section, we describe these simulations that are, to the best of our knowledge, an innovative approach that hasn't been used before in this context.

First Step: Getting the Sets of Connected Components. The first step for comparing experiments to the conjecture is computing the set of the connected components of two or more variables. We recall that there is an edge between the variables X_i and Y_j if and only if there is the equation $X_i \oplus Y_j = \delta_{i,j}$. First we make a list of equations sorted by index i and one sorted by index j for retrieving the edges quickly, then we apply a classic Breadth-First Traversal of the graph. The procedure to get it is described in Algorithm 6.2 below:

Algorithm 6.2 Retrieving the sets on connected components

Input: List of equations of the form $X_i \oplus Y_j = \delta_{i,j}$

Output: The list of connected components

- 1: Sort the equations $X_i \oplus Y_j = \delta_{i,j}$ by i the result is named L_X
 - ▷ There needs to have a place to mark the different i .
 - 2: Sort the equations $X_i \oplus Y_j = \delta_{i,j}$ by j the result is named L_Y
 - ▷ There needs to have a place to mark the different j .
 - 3: **for all** i **do**
 - 4: Start a pile with the element $(X, i, 0)$
 - 5: Start a list for recording the elements of the current connected component with the root element $(X, i, 0)$
 - 6: **while** the pile is not empty **do**
 - 7: Pop the first element (Z, l, Δ)
 - 8: **if** l is not marked in the list L_Z **then**
 - 9: **for all** $X_i \oplus Y_j = \delta_{i,j}$ in L_Z with $i = l$ if $X = Z$ and $j = l$ otherwise **do**
 - 10: Add $(Y, j, \Delta \oplus \delta_{i,j})$ to the pile and to the component if $X = Z$ and $(X, i, \Delta \oplus \delta_{i,j})$ otherwise
 - 11: Mark l in the list L_Z
 - 12: Register the list if it contains more than one element
 - ▷ This connected component has either no elements or is a isolated point.
 - 13: Return the lists of connected components
-

Naive Approach. The most natural way of doing once obtained the connected components is to generate the solutions. This can be done by initializing two sets of remaining places to $\{0, 1\}^n$ (S_X for the variables X and S_Y for the variables Y). We take the largest component, find a place α for its root and for every points (Z, l, Δ) of this component ruling out $\alpha \oplus \Delta$ from the the corresponding variable (S_X if $Z = X$ and S_Y otherwise). We take the second largest, find a place for the second root β such that for every point (Z, l, Δ) of this second component $\beta \oplus \Delta$ is in the set of remaining places of corresponding variable and once a β is found we rule them out. We continue for the remaining components until there is either no more component (making a solution) or there is no viable place for a root (the earlier choices did not lead to a solution). While this method is not practical as it generates every solution which is double exponential on the size of the input, it is the only one (to our knowledge) to give the exact number of solutions.

Approximation. In this paragraph, we describe a method that allows to propose an approximation of the number of solutions of the system given by the equations $X_i \oplus Y_j = \delta_{i,j}$. For a given set of equations we want to obtain the connected components (C_i) as described before and m their number and their

size. We note $\Delta_{i,j}$ the set of places of the roots of the connected components such that the values taken by C_i and C_j collide. Then by the formula of the cardinal of a union, we get

$$2^{nm} - |\text{solutions}| = \sum_{k=1}^m (-1)^{k-1} \sum_{\{i_1, j_1\} > \dots > \{i_k, j_k\}} |\Delta_{i_1, j_1} \cap \dots \cap \Delta_{i_k, j_k}|$$

where $>$ is an order.

A first observation is that for all $\{i_1, j_1\} \neq \{i_2, j_2\}$,

$$|\Delta_{i_1, j_1} \cap \Delta_{i_2, j_2}| \times 2^{nm} = |\Delta_{i_1, j_1}| \times |\Delta_{i_2, j_2}|$$

as the difference between the value of the roots of C_{i_1} and C_{j_1} and between C_{i_2} and C_{j_2} are independent even for cases $\{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset$. For further use, for a set $\{i_1, j_1\} > \dots > \{i_k, j_k\}$, we define the graph $G_{\{i_1, j_1\} > \dots > \{i_k, j_k\}}$ with vertices $1, \dots, m$ and there is an edge between vertices a and b if and only if there exist l such that $\{i_l, j_l\} = \{a, b\}$. For a graph G on vertices $1, \dots, m$, we define

$$S_G = \frac{1}{2^{nm}} \sum_{G' \cong G} \left| \bigcap_{i, j \in G'} \Delta_{i, j} \right|,$$

where \cong denotes graph isomorphism. We let C_G be the number of connected components of G and P_G the number of non-isolated points.

This first observation extends to the computation of any S_G where G has no cycle and to unions of not connected sub-graphs. S_G can be bounded by $(\sum_i |C_i|^2)^{P_G} / 2^{n(m-C_G)}$. This means that this method of approximation is suited for cases where the value $\sum_i |C_i|^2$ is controlled. For random systems, $\sum_i |C_i|^2 = O(q)$. Then a first approximation can be made by taking

$$\frac{|\text{solutions}|}{2^{nm}} = \prod_{i, j} \left(1 - \frac{|\Delta_{i, j}|}{2^n} \right) + O\left(\frac{q^3}{2^{2n}} \right).$$

More advanced approximations can be made by considering cycles of successively bigger sizes. (For example, by considering the cycles of size 3, we get a better approximation with an error in $O(q^4/2^{3n})$.)

The figure 6.3 shows the different simulations of the different approximations. It took 10 hours on an Intel i5-6500U CPU to compute. The approximations tend to get more solutions than the conjecture, but remaining in the expected error.

Conclusion. Overall our simulations confirm that the tighter version of the mirror theory holds for small bit sizes ($n \leq 5$ for the exact approach Figure 6.3a, $n \leq 8$ for the better approximation Figure 6.3b, and $n \leq 11$ for the first approximation Figure 6.3c). While it becomes infeasible to run the simulations for usable values of n , since we did not use any special properties of small n -values, our results seem to suggest that it should also hold for larger values of n , making our conjecture a reasonable one.

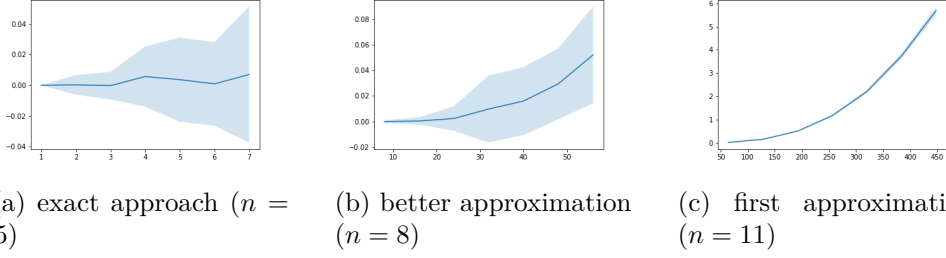


Figure 6.3: Results of simulations: logarithmic difference between simulation and conjectural prediction by number of equations (line is mean over 100 tries, area is variability)

6.7 An $O(2^n)$ -Distinguisher on our Scheme

We have black box access to a function f st. $f \stackrel{\$}{\leftarrow} P_{2n}$ or $f \leftarrow \text{QuEME}^\pi$, and we want to distinguish in which case we are. We present a distinguisher that performs only forward queries to f . Assume on query $(x_1||x_2)$ we get output $(y_1||y_2)$. In the first case, $(y_1||y_2)$ is a uniformly random string that has not been outputted before for any $(x_1||x_2)$ that has not been queried before. In the second case, there exists permutations $\pi_1, \pi_2, \pi_3, \pi_4$ such

$$\pi_1(x_1) \oplus \pi_2(x_2) = \pi_3^{-1}(y_1) \oplus \pi_4^{-1}(y_2). \quad (6.30)$$

Our proof will use linear algebra techniques. Let $N = 4 \cdot 2^n$. Let $x_1, x_2, y_1, y_2 \in \{0, 1\}^n$ that we interpret as integers in $[0, 2^n - 1]$, and let $\vec{e}_{x_1 x_2 y_1 y_2} \in \mathbb{F}_2^N$ be the binary column vector where the i^{th} coordinate of $\vec{e}_{x_1 x_2 y_1 y_2}$ is equal to 1 iff. $i = x_1$, $i = x_2 + 2^n$, $i = y_1 + 2 \cdot 2^n$ or $i = y_2 + 3 \cdot 2^n$ and is equal to 0 otherwise. This means each $\vec{e}_{x_1 x_2 y_1 y_2} \in \mathbb{F}_2^N$ has weight 4, meaning four non-zero coordinates.

The idea of the distinguisher is that if the following: perform M queries $\{x_1^i x_2^i || y_1^i y_2^i\}_{i \in [M]}$, let $H = \text{span}\{\vec{e}_{x_1^i x_2^i y_1^i y_2^i}\}_{i \in [M]}$. For M large enough (but linear in N), we will show that if we queried our QuEME^π construction, we have $\dim(H) \leq N - 2$ with overwhelming probability, which we prove using equation 6.30. On the other hand, if we start from a random permutation, then we can show that $\dim(H) \geq N - 1$ since the $\vec{e}_{x_1^i x_2^i y_1^i y_2^i}$ will essentially be random vectors of \mathbb{F}_2^N of weight 4.

We consider the following adversary

Adversary \mathfrak{a} for distinguishing the QuEME^π construction from a random permutation

- Perform $M = 4N$ random different queries $(x_1^i || x_2^i)$ for $i \in [M]$ and get respective outputs $(y_1^i || y_2^i)$.
- Let $H = \text{span}\{\vec{e}_{x_1^i x_2^i y_1^i y_2^i}\}_{i \in [M]}$. Compute $\dim(H)$.
- If $\dim(H) \leq N - 2$, return "EME" else return "random permutation".

Proposition 6.1. *If the adversary queries a function $f \leftarrow \text{QuEME}^\pi$, we have $\dim(H) \leq N - 2$ with overwhelming probability.*

Proof. Take $f \leftarrow \text{QuEME}^\pi$. This means in particular we choose random permutations $\pi_1, \pi_2, \pi_3, \pi_4$ and for each query x_1^i, x_2^i that gives output y_1^i, y_2^i , we have

$$\pi_1(x_1^i) \oplus \pi_2(x_2^i) = \pi_3^{-1}(y_1^i) \oplus \pi_4^{-1}(y_2^i).$$

Consider the following matrix $M \in F_2^{n \times 4 \cdot 2^n}$: the first 2^n columns of M are

the columns $\begin{pmatrix} [\pi_1(x)]_1 \\ \vdots \\ [\pi_1(x)]_n \end{pmatrix}$ for $x \in \{0, 1\}^n$. Then, the next 2^n columns are the same but we replace π_1 with π_2 , and similarly with the third and last where we have π_3^{-1} and π_4^{-1} respectively instead of π_1 . So we can write

$$M = \left(\begin{pmatrix} [\pi_1(0)]_1 \\ \vdots \\ [\pi_1(0)]_n \end{pmatrix} \cdots \begin{pmatrix} [\pi_2(0)]_1 \\ \vdots \\ [\pi_2(0)]_n \end{pmatrix} \cdots \begin{pmatrix} [\pi_3^{-1}(0)]_1 \\ \vdots \\ [\pi_3^{-1}(0)]_n \end{pmatrix} \cdots \begin{pmatrix} [\pi_4^{-1}(0)]_1 \\ \vdots \\ [\pi_4^{-1}(0)]_n \end{pmatrix} \cdots \right)$$

Because $\pi_1, \pi_2, \pi_3, \pi_4$ are permutations, the matrix M contains at least 2 different non-zero lines, therefore $\dim(M) \geq 2$. Also notice that

$$M \cdot \vec{e}_{x_1 x_2 y_1 y_2} = \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3^{-1}(y_1) \oplus \pi_4^{-1}(y_2).$$

so $H \subseteq \text{Ker}(M)$ and $\dim(\text{Ker}(M)) = N - \dim(M) \leq N - 2$ from which we conclude $\dim(H) \leq N - 2$. \square

Proposition 6.2. *If the adversary queries a random permutation $f \xleftarrow{\$} P_{2n}$, we have $\dim(H) = 4 \cdot 2^n - 1$ with overwhelming probability.*

Proof. Let $H_j = \text{span}\{\vec{e}_{x_1^i x_2^i y_1^i y_2^i}\}_{i \in [j]}$. We will show that if $\dim(H_j) \leq N - 2$ then with constant probability, $\dim(H_{j+1}) = \dim(H_j) + 1$. Let H_j^\perp be the dual of H_j , so

$$x \in H_j \Leftrightarrow \forall y \in H_j^\perp, \langle x, y \rangle = 0.$$

We have $\dim(H_j) + \dim(H_j^\perp) = N$ so in particular $\dim(H_j^\perp) \geq 2$. This means in particular that there exists two distinct non-zero vectors $z_1, z_2 \in H_j^\perp$. This again implies that there exists $z^* \in H_j^\perp$ st. $|z^*| \leq 2N/3$. One can indeed easily check that if $|z_1|, |z_2| > 2N/3$ then $|z_1 + z_2| \leq 2N/3$. For a random tuple x_1, x_2, y_1, y_2 , we have

$$\begin{aligned} \Pr[\vec{e}_{x_1 x_2 y_1 y_2} \in H_j] &\leq \Pr[\langle \vec{e}_{x_1 x_2 y_1 y_2}, z^* \rangle = 0] \\ &\leq \left(\frac{1}{3}\right)^4 + 6 \left(\frac{1}{3}\right)^2 \left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^4 = \frac{41}{81}. \end{aligned}$$

This gives $\Pr[\vec{e}_{x_1 x_2 y_1 y_2} \notin H_j] \geq 40/81$. In reality, the tuple $(x_1^{j+1} x_2^{j+1} y_1^{j+1} y_2^{j+1})$ is not entirely random. Indeed, while x_1^{j+1}, x_2^{j+1} are chosen uniformly at random, the outputs must satisfy the permutation constraints, meaning that if $(x_1^{j+1} || x_2^{j+1})$ has not been queried then the output $(y_1^{j+1} || y_2^{j+1})$ must be different from the previous outputs. For a fixed query, this changes the output distribution by at most $O(j/2^{2n}) = O(1/2^n)$ (since there are $O(N) = O(2^n)$ queries in total, so $j \leq O(2^n)$). From there, we get

$$\Pr[\vec{e}_{x_1^{j+1} x_2^{j+1} y_1^{j+1} y_2^{j+1}} \notin H_j] \geq \frac{40}{81} - O\left(\frac{1}{2^n}\right).$$

when the above holds, this immediately implies that $\dim(H_{j+1}) = \dim(H_j) + 1$. Since $M = 4N$, this then implies that with overwhelming probability $\dim(H_M) \geq N - 1$. \square

Quantising the distinguisher. So far we have not found any quantum versions improving the complexity, and we do not believe that this distinguisher can benefit of any speed-up in the quantum setting.

6.8 Quantum Security

In this section, we study the quantum security of our constructions. We will show that the QuEME^π has $n/6$ bits of quantum security. While this is not an as strong statement as in the classical setting, it implies at least that the security does not totally collapse when we consider quantum adversaries as is the case for the 3-round Luby-Rackoff construction [84].

To prove this statement, we will actually reduce the quantum security to classical security (in a non-tight way) using Zhandry's lower bound on small range functions. We then show how to prove our quantum security statement in this framework. Finally, we discuss our results and ways of improving these quantum security claims.

6.8.1 Hardness of Distinguishing a Random Permutation from a Random Function with Small Range

Our proof will use a quantum lower bound on distinguishing a random permutation from a random function with small range proven in [127]. We first define a distribution $S_n(r)$ of small range functions:

Definition 6.1. $S_n(r)$ is a distribution on functions from $\{0, 1\}^n$ to $\{0, 1\}^n$ sampled as follows:

- Draw a random function g from $\{0, 1\}^n \rightarrow [r]$.
- Draw a random injective function h from $[r] \rightarrow \{0, 1\}^n$.
- Output the composition $h \circ g$.

Notice that any function f drawn from $S_n(r)$ satisfies $|Im(f)| \leq r$. Let also P_n be the uniform distribution on permutations on $\{0, 1\}^n$. Zhandry's lower bound can be stated as follows:

Proposition 6.3 ([127]). *For any r , for any quantum adversary \mathcal{A}^f performing q queries to f , we have*

$$\mathbf{Adv}_{qpp}^{S_n(r)}(\mathcal{A}) = O\left(\frac{q^3}{r}\right).$$

6.8.2 Quantum Security Statement

As in the classical setting, we consider the prp advantage against quantum adversaries that can query the whole function QuEME^E but also the inner function E . In the quantum setting, both these queries can be quantum queries.

Theorem 6.10. *For any quantum adversary \mathcal{A} playing a q -query IND-CPA game against QuEME^E with q' offline (quantum) queries to E , we have*

$$\mathbf{Adv}_{prp}^{\text{QuEME}^E}(\mathcal{A}) \leq O\left(\frac{(q + q')^2}{2^{n/3}}\right) + O\left(\left(\frac{(q + q')^2}{2^{n/3}}\right)^2\right) + \alpha^E(r^2, r^2).$$

with $r = O((q + q')2^{n/3})$, where recall that $\alpha^E(x, x')$ be an upper-bound on the advantage of an adversary who has learned x' key-plaintext-ciphertext triples for E and is trying to distinguish x other key-plaintext-ciphertext triples from random. This implies in particular that the advantage is small essentially up to $(q + q') = 2^{n/6}$.

Proof. We consider a quantum adversary \mathcal{A} that performs q quantum queries to QuEME^E and q' quantum queries to E . We perform a game based proof to prove our statement. We start from Game 1 which corresponds to the prp-advantage.

Game1 \rightarrow **Game2**. We transform Game1 into Game2 by adding two random permutations j_1, j_2 in the key and in the input of E .

Game1: prp-game(\mathcal{A})
$b \xleftarrow{\$} \{0, 1\}$
$E \xleftarrow{\$} TBC_n$
$f \leftarrow \begin{cases} \text{QuEME}^E & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$
$b' \leftarrow A^{f,E}(\cdot)$
Win if $b = b'$

Game2: adding permutations to E
$b \xleftarrow{\$} \{0, 1\}$
$E \xleftarrow{\$} TBC_n$
$j_1, j_2 \xleftarrow{\$} P_n$
Let E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$.
$f \leftarrow \begin{cases} \text{QuEME}^{E'} & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$
$b' \leftarrow A^{f,E'}(\cdot)$
Win if $b = b'$

Transforming E into E' does not change its distribution. It is still a random tweakable permutation (TBC_n). Therefore, this does not change the value of the game.

$$\Pr[\mathcal{A} \text{ wins Game1}] = \Pr[\mathcal{A} \text{ wins Game2}].$$

Game2 \rightarrow **Game3**. We transform Game2 into Game3 by adding two random permutations on the top of $\text{QuEME}^{E'}$.

Game2: adding permutations to E
$b \xleftarrow{\$} \{0, 1\}$
$E \xleftarrow{\$} TBC_n$
$j_1, j_2 \xleftarrow{\$} P_n$
Let E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$.
$f \leftarrow \begin{cases} \text{QuEME}^{E'} & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$
$b' \leftarrow A^{f,E'}(\cdot)$
Win if $b = b'$

Game3: adding permutations to $\text{QuEME}^{E'}$
$b \xleftarrow{\$} \{0, 1\}$
$E \xleftarrow{\$} TBC_n$
$j_1, j_2 \xleftarrow{\$} P_n$
Let E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$.
$f \leftarrow \begin{cases} \text{QuEME}^{E'} & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$
$h_L, h_R \xleftarrow{\$} P_n$
$f' := f \circ (h_L h_R)$.
$b' \leftarrow A^{f',E'}(\cdot)$
Win if $b = b'$

In our encrypt then mix construction, we start already by 2 random permutations so adding an extra layer of permutations will not change the distribution $\text{QuEME}^{E'}$. This of course does not change the distribution of random permutation as well hence

$$\Pr[\mathcal{A} \text{ wins Game2}] = \Pr[\mathcal{A} \text{ wins Game3}].$$

Game3 \rightarrow **Game4**. We now replace all the permutations with random small-range functions.

Game3: adding permutations to QuEME ^{E'}	Game4: adding permutations to QuEME ^{E'}
$b \xleftarrow{\$} \{0, 1\}$ $E \xleftarrow{\$} TBC_n$ $j_1, j_2 \xleftarrow{\$} P_n$ Let E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$. $f \leftarrow \begin{cases} \text{QuEME}^{E'} & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$ $h_L, h_R \xleftarrow{\$} P_n$ $f' := f \circ (h_L h_R)$. $b' \leftarrow A^{f', E'}(\cdot)$ Win if $b = b'$	$b \xleftarrow{\$} \{0, 1\}$ $E \xleftarrow{\$} TBC_n$ $j_1, j_2 \xleftarrow{\$} S_n(r)$ Let E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$. $f \leftarrow \begin{cases} \text{QuEME}^{E'} & \text{if } b = 0 \\ P_{2n} & \text{if } b = 1 \end{cases}$ $h_L, h_R \xleftarrow{\$} S_n(r)$ $f' := f \circ (h_L h_R)$. $b' \leftarrow A^{f', E'}(\cdot)$ Win if $b = b'$

From Zhandry's bound on small range functions, we have

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins Game3}] &\leq \Pr[\mathcal{A} \text{ wins Game4}] + O\left(\frac{q^3}{r}\right) + O\left(\frac{q'^3}{r}\right) \\ &\leq \Pr[\mathcal{A} \text{ wins Game4}] + O\left(\frac{(q + q')^3}{r}\right). \end{aligned}$$

Game4: Classical Emulation. We consider the following classical adversary.

Classical adversary $\mathcal{B}^{f, E}$
<ol style="list-style-type: none"> 1. Pick $j_1, j_2, h_L, h_R \xleftarrow{\\$} S_n(r)$. Let Y_1, Y_2 be the ranges of j_1, j_2 respectively, and Z_L, Z_R be the ranges of h_L, h_R respectively, so they are each subsets of $\{0, 1\}^n$ of size r. 2. Define E' st. $E'_k(x) = E_{j_1(k)}(j_2(x))$ and $f' := f \circ (h_L h_R)$. 3. Query $f(x, y)$ for each $(x, y) \in Z_L \times Z_R$ and query $E_k(x)$ for each $(k, x) \in Y_1 \times Y_2$, for a total of $2r^2$ queries. From these queries, recover the truth table of f' and E'. 4. Emulate the quantum circuit $\mathcal{A}^{f', E'}(\cdot)$ and output $b' \leftarrow \mathcal{A}^{f', E'}(\cdot)$.

$\mathcal{B}^{f, E}$ outputs exactly the same output as $A^{f', E'}$ so by definition, we have

$$\Pr[\mathcal{B} \text{ wins Game1}] = \Pr[\mathcal{A} \text{ wins Game4}].$$

and moreover, \mathcal{B} is a classical algorithm that performs $2r^2$ queries to f . We still add a few remarks on the adversary \mathcal{B} . Notice that we limit the number of queries of \mathcal{B} but not its running time so we do not need to perform the different

steps of the algorithm efficiently. From this, it makes it much easier to see how \mathcal{B} performs the different steps of the algorithm. In particular, he chooses h_L, h_R, j_1, j_2 at random and knows the full description of these functions, so he knows Y_1, Y_2, Z_L, Z_R . Also, after his $2r^2$ queries, he can know the full truth table of f' and E' , so he knows the full description of $A^{f', E'}$. A quantum algorithm on n qubits with m gates, can be emulated by a classical algorithm running in time exponential in n, m . However, he does not need any extra queries to f', E' since he already knows the full description of f' and E' . What is important here is that even though the running time is large, the amount of queries done to f and E is limited to r^2 .

We can now conclude

$$\begin{aligned}
\mathbf{Adv}_{qrp}^{\text{QuEME}^E}(\mathcal{A}) &= \Pr[\mathcal{A} \text{ wins Game1}] \\
&\leq \Pr[\mathcal{A} \text{ wins Game4}] + O\left(\frac{(q+q')^3}{r}\right) \\
&= \Pr[\mathcal{B} \text{ wins Game1}] + O\left(\frac{(q+q')^3}{r}\right) \\
&= \mathbf{Adv}_{prp}^{\text{QuEME}^E}(\mathcal{B}) + O\left(\frac{(q+q')^3}{r}\right).
\end{aligned}$$

In order to conclude, take $r = O((q+q')2^{n/3})$. From Theorem 6.7, we have

$$\begin{aligned}
\mathbf{Adv}_{prp}^{\text{QuEME}^E}(\mathcal{B}) &\leq O\left(\frac{r^2}{2^n}\right) + O\left(\frac{r^4}{2^{2n}}\right) + \alpha^E(r^2, r^2) \\
&= O\left(\frac{(q+q')^2}{2^{n/3}}\right) + O\left(\left(\frac{(q+q')^2}{2^{n/3}}\right)^2\right) + \alpha^E(r^2, r^2).
\end{aligned}$$

and from the choice of r , we have $O((q+q')^3/r) = O((q+q')^2/2^{n/3})$. From there, we conclude

$$\begin{aligned}
\mathbf{Adv}_{qrp}^{\text{QuEME}^E}(\mathcal{A}) &\leq \mathbf{Adv}_{prp}^{\text{QuEME}^E}(\mathcal{B}) + O\left(\frac{(q+q')^3}{r}\right) \\
&\leq O\left(\frac{(q+q')^2}{2^{n/3}}\right) + O\left(\left(\frac{(q+q')^2}{2^{n/3}}\right)^2\right) + \alpha^E(r^2, r^2).
\end{aligned}$$

□

6.8.3 Discussion

Our proof is very generic and we can relate the quantum security to the classical security for any construction that start by encrypting the left and right halves

of the input. The drawback of this strategy is that it seems to be far from tight. Indeed, when looking at our construction and the attack running with $O(2^n)$ queries it's not clear how to use quantum queries to improve this attack. We expect our construction to have much more than $n/6$ bits of quantum security, maybe its quantum security is actually n bits.

In order to improve these bounds, one natural path would be to look at Zhandry's quantum query recording technique. However, in our case, we need to consider random permutation and not random functions and this is notoriously hard, as some of the proposals for this turned out to be incorrect (see for instance [116]). As this topic becomes more mature, we hope that this tool will be available for proving tight quantum security bounds for our construction.

6.9 Proposing a Concrete Instance: Double-AES

The aim of this section is to propose a concrete instance that would motivate cryptanalysis on these constructions. For this, we propose some variants that claim the same quantum security as the Saturnin block cipher [33], that was conceived with the objective of proposing resistance against quantum-attackers. In particular, we also claim that:

There exists no quantum attack in the single-key setting with $T^2/p < 2^{224}$, where T is the time complexity and p the success probability. We do not provide security against related-key superposition attacks (as is the case of all known block ciphers).

Therefore, in this section we propose a concrete construction based on AES-128 [40], showing how to use 256-bit keys and domain extensions in the different blocks, and proposing several possible variants depending on how many AES rounds are considered in each block, and we announce their corresponding (quantum) security claims. We have chosen the AES-128 block cipher to profit from its large amount of cryptanalysis on reduced round versions, that will simplify analyzing the instantiation of the new construction. We also briefly describe the best attacks we have found so far in these family of constructions. As we will see, these attacks motivated us to include the last MC transformation on each block cipher call, but on the two ones from the final layer. A brief description of AES-128 and of the best known attacks on it can be found in Section 5.5.1.

The main aim of these instantiations is to motivate cryptanalysis and comparison with other constructions as well as further research.

6.9.1 How to Extend the Keys?

Independently of the blocks ciphers used, we can decide how to choose the keys used in each block. As we expect that a $2n$ key can have the same security as a $4n$ one, we will choose the keys for E_3 and E_4 to be dependent on the ones

from E_1 and E_2 for the sake of simplicity. For this, we propose to have a $2n$ key formed by $(k_1||k_2)$. These keys will be used as the key of the two blocks applied in the input of the construction: k_1 for E_1 and k_2 for E_2 . We have to define now the keys k_3 and k_4 for the two output blocks E_3 and E_4 . The idea is that, from the knowledge from one input key we should not be able to retrieve any information from any other key. For this, we propose to define:

$$k_3 = k_1 \oplus k_2 \text{ and } k_4 = k_1 \oplus (k_2 \lll 1).$$

Though more robust and complicated key-extensions could be proposed, like for instance $K_1, K_2, E_5^{K_2}(K_1), E_6^{K_1}(K_2)$, we believe ours has the advantage of being simpler and very efficient, and should have an equivalent security as long as the used block cipher is secure.

Property : guessing one full key (k_1 for instance) and x bits of any other key (*i.e.* k_2), allows to compute x and $x - 1$ bits of the other two keys (k_3 and k_4 in the example). We could have chosen other simple configurations where the bits determined in the two last subkeys wouldn't belong mainly to the same bytes, but we believe this configuration is interesting, for the sake of simplicity of implementation and analysis.

6.9.2 Introducing a Domain in AES for Defining E_j

We present in Section 5.5.1 the specifications of AES-128. For defining 4 additional different instances E_1 to E_4 , we propose to use a different constant for each block. All the remaining parts stay the same and we only change the constant, as: with

$$rc_{i,j} = \begin{pmatrix} X^i \bmod X^8 + X^4 + X^3 + X + 1 \\ j \\ 0 \\ 0 \end{pmatrix}.$$

For the middle call to the block cipher we consider the original AES definition, where the input arriving as a parameter will have the role of the secret key.

6.9.3 Double-AES Concrete Proposals

Double-AES-10. A very conservative approach would be to consider a full AES encryption inside each block.

Conjecture: the 7-round attacks cannot be exploited when using AES in our construction. We have tried to build attacks in our construction exploiting the best known attacks on 7-rounds when we consider the blocks

used in the instantiation reduced to 7 rounds and we have not been able to do so. We conjecture that these attacks cannot apply when using our construction, and we propose therefore Double-AES-7, that instead of 10 rounds in each block only used 7. We also believe that it is a quite conservative approach, given the best found attacks in the next section.

Variants Including Last MC: Double-AES-6-MC. We propose a variant where the last MC operation is considered in the block call that do not belong to the last layer (E_3 and E_4). We encourage the cryptanalysis of Double-AES-5-MC, for which we think an attack might exist, and we conjecture that Double-AES-6-MC should provide a similar security than Double-AES-10.

Best Attacks. In Section 6.10, we present the best attacks we have found. In order to reflect that different number of rounds can be considered per block, we call an attack on r_1 - r_2 - r_3 when it covers r_1 rounds for E_1 and E_2 , r_2 rounds for E and r_3 rounds for E_3 and E_4 . Our best attacks cover X-3-3 and X-2-X, without the last MC in E_1 , E_2 and E .

6.9.4 Security Claims

The final construction has a 256-bit state and key. We provide an unique security claim, not distinguishing between classical and quantum attacks: we claim that our extended block cipher Double-AES, with versions Double-AES-10, Double-AES-7 and Double-AES-5-MC provide around 128 bits of security against any type of attacker.⁴

In addition, we claim that when plugged in secure modes, any attack that requires a collision on the state would require at least the generic complexity for generating a collision, that is, no attack significantly better than $\mathcal{T}^5 \times M_q = 2^{512}$ exists, where \mathcal{T} represents the time complexity, and M_q the quantum memory (that includes the classical memory).

6.9.5 Discussion

As already said, we considered Saturnin [33], the first conceived block cipher aiming at security against all quantum attackers, as a model for our quantum security claims. Given that that construction also inherits a lot from the AES one, we consider the comparison interesting. The same goes for the cipher AES-256 [41]. In particular, if we consider the Double-AES-7, as the two upper and the two lowers can be done in parallel, the time of one encryption should take an equivalent of $3 \times 7 = 21$ AES rounds, while AES-256 takes 14 rounds for encrypting half of the state. Rijndael-256 [40] is a similar case than the AES-256, but with a state of 256 bits this time, as Double-AES. Nevertheless,

⁴As all known block ciphers, we cannot provide security against superposition related-key attacks.

this constructions, that was not standardized, has been much less studied by the community and benefits less from the cryptanalysis knowledge of the community.

In all the three cases, our construction has the advantage of being a generic way for doubling block ciphers, and therefore, we believe further study would be useful for understanding the properties of the underlying block cipher that can generate an attack or that can not.

We expect further cryptanalysis to show if we can reduce the number of rounds in Double-AES-7 or in Double-AES-5-MC further in order to gain in interest with respect to these previous constructions while staying secure.

6.10 Best Attacks Found of Double-AES

The two best attacks we have found work on 3 rounds in E , E_3 and E_4 blocks and any number of rounds in the two first ones (X-3-3); and on 2 rounds in E while having any number of rounds in the upper and lower blocks (X-2-X) and are, logically, quantum attacks. They both use a step consisting on guessing at least a whole 128-bit key.

They also exploit the following property:

First Middle Round Canceled. Given the key-addition operation of the AES, the input of the middle block after the first key addition will be equal to the output of E_2 . This property is very interesting. As an example, if we consider differences and if the input of the right part of the state is fixed, the first SB transfer of the middle round will have no active sboxes,

6.10.1 Attack on X-3-3 Without Last MC in Blocks E_1 , E_2 and E .

This attack is based on the square attack [55]. From it we know that, if the input is an active diagonal only in the left part of the plaintext, we will recover 2^{24} sets that verify that the state 4 rounds later is an state where all of its bytes take all the possible values independently. The aim is to generate a square state thanks to the left input at the beginning of the middle E , but we have to be careful, as this input state will also influence its subkeys. In order to make it work, we will consider a byte (in the before last column, not to affect too much further subkeys due to key-schedule), that takes all the 2^8 possible values in the input instead of structures of 2^{32} , and we will then cover with the distinguisher one less round than the original square attack.

We guess k_1 of E_1 , in order to generate an input to E with two fully active bytes, as shown in Figure 6.4. Therefore, any number of rounds in E_1 would allow the attack to work. In addition, we guess the subkeys from k_3 associated to: 32 bits of antidiagonal for the last subkey, and 8 bits of subkey

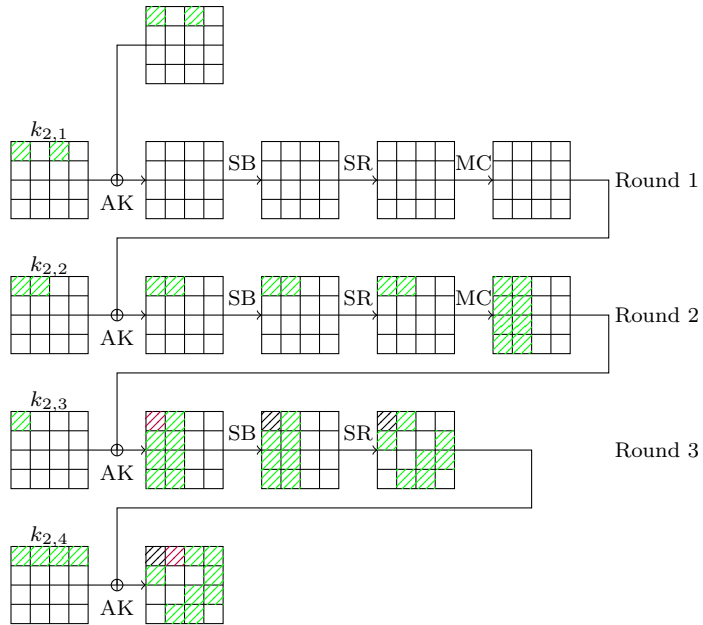


Figure 6.4: Square-like property on the middle part. We use green for bytes that take every values, purple for balanced bytes and black for ignored bytes.

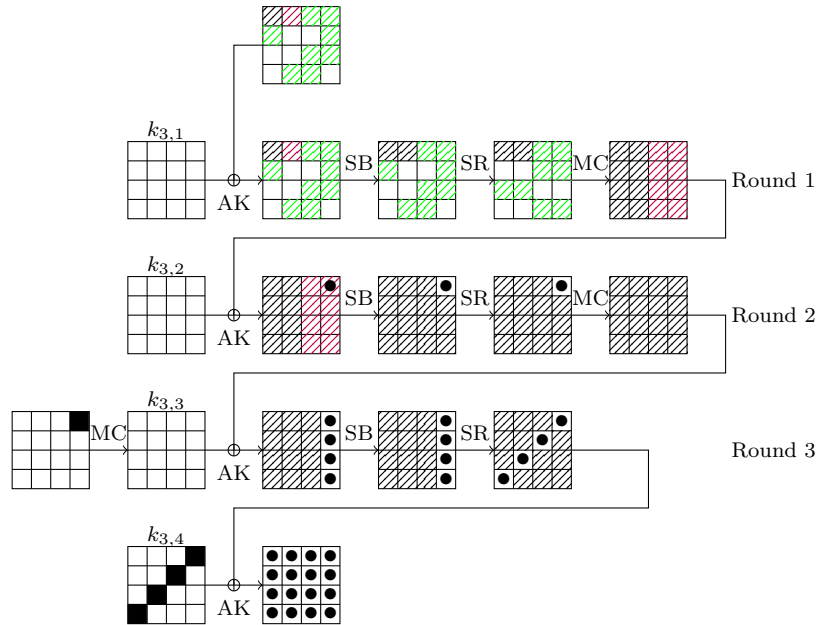


Figure 6.5: Recovery on the bottom part. We use green for bytes that take every value, purple for balanced bytes, black for ignored bytes, ■ for guessed bytes and ● for deduced and known bytes.

for the before-last equivalent subkey, as shown in Figure 6.5. This allows us to compute, for a given left output, a byte after the first MC transformation in E_3 and check whether the sum of the resulting bytes is 0 or not. This filters one guess out of 256.

In order to increase this sieving, we choose, instead of one fixed state for the right input, 21, which would provide a sieving of $2^{-8 \times 21}$ to have balanced bytes each time, leaving approximately $2^{128+5 \times 8} \times 2^{-8 \times 21} = 1$ key guess.

Complexity will be $21 \times 2^8 \times 2^{(128+32+8)/2} = 2^{96.5}$ data and time.

We expect that these attacks might be extended to 4-4-4, or to 4-3-4 with MC by having a closer look at the properties generated by the key-schedule of the middle block.

6.10.2 Attack on X-2-X Without Last MC in Blocks E_1 , E_2 and E .

Guessing k_3 . We start with a fixed pair (P_1, P_2) of left blocks of plaintexts and perform the encryption through Double-AES-X-2-X of (P_1, R) , (P_2, R) , for a fixed value R . We consider exclusively the left part of the output, we obtain C_1 and C_2 . For each guess of key k_3 from E_3 , we will try a decryption through E_3 of C_1 and C_2 and note the difference δ .

Guessing k_1 and Some Middle Subkeys. For each guess of key k_1 from E_1 , we will try a encryption through E_1 of P_1 and P_2 . This will produce values x_1 and x_2 that correspond to the values that should enter the middle part E .

Then, we will experience the cancellation of the first round as described earlier. The second round starts by a key addition, and we can get to know the differences on the bytes 0,4,8 and 12 (the first line) before the second SB for one additional guess of the byte 13 of $E_2(R)$. Each one of this differences can be associated to $2^{32-4} = 2^{28}$ output differences through the DDT of these four sboxes. The output differences of these second SB will be determined by δ xored to the last subkey of the middle round. In order to compute the difference of this subkey for the first line, and therefore the possible values for finding a match of this first line with the δ s, we can perform an additional guess guess of the byte 14 of $E_2(R)$ and the xor of the bytes 1,5,9 and 13 of $E_2(R)$. We therefore get to compute the possible output differences of E and compare them to the differences δ we described earlier.

As we just said, the probability of this sieving is of 2^{-4} because of the DDT.

More Pairs. In order to sieve more guesses, we use 70 pairs instead of one, which leaves approximately $2^{128+128+3 \times 8} \times 2^{-4 \times 70} = 1$ combination.

Complexity. We can then use an element distinctiveness algorithm to find the correct combination of (k_1, k_3) . Thanks to Ambainis algorithm [5], the cost of this attack will be about $70 \times (2^{128} + 2^{128+24})^{2/3} = 2^{107.5}$ time and memory.

6.10.3 Best known attacks on AES

List of Round-reduced Attacks We expose a quick list of the best known attack against round-reduced AES-128 in the different settings.

Attack	Rounds	Time	Data	Reference
Mixture Differential	5	$2^{21.5}$	$2^{21.5}$	[8]
Yoyo	5	2^{33}	$2^{13.3}$	[105]
Partial Sum	5	2^{40}	2^8	[115]
Rectangle	5	2^{23}	2^9	[49]
Rectangle	5	$2^{16.5}$	2^{15}	[49]
Improved Square	5	2^{35}	2^{33}	[55]
Boomeyong	5	2^{49}	2^{49}	[104]
Rectangle	6	2^{80}	2^{26}	[49]
Partial Sum	6	2^{44}	$2^{34.5}$	[55]
Truncated Differential	6	$2^{78.7}$	$2^{71.3}$	[7]
Boomeyong	6	$2^{79.72}$	$2^{79.72}$	[104]
Impossible Differential	7	$2^{117.2}$	$2^{112.2}$	[88]
Meet-in-the-Middle	7	2^{116}	2^{116}	[50]
Impossible Differential	7	2^{113}	$2^{105.1}$	[30]
Impossible Differential	7	$2^{110.9}$	$2^{104.9}$	[87]
Meet-in-the-Middle	7	2^{99}	2^{97}	[44]

Table 6.1: Current cryptanalysis for round-reduced AES-128 in the secret-key model.

6.11 Conclusion

In this chapter, we provide the first proposal of a generic way for extending both the key and the state size, with quantum security arguments and significant classical proofs. In addition we have proposed a new type of superposition attacks on the EME construction, an original distinguisher matching the bound of our construction, and a method considering simulations for supporting conjectures from proofs.

On concrete instances and cryptanalysis. Concrete instances of our construction combined with AES-128 have been proposed, with the aim of

Attack	Rounds	Time	Data	Reference
Related-key				
RK Boomerang	7	2^{97}	2^{97}	[23]
Chosen-key				
Multi-collision	9	2^{55}	2^{55}	[56]
Multiple-of-n	9	2^{64}	2^{64}	[62]
Known-key				
Uniform Distribution	10	2^{64}	2^{64}	[60]
Uniform Distribution	12	2^{82}	2^{82}	[63]

Table 6.2: Current cryptanalysis for AES-128 in the related-key/chosen-key/known-key model.

motivating its study, with a unified security claim regarding classical and quantum attackers. We believe the number of rounds of AES-128 considered in the building blocks can be reduced. We propose 7 rounds and 5 rounds if the final *MC* is not omitted in E_1 , E_2 and E , where we claim equal security than with 10 rounds, but we believe an interesting question would be whether we manage to attack a variant with less rounds, as our best attack reaches X-3-3 rounds (so any number of rounds in the first layer, and three in the middle and final layers), or 2 rounds in the middle one, for any number of rounds in the upper and lower instances. Related-key attacks on AES might also have an interesting application in this case because of the middle block. We leave this as an interesting open question to break a variant with 5-5-5 rounds, or 6-6-6, that should be harder, as the structural distinguishers could be exploited. Another option would be to consider variants with less rounds in the middle block than in the four external ones. How low could we go? We know attacks exists with only 2 rounds in the middle, for any number of rounds in the external applications.

Chapter 7

Quantum Security of UMTS-AKA

In this chapter, we first define a security model that does not rely on unconditional security and allows honest parties and attackers to use superposition messages. In particular, assuming that we are in a quantum world, we provide a discussion on what can be put into superposition by the client, the server and the operator during a honest execution of such *quantum UMTS-AKA* protocol. More precisely, the quantum version of the UMTS-AKA that we describe assumes quantum computations as well as quantum communications (i.e., the messages that are exchanged between parties are in a superposition of quantum states).

We then formally prove that this quantum version of the UMTS-AKA is as secure in this quantum model as it is in the classical setting [4]. As the quantum version we built is an extension of the classical one, the attack by Zhang is still valid [128]. It consists in using the extra authentication vectors of a corrupted server by the attacker elsewhere to make an unauthorized authentication. As a supplementary contribution, we also exhibit a new attack against the state confidentiality of a mobile user. This attack holds in the quantum but also in the classical setting.

We finally study the security of the underlying primitives that can instantiate the UMTS-AKA, Milenage and TUAK. We show an attack on Milenage as a qPRF, however the context of the UMTS-AKA makes this attack unrealistic and we further prove the security of Milenage based on the security of AES. We also show a reduction from the security of TUAK to the security of Keccak-f.

It is based on a joint work with Sébastien Canard and Loïc Ferreira that is currently in a review process.

Contents

7.1	The AKA Protocol	158
7.1.1	The Classic Version of AKA	158
7.1.2	Quantum AKA Protocol	162
7.2	Adversarial Model and Building Blocks	165
7.2.1	Oracles	166
7.2.2	Targeted Notions	168

7.2.3	Pseudo-random Function	169
7.2.4	Pseudo-random Key-derivation	170
7.3	Quantum Security of the UMTS-AKA Protocol	170
7.3.1	Session Keys Indistinguishability	170
7.3.2	Client-impersonation Resistance	174
7.3.3	Weak Server-impersonation Resistance	176
7.3.4	Soundness	179
7.3.5	State Confidentiality	181
7.4	Quantum Security of Milenage	184
7.4.1	Description of Milenage	184
7.4.2	Attack on Milenage as a Pseudo-Random Function	184
7.4.3	Security Proof of Milenage in AKA	185
7.5	Quantum Security of TUAK	190
7.5.1	Description of TUAK.	190
7.5.2	Security Arguments of TUAK.	191
7.6	Conclusion	191

7.1 The AKA Protocol

In this section, we present the standard protocol AKA used in the 3G and 4G infrastructure, and which is also the basis of the 5G AKA. The we describe a quantum variant supposed to be run between quantum computers and through superposition-allowing channels.

7.1.1 The Classic Version of AKA

The AKA procedure consists of an exchange of Message Authentication Codes (MAC) and key derivations from a fresh random value, and static secret keys shared by the *client* (Mobile Equipment/User Subscriber Identity Module, ME/ USIM) and the *operator* (Home Location Register, HLR). It is executed as a challenge response through a *server* (Visited Location Register, VLR).

7.1.1.1 Elements of the Protocol.

The main elements are the following.

Pseudo-random Functions. The protocol defines seven functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ which take as input the secret keys and the random value R . f_1 and f_1^* take as additional inputs the sequence number (Sqn_C or $Sqn_{Op,C}$), and an Authentication Management Field (AMF). f_1 outputs the MAC tag Mac_S sent to the client. f_2 outputs the MAC tag Mac_C sent to the server. f_3 and f_4 output respectively the session keys CK (Confidentiality Key) and

IK (Integrity Key). f_5 outputs an Anonymity Key (*AK*) used to mask Sqn . f_1^* and f_5^* intervene in place of f_1 and f_5 during the the re-synchronization procedure. In practice, these functions are instantiated by either Milenage [1] or TUAK [112].

Static Secret Keys. The client key (sk_C) is known by the client and the operator. From the latter and the operator key (sk_{Op}) an intermediate value ($sk_{Op,C}$) is derived as $sk_{Op,C} = KD(sk_C, sk_{Op})$.¹ The client stores $sk_{Op,C}$ while sk_{Op} is known only to the operator. We note $sk_s = sk_C || sk_{Op,C}$.

Random Value. A fresh random value R is generated during each session, and used as challenge for the client. It aims at protecting the client and the operator from replay attacks.

Sequence Number. The client and the operator keep respectively the secret values Sqn_C and $Sqn_{Op,C}$, which number the executions of the protocol, and prevent replay attacks against the client. A re-synchronization procedure may be executed in case of discrepancy.

We consider that the values Sqn_C do not repeat (otherwise the client would accept values from a previous execution of the protocol). To be sure the operators do not allow for a repetition of Sqn_C , we restrict the number of authentication vector all operators can produce with the same sk_C to be at most $\frac{2^{|Sqn_C|}}{\Delta}$ (since a client uses the same Sqn_C for every operator), where Δ , chosen by the operator, defines an acceptance interval.

7.1.1.2 Description of an Execution.

The main steps are the following (see Figure 7.1).

Generation of the Challenge. First the server requests from the client an *UID*, which is an IMSI (International Mobile Subscriber Identify) or a TMSI (Temporary Mobile Subscriber Identity), a value agreed in a previous session. The server forwards it to the operator. The operator produces n Authentication Vectors (*AV*), sends them to the server, and updates $Sqn_{Op,C}$ to $Sqn_{Op,C} + n$. Each vector AV^i , $1 \leq i \leq n$, is computed from a fresh random

¹ sk_{Op} is denoted as OP_C in Milenage and TOP_C in TUAK.

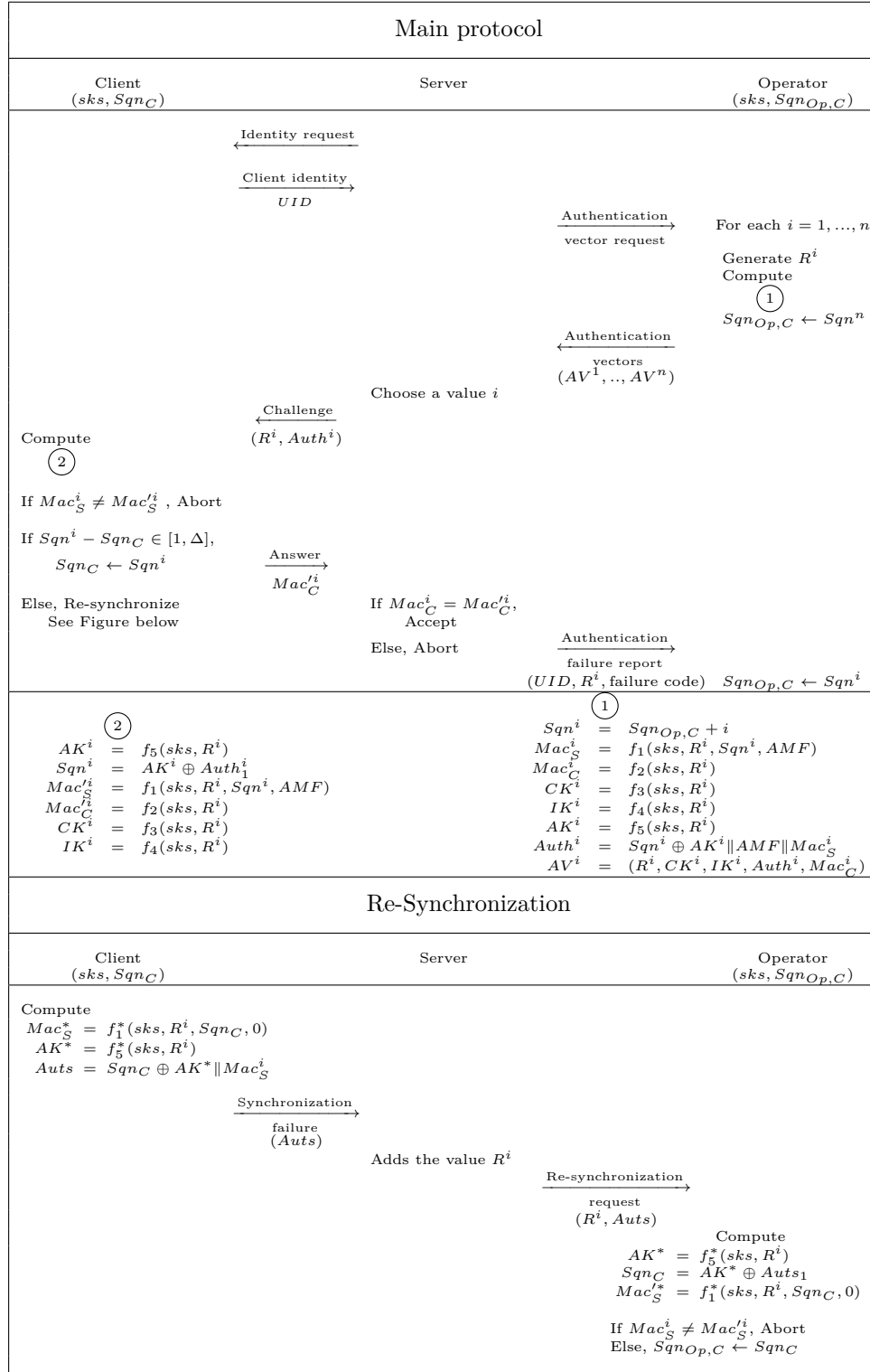


Figure 7.1: AKA protocol and re-synchronization procedure

value R^i as follows:

$$\begin{aligned}
Sqn^i &= Sqn_{Op,C} + i \\
Mac_S^i &= f_1(sk_s, R^i, Sqn^i, AMF) \\
Mac_C^i &= f_2(sk_s, R^i) \\
CK^i &= f_3(sk_s, R^i) \\
IK^i &= f_4(sk_s, R^i) \\
AK^i &= f_5(sk_s, R^i) \\
Auth^i &= Sqn^i \oplus AK^i \| AMF \| Mac_S^i \\
AV^i &= (R^i, CK^i, IK^i, Auth^i, Mac_C^i)
\end{aligned}$$

Challenge-Response. The server picks a vector AV^i and sends the corresponding R^i to the client. In turn, the latter computes

$$\begin{aligned}
AK^i &= f_5(sk_s, R^i) \\
Sqn^i &= AK^i \oplus Auth_1^i \\
Mac_S^i &= f_1(sk_s, R^i, Sqn^i, AMF) \\
Mac_C^i &= f_2(sk_s, R^i) \\
CK^i &= f_3(sk_s, R^i) \\
IK^i &= f_4(sk_s, R^i)
\end{aligned}$$

The client verifies that $Mac_S^i = Mac_S^i$, sends an abort message if not, and checks if the challenge is fresh, i.e., $Sqn^i \in [Sqn_C + 1, Sqn_C + \Delta]$. Next it sets Sqn_C to Sqn^i , and replies to the server with Mac_C^i . If this answer is incorrect, the server sends to the operator a report containing the R^i of the used AV^i , the client's UID , and a failure code. In such a case the operator updates $Sqn_{Op,C}$ to Sqn^i .

Re-synchronization. If $Sqn^i \notin [Sqn_C + 1, Sqn_C + \Delta]$ the client triggers a re-synchronization procedure (see Figure 7.1). First it computes

$$\begin{aligned}
AK^* &= f_5^*(sk_s, R^i) \\
Mac_S^* &= f_1^*(sk_s, R^i, Sqn_C, 0) \\
Auts &= (Sqn_C \oplus AK^*) \| Mac_S^*
\end{aligned}$$

where R^i corresponds to the current challenge. The client sends $Auts$ to the server which forwards it along with R^i to the operator. Then the operator computes

$$\begin{aligned}
AK^* &= f_5^*(sk_s, R^i) \\
Sqn_C &= AK^* \oplus Auts_1 \\
Mac_S^* &= f_1^*(sk_s, R^i, Sqn_C, 0)
\end{aligned}$$

and verifies if $Mac'_S = Mac^*_S$. If so, it updates $Sqn_{Op,C}$ to Sqn_C , otherwise it aborts the procedure.

7.1.2 Quantum AKA Protocol

In this subsection we describe a quantum variant of the AKA protocol. More precisely, we consider a version within a full quantum world, where both honest and dishonest parties are given quantum capabilities. Hence, each party is able to perform quantum computations, and communication allows transferring qubits in superposition. Since the values can still be computed classically (measurements are still possible), this quantum variant is obviously an extension of the classical AKA.

7.1.2.1 Modelization of Quantum Communications.

We base our modelization of quantum communications on the work of Yao [123]. For a communication between Alice and Bob, the total state $|a\rangle|c\rangle|b\rangle$ is composed of three parts:

- $|a\rangle$ is Alice's secret part and modifiable only by her;
- $|b\rangle$ is Bob's secret part and only modifiable by him;
- $|c\rangle$ is the state of the channel, it is modifiable by everyone, or only Alice and Bob if the channel is secure (like the ones between operators and servers).

Other modelizations. Other modelizations of quantum communications exists, like [36], where communications are made using EPR-pairs, pairs of entangled qubits shared by Alice and Bob. While this englobes our modelization, it implies making and storing those pairs. The latter seems highly unpractical.

7.1.2.2 Superposition or Not

We detail below whether the parameters of the protocol may be used as values in superposition in the quantum variant.

Static Secret Keys. These keys are long-term values whose lifespan depends only on the time needed by an attacker to find them. Having long-term keys maintained in superposition seems obviously unpractical. From now on, we consider client and operator keys (sk_C, sk_{Op}) to be in a classical state. Consequently $sk_{Op,C}$ is also in classical state.

Sequence Number. We consider sequence numbers to be in classical state.

Random Value. The random R does not need to be stored from one session to another. Therefore it is a matter of choice to keep it in classical state or superposition state. As the latter is a generalization of the former, we allow for the random values to be used in superposition. The operator can generate authentication vectors in superposition, and the attacker is allowed to send superposition queries to client and server parties.

Confidentiality and Integrity Keys. As R is allowed to be in superposition, we expect the session keys CK and IK to be in superposition too.

7.1.2.3 Challenge Generator.

In the classical case [4], R is expected to be an unpredictable value from $\{0, 1\}^{|R|}$. In a quantum setting, R are elements of $\mathbb{C}^{2^{|R|}}$. Therefore our expectations of its randomness are meant to change. While the pseudo-random functions $f_1, \dots, f_5, f_1^*, f_5^*$ now have to resist quantum attacks, one could think the generation of R would face a similar challenge. Yet our security proofs only involve few properties of such a generator.

Expectations. We expect the function that generates $R = \sum_{i=0}^{2^{|R|}-1} a_i |i\rangle$ to have the following properties:

- a newly generated R is not entangled to any other R ;
- for all $i \in \{0, 1, \dots, 2^{|R|} - 1\}$, $\mathbb{E}[|a_i|^2] = 2^{-|R|}$.

We want the random generator to produce independent uncorrelated values as we want to take care of minimal long-term values (we have the keys for black-boxing $f_1, \dots, f_5, f_1^*, f_5^*$ and Sqn for resistance toward replay attack). We want also the random generator to have a good distribution to make use of the full space of possible challenges.

Useful Property for Security Proofs in Later Sections. Then we have the following property: with n values R^1, \dots, R^n , the probability to measure a collision among R^1, \dots, R^n is less than $\frac{n^2}{2^{|R|}}$.

Possible Construction. Note that $|R\rangle = \frac{1}{\sqrt{2^{|R|}}} \sum_{i=0}^{2^{|R|}-1} |i\rangle$ is a possible option of generation that respects our expectations. While this challenge generation does not contain any randomness, it still has the properties we need. Indeed, as the further computations of $f_1, \dots, f_5, f_1^*, f_5^*$ entangle R with other elements of the protocol that are uncomputable by the adversary, the underlying measurement results in the variability of the protocol.

7.1.2.4 Description of a Quantum Execution

The main steps of the quantum version of the protocol are the following (see Figure 7.2).

Generation of the Challenge. For $1 \leq i \leq n$, the operator generates each superposition $\sum_{R^i} a_{R^i} |R^i\rangle$, increments $Sqn^i = Sqn_{Op,C} + i$, and computes the parameters entangled with R^i . The operator gets the superposition:

$$\sum_{R^i} a_{R^i} |R^i, Mac_S^i, Mac_C^i, CK^i, IK^i, AK^i\rangle$$

$$\begin{aligned} Mac_S^i &= f_1(sks, R^i, Sqn^i, AMF) \\ Mac_C^i &= f_2(sks, R^i) \\ CK^i &= f_3(sks, R^i) \\ IK^i &= f_4(sks, R^i) \\ AK^i &= f_5(sks, R^i) \end{aligned}$$

The operator computes $Auth^i = |Sqn^i \oplus AK^i \parallel AMF \parallel Mac_S^i\rangle$ and $AV^i = |R^i, CK^i, IK^i, Auth^i, Mac_C^i\rangle$, and sends the authentication vectors AV^i to the server.

Challenge-Response. The server picks a vector i and sends the corresponding challenge to the client. The client computes

$$\begin{aligned} AK^i &= f_5(sks, R^i) \\ Sqn^i &= AK^i \oplus Auth_1^i \\ Mac_S^i &= f_1(sks, R^i, Sqn^i, AMF) \\ Mac_C^i &= f_2(sks, R^i) \\ CK^i &= f_3(sks, R^i) \\ IK^i &= f_4(sks, R^i) \end{aligned}$$

At this point, several qubits are shared between the operator, the server and the client. They are entangled, and every state value depends only on the value R^i it was computed from. The quantum state of the system can be represented as

$$\sum_{R^i} a_{R^i} \left(\begin{array}{c} R^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \\ Sqn_{Op,C} \end{array} , \begin{array}{c} R^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \oplus Sqn_{Op,C} \end{array} , \begin{array}{c} R^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \end{array} \right)$$

The first, second and third columns correspond respectively to the client's, server's, and operator's view.

The end of the protocol goes as follows. The client measures $Mac_S^i \oplus Mac_C^i$ to verify that it equals 0, and sends an abort message if not. Then it measures Sqn^i and checks if the challenge is fresh, i.e., $Sqn^i \in [Sqn_C + 1, Sqn_C + \Delta]$. Finally the client answers the challenge, and sets Sqn_C to Sqn^i . The server checks the answer by measuring $Mac_C^i \oplus Mac_S^i$, and proceeds as in the classical protocol.

Re-synchronization. If $Sqn^i \notin [Sqn_C + 1, Sqn_C + \Delta]$, the client triggers a re-synchronization procedure. First it computes the superposition $\sum_{R^i} a_{R^i} |R^i, Mac_S^*, AK^*\rangle$ where R^i comes from the current challenge and

$$\begin{aligned} AK^* &= f_5^*(sks, R^i) \\ Mac_S^* &= f_1^*(sks, R^i, Sqn_C, AMF) \end{aligned}$$

The client generates $Auts = |(Sqn_C \oplus AK^*) \parallel Mac_S^*\rangle$ and sends it to the server which forwards it as $\sum_{R^i} a_{R^i} |R^i, Auts\rangle$ to the operator. Then the operator computes

$$\begin{aligned} AK^* &= f_5^*(sks, R^i) \\ Sqn_C &= AK^* \oplus Auts_1 \\ Mac_S^* &= f_1^*(sks, R^i, Sqn_C, AMF) \end{aligned}$$

and verifies if the measure of $Mac_S^* \oplus Mac_C^*$ equals 0. If so, it updates $Sqn_{Op,C}$ to Sqn_C , otherwise it aborts the procedure.

7.1.2.5 Notes on the Measurements.

During the protocol, we measure Sqn_C as a value in classical state. We also need to measure $Mac_S \oplus Mac_C'$ and $Mac_C \oplus Mac_S'$ as it decides if the protocol continues or is interrupted.

Compared to the classical protocol, allowing for answers in superposition may mislead us into think that an adversary can have a higher probability to get an accepting state by entering random values. As seen in the description of the protocol, every value is determined by R^i . For each R^i , the probability of acceptance being identical, the same holds regarding the superposition of R^i . Another way to see this phenomenon is to pass the deciding measurements, the answer needs to be measured as the expected superposition which is entangled with R .

7.2 Adversarial Model and Building Blocks

In this section, we define the quantum adversarial model for a quantum variant of the UMTS-AKA protocol, and some building blocks we need. We here adapt the work of Alt, Fouque, Macario-Rat, Onete and Richard [4] (done in a classical context) to our quantum setting.

7.2.1 Oracles

The adversary interacts with the system by means of the following oracles, in addition to the functions $f_1, \dots, f_5, f_1^*, f_5^*$ and KD through their specifications.

$CreateClient(Op) \rightarrow (sk_C, ID_C, Sqn_C)$: this oracle (used by the challenger) creates a client C with unique identifier ID_C , the client's secret keys sk_C and $sk_{Op,C} = KD(sk_C, sk_{Op})$ and the sequence number Sqn_C . The tuples $(ID_C, sk_C, sk_{Op,C}, Sqn_C)$ are associated with the client ID_C and with the corresponding operator Op (i.e., each “copy” of Op in each server does this). The operator sets $Sqn_{Op,C} := Sqn_C$ and then keeps track of $Sqn_{Op,C}$. The adversary is given ID_C .

$CreateOperator() \rightarrow (sk_{Op}, ID_{Op})$: this oracle (used by the challenger) creates an operator Op with unique identifier ID_{Op} , the operator's secret keys sk_{Op} . The adversary is given ID_{Op} .

$CreateServer(Op) \rightarrow (ID_S)$: this oracle (used by the challenger) creates a server S with unique identifier ID_S . The challenger makes a secure channel between the new server and the operator Op . The adversary is given ID_S .

$NewInstance(P) \rightarrow (P_j, m)$: this oracle instantiates the new instance P_j , of party P , which is either a client or a server. Furthermore, the oracle also outputs a message m , which is either the first message in an honest protocol session (if P is a server) or \perp (if P is a client).

$Execute(C, i, S, j) \rightarrow \tau$: selects the client C and the server S , creates (fresh) instances C_i, S_j , allocates the necessary quantum memory for these instances, then runs the protocol between them using the allocated quantum memory. The adversary has access to the transcript of the protocol instance τ via the channel state.

As for the immediate description of the protocol, there is no use of the results CK, IK , the adversary gets maximum control of the transcript by making the client uncompute its quantum values of the protocol (except R as it comes from the challenge sent by the server) and the challenger uncompute the quantum values of the protocol in the sever and in the operator (except R , as it links the quantum state of the protocol, and the ones concerned by a *RevealSession* or *Test* query, see below) in the clients, servers and operators. Then the only remaining value in superposition are the R in the client and in the operator which are sent to the attacker (it already has access to it).²

²This process separates the used memory of previous executions entangled with the attacker and the memory in new executions of the protocol.

(One could consider the whole system to be a simulation and then executing the protocol on superpositions of clients, servers and operators but this is not our interest here. As such, this oracle takes classical values but returns a script written on qubits and most often in superposition, as specified in Section 7.1.2).

$Send(P, i, m) \rightarrow m'$: simulates sending the message m to the instance P_i of P . The output is a response message m' (which is set to \perp in case of an error or an abort). As our protocol is meant to run on superposition allowing channels, the messages m and m' use qubits and can be in superposition.

$RevealSession(P, i) \rightarrow \{K, \perp\}$: this oracle can be used just after an execution of the protocol. If the party has not terminated in an accepting state, this oracle outputs \perp , else it outputs the session keys $K = (CK, IK)$ computed by P_i on the last execution involving P_i (as explained in Section 7.1.2 the session keys are expected to be a superposition of values).

$CorruptClient(C) \rightarrow (sk_C, \{sk_{Op,C}\})$: this oracle corrupts C and returns the long-term client key sk_C and $sk_{Op,C}$ (not in superposition), but not sk_{Op} as the client is not given the operator keys.

$CorruptServer(S)$: This oracle corrupts the server S and gives the adversary access to a special oracle $OpAccess$.

$OpAccess(S, C) \rightarrow m$: for a corrupted server S , this oracle gives the adversary access to the server's local copy of all the operators, in particular returning the message that the operator Op would output to this server for initializing an execution of the protocol with the client C . (We do not consider superpositions of parties, and the output message can be a superposition.)

$RevealState(C, i, b) \rightarrow Sqn_C$: for a client C , if $b = 0$, then this oracle reveals the current state of C_i , else, if $b = 1$, then the oracle returns the state the operator stores for C (Sqn_C is not a superposition).

$Test(P, i) \rightarrow K$: this oracle can be used just after an execution of the protocol. It is initialized with a secret random bit b and secret random functions f'_3 and f'_4 whose outputs are in the same space as CK and IK respectively. It returns \perp if the instance P_i is not fresh or if it has not terminated in an accepting state (with a session key CK, IK). If $b = 0$, then the oracle returns $CK = f_3(R), IK = f_4(R)$, else it returns $CK' = f'_3(R), IK' = f'_4(R)$. We assume that the adversary makes a single $Test$ query.

Partners. Two instances P_i and P'_j are partners if:

- one is from a server and one is from a client;

- both instances are in accepting state;
- they share the same entangled superposition $sid = (R, AK \oplus Sqn_C = AK \oplus Sqn_{Op,C})$.

Note that sid is a superposition included in the challenge transmitted by the server to the client. sid still contains information corresponding to the operator and the intended client as it is computed using their keys.

The sid are entangled values not only in themselves but also between the partnered instances as seen in the description of the quantum protocol. Then, deciding whether two instances are partnered can be done by xoring their sid . The result is 0 if they are effectively partnered and not 0 with overwhelming probability otherwise.

7.2.2 Targeted Notions

We here only give a short description of each security properties. Detailed definitions (together with proofs) are given in Section 7.3.

Indistinguishability of the output of the protocol: a man-in-the-middle (MitM) attacker should not be able to distinguish (with significative probability) the output of an execution of the protocol from random values, even with corruptions of other clients and servers (strong indistinguishability).

Resistance to client impersonation: a MitM attacker should not be able to make an uncorrupted server accept an execution (with significative probability) without relaying expected messages, even with corruptions of other clients and servers (strong resistance).

Resistance to server impersonation: a MitM attacker should not be able to make an uncorrupted client accept an execution (with significative probability) without relaying expected messages.

Soundness: a malicious server should not be able to make more authentication (with significative probability) with uncorrupted clients than what operators have allowed.

State-confidentiality: a malicious server should not be able to recover the values sk_{Op} , $sk_{Op,C}$, sk_C or Sqn_C significantly faster than guessing them.

Communications between Operators and Servers. Many elements that could easily break the security of the authentication are communicated between the operators and the servers. As such we consider the channels of communications between the operators and the servers to be secure.

7.2.3 Pseudo-random Function

As seen in Section 7.1, the protocol's security depends on the functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ implemented with either Milenage or TUAK. We need to assess how they impact the protocol security. For convenience, we denote $G = (f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*)_{sk_{Op,C}, sk_C}$.

$$G : (R, Sqn_{Op,C}, AMF) \mapsto (Mac_S, Mac_C, CK, IK, AK, Mac_S^*, AK^*)$$

Following Alagic, Broadbent, Fefferman, Gagliardini, Schaffner, Jules [3] and Zhandry [125], we estimate the protocol's security based on the security of G . As such, the latter is defined by the ability to distinguish a group of implementations of G from random functions with the property that only the outputs of f_1 and f_1^* depend on $Sqn_{Op,C}$ and AMF .

Quantum Advantage on a Set of Pseudo-random Functions. More precisely, we define the game for the quantum pseudo-randomness of G as follows.

1. The challenger generates random independent classical keys $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ to key the pseudo-random functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$, and $7n_{Op}n_C$ random functions.³ The challenger gives the attacker either the random functions or the pseudo-random ones as a quantum black box $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}$ such that $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus f_{i, sk_{C,j}, sk_{Op,C,j,j'}}(x)\rangle$.
2. The attacker can use the given function in quantum circuits.
3. The attacker guesses if the given black box is an implementation of the pseudo-random functions or random ones, and wins if is is right.

The pseudo-random function is $(q, t, n_{Op}, n_C, \epsilon)$ -indistinguishable if no attacker \mathcal{A} running in time t with q uses of any black box $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}$ has an advantage $\mathbf{Adv}(\mathcal{A}) = 2(\mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2})$ more than ϵ . We also define $\mathbf{Adv}(G)_{n_{Op}, n_C}^{q,t} = \sup_{\mathcal{A}} \{\mathbf{Adv}(\mathcal{A})\}$.

Note that our definition is similar to what is usually depicted as advantage on a secret-key quantum pseudo-function. The main difference is the reuse of sk_C with different $sk_{Op,C}$. This notion makes related-key attacks happen faster than the standard case having the full keys being random when the key difference is on $sk_{Op,C}$.

³This is the amount of generated functions since for each couple (C, Op) there is a generation of G .

7.2.4 Pseudo-random Key-derivation

Through corruption, one can reasonably expect some keys sk_C and $sk_{Op,C}$ to get leaked. Such an event can occur but should not reveal anything about any sk_{Op} or unlearned sk_C and $sk_{Op,C}$. As said in Section 7.1, the $sk_{Op,C}$ are obtained through a key derivation function KD . We define the advantage on KD against both of these issues.

Advantage on a Pseudo-random Key-derivation Function. We define the game about the pseudo-randomness of KD as a key-derivation function as follows.

1. The challenger generates random independent classical keys $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,1}, \dots, sk_{Op,n_{Op}}$. The challenger generates the derived keys $sk_{Op,C,i,j} = KD(sk_{C,i}, sk_{Op,j})$ for $i \leq n_C$ and $j \leq n_{Op}$. The challenger gives the attacker the client keys and either the derived keys or random values of the same length.
2. The attacker guesses if the given keys are generated with KD or random values. The attacker wins if the guess is right.

The pseudo-random key derivation is $(t, n_{Op}, n_C, \epsilon)$ -indistinguishable if no attacker \mathcal{A} running in time t has an advantage $\mathbf{Adv}(\mathcal{A}) = 2(\mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2})$ more than ϵ . We also define $\mathbf{Adv}(KD)_{n_{Op}, n_C}^t = \sup_{\mathcal{A}} \{\mathbf{Adv}(\mathcal{A})\}$.

Note that guessing a value sk_{Op} is an attack. For one operator, this game can be seen as a known-plaintext game for the pseudo-random function $sk_C \mapsto KD(sk_C, sk_{Op})$.

7.3 Quantum Security of the UMTS-AKA Protocol

In this section, we present detailed definitions of the security properties by means of games executed between a challenger and an attacker. Then, we prove the attacker's advantages on those games to be negligible with the use of secure primitives. In this section, we consider a generic secure pseudo random function G . As said above, it can be instantiated using either Milenage or TUAK. The exact quantum security of those two instances are studied in Sections 7.4 and 7.5 respectively.

7.3.1 Session Keys Indistinguishability

Freshness: session keys indistinguishability. An instance P_i is fresh if the adversary has not used the oracles *CorruptClient*, *CorruptServer* or *RevealSession* on P_i or on any of its partners.

7.3.1.1 Session Keys Indistinguishability Game.

We define the session keys indistinguishability game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target (a server or a client) and an operator.
3. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
4. The attacker uses the *Test* query on a fresh instance of the target.
5. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker (second phase).
6. The attacker guesses the secret bit b of the *Test* query. The attacker wins if the guess is right and the target instance is still fresh.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly key-indistinguishable if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server (outside an execution of the protocol) has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2}$ more than ϵ .

Theorem 7.1 (Session Keys Indistinguishability). *The advantage on the session keys indistinguishability game for the UMTS-AKA protocol is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{1}{2^{MacS}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{(q_{exec} + q_{Op})^2}{2^{|R|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof.

GAME \mathbb{G}_0 : this is the game of the definition.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbf{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : we define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: an attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD . To put it in another way, one can use an attack that is successful against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' . More formally, using the best adversary \mathcal{A}_0 for \mathbb{G}_0 , we can build an adversary \mathcal{A}_{KD} for the game against KD .

\mathcal{A}_{KD} will take the given values as keys sk_C and $sk_{Op,C}$ to generate the clients and operators. Then it will run the adversary \mathcal{A}_0 with the generated clients and operators then if \mathcal{A}_0 is right, \mathcal{A}_{KD} will answer “the keys are generated” and if \mathcal{A}_0 is wrong, \mathcal{A}_{KD} will answer “the keys are random”.

In the case the tuple of keys is the generated one, \mathcal{A}_0 works normally. If it is the random one \mathcal{A}_0 will work as an adversary for the game \mathbb{G}_1 . The probability of the described events is reported as follows:

\mathcal{A}_0	right	wrong
keys generated	$\frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t})}{2}$
random	$\leq \frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\geq \frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$

$$\mathbf{Adv}(KD)_{n_{Op}, n_C}^t \geq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

GAME \mathbb{G}_2 : we define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but we replace G with a perfect pseudo-random function G' on which there is no advantage.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: an attacker for \mathbb{G}_1 either attacks \mathbb{G}_2 or uses a property of G . To put it in another way, one can use an attack against \mathbb{G}_1 but fails against \mathbb{G}_2 to distinguish between G and G' . More formally, using the best adversary \mathcal{A}_1 for \mathbb{G}_1 , we can build a quantum adversary \mathcal{A}_G for the game against G with a total of $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests⁴ for keyed G as a black box.

\mathcal{A}_G will take the black boxes to generate the clients and operators. Then it will run the adversaries \mathcal{A}_1 with the generated clients and operators then if \mathcal{A}_1 is right, \mathcal{A}_G will answer “the functions are generated” and if \mathcal{A}_1 is wrong, \mathcal{A}_G will answer “the functions are random”.

In the case the functions are the generated one, \mathcal{A}_1 works normally. If they are the random one \mathcal{A}_1 will work as an adversary for the game \mathbb{G}_2 . The probability of the described events is reported as follows:

\mathcal{A}_1	right	wrong
functions generated	$\frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$
random	$\leq \frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\geq \frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})}{2}$

$$\mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} \geq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

⁴This is the number of queries needed to simulate the games. During an execution, the operator and the client each compute f_1, f_2, f_3, f_4 and f_5 . For a re-synchronization, the client and the operator each compute f_1^* and f_5^* . For a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_3 : we define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but there is only one client and one operator.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: as G' and KD' are perfect, the keyed G' are uncorrelated. Thus the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_3 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$ or $\mathbb{G}_1 \rightarrow \mathbb{G}_2$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, q_{Op}, t})$$

GAME \mathbb{G}_4 : we define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but there is only one server now.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: as servers are interchangeable in the sense that the values exchanged do not depend on the server, we essentially make the attacker unable to corrupt servers. Since corruption of servers only gives access to authentication vectors which would be given during an execution and reveal, the queries from a corrupt server are now counted as additional executions of the protocol: $q'_{exec} = q_{exec} + q_{Op}$.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t})$$

GAME \mathbb{G}_5 : we define the game \mathbb{G}_5 as the same as \mathbb{G}_4 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_4 \rightarrow \mathbb{G}_5$: with our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q'^2_{exec}}{2^{|R|}}$.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t}) &= \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t} \text{ and some } R^i \text{ are equal}) \\ &\quad + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t} \text{ and the } R^i \text{ are all different}) \end{aligned}$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t}) \leq \frac{(q'_{exec})^2}{2^{|R|}} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_5^{q'_{exec}, q_{res}, t})$$

ENDING: the attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. It has to distinguish between values generated by a perfect pseudo-random function from different R^i and true random values. To attack a client, the attacker can also try to send a different challenge which means guessing the output Mac_S of a perfect pseudo-random function with a value Sqn_C never seen before.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_5^{q'_{exec}, q_{res}, t}) \leq \frac{1}{2} + \frac{1}{2^{Mac_S}}$$

Then by recalling the previous transitions, we get:

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{1}{2^{Mac_S}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{(q_{exec} + q_{Op})^2}{2^{|R|}}$$

□

7.3.2 Client-impersonation Resistance

Freshness: client-impersonation resistance. An instance P_i is considered fresh if the adversary has not used the oracles *CorruptClient* or *CorruptServer* on P_i or on any of its partners.

Relay: client-impersonation. The attacker is considered to be a simple relay during an execution of the protocol between a server instance S_j and a client instance C_i if the following events happened in the following order:

1. use of the oracle $Send(S, j, m)$ where m is the *UID* of the client C , initializing the execution of the protocol;
2. use of the oracle $Send(C, i, m')$ where m' is a projection of the output of $Send(S, j, m)$;
3. use of the oracle $Send(S, j, m'')$ where m'' is a projection of the output of $Send(C, i, m')$.

7.3.2.1 Client-impersonation Resistance Game.

We define the client-impersonation resistance game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target (a server) and an operator.
3. The attacker is allowed to corrupt any number of clients and servers except the target.
4. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
5. The attacker wins if he successfully breaks the client-impersonation resistance for the target server, i.e., makes the target server instance accept, is still fresh and either the server instance is not partnered with the intended client, the server instance is partnered with a non intended client or the attacker has not been a simple relay.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly client-impersonation resistant if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 7.2 (Client-impersonation resistance). *The advantage on the client-impersonation resistance game for the UMTS-AKA protocol is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{(q_{exec} + q_{Op})^2}{2^{|R|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{q_{exec}}{2^{|Mac_C|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof.

GAME \mathbb{G}_0 : this is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbf{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : we define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: an attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section 7.3.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests⁵.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_2 : we define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: as G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

⁵See footnote 4.

GAME \mathbb{G}_3 : we define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but there is only one server now.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: as servers are interchangeable in the sense that the values exchanged do not depend on the server, we essentially make the attacker unable to corrupt servers. Since corruption of servers only gives access to authentication vectors which would be given during an execution and reveal, the queries from a corrupt server are now counted as additional executions of the protocol: $q'_{exec} = q_{exec} + q_{Op}$. For the next games, those additional executions will be marked as not valid for winning the game, $q_{valid} = q_{exec}$ is the number of valid executions for winning. The first executions are the marked ones and the valid ones are last, doing so does not remove generality.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t})$$

GAME \mathbb{G}_4 : we define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: with our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q'^2_{exec}}{2^{|R|}}$.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t}) &= \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t} \text{ and some } R^i \text{ are equal}) \\ &\quad + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t} \text{ and the } R^i \text{ are all different}) \end{aligned}$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t}) \leq \frac{(q'_{exec})^2}{2^{|R|}} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, q_{valid}, t})$$

ENDING: The attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. Since the uncorrupted server needs the first and the last message in that order to be able to accept the execution, the attacker cannot ask the answer to the client or it will be a simple relay and then has to guess values Mac_C generated by a perfect pseudo-random function with a R^i different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, q_{valid}, t}) = \frac{q_{valid}}{2^{|Mac_C|}}$$

Then by recalling the previous transitions, we get:

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{(q_{exec} + q_{Op})^2}{2^{|R|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{q_{exec}}{2^{|Mac_C|}}$$

□

7.3.3 Weak Server-impersonation Resistance

Freshness: server-impersonation resistance. An instance P_i is considered fresh if the adversary has not used the oracles *CorruptClient* or *CorruptServer* on P_i or on any of its partners.

Relay: server-impersonation. The attacker is considered to be a simple relay during an execution of the protocol between a server instance S_j and a client instance C_i if the following events happened in the following order:

1. use of the oracle $Send(S, j, m)$ where m is the *UID* of the client C , initializing the execution of the protocol;
2. use of the oracle $Send(C, i, m')$ where m' is a projection of the output of $Send(S, j, m)$.⁶

7.3.3.1 Weak Server-impersonation Resistance Game.

We define the weak server-impersonation resistance game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target (a client) and an operator.
3. The attacker is allowed to corrupt any number of clients except the target.
4. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
5. The attacker wins if he successfully breaks the server-impersonation resistance for the target client, i.e.s makes the target client instance accept, is still fresh and either the client instance is not partnered with the intended server, or is partnered with an other server, or the attacker has not been a simple relay.

The protocol is $(q_{exec}, q_{res}, t, \epsilon)$ -weakly server-impersonation resistant if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol and q_{res} re-synchronizations has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 7.3 (Weak server-impersonation resistance). *The advantage on the server-impersonation resistance game for the UMTS-AKA protocol is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{q_{exec}}{2^{|Mac_S|}} + \frac{q_{exec}^2}{2^{|R|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 10q_{exec} + 4q_{res}$.

⁶We do not consider the last message of the protocol in the client-impersonation game as it does not interfere with the client acceptance.

Proof.

GAME \mathbb{G}_0 : this is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, t}) = \mathbf{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : we define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: an attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section 7.3.1 with $q_{req} = 10q_{exec} + 4q_{res}$ requests ⁷.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) &\leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t \\ &\quad + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} \end{aligned}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_2 : we define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: as G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

GAME \mathbb{G}_3 : we define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: with our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q_{exec}^2}{2^{|R|}}$.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) &= \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t} \text{ and some } R^i \text{ are equal}) \\ &\quad + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t} \text{ and the } R^i \text{ are all different}) \end{aligned}$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) \leq \frac{(q_{exec})^2}{2^{|R|}} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, t})$$

⁷See footnote 4. There is no server corruption (i.e., $q_{Op} = 0$).

GAME \mathbb{G}_4 : we define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but there is only one server.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: as servers are incorruptible, they can only partner with a client that shares the same R they got from an operator. Then, as the R are all different, an unintended server will not partner.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, t})$$

ENDING: the attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. Since the uncorrupted client needs the second message to accept, the attacker cannot ask the answer to the server or it will be a simple relay and then has to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, q_G}) = \frac{q_{exec}}{2^{|Mac_S|}}$$

Then by recalling the previous transitions, we get:

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{q_{exec}}{2^{|Mac_S|}} + \frac{q_{exec}^2}{2^{|R|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

□

7.3.4 Soundness

7.3.4.1 Soundness Game

We define the soundness game as follows.

1. The challenger generates n_C clients and n_{Op} operators and their respective keys. Then the challenger generates q_{Op} authentication vectors and reveal them to the attacker.
2. The attacker is the server for executions of the protocol with the clients decided by the attacker.
3. The attacker wins if he makes $q_{Op} + 1$ executions of the protocol accept.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -server-sound if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol and q_{res} re-synchronizations has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 7.4 (Soundness). *The advantage on the soundness game for the UMTS-AKA protocol is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof.

GAME \mathbb{G}_0 : this is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbf{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : we define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: an attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section 7.3.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests ⁸.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client and one operator.

GAME \mathbb{G}_2 : we define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: as G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

ENDING: then the attacker has to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}}$$

Then by recalling the previous transitions, we get:

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}} + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

□

⁸See footnote 4.

7.3.5 State Confidentiality

7.3.5.1 Distinguishing Attack against State Confidentiality

During the process of establishing a proof of security for the state confidentiality, we discovered an attack which is able to distinguish the real value of Sqn_C from a random one by recovering at least the last bit of Sqn_C .

The process is the following.

1. The operator and client are synchronized.
2. The (malicious) server queries two following authentication vectors AV_1 and AV_2 ($Sqn^2 = Sqn^1 + 1$).
3. The client initiates an authentication request.
4. The server authenticates the client using AV_1 (as normal).
5. The server ends the conversation causing the client to request an authentication.
6. The server authenticates the client using AV_1 causing a re-synchronization procedure.
7. The server registers the value $AK^*(R_1) \oplus Sqn^1$.
8. A new authentication procedure is immediately started.
9. The server authenticates the client using AV_2 (as normal).
10. The server ends the conversation causing the client to request an authentication.
11. The server authenticates the client using AV_1 causing a re-synchronization procedure.
12. The server registers the value $AK^*(R_1) \oplus Sqn^2$.
13. From those two values, the server gets $Sqn^2 \oplus (Sqn^2 - 1) = AK^*(R_1) \oplus Sqn^1 \oplus AK^*(R_1) \oplus Sqn^2$. This new value is of the form $0 \dots 01 \dots 1$ where the number of 1 is one more than the 2-adic valuation of Sqn^2 . Then, if there is exactly one “1”, the last bit of Sqn^2 is a “1”, otherwise it is a “0”.

Freshness: state confidentiality. An entity P is fresh if the adversary has not used the oracles *CorruptClient* or *RevealState* on any instance of P .

7.3.5.2 State Confidentiality Game

We define the state confidentiality game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target (a client) and an operator.
3. The attacker corrupts any number of clients except the target.
4. The attacker is the server for executions of the protocol with the clients decided by the attacker.
5. The attacker sends a tuple $(sk_C, sk_{Op,C}, sk_{Op}, n, Sqn_{C,n})$, wins if he guesses right any of the values sk_C , $sk_{Op,C}$, sk_{Op} or $Sqn_{C,t}$ which is the value Sqn_C at the end of the execution n of the target client, and the client is still fresh.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -state-confidential if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 7.5 (State confidentiality). *The advantage on the state confidentiality game for the UMTS-AKA protocol is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof.

GAME \mathbb{G}_0 : this is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbf{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : we define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: an attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section 7.3.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests⁹.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

⁹See footnote 4.

Our next step is to reduce the configuration to one client and one operator.

GAME \mathbb{G}_2 : we define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: as G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

ENDING: the attacker can only win by sending the correct value for $sk_C, sk_{Op,C}$ or sk_{Op} or getting a right couple $(n, Sqn_{C,n})$.

For obtaining the secret keys, since G' and KD' are perfect, the attacker can only guess them.

Since the attacker is the server, it controls and knows the differences $Sqn_{C,n+1} - Sqn_{C,n}$ of the value Sqn_C at times n and n' (which by design is between 1 and Δ).

From executions and synchronizations, the attacker only gets $Sqn_{Op,C,n} \oplus AK^n$ and $Sqn_{C,n} \oplus AK^{*n}$. Since G' and KD' are perfect, the attacker can only learn $Sqn_{C,n} \oplus Sqn_{C,n'}$ when R^n is the same as $R^{n'}$ (which can happen since the attacker can use an old authentication vector and get the result from the synchronization procedure).

The attacker then can learn the value of all the bits of $Sqn_{C,n}$ that changes throughout the game. For example, $Sqn_C \oplus (Sqn_C + 1)$ is of the form $0 \dots 01 \dots 1$ where the number of 1 is one more than the 2-adic valuation of $Sqn_C + 1$ (the details of this property change with the value of the difference between the successive values but the principle remains the same). As we expect around $\log(\Delta q_{exec}) + 3$ bits to be affected, the probability for the attacker of guessing a good couple (n, Sqn_C) is at most $\frac{8\Delta q_{exec}}{2^{|Sqn_C|}}$.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) \leq \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

Then by recalling the previous transitions, we get:

$$\mathbf{Adv}(\mathcal{A}) \leq \mathbf{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t + \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

□

In the above security analysis, we have idealized the core pseudo-random function and our purpose in the sequel is to study the quantum security of the two standardized implementations: Milenage and TUAK.

7.4 Quantum Security of Milenage

In the last section, we proved security properties of the UMTS-AKA protocol assuming the security of the primitives underneath. We resolve this point in this section, discussing the quantum security of Milenage, and the next one, TUAK.

7.4.1 Description of Milenage

Milenage is a construction based on a block cipher noted E that acts on 128-bit messages (AES).

Key Derivation. The keys $sk_{Op,C}$ are computed as $sk_{Op,C} = E_{sk_C}(sk_{Op}) \oplus sk_{Op}$.

Functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$. From the input R (128 bits), Sqn (48 bits), AMF (16 bits), intermediate values T and I are computed as $T = E_{sk_C}(R \oplus sk_{Op,C})$ and $I = Sqn \parallel AMF \parallel Sqn \parallel AMF$.

Then five values are computed as follows:

- $out_1 = E_{sk_C}(T \oplus rot_{64}(I \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_2 = E_{sk_C}(c_2 \oplus rot_0(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_3 = E_{sk_C}(c_3 \oplus rot_{32}(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_4 = E_{sk_C}(c_4 \oplus rot_{64}(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_5 = E_{sk_C}(c_5 \oplus rot_{96}(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- f_1 outputs the first 64 bits of out_1 .
- f_1^* outputs the last 64 bits of out_1 .
- f_2 outputs the last 64 bits of out_2 .
- f_3 outputs out_3 .
- f_4 outputs out_4 .
- f_5 outputs the first 48 bits of out_2 .
- f_5^* outputs the first 48 bits of out_5 .

7.4.2 Attack on Milenage as a Pseudo-Random Function

We show an attack against Milenage as a pseudo-random function using the Grover-meets-Simon framework (Section 3.4.4.2).

7.4.2.1 Idea of the Attack

The attack relies on the following property of Milenage.

For any value R_0 and R_1 for R , let us consider $E(R_0 \oplus sk_{Op,C}) = A\|B$, $E(R_1 \oplus sk_{Op,C}) = C\|D$, if $A \oplus B = C \oplus D$ (probability 2^{-64} to happen), then for $x = SQN\|AMF$ and $b \in \{0, 1\}$, $F : b\|x \mapsto f_1(R_b, Sqn, AMF)$ has a period $1\|(A \oplus C)$. (This should not happen for a random function.)

We will fix R_0 and search for a R_1 such that F has a period.

(Then querying F on $0\|x$ will give the value of F on $1\|(x \oplus A \oplus C)$, which also breaks the security of f_1 as a MAC function.)

7.4.2.2 Algorithm of the Attack

We present a quantum attack on Milenage that uses 2^{57} computations (while searching the keys uses 2^{128} computations as there are two 128-bit keys to recover).

Algorithm 7.1 Quantum attack on Milenage

Input: *superposition* oracle access to G , or a random function

Output: “Milenage” or “Random”

- 1: Fix a value R_0
 - 2: **Grover search** on R_1 with $\frac{\pi}{4}2^{64/2}$ turns using the following oracle :
 - 3: Apply Simon’s algorithm on the function $F : b\|x \mapsto f_1(R_b, Sqn, AMF)$
 ▷ costs $(5 \times 64)^3 = 2^{25}$ computations per use
 - 4: **If** F is found to be periodic **then**
 - 5: the state (R_1) is “good” **Else** the state (R_1) is “bad”
 - 6: **EndGrover**
 - 7: Measure and check if a solution was found
 - 8: **If** it was found **return** “Milenage” **Else return** “Random”
-

7.4.3 Security Proof of Milenage in AKA

The attack we showed has some extensive use of adaptive superposition queries on Milenage. However, the AKA protocol severely limits the possibilities for such queries. Thus, the above attack can be seen as only theoretical since hard to be put in practice in UMTS-AKA. Considering that, we below define a new game for the security of the primitive that reflects those constraints. We then prove that Milenage is quantum-secure using that “practical in-AKA” game.

7.4.3.1 In-AKA-distinguishing Game

We define the G -in-AKA-distinguishing game:

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ for the studied pseudo-random

functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ and $7n_{Op}n_C$ random functions¹⁰. The challenger will use either the random functions or the pseudo-random ones as a quantum black box $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}$ such that $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus f_{i,sk_{C,j},sk_{Op,C,j,j'}}(x)\rangle$.

2. The challenger now simulates the AKA protocol with the chosen oracles.
3. The attacker can ask for executions of the protocol where he is a Man-In-the-Middle and has access to corrupted servers.
4. The attacker guesses whether the protocol is using G or random functions and wins if the guess is right.

A pseudo-random function is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly in AKA-indistinguishable if no attacker \mathcal{A} running in time t with q_{exec} instances of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server (outside an execution of the protocol) has an advantage $\mathbf{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2}$ more than ϵ .

From the structure of UMTS-AKA, an execution of the protocol uses only G twice: once for generating the authentication vectors and once to verify them. Note that in order to get an interesting answer from the client, the attacker needs to pass through the client verification with a different element. The verification of the server is not an oracle since it exclusively uses the authentication vector it is given, which the attacker has also access to). For simplicity, we give the attacker the values $out_1, out_2, out_3, out_4, out_5$ instead of $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ with no loss of generality (we actually give more to the attacker this way). We recall that Sqn and AMF are classical values. Then we can replace the previous game with the following (named \mathbb{G}_0) which is obviously exactly equivalent to the one above. There is no difference in winning one or the other game.

1. The challenger generates random independents keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ for the studied pseudo-random functions $out_1, out_2, out_3, out_4, out_5$ and $5n_{Op}n_C$ random functions. The challenger will use either the random functions or the pseudo-random ones as a quantum black box $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}$ such that $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus out_{i,sk_{C,j},sk_{Op,C,j,j'}}(x)\rangle$.
2. The challenger now gives the attacker authentication vectors and the associated answers (i.e., elements $\sum_R a_R |R, out_1, out_2, out_3, out_4, out_5\rangle$) and access to black-box verification functions F_1 and F_1^* that take quantum superpositions of states $|x, y\rangle$ and measures whether $y =$

¹⁰This is the number of generated functions since for each couple (C, Op) there is a generation of G .

$f_1(x, Sqn, AMF)$ and $y = f_1^*(x, Sqn, AMF)$ respectively and give the result on a classic bit.

3. The attacker is free to interact with those elements.
4. The attacker guesses whether the challenger is using G or random functions and wins if the guess is right.

The number of given authentication vectors is bounded by $q_{vect} = q_{exec} + q_{Op}$, the number of calls to F_1 is bounded by $q_{F_1} = q_{exec}$ and, the number of calls to F_1^* is bounded by $q_{F_1^*} = q_{res}$.

We now show that Milenage is G -in-AKA indistinguishable by proving the following theorem.

Theorem 7.6 (*G -in-AKA indistinguishability*). *The advantage on the G -in-AKA indistinguishability game for Milenage is bounded as follows:*

$$\mathbf{Adv}(\mathcal{A}) \leq \frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|\mathcal{R}|}} + \frac{50q_{vect}^2}{2^{|\mathcal{R}|}} + \frac{q_{F_1} + q_{F_1^*}}{|\mathit{Mac}_S|} + \mathbf{Adv}(E)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 6q_{vect} + 2q_{F_1} + 2q_{F_1^*}$.

The rest of the subsection is dedicated to proving this theorem.

Use of E . We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 except in Milenage we use a perfect quantum pseudo-random permutation E' instead of E and a perfect key derivation KD' instead of KD .

Lemma 7.6.1.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) + \mathbf{Adv}(E)_{n_{Op}, n_C}^{q_{req}, t} + \mathbf{Adv}(KD)_{n_{Op}, n_C}^t$$

The attacker either uses a property of E with at most $q_{req} = 6q_{vect} + 2q_{F_1} + 2q_{F_1^*}$ queries or it attacks \mathbb{G}_1 .

Use of Multiple Clients. As we now deal with a perfect block cipher, the different elements obtained through the different clients are unrelated. The total advantage is the sum of the advantages against each client. We show that for one client, the advantage is less than $\frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|\mathcal{R}|}} + \frac{50q_{vect}^2}{2^{|\mathcal{R}|}} + \frac{q_{F_1} + q_{F_1^*}}{|\mathit{Mac}_S|}$. The resources of the attacker (access to authentication vectors, calls to F_1 and F_1^*) are spread between the clients ($q_{vect} = \sum_{clients} q_{vect, client}$, $q_{F_1} = \sum_{clients} q_{F_1, client}$ and, $q_{F_1^*} = \sum_{clients} q_{F_1^*, client}$).

Then the total advantage is less than $\sum_{clients} \frac{24q_{vect,client}(q_{F_1,client} + q_{F_1^*,client})}{2^{|R|}} + \frac{50q_{vect,client}^2}{2^{|R|}} + \frac{q_{F_1,client} + q_{F_1^*,client}}{|Mac_S|}$, which is still less than $\frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{50q_{vect}^2}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|Mac_S|}$. We only consider the case of one client for the rest of the proof.

Use of Verification Functions. We define an intermediate game $\mathbb{G}_{2,R',Sqn,AMF}$ as the following:

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ for the studied pseudo-random function $out_1, out_2, out_3, out_4, out_5$ and $5n_{Op}n_C$ random functions. The challenger will use the pseudo-random ones as a quantum black box $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}$ such that $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus out_{i, sk_{C,j}, sk_{Op,C,j,j'}}(x)\rangle$.
2. The challenger now generates authentication vectors and the associated answers (i.e., elements $\sum_{R^i} a_{R^i} |R^i, out_1^i, out_2^i, out_3^i, out_4^i, out_5^i\rangle$) and measures whether any R^i equal R' and abort in this case
3. The challenger gives the attacker the authentication vectors and access to black-box verification functions F_1 and F_1^* that take quantum superpositions of states $|x, y\rangle$ and measures whether $y = f_1(x, Sqn, AMF)$ and $y = f_1^*(x, Sqn, AMF)$ respectively and gives the result on a classical bit.
4. The challenger gives the attacker a value $\sum_{R',h} a_{R',h} |R', h\rangle$ such that R' is measured to never be equal to a previous R^i .
5. The attacker is free to interact with those elements.
6. The attacker guesses the value of any part of $G(R', Sqn, AMF)$ ($f_1, f_1^*, out_2, out_3, out_4, out_5$) and wins if the guess is right.

Lemma 7.6.2. *For every R' ,*

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq \frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|Mac_S|}$$

Proof. As we want to consider all possible R' , the attacker uses F_1 or F_1^* either on an R' or on a previous R^i . However, the attacker gains nothing by querying F_1 or F_1^* on a previous R^i (the result is already known). Then we consider the attacker to only query F_1 or F_1^* on a R' . Then on a R' , either the result is “true” and we can declare the attacker to win, or the result is false and it continues. Either way, the measurement can be done at the end of the game

with the final guess. Then having q_{F_1} access to F_1 and $q_{F_1^*}$ access to F_1^* only multiplies the probability of success.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq q_{F_1} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, 1, 0, t}) + q_{F_1^*} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, 0, 1, t})$$

There are two cases, either one of the internal value of the computation matches one that appeared for one of the previous R^i , which happens with probability less than $\frac{12q_{vect}}{2^{|R|}}$,¹¹ or there is no collision and the attacker has to guess a random value such that there is no internal collision between R' and any R^i . \square

Note that there is no gain to take more R' or a smaller part. In the first case, the attacker would have to spread the calls to F_1 or F_1^* between the different values R' to attack. In the second case, F_1 and F_1^* would only give a partial answer and the advantage of the attacker compared to a perfect G would not grow.

Randomness of the Initial Vectors. We showed that vectors outside the initial ones cannot be used efficiently. We now focus on the initial vectors. The verification function are trivial on those ones, and we can then dispose of them. We define the game \mathbb{G}_3 as distinguishing the authentication vectors given to the attacker from vectors generated by a perfect pseudo-random function.

Lemma 7.6.3.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{vect}, t}) \leq \frac{1}{2} + \frac{50q_{vect}^2}{2^{|R|}}$$

Proof. There are two cases, either one of the internal value of the computation matches one that appeared for one of the previous R^i or there is no collision. The first one happens with probability less than $\frac{25q_{vect}^2}{2^{|R|}}$. The second one means the attacker has only the fact that there is no internal collision between the computations of $out_1, out_2, out_3, out_4, out_5$ (which gives the same advantage). \square

Note on AES. AES has been studied in the classical case for over 20 years. While the same cannot be said for the quantum case, the work of Bonnetain, Naya-Plasencia and Schrottenloher [27] seems convincing that AES can be considered quantum-safe for now.

Security of the Key Derivation. As stated in 7.2.4, the model for evaluating the key derivation is a known-plaintext model on AES. It can be considered safe.

¹¹There is two values for R' as only Out_1 is computed

Note on the Advantage on E . The use of fixed superposition may mislead us to think only the classical security of E is needed. However, an example for a quantum attack happening is the one-time pad distinguisher (Section 3.4.1). This distinguisher happens with only one query on $H|0\rangle$ against a one-time pad (where a classical distinguisher would need at least two queries) as G is a kind of one-time pad when E is a one-time pad.

7.5 Quantum Security of TUAk

7.5.1 Description of TUAk.

In this section we discuss the security of TUAk, which is based on a permutation noted P (in practice, it is Keccak-f[1600] [112]). The following description concerns the highest size of elements (keys and MACs are 256-bit long). The lower sizes only have some tweaks on the *INSTANCE* value and only output a subset of their higher size counterparts. We note $rev(M)$ the reversion of M , $a * M$ to be the message M repeated a times and *ALGONAME* to be the 56-bit value of the ASCII representation of “TUAk1.0”.

Key Derivation. *INSTANCE* is set to 00000001,
 IN is $rev(sk_{Op})||rev(INSTANCE)||rev(ALGONAME)||192 * (0)||$
 $rev(sk_C)||5 * (1)||314 * (0)||1||512 * (0)$,
 $sk_{Op,C}$ is defined as the reverse of the first reversed 256 bits of $P(IN)$.

Functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$.

For f_1 , *INSTANCE* is 00100001,
 IN is $rev(sk_{Op})||rev(INSTANCE)||rev(ALGONAME)||rev(R)||rev(AMF)||$
 $rev(Sqn)||rev(sk_C)||5 * (1)||314 * (0)||1||512 * (0)$,
 f_1 outputs the reverse of the first reversed 256 bits of $P(IN)$.

For f_1^* , *INSTANCE* is 10100001,
 IN is $rev(sk_{Op})||rev(INSTANCE)||rev(ALGONAME)||rev(R)||rev(AMF)||$
 $rev(Sqn)||rev(sk_C)||5 * (1)||314 * (0)||1||512 * (0)$,
 f_1^* outputs the reverse of the first reversed 256 bits of $P(IN)$.

For f_2, f_3, f_4, f_5 , *INSTANCE* is set to 01100111,
 IN is $rev(sk_{Op})||rev(INSTANCE)||rev(ALGONAME)||rev(R)||64 * (0)||$
 $rev(sk_C)||5 * (1)||314 * (0)||1||512 * (0)$,
 f_2 outputs the reverse of the first 256 bits of $P(IN)$,
 f_3 outputs the reverse of the bits 256 to 511 of $P(IN)$,
 f_4 outputs the reverse of the bits 512 to 767 of $P(IN)$,
 f_5 outputs the reverse of the bits 768 to 815 of $P(IN)$.

For f_5^* , *INSTANCE* is set to 11000001,
 IN is $rev(sk_{Op})||rev(INSTANCE)||rev(ALGONAME)||rev(R)||64 * (0)||$
 $rev(sk_C)||5 * (1)||314 * (0)||1||512 * (0)$,
 f_5^* outputs the reverse of the bits 768 to 815 of $P(IN)$.

7.5.2 Security Arguments of TUAk.

From a security perspective, we look at P as a pseudo-random function taking on input the key $sk_{Op,C}$ on positions 0 to 255 and the key sk_C on positions 512 to 767. Any attack on P would be considered as an attack against the permutation.

With this mindset, we can obviously reduce the security of TUAk to the one of P with at most $4q$ queries.

Note on Keccak-f. We now have to give some evidence that the underlying permutation, namely Keccak-f, can be considered as quantum-secure. As Keccak-f[1600] is a permutation on 1600 bits, it means that the best general attack as a qPRF is to try and fail at finding a collision which takes at least 2^{533} computations to do [127] compared to the 2^{256} computations to brute-force the keys. Keccak-f has been studied in the classical case since the NIST SHA-3 competition in 2008. While the same cannot be said for the quantum case, the fact that no practical classical attack breaks more than 9 rounds out of 24 [113] (the best classical attack on the full permutation to our knowledge uses 2^{1573} computations [122]) seems convincing that Keccak-f[1600] can be considered quantum-safe for now.

7.6 Conclusion

The UMTS-AKA protocol was designed in 1999 and intended to a classical world. With the advent of quantum computers, that multiple experts consider as imminent, it is necessary to take a new look at such protocols deployed in real life, which are used to protect the communication of billions of users. Assessing their security anew is therefore implied by the coming quantum era.

In this paper we have focused our attention on the UMTS authenticated key agreement, designed for the 3G telecommunication technology, and still at the basis of both 4G and 5G standards. This protocol is used every day by numerous users to protect their voice and data mobile communications.

We have defined a quantum version of the genuine AKA protocol. Starting from the work of Alt et al. we have derived a stronger quantum model which grants the adversary quantum computations as well as superposition queries. Then we have provided detailed security proofs of the quantum UMTS-AKA, showing that the quantum security of the protocol relies upon that of the underlying pseudo-random functions (f_1, \dots, f_5^*) . Therefore, under the assumption that they are quantum-secure, the UMTA-AKA remains a secure scheme to protect users' communications. To the best of our knowledge this paper provides the first rigorous proof of the UMTS-AKA in a strong quantum setting.

As supplementary contributions, we also exhibit a new attack against the state confidentiality of a mobile user. This attack holds in the quantum as

well as in the classical setting. We also describe a quantum existential forgery attack against the standalone f_1 function when instantiated with Milenage. We analyzed the quantum security of the underlying primitives of UMTS-AKA, Milenage and TUAK, and conclude that, if keyed with a 256-bit key, they are quantum secure as core functions of the UMTS-AKA protocol.

As a next work we aim at studying in a quantum setting the new properties of user's privacy and network's "awareness" that are provided in the 5G technology. With in mind the threat posed by quantum computing, the goal of the NIST's post-quantum competition is to motivate the design of quantum-secure asymmetric schemes. Likewise we hope that our work will contribute to the systematic analysis of the currently deployed protocols with regard to the quantum threat they will soon face, and help design secure quantum networks.

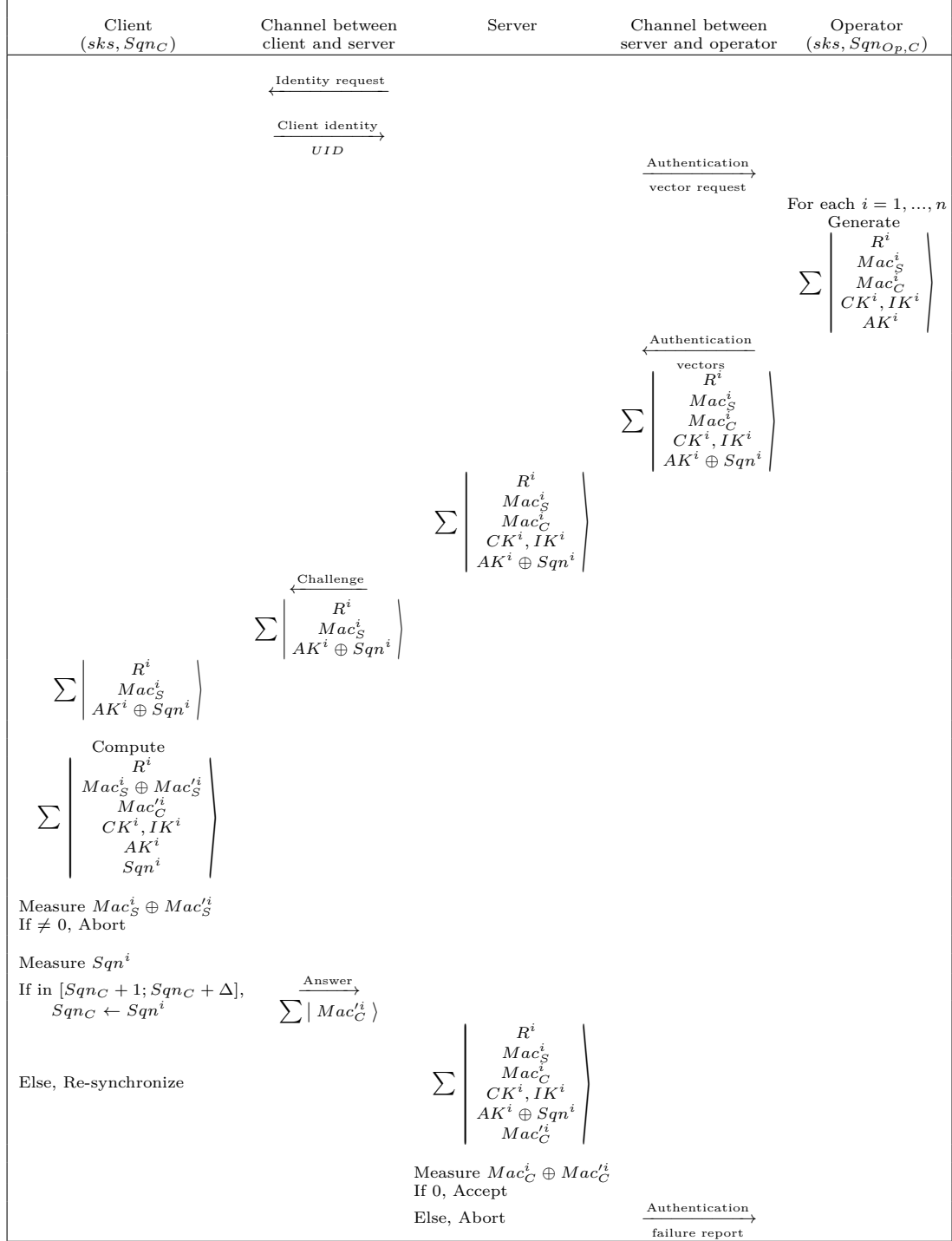


Figure 7.2: Quantum version of the AKA protocol

Chapter 8

Conclusion

In this conclusion, we recap the work done in those three years of thesis and give perspectives and open problems.

Contents

8.1	Summary of Results	195
8.2	Open Problems	196

8.1 Summary of Results

The contributions made in this thesis are quite diverse.

Quantum Algorithm. We developed the first instance of an offline Kuperberg algorithm i.e., a quantum algorithm that from classical queries, recovers a hidden shift using Kuperberg algorithm. We applied it to the Shifted Legendre Symbol problem successfully. This method is interesting as it rivals asymptotically the table-based collision search, which needs to use QRAM, a supplementary assumption. It is also the only known algorithm of this kind besides is the Offline-Simon algorithm.

Quantum Cryptanalysis. Our contribution in this domain is twofold.

Quantization of Classical Attacks. We proposed an efficient quantum boomerang attack, as well as a variant for mixing boomerang attacks. We proposed several improvements for different cases, as reducing the quantum RAM needed or making $Q2$ attacks work in $Q1$ under certain circumstances. In some cases, our attacks reach a quadratic speedup with respect to classical attacks. This shows that boomerang attacks are also performant cryptanalysis tools for the post-quantum world, that will be needed for correctly determining the best security margins.

New Quantum Attack. We presented a new kind of Simon-based attack on the EME construction. While a period does not appear in

the different functions, which was the focus of the previous Simon-based attacks, we found a periodicity in the set of collisions, and made a key-recovery attack.

New Construction. We proposed a new construction QuEME for doubling key sizes and state sizes.

(Quantum) Security Proofs. Our contribution in this domain is threefold.

- We obtained a IND-CCA security proof for our new construction that allows an (classical) attacker to use enough queries to hold a significant part of the sub-permutation codebook, a bound not common to achieve.
- We managed to make the first security proof against a quantum attacker which has access to encryptions and decryptions, making QuEME the first quantum-proof construction for block ciphers.
- We proved the security of Milenage and TUAk as primitives of a quantum UMTS-AKA.

Security Proofs for Quantum Protocols. We proposed a quantum version of the UMTS-AKA protocol and prove its security. To the best of our knowledge, this is the first rigorous proof of security of this kind of protocols in a full-quantum setting.

8.2 Open Problems

We end by proposing some open problems.

New Quantum Algorithms. Let us address the elephant in the room. While this thesis does not show any new quantum algorithmic brick, it certainly does not rule out the possibility of a new one changing the landscape of quantum cryptanalysis in a few years.

Even without inventing a such devastating algorithm, progress in known quantum algorithms is not out of the mind. An example may be Kuperberg's algorithm, which has seen many iterations of improvement since its discovery.

Quantum Cryptanalysis. Speaking of quantum cryptanalysis, there is still a lot to discover on hidden shift algorithm and their applications. For example, we exposed in this thesis a new way to use Simon's algorithm with substructures and broadening the type of properties that can be exploited. This technique was developed for the case of the Encrypt-Mix-Encrypt construction with the collision set. It could be useful to look for other structures (leading to more properties to be exploited) and other constructions (making it a more general approach).

“Offline” Quantum Algorithms. While the current “offline” quantum attacks require the knowledge of the intricacies of the original attack to realize, these original attacks are a direct application of Simon’s or Kuperberg’s algorithm. It would be interesting to look at other quantum attacks relying on those algorithms.

Another approach is to look for other quantum algorithms to make offline algorithms. A possibility could be lying with the linearization attacks.

Quantum Tools for Security Proofs. While the field of security proofs against quantum adversaries has evolved to adapt many tools from the security proofs against classical adversaries, many are still missing. The most immediate step is to make a “recording” oracle for random permutations on the base of what exists for random functions.

Quantum Protocols. Our work on the UMTS-AKA protocol is a beginning for quantum protocols relying on the security of smaller primitives. A possibility of continuation is to look for other properties, like privacy claimed for the 5G version of the UMTS-AKA protocol. Another possibility is looking for other protocols.

Bibliography

- [1] *3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f_1 , f_1^* , f_2 , f_3 , f_4 , f_5 and f_5^* ; Document 2: Algorithm specification*. Rev. 16.0.0. 3GPP. 1999.
- [2] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification*. Rev. 16.0.0. 3GPP. 1999.
- [3] Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardini, Christian Schaffner, and Michael St. Jules. “Computational Security of Quantum Encryption”. In: *Information Theoretic Security – 9th International Conference, ICITS 2016*.
- [4] Stéphanie Alt, Pierre-Alain Fouque, Gilles Macario-Rat, Cristina Onete, and Benjamin Richard. “A Cryptographic Analysis of UMTS/LTE AKA”. In: *Applied Cryptography and Network Security – 14th International Conference, ACNS 2016*.
- [5] Andris Ambainis. “Quantum walk algorithm for element distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239.
- [6] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. “PHOTON-beetle authenticated encryption and hash family”. In: *NIST Lightweight Compet. Round 1* (2019), p. 115.
- [7] Zhenzhen Bao, Jian Guo, and Eik List. “Extended Truncated-differential Distinguishers on Round-reduced AES”. In: *IACR Trans. Symmetric Cryptol.* (2020).
- [8] Achiya Bar-On, Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. “Improved Key Recovery Attacks on Reduced-Round AES with Practical Data and Memory Complexities”. In: *CRYPTO 2018, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*. 2018.

- [9] Kenneth E. Batchler. “Sorting Networks and Their Applications”. In: *AFIPS Spring Joint Computing Conference*. Vol. 32. AFIPS Conference Proceedings. Thomson Book Company, Washington D.C., 1968, pp. 307–314.
- [10] Robert Beals, Stephen Brierley, Oliver Gray, Aram W Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. “Efficient distributed quantum computing”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469.2153 (2013), p. 20120686.
- [11] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, and Alex Biryukov. “Schwaemm and Esch: lightweight authenticated encryption and hashing using the sparkle permutation family”. In: *NIST round 2* (2019).
- [12] Mihir Bellare, David Pointcheval, and Phillip Rogaway. “Authenticated Key Exchange Secure against Dictionary Attacks”. In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 139–155.
- [13] Mihir Bellare and Phillip Rogaway. “Entity Authentication and Key Distribution”. In: *Advances in Cryptology - CRYPTO ’93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Springer, 1993, pp. 232–249.
- [14] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. “Strengths and Weaknesses of Quantum Computing”. In: *SIAM J. Comput.* 26.5 (1997), pp. 1510–1523.
- [15] G Bertoni, J Daemen, M Peeters, and G Van Assche. *The Keccak reference. SHA-3 competition (round 3), 2011*.
- [16] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. “Farfalle: parallel permutation-based cryptography”. In: *IACR Transactions on Symmetric Cryptology* (2017), pp. 1–38.
- [17] Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. “Cryptanalysis of the Legendre PRF and Generalizations”. In: *IACR Trans. Symmetric Cryptol.* 2020.1 (2020), pp. 313–330.
- [18] Ward Beullens and Cyprien Delpèch de Saint Guilhem. “LegRoast: Efficient Post-quantum Signatures from the Legendre PRF”. In: *PQCrypto*. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 130–150.

- [19] Eli Biham, Orr Dunkelman, and Nathan Keller. “A Related-Key Rectangle Attack on the Full KASUMI”. In: *ASIACRYPT*. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 443–461.
- [20] Eli Biham and Adi Shamir. *Differential cryptanalysis of the data encryption standard*. Springer Science & Business Media, 2012.
- [21] Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. “Cryptanalysis of SAFER++”. In: *CRYPTO*. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 195–211.
- [22] Alex Biryukov and Dmitry Khovratovich. “Related-Key Cryptanalysis of the Full AES-192 and AES-256”. In: *ASIACRYPT*. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 1–18.
- [23] Alex Biryukov and Ivica Nikolic. “Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others”. In: *Advances in Cryptology - EUROCRYPT 2010, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*. 2010.
- [24] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. “Quantum Attacks Without Superposition Queries: The Offline Simon’s Algorithm”. In: *ASIACRYPT (1)*. Vol. 11921. Lecture Notes in Computer Science. Springer, 2019, pp. 552–583.
- [25] Xavier Bonnetain, Gaëtan Leurent, María Naya-Plasencia, and André Schrottenloher. “Quantum linearization attacks”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2021, pp. 422–452.
- [26] Xavier Bonnetain and María Naya-Plasencia. “Hidden Shift Quantum Cryptanalysis and Implications”. In: *ASIACRYPT (1)*. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 560–592.
- [27] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. “Quantum Security Analysis of AES”. In: *IACR Trans. Symmetric Cryptol.* 2019.2 (2019), pp. 55–93.
- [28] Xavier Bonnetain and André Schrottenloher. “Quantum Security Analysis of CSIDH”. In: *EUROCRYPT (2)*. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 493–522.
- [29] Xavier Bonnetain, André Schrottenloher, and Ferdinand Sibleyras. “Beyond quadratic speedups in quantum attacks on symmetric schemes”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 315–344.
- [30] Christina Boura, Virginie Lallemand, Valentin Suder, and María Naya-Plasencia. “Making the Impossible Possible”. In: *Journal of Cryptology* (2018).

- [31] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. “Quantum amplitude amplification and estimation”. In: *Contemporary Mathematics* 305 (2002), pp. 53–74.
- [32] Gilles Brassard, Peter Høyer, and Alain Tapp. “Quantum Cryptanalysis of Hash and Claw-Free Functions”. In: *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*. Ed. by Claudio L. Lucchesi and Arnaldo V. Moura. Vol. 1380. Lecture Notes in Computer Science. Springer, 1998, pp. 163–169.
- [33] Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. “Saturnin: a suite of lightweight symmetric algorithms for post-quantum security”. In: *IACR Trans. Symmetric Cryptol.* (2020).
- [34] André Chailloux, María Naya-Plasencia, and André Schrottenloher. “An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography”. In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10625. Lecture Notes in Computer Science. Springer, 2017, pp. 211–240.
- [35] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. “Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives”. In: *CCS*. ACM, 2017, pp. 1825–1842.
- [36] Richard Cleve and Harry Buhrman. “Substituting quantum entanglement for communication”. In: *Physical Review A* ().
- [37] Benoit Cogliati, Rodolphe Lampe, and Jacques Patarin. “The Indistinguishability of the XOR of k Permutations”. In: *Fast Software Encryption*. 2015.
- [38] Joan Daemen, Seth Hoeffert, Michaël Peeters, G Van Assche, and R Van Keer. “Xoodoo, a lightweight cryptographic scheme”. In: (2020).
- [39] Joan Daemen and Vincent Rijmen. *AES proposal: Rijndael*. Submission to NIST AES competition. 1999.
- [40] Joan Daemen and Vincent Rijmen. “Rijndael for AES”. In: *The Third Advanced Encryption Standard Candidate Conference, April 13-14, 2000, New York, New York, USA*. 2000.
- [41] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN: 3-540-42580-2.

- [42] Wim van Dam, Sean Hallgren, and Lawrence Ip. “Quantum Algorithms for Some Hidden Shift Problems”. In: *SIAM J. Comput.* 36.3 (2006), pp. 763–778.
- [43] Ivan Damgård. “On the Randomness of Legendre and Jacobi Sequences”. In: *CRYPTO*. Vol. 403. Lecture Notes in Computer Science. Springer, 1988, pp. 163–172.
- [44] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. “Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting”. In: *EUROCRYPT*. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 371–387.
- [45] David Deutsch and Richard Jozsa. “Rapid solution of problems by quantum computation”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.
- [46] W Diffie and ME Hellman. “New directions in cryptography. IEEE Information theory June 23-25 (1975)”. In: *IEEE International Symposium on Information Theory, Sweden*. 1976.
- [47] CE Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, BJM Mennink, Robert Primas, and Thomas Unterluggauer. “Isap v2.0”. In: (2020).
- [48] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schl affer. “Ascon v1. 2: Lightweight authenticated encryption and hashing”. In: *Journal of Cryptology* 34.3 (2021), pp. 1–42.
- [49] Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. “The Retracing Boomerang Attack”. In: *EUROCRYPT (1)*. Vol. 12105. Lecture Notes in Computer Science. Springer, 2020, pp. 280–309.
- [50] Orr Dunkelman, Nathan Keller, and Adi Shamir. “Improved Single-Key Attacks on 8-Round AES-192 and AES-256”. In: *ASIACRYPT 2010 Singapore, December 5-9, 2010. Proceedings*. 2010.
- [51] Orr Dunkelman, Nathan Keller, and Adi Shamir. “A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony”. In: vol. 27. 4. 2014, pp. 824–849.
- [52] Avijit Dutta, Mridul Nandi, and Abishanka Saha. “Proof of Mirror Theory for $\xi_{\max} = 2$ ”. In: *IEEE Transactions on Information Theory* (2022).
- [53] Shimon Even and Yishay Mansour. “A construction of a cipher from a single pseudorandom permutation”. In: *Journal of cryptology* 10.3 (1997), pp. 151–161.
- [54] Dankrad Feist. *Legendre pseudo-random function*. Accessed: 2021-01-19. 2019.

- [55] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David A. Wagner, and Doug Whiting. “Improved Cryptanalysis of Rijndael”. In: *FSE*. Vol. 1978. Lecture Notes in Computer Science. Springer, 2000, pp. 213–230.
- [56] Pierre-Alain Fouque, Jérémy Jean, and Thomas Peyrin. “Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128”. In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. 2013.
- [57] Paul Frixons, María Naya-Plasencia, and André Schrottenloher. “Quantum Boomerang Attacks and Some Applications”. In: *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*. Ed. by Riham AlTawy and Andreas Hülsing. Vol. 13203. Lecture Notes in Computer Science. Springer, 2021, pp. 332–352.
- [58] Paul Frixons and André Schrottenloher. “Quantum security of the legendre prf”. In: *Mathematical Cryptology* 1.2 (2021), pp. 52–69.
- [59] Michael Randolph Garey and David S Johnson. “A Guide to the Theory of NP-Completeness”. In: (1978).
- [60] Henri Gilbert. “A Simplified Representation of AES”. In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. 2014.
- [61] *Google wants to create a quantum computer with a million qubits.*
- [62] Lorenzo Grassi, Gregor Leander, Christian Rechberger, Cihangir Tezcan, and Friedrich Wiemer. “Weak-Key Distinguishers for AES”. In: *SAC 2020, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*. 2020.
- [63] Lorenzo Grassi and Christian Rechberger. “Revisiting Gilbert’s known-key distinguisher”. In: *Des. Codes Cryptogr.* (2020).
- [64] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. “MPC-Friendly Symmetric Key Primitives”. In: *CCS*. ACM, 2016, pp. 430–443.
- [65] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219.
- [66] Shai Halevi and Phillip Rogaway. “A Parallelizable Enciphering Mode”. In: *Topics in Cryptology – CT-RSA 2004*. 2004.

- [67] Akinori Hosoyamada and Tetsu Iwata. “4-Round Luby-Rackoff Construction is a qPRP”. In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. 2019.
- [68] Akinori Hosoyamada and Yu Sasaki. “Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2018, pp. 198–218.
- [69] *IBM plans a huge leap in superfast quantum computing by 2023*.
- [70] *IBM quantum roadmap*.
- [71] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. “Quantum chosen-ciphertext attacks against Feistel ciphers”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2019, pp. 391–411.
- [72] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. “Duel of the titans: the romulus and remus families of lightweight AEAD algorithms”. In: *IACR Transactions on Symmetric Cryptology* (2020), pp. 43–120.
- [73] Samuel Jaques and John M. Schanck. “Quantum Cryptanalysis in the RAM Model: Claw-Finding Attacks on SIKE”. In: *CRYPTO (1)*. Vol. 11692. Lecture Notes in Computer Science. Springer, 2019, pp. 32–61.
- [74] Novak Kaluderovic, Thorsten Kleinjung, and Dusan Kostic. “Improved key recovery on the Legendre PRF”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 98.
- [75] Novak Kaluderović, Thorsten Kleinjung, and Dušan Kostić. “Cryptanalysis of the generalised Legendre pseudorandom function”. In: *ANTS*. Vol. 4. Open Book Series 1. Mathematical Sciences Publishers, 2020, pp. 267–282.
- [76] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. “Breaking Symmetric Cryptosystems Using Quantum Period Finding”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Springer, 2016, pp. 207–237.
- [77] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. “Quantum Differential and Linear Cryptanalysis”. In: *IACR Trans. Symmetric Cryptol.* 2016.1 (2016), pp. 71–94.
- [78] Auguste Kerckhoffs. “La cryptographie militaire”. In: *Journal des sciences militaires* (1883), pp. 5–38.

- [79] Auguste Kerckhoffs. “La cryptographie militaire (part ii)”. In: *Journal des sciences militaires* 9.2 (1883), pp. 161–191.
- [80] Dmitry Khovratovich. “Key recovery attacks on the Legendre PRFs within the birthday bound”. In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 862.
- [81] Joe Kilian and Phillip Rogaway. “How to protect DES against exhaustive key search”. In: *Annual International Cryptology Conference*. Springer, 1996, pp. 252–267.
- [82] Arun Kumar Pati and Samuel L Braunstein. “Impossibility of deleting an unknown quantum state”. In: *Nature* 404.6774 (2000), pp. 164–165.
- [83] Greg Kuperberg. “A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem”. In: *SIAM J. Comput.* 35.1 (2005), pp. 170–188.
- [84] Hidenori Kuwakado and Masakatu Morii. “Quantum distinguisher between the 3-round Feistel cipher and the random permutation”. In: *ISIT*. IEEE, 2010, pp. 2682–2685.
- [85] Hidenori Kuwakado and Masakatu Morii. “Security on the quantum-type Even-Mansour cipher”. In: *2012 International Symposium on Information Theory and its Applications*. IEEE, 2012, pp. 312–316.
- [86] Gregor Leander and Alexander May. “Grover Meets Simon - Quantumly Attacking the FX-construction”. In: *ASIACRYPT (2)*. Vol. 10625. Lecture Notes in Computer Science. Springer, 2017, pp. 161–178.
- [87] Gaëtan Leurent and Clara Pernet. “New Representations of the AES Key Schedule”. In: *Advances in Cryptology - EUROCRYPT 2021 - Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*. 2021.
- [88] Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. “New Impossible Differential Attacks on AES”. In: *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*. 2008.
- [89] Michael Luby and Charles Rackoff. “How to construct pseudorandom permutations from pseudorandom functions”. In: *SIAM Journal on Computing* 17.2 (1988), pp. 373–386.
- [90] James L. Massey. “SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm”. In: *FSE*. Vol. 809. Lecture Notes in Computer Science. Springer, 1993, pp. 1–17.
- [91] James L. Massey, Gurgen H. Khachatrian, and Melsik K. Kuregian. *Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES)*. 1998.

- [92] James L. Massey, Gurgun H. Khachatryan, and Melsik K. Kuregian. *Nomination of SAFER++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE)*. 2000.
- [93] Mitsuru Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 386–397.
- [94] Mitsuru Matsui and Atsuhiro Yamagishi. “A New Method for Known Plaintext Attack of FEAL Cipher”. In: *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*. Ed. by Rainer A. Rueppel. Vol. 658. Lecture Notes in Computer Science. Springer, 1992, pp. 81–91.
- [95] Frank Miller. *Telegraphic code to insure privacy and secrecy in the transmission of telegrams*. CM Cornwell, 1882.
- [96] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard*. FIPS Publication 197. 2001.
- [97] Jacques Patarin. “Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security”. In: *Advances in Cryptology - CRYPTO 2003*. 2003.
- [98] Jacques Patarin. “A Proof of Security in $O(2n)$ for the Xor of Two Random Permutations”. In: *Information Theoretic Security*. 2008.
- [99] Jacques Patarin. “The “Coefficients H” Technique”. In: *Selected Areas in Cryptography*. 2009.
- [100] Jacques Patarin. *Security of balanced and unbalanced Feistel Schemes with Linear Non Equalities*. Cryptology ePrint Archive, Paper 2010/293. 2010.
- [101] John M Pollard. “Monte Carlo methods for index computation”. In: *Mathematics of computation* 32.143 (1978), pp. 918–924.
- [102] *Post Quantum Cryptography Standardization*.
- [103] FIPS Pub. “Data encryption standard (des)”. In: *FIPS PUB* (1999), pp. 46–3.
- [104] Mostafizar Rahman, Dhiman Saha, and Goutam Paul. “Boomeyong: Embedding Yoyo within Boomerang and its Applications to Key Recovery Attacks on AES and Pholkos”. In: *TOSC* (2021).
- [105] Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. “Yoyo Tricks with AES”. In: *ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*. 2017.

- [106] Martin Rötteler and Rainer Steinwandt. “A note on quantum related-key attacks”. In: *Inf. Process. Lett.* 115.1 (2015), pp. 40–44.
- [107] *RSA Is Dead — We Just Haven’t Accepted It Yet.*
- [108] István András Seres, Máté Horváth, and Péter Burcsi. “The Legendre Pseudorandom Function as a Multivariate Quadratic Cryptosystem: Security and Applications”. In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 182.
- [109] Claude E Shannon. “Communication theory of secrecy systems”. In: *The Bell system technical journal* 28.4 (1949), pp. 656–715.
- [110] Peter W. Shor. “Polynomial time algorithms for discrete logarithms and factoring on a quantum computer”. In: *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings.* Ed. by Leonard M. Adleman and Ming-Deh A. Huang. Vol. 877. Lecture Notes in Computer Science. Springer, 1994, p. 289.
- [111] Daniel R. Simon. “On the Power of Quantum Computation”. In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994.* IEEE Computer Society, 1994, pp. 116–123.
- [112] *Specification of the TUAk algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 1: Algorithm specification.* Rev. 16.0.0. 3GPP. 2015.
- [113] Sahiba Suryawanshi, Dhiman Saha, and Satyam Sachan. “New Results on the SymSum Distinguisher on Round-Reduced SHA3”. In: *International Conference on Cryptology in Africa.* Springer. 2020, pp. 132–151.
- [114] *The 2021 TLS Telemetry Report.*
- [115] Michael Tunstall. “Improved "Partial Sums"-based Square Attack on AES”. In: *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT.* 2012.
- [116] Dominique Unruh. *Compressed Permutation Oracles (And the Collision-Resistance of Sponge/SHA3).* Cryptology ePrint Archive, Report 2021/062. 2021.
- [117] Gilbert Sandford Vernam. “Secret signaling system, July 1919”. In: *US Patent 1* ().
- [118] David A. Wagner. “The Boomerang Attack”. In: *FSE.* Vol. 1636. Lecture Notes in Computer Science. Springer, 1999, pp. 156–170.
- [119] Ronald de Wolf. “Quantum Computing: Lecture Notes”. In: *CoRR* abs/1907.09415 (2019). arXiv: 1907.09415.

- [120] William K Wootters and Wojciech H Zurek. “A single quantum cannot be cloned”. In: *Nature* 299.5886 (1982), pp. 802–803.
- [121] Hongjun Wu and Tao Huang. “TinyJAMBU: A family of lightweight authenticated encryption algorithms (version 2)”. In: *Submission to the NIST Lightweight Cryptography Standardization Process* (2021).
- [122] Hailun Yan, Xuejia Lai, Lei Wang, Yu Yu, and Yiran Xing. “New zero-sum distinguishers on full 24-round Keccak-f using the division property”. In: *IET Inf. Secur.* ().
- [123] Andrew Chi-Chih Yao. “Quantum Circuit Complexity”. In: *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*.
- [124] Christof Zalka. “Grover’s quantum searching algorithm is optimal”. In: *Physical Review A* 60.4 (1999), p. 2746.
- [125] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *J. ACM* ().
- [126] Mark Zhandry. “How to construct quantum random functions”. In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 2012, pp. 679–687.
- [127] Mark Zhandry. “A Note on the Quantum Collision and Set Equality Problems”. In: *Quantum Info. Comput.* 15.7?8 (2015), 557?567. ISSN: 1533-7146.
- [128] Muxiang Zhang. “Provably-Secure Enhancement on 3GPP Authentication and Key Agreement Protocol”. In: *IACR Cryptol. ePrint Arch.* 2003 ().