Quantum information in security protocols

by

Sebastian Reynaldo Verschoor

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science (Quantum Information)

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Harry Buhrman Professor, Faculty of Science, Institute for Logic, Language and Computation University of Amsterdam
Supervisor:	Michele Mosca Professor, Department of Combinatorics and Optimization, University of Waterloo
Internal Member:	Ian Goldberg Professor, David R. Cheriton School of Computer Science, University of Waterloo
Internal-External Member:	Douglas Stebila Associate Professor, Department of Combinatorics and Optimization, University of Waterloo
Other Member(s):	John Watrous Professor, David R. Cheriton School of Computer Science, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Information security deals with the protection of our digital infrastructure. Achieving meaningful real-world security requires powerful cryptographic models that can give strong security guarantees and it requires accuracy of the model. Substantial engineering effort is required to ensure that a deployment meets the requirements imposed by the model.

Quantum information impacts the field of security in two major ways. First, it allows more efficient cryptanalysis of currently widely deployed systems. New "post-quantum" cryptographic algorithms are designed to be secure against quantum attacks, but do not require quantum technology to be implemented. Since post-quantum algorithms have different properties, substantial effort is required to integrate these in the existing infrastructure. Second, quantum cryptography leverages quantum-mechanical properties to build new cryptographic systems with potential advantages, however these require a more substantial overhaul of the infrastructure.

In this thesis I highlight the necessity of both the mathematical rigour and the engineering efforts that go into security protocols in the context of quantum information. This is done in three different contexts.

First, I analyze the impact of key exhaustion attacks against quantum key distribution, showing that they can lead to substantial loss of security. I also provide two mitigations that thwart such key exhaustion attacks by computationally bounded adversaries, without compromising the information theoretically secure properties of the protocol output. I give various security considerations for secure implementation of the mitigations.

Second, I consider how quantum adversaries can successfully attack quantum distance bounding protocols that had previously been claimed to be secure by informal reasoning. This highlights the need for mathematical rigour in the analysis of quantum adversaries.

Third, I propose a post-quantum replacement for the socialist millionaire protocol in secure messaging. The protocol prevents some of the usability problems that have been observed in other key authentication ceremonies. The post-quantum replacement utilizes techniques from private set intersection to build a protocol from primitives that have seen much scrutiny from the cryptographic community.

Acknowledgements

Thanks to Michele Mosca, for his continued support throughout this program, especially near the end as the mental burden during the lockdowns almost made my progress come to a standstill. Thanks to my committee, Harry Buhrman, Ian Goldberg, Douglas Stebila, and John Watrous, for taking the time to read and discuss my work. Their feedback has allowed many improvements to the quality of this thesis.

Thanks to all those that have helped me improve upon these chapters through helpful discussions and feedback. Thanks to all my colleagues at evolutionQ for their help with Chapter 2, in particular I want to thank James Godfrey, Thomas Parry, and Geovandro Pereira. I want to thank Ian Goldberg, Daan Leermakers, Boris Škorić, and John Watrous for helpful feedback on Chapter 3. I want to thank Douglas Stebila for helpful discussions, and Daniel Masny for helping me understand the nuances of the security model in Chapter 4.

Thanks to the staff from the University of Waterloo for their help and support throughout my graduate studies. Special thanks to Chin Heng Lee, Shannon Chung, the CS graduate office and the UW counselling services.

Thanks to all the friends that have made me feel at home at both the IQC and in Kitchener. Special thanks to Dan Allard, Júlia Amorós Binefa, Matthew Amy, Stefanie Beale, Nina Bindel, Olivia Di Matteo, Vlad Gheorghiu, Geovandro Pereira, Ramy Tannous, Sara Zafar Jafarzadeh.

Thanks to my family for their support. Thanks to my mother, José, for her support and for being a role-model of strength, no matter what life throws at you. Thanks to my father, Ton, for always listening with pride, even if he did not understand everything I told.

Thanks to Silke, for marrying me and joining me on this five year adventure across the ocean. Her friendship, patience, and love have given me the strength to complete this thesis.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AES Advanced Encryption Standard 20, 57, 86, 95

AKE authenticated key exchange 102, 103, 105, 115, 124, 140, 141, 148

CA certificate authority 103

CFRG Crypto Forum Research Group 19

CIA confidentiality, integrity and availability 9

DAKE deniable authenticated key exchange 105, 108

DDH decisional DH 130

DF distance fraud 65, 67, 76, 77, 86, 91, 92, 95–97, 99, 100

DH Diffie-Hellman xi, xiii, 7, 104, 106, 125, 129–131, 143, 144

DoS Denial-of-Service 4, 9, 11, 20, 23–25, 54

ECDH elliptic curve DH 104, 105, 131, 143, 148

EUF-CMA existential unforgeability under chosen message attack xv, 15, 50, 140

GNSS Global Navigation Satellite System 67

HBS hash-based signatures 11, 18, 19, 25, 51

HTTP Hypertext Transfer Protocol xiii

HTTPS HTTP/TLS 103

IRAD improved RAD 57, 58, 75, 76, 93, 97–100

ISO International Organisation for Standardization 1

ITI ITM instance 111–115

ITM interactive Turing machine xiv, 111, 112

ITS information theoretically secure iv, 3, 6, 9, 10, 12, 14, 16–18, 20–33, 38–45, 47, 48, 50–54, 149

KDF key derivation function 105, 131, 144

KEM key encapsulation mechanism 7, 104, 105, 120, 124, 125, 128, 129, 131, 142–144, 147, 150

KEX key exchange xi, 3, 4, 9, 11, 13, 50, 104, 106, 125–128, 131, 141, 145, 147, 148

MAC message authentication code 14–18, 24, 25, 45, 50, 59, 64, 65, 84, 87, 97

MF mafia fraud 62, 65–67, 69, 76, 77, 87, 91, 92, 96, 97, 99, 100

MPC secure multi-party computation 119, 120

NFC near-field communication 67

NIST National Institute for Standards and Technology 51, 104, 124, 145

OSI Open Systems Interconnection 4

OT oblivious transfer xi, xii, 7, 104, 105, 109, 119, 120, 124–137, 142–148, 150

OTR Off-the-Record Messaging xiv, 42, 102–107, 109, 115, 131, 140, 141, 145, 147, 174

OTRv4 OTR version 4 105, 131, 141, 143

PAKE password authenticated key exchange 105, 124, 140, 148

PEC private equality confirmation xi, 132, 140, 143, 147, 150

PET private equality test 104, 105, 109, 124, 131, 140, 147, 148

PGP Pretty Good Privacy 103, 105

PITM person-in-the-middle 13, 22, 24, 33, 94, 103, 131

PKE public key encryption 128, 129

PKI public key infrastructure 45, 103, 106, 120, 141

PPT probabilistic polynomial time 33, 34, 112, 113, 132, 133, 139

PRF pseudo-random function 63, 64, 76, 79, 86, 91, 95, 96, 124

PSI private set intersection iv, 7, 132

QKD quantum key distribution iv, 3, 6, 9–14, 17–21, 23–31, 33–35, 38–47, 50–52, 54, 55, 76, 149

QROM quantum ROM 119, 130, 131

RAD relay attack detection xiv, 57

RFID radio-frequency identification 56, 59, 63, 68, 100

ROM random oracle model xi, xv, 118, 119

RTT round trip time xi, 125, 126, 142, 148

SDP semidefinite program 74, 75

sEUF-CMA strong EUF-CMA 16, 50

SFE secure function evaluation 109, 119, 125, 131

SMP socialist millionaire protocol iv, xi, 7, 103, 104, 106–109, 147, 150

SUC simple UC 115–118, 127, 128, 131, 133, 134, 139, 147

TCP Transmission Control Protocol 4

TF terrorist fraud 6, 57, 58, 62, 63, 65, 66, 69, 76, 77, 85, 86, 89, 91, 92, 95, 97, 99, 100, 150

TLS Transport Layer Security xiii, 103

TOFU trust on first use 103

UC universal composability xi, xv, 7, 33, 104, 109–118, 120, 122–124, 128, 131, 139–141, 148

WOT web of trust 103, 106

XMSS extended Merkle signature scheme 19, 26

XOF extendable-output function 144, 145

Chapter 1

Introduction

Information security and cybersecurity are terms that appeal to the intuitive notions we have about the protection of the digital infrastructure which forms a cornerstone of our modern society. Although the above sentence captures the importance of the field, its imprecision and broadness make the terms almost meaningless. Despite a general lack of consensus about the precise meaning and role of security, many have attempted to pin down this intuition. For example, the International Organisation for Standardization (ISO) defines information security as the "preservation of confidentiality, integrity and availability of information" [ISO18], also know as the CIA triad. Any definition that is based on natural language will run into the problem that it appeals to the connotations held by the reader. To avoid such ambiguity, we generally turn to mathematics.

Much has been said about the role that mathematics can have in the context of information security (see the systematization of knowledge by Herley and van Oorschot [HO17] for an overview). Cryptography operates on mathematical abstractions of the real world to which we refer as the security model. Besides deductive reasoning within the model, security research should apply inductive reasoning to assess the validity of that model. Stated differently, an adversary might be able to compromise security by exploiting the gap that exists between the theory of cryptography and the reality of implementations in the form of assumptions in the model that are not met. Meaningful real-world security can only be achieved if meaningful security results can be derived in the model *and* the model is an accurate representation of reality.

Some examples make that concrete. There is the assumption of *secure local environments*: almost all cryptography operates under the assumption that devices of communicating parties are not completely compromised. Without this assumption it would be

difficult to give any guarantee of security, but in practice it is very hard to secure devices. Another complicating factor is provided by side-channel attacks [Koc96]: measurements of computations, such as the required amount of time or consumed power, can completely leak what is being computed upon. Models for protecting against side-channels are becoming increasingly sophisticated, but currently the protection is mostly provided by both software and hardware engineering efforts.

An example more relevant to the thesis is the following assumption about the physical environment. When you have your wireless payment card in your wallet, you do not want somebody else to be able to pay with that card at some other location. Standard cryptography can not help us here, instead we require *distance bounding protocols*: these ensure that the card will only work if it is in close proximity to the payment terminal.

Another gap between the theory and practice of security is that practical security has to deal with humans. Human beings make mistakes, but these should not lead to devastating loss of security. The easiest solution to this problem is to avoid human involvement in security critical processes when possible. However for the system to be usable, the user will have to be able to interact with it somehow.

Finally we consider the engineering challenge of securely implementing and deploying information systems. Complexity is the enemy of security: every mistake that exists within a security system has the potential to lead to real-world security loss and complex systems have more room for mistakes. Also, hardware needs to be maintained, software must be updated and everything must be integrated into a larger network. Designing protocols with few resource requirements and little room for (catastrophic) failure help to keep systems simple and thereby secure.

1 Quantum information

Quantum mechanical processes behave very differently from the classical phenomena that humans experience on a macroscopic level, which makes reasoning about those processes somewhat non-intuitive. Physical experiments confirm to a high degree of accuracy that the physical laws of quantum mechanics can be derived from some simple mathematical postulates. The implications of those laws turn out to be significant to cryptography.

Contemporary cryptography has seen significant impact from developments in quantum computation. Shor's algorithm [Sho94] threatens to break the security guarantees of widely deployed public-key (asymmetric) cryptographic primitives. The algorithm can efficiently factor large numbers and find discrete logarithms, which represents the bulk of widely

deployed cryptography. Grover's algorithm [Gro96] threatens to weaken symmetric cryptography, as it allows a quadratic speedup in search problems such as finding pre-images of a cryptographic hash function. Although both algorithms require large-scale fault-tolerant quantum computers that currently do not exist, their implications greatly accelerated the development of post-quantum cryptography, both in academia [BBD09; ABB+15; BL17] and in standardization efforts in the industry [NIST17; ETSI].

Post-quantum (also called quantum-resistant) cryptography has the goal of creating classical primitives that are quantum-secure: they resist the increase in computational power gained by quantum adversaries. Against Grover's algorithm it is sufficient to double the key sizes, but against Shor's algorithm it is necessary to deploy new asymmetric primitives that are based on other mathematical problems that are (conjectured to be) hard. The currently known post-quantum primitives differ from the pre-quantum ones in several different ways, ranging from slightly larger keys to the fact that certain useful operations on them are not possible. Directly replacing pre-quantum with post-quantum primitives is not always possible, posing challenges to designers of post-quantum protocols.

On the constructive side, a quantum channel can be used as a cryptographic primitive. Data on this channel has some interesting properties, such as the inability to be perfectly cloned [WZ82] or being measured undetected. This can be leveraged to build cryptographic protocols. Most notably quantum key distribution (QKD) [BB84] uses a quantum channel to expand a small shared key into an arbitrary sized key, as long as the effect of tampering and eavesdropping is below a certain threshold. QKD can provide information theoretically secure (ITS) guarantees about its output, meaning that it does not depend on some computational complexity assumption. However it should be noted that quantum cryptography is not exempt from the earlier mentioned relation between security models and the real world.

2 Security protocols

This thesis will focus on the security provided by cryptographic protocols. Protocols combine cryptographic primitives with interactivity to provide security guarantees. For example, a typical key exchange (KEX) sets up a secure shared key between two users that only know each other's public key. This can be used to encrypt messages back and forth, so that effectively a secure channel has been set up. Many tools and frameworks exist to support the design and analysis of cryptographic protocols, making this the highest layer that can still be analyzed with mathematical rigour.

2.1 Ceremonies

Ceremonies are an extension of network protocols [Ell07]. Whereas network protocols might consider some communication to be out-of-band, nothing is out-of-band for a ceremony, including interaction with humans. For example, a protocol can assume that a public key for the KEX is simply provided to the device initially, but a ceremony must specify how it gets there. Ceremony analysis can be a helpful tool in analyzing protocols: if a ceremony cannot be made secure or if it heavily relies on humans successfully completing a difficult task, the protocol may have to be changed accordingly.

2.2 Protocol aborts

I will consider protocol aborts in detail, as it will be relevant in multiple locations of the thesis.

Security protocols must deal with aborts: a party can always deviate from a protocol by halting. A complete formal specification of a security protocol must include a specification of how to handle aborts [Gol04, Section 7.2.3], but specifying such details can rapidly become tedious, so it is rarely made explicit in much of the literature. The method for aborting can be significant for the security of the protocol in subtle ways, as highlighted by this work, so in this thesis I argue that aborts and their handling should be made explicit.

An underlying assumption throughout this thesis is that classical (non-quantum) messages are delivered over a message *transport layer*. The purpose of such a layer is to preprocess incoming signals into bytestrings that can be delivered to the security protocol. This layer roughly corresponds to the Level 4 layer in the Open Systems Interconnection (OSI) network stack (think of the Transmission Control Protocol (TCP) layer on the internet), but with some specific assumptions. The message transport is assumed to be sufficiently robust against environmental noise, meaning that any non-malicious message alterations are delivered with low probability. Such mechanisms include error correcting codes, message acknowledgements, and even sending messages over redundant physical and/or logical channels.

In order to achieve the above mentioned robustness, the transport layer locally has to maintain some state, for example to reorder packets that arrive out of order. A program that allocates many resources of its executing device will become vulnerable to Denial-of-Service (DoS) attacks, where an adversary can initiate many connections in order to exhaust the resources of the system. This problem is exacerbated by protocols that require devices to maintain a large state on top of the state required for the transport layers and

underlying layers. Thus the message transport layer could ignore some invalid messages and optionally send resend requests, but eventually it may conclude that no valid message is incoming, at which point it delivers a special abort symbol ($\perp$). It is important to recognize that no implementation of the message transport layer can guarantee delivery of messages in the presence of a sufficiently powerful adversary and the impact of message delivery failure must be assessed accordingly.

If the transport layer does deliver a message, I distinguish two reasons why the cryptographic layer might abort. First, incoming messages could be invalid. Most models of protocols operate on messages in a certain domain $\mathcal{M}$, but implementations usually operate on bytestrings. Let $\mathcal{B} = \{0, \ldots, 255\}$ be the set of bytes. Then there should be an effective subroutine $P : \mathcal{B}^* \to \mathcal{M} \cup \{\perp\}$, which I will call the message *parser*. The parser P can return a special symbol $\perp$ as a parser error, indicating that the bytestring does not have a value in $\mathcal{M}$. When a parser operates on public values (directly on the full message received from the transport layer) a parser abort cannot leak any information, but there are subtle issues surrounding parsing that can lead to security failures. For example, if a secret local value determines which part of a message is parsed, parser aborts can leak that secret. Since parsing details differ per protocol, we consider parsing as part of the protocol and not as part of the message transport layer.

A second reason for aborting is that a local subroutine can fail. For example, the decoder in code-based cryptography can fail to decode some ciphertext. Since the subroutine outcome can depend on secret data, protocols should always ensure that none of that secret data is leaked through the act of aborting.

Note we have not yet specified *how* a protocol deals with aborts. There are two commonly used methods. The first is to replace abort symbols with a default value. This can simplify the protocol description and analysis, and it may simplify the application layer built on top of the security protocol. The second method is to abort the protocol itself, which includes extending protocol output domains with a dedicated abort output, and optionally extending all protocol message domains with a dedicated abort message.

When the cryptographic layer aborts, the application layer will have to deal with it, which may result in exposing the user to an error message. Since these aborts often indicate some sort of security failure, proper handling is required. Designing adequate ceremonies for this purpose is not an easy task, so where possible, exposing users to such security details is not advisable.

I will abuse notation and overload the $\perp$ symbol in the context of aborts. It represents protocol outputs after abort, abort protocol messages, and failed subroutine outputs, where the exact meaning should be clear from the context.

A final consideration regarding aborts is that real-world systems are not infallible. For example, loss of power or crashes can force system reboots with the consequence that any state stored in non-persistent memory is lost. Any protocol that was running at the time will implicitly have been aborted, but such aborts should not lead to significant loss of security. Any state that is essential for security should therefore be kept in non-volatile storage. Security-critical updates to this storage should be atomic, minimizing the probability that system failures result in partial updates that leave an invalid state. This probability may be large in the context of active side channel attacks: an adversary with some influence over system crashes might be able to lower security.

3 Contributions

Thesis statement

Information security in the context of quantum information has a strong dependency on mathematical definitions of security, yet sound engineering practices remain unavoidable in order to construct meaningfully secure cryptographic protocols.

This thesis has three main contributions to support the above thesis statement. Each contribution is presented in its own chapter.

Chapter 2 discusses key exhaustion in QKD, a problem that is known to exist, but that I believe has not been given sufficient attention. In the chapter, I investigate whether a protocol that is vulnerable to key exhaustion can provide meaningful security in real-world scenarios. I also present two countermeasures that combine computational and ITS cryptography: the result is a protocol with computational protection against key exhaustion, without any compromise to the ITS guarantees that QKD can provide.

Chapter 3 contains a cryptanalysis of (all three existing) quantum distance bounding protocols [AMSP17; Abi19; Abi20]. The protocols' structure is similar to that of classical distance bounding protocols, but the rapid phase exchanges qubits instead of classical bits. In this chapter, I show that the proposed countermeasure against terrorist fraud (TF) (a type of collaborative attack) is ineffective at best and can even leak the long-term key. Since such a countermeasure is not always required, I also analyze the protocol without it, finding that previous analyses were either flawed or overestimated the achievable security.

Chapter 4 provides a post-quantum replacement for the socialist millionaire protocol (SMP) [BST01], which is used for key authentication in secure messaging. The SMP depends on properties of the Diffie-Hellman (DH) primitive and cannot be directly replaced by current post-quantum primitives. I propose a protocol that is an adaptation of an existing private set intersection (PSI) solution [RR17]. It is built on top of oblivious transfer (OT), which can be built from post-quantum key encapsulation mechanisms (KEMs) [MR21]. Besides a reductionist security argument in the universal composability (UC) framework [Can01], I also discuss the design decisions I made to ensure the protocol could provide meaningful security.

3.1 Terminology

Throughout the thesis I comment on the confusing (and sometimes incorrect) terminology that I have found in the literature. This might be a little pedantic, but I believe that clear and non-ambiguous communication is essential in order to ensure that academic results can have a positive impact in the real world. The tools we propose should come with clear statements of both their capabilities and limitations, helping engineers to construct the right context for deploying them. I am concerned that presenting exaggerated results, whether unknowingly or purposefully, can damage the credibility of the field. Specifically in cryptography, where seemingly minute details can have a large impact, such confusion could lead to loss of meaningful security. Some might say that it already has [KM07].

4 Conventions

To enhance the clarity of protocol descriptions, I refer to the communicating parties as Alice (she/her) and Bob (he/him), whereas Mallory (she/her) is an (active) adversarial outsider party. This is an anthropomorphic description: the parties are (idealized) devices that act precisely according to the protocol specification. Where necessary, I distinguish *users* from their *devices*.

I will adhere to standard terminology and notation as much as possible. The notation $x \xleftarrow{\$} X$ represents sampling x according to X, where X is a probability distribution, a randomized subroutine, or a set (in which case x is sampled uniformly). The notation $x := X$ means that the value X is assigned to the variable x, distinguishing it from a test for equality where necessary.

An important concept in security is that of negligible functions, defined as follows.

Definition 1.1 (Negligible)**.** *A function* $\mu : \mathbb{N} \to \mathbb{R}^+$ *is* negligible in λ *if for every polynomial* p *there exists a* λ_0 *such that for all* $\lambda > \lambda_0$*:*

$$\mu(\lambda) < \frac{1}{p(\lambda)}. \tag{1.1}$$

The set of all functions that are negligible in λ *is denoted* $\mathrm{negl}(\lambda)$*.*

We follow a few conventions with respect to this definition. First, when a function is claimed to be negligible, it should be understood as being negligible *in* λ (the security parameter). Second, the phrase "for sufficiently large λ" should be interpreted as "there exists a λ_0 such that for all $\lambda > \lambda_0$". Third, when we state that some event X occurs *with overwhelming probability*, that means the probability that the event does not occur is negligible: $(1 - \Pr[X]) \in \mathrm{negl}(\lambda)$.

A function that does not satisfy Definition 1.1 is called *non-negligible*: $f(\lambda)$ is non-negligible if there exists a polynomial p such that for all λ_0 there exists a $\lambda > \lambda_0$ such that $f(\lambda) \geq 1/p(\lambda)$. By convention the phrase "for infinitely many λ" replaces "for all λ_0 there exists a $\lambda > \lambda_0$".

Chapter 2

Preventing key exhaustion in quantum key distribution

1 Introduction

Many attempts at defining information security list the CIA triad among the core concepts: confidentiality, integrity and availability. Whereas cryptography provides primitives and protocols for protecting confidentiality and integrity, it usually has an adverse effect on availability: cryptography requires time and other resources of the executing device, amplifying the effect of Denial-of-Service (DoS) attacks. In order to minimize the impact of cryptography on availability, primitives should be fast and have low resource requirements, while protocols should detect illegitimate behaviour early and abort in order to allow rapid continuation of legitimate communication.

Quantum key distribution (QKD) protocols establish fresh shared key material. When participants start with a small shared secret, for example a secret that was established through some uncompromised key exchange (KEX), then the confidentiality and integrity of the QKD output does not rely on any complexity assumptions, so it is information theoretically secure (ITS). This means that QKD can provide *everlasting security* [Unr13]: keys remain secret, even against adversaries that become computationally unbounded in the future. Despite this theoretical advantage, the NSA recently recommended against using QKD in general [NSA20], listing the increased risk of DoS as one of five technical limitations.

Any cryptographic system is vulnerable to DoS attacks, but a problem that is specific to QKD is that of key exhaustion. QKD requires communication over a public quantum

channel and an authenticated classical (non-quantum) channel. In practice the latter is built by adding cryptographic authentication to messages on a *public* classical channel. The integrity of the QKD output depends on that of the classical channel: ITS QKD requires ITS authentication. There exists classical cryptography that achieves this [WC81], but it comes at a price compared to computationally secure authentication: key material (or some of it) has to be discarded after it has been used. Since QKD outputs more bits than are used for authentication, the consumed key bits can be replenished from this output. However if an adversary can consistently force the QKD protocol to abort, after key material has been consumed but before providing new output, they can exhaust all key material.

This chapter presents two main contributions. First we consider key exhaustion on QKD protocols in greater detail than has been done before. The attack applies to all current QKD protocols and allows an adversary to completely exhaust the shared key material. Attacks on availability are often dismissed in the QKD literature, often with the argument that no cryptography can protect you from an adversary that simply cuts the wire. We show that the impact of the described attack is much more significant and undermines some of the security properties ascribed to QKD: it can even lead to loss of integrity and confidentiality of future sessions. Second we propose two mitigations that prevent key exhaustion in practice, by combining computationally secure authentication with ITS authentication. The resulting protocol provides computational protection against key exhaustion without compromising the confidentiality and integrity of the QKD output key material. We provide arguments for both the security and the necessity of this mitigation for any real-world deployment of a QKD protocol.

1.1 Outline

In Section 2 I review the necessary background for this chapter. Section 3 discusses key exhaustion in detail, describing its impact and the importance of preventing it. Sections 4 and 5 describe and analyze the two mitigations we propose, while Section 6 compares the two and suggests a method for combining them. A deployment of these mitigations is not necessarily secure, so Section 7 highlights some security considerations that apply. I conclude the chapter with a discussion in Section 8.

2 Background

In this section I consider the cryptography relevant to key exhaustion and our mitigations. Section 2.1 provides a brief description of QKD, where I note that a high level overview is

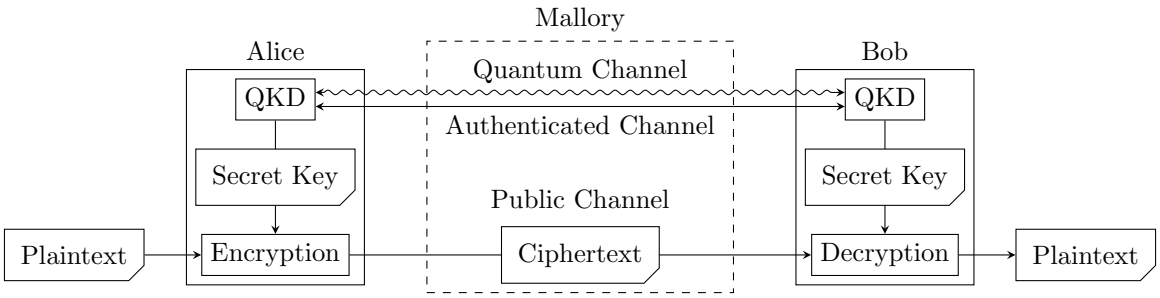


Figure 2.1: QKD with the application of message encryption.

sufficient for this chapter: no detailed description of the quantum channel or any quantum information is required. I then discuss computational authentication, focussing on symmetric cryptography in Section 2.2 and asymmetric cryptography (specifically: hash-based signatures (HBS)) in Section 2.3. Finally, Section 2.4 discusses a recent proposal that provides DoS protection to *computationally secure* QKD.

2.1 Quantum key distribution

In quantum key distribution (QKD), Alice and Bob aim to derive a fresh shared secret key. They will communicate over a public (non-confidential and non-authenticated) quantum channel and a public classical channel, which is authenticated with classical cryptography (see Section 2.2). I assume that the QKD protocol is "secure", meaning that the implementing devices satisfy the requirements imposed by the security model, such as having a local secure environment/laboratory. The guarantee of QKD is that if neither Alice nor Bob aborts, then the protocol outputs a shared secret bitstring to both of them. Just like the output of other KEX protocols, the resulting key can be used for arbitrary purposes, for example to encrypt a message as shown in Figure 2.1.

For the security of QKD we consider outsider adversaries. We distinguish passive adversaries that only eavesdrop on the communication from active adversaries that can also tamper with the communication. In the presence of a passive adversary the protocol will output a key[1] and that key is statistically indistinguishable from a random key. Note that eavesdropping on a quantum channel is generally impossible to do without altering the communication itself, so that is classified as an active attack. Under the assumption that active adversaries cannot tamper with messages over the classically authenticated channel

[1]We assume that the QKD protocol is robust against environmental noise.

undetected, all they can do is force the protocol to abort, but if neither party aborts, then the output is an ITS key. Stated differently, QKD can be as secure as the authentication of the classical channel.

2.1.1 Purpose of QKD

QKD is often dismissed by its critics, but this dismissal is not always justified. Part of the misunderstandings surrounding QKD are driven by misinterpretation of technical terms: this mistake is not only made by science journalists,[2] but also in the academic literature itself. Terminology in the context of quantum cryptography can be confusing, especially when standard technical terms from security research get a slightly different meaning.

It has often been said that QKD provides *unconditional* security [May01], which has often lead to false expectations about QKD [PPS07]. In this context the term unconditional has a narrow technical meaning, namely the indistinguishability of the key is not conditional on the computational intractability of some underlying mathematical problem. There are however several assumptions underlying the real-world security of any QKD deployment.

One assumption is the real-world security of the QKD hardware and software: adversaries should not be able to extract secret information from it and should not be able to influence its functionality. The research area of *quantum hacking* [JSK+16] provides many attacks on QKD deployments that fail to meet this assumption. Some protection can be provided by so-called device-independent QKD [PAB+07]:

> It is therefore "device independent" in the sense that it needs no knowledge of the way the QKD devices work, provided quantum physics is correct and provided Alice and Bob do not allow any unwanted signal to escape from their laboratories.

The last part of this quote makes it clear that the protocol is secure *assuming* that there are no *other* side-channels. Preventing all side-channels may be extremely difficult to realize [Rud02; Ber18], although theoretically not entirely impossible [RR20].

Another assumption for security of QKD is that Alice and Bob communicate directly over a quantum channel. Current implementations of QKD are limited in their distance and therefore require trusted relays [SPD+10]. Trusted relays only provides security against adversaries that do not control (some of) those relays, undermining the end-to-end principle

[2] Searching online for the term "unhackable internet" provides some insight into how wide this is spread.

of modern encryption standards. Quantum repeaters [BDCZ98] can extend the quantum channel so that this is no longer an issue, but further engineering efforts are required before these repeaters can be deployed in practice.

Despite the above *practical* issues, there is an advantage that a secure implementation of QKD can provide over computational cryptography, namely *everlasting security* [Unr13]. This is a powerful property that protects against harvest-and-decrypt attacks. Since the QKD output is effectively uniformly random and independent of earlier key material, an adversary that recorded the conversation cannot guess the key even if they become computationally unbounded after the conversation happened. In contrast, output from a computationally secure KEX can always get compromised by an adversary that becomes unbounded later (or more realistically: one that breaks the computational assumption later).

Everlasting security is sometimes mistakenly called forward secrecy,[3] but the former protects against cryptanalytic advances while the latter protects against *device* compromise. Forward secrecy *allows* users to erase keys from their devices after they have been used. A device compromise does not leak old message contents, if at least both the key, the plaintext message itself, and all other information that could lead to recovery were removed from the device. Unless device compromise is not considered a realistic threat, everlasting security is only meaningful if the QKD output is used in a forward secure system and all local data is completely removed afterwards on both endpoints.

2.1.2 Phases of QKD

All known QKD protocols consist of three phases. The first phase describes the communication over the public quantum channel. It is assumed that Mallory, the adversary, has full control over this channel: she can intercept, resend, alter, inject or block any message on the channel. However Mallory has to follow the laws of quantum mechanics, for example the no-cloning-theorem applies so that she cannot perfectly clone qubits. Note that we are not distinguishing prepare-and-measure protocols (in which Alice sends qubits to Bob [BB84]) from entanglement-based protocols (in which Alice and Bob receive entangled qubits from an untrusted third party [Eke91]). At the end of this phase, Alice and Bob measure the quantum states, so that they both have some (classical) key material. However initially there are few guarantees about this material, for example they may have measured in incompatible bases or a person-in-the-middle (PITM) could have been present.

[3] Forward secrecy is also known as perfect forward secrecy, where the term "perfect" is meaningless.

The second phase describes the *post-processing* of the key material from the first phase, which uses communication over the authenticated classical channel. The following description of phase two is based on [SML09]. The first step is called reconciliation, where Alice and Bob sift out the key bits resulting from measurements in incompatible bases. After some error correction on the result, Alice and Bob estimate the security parameter, which provides an upper bound on the information that Mallory may have about the remaining shared key material. If there is enough secrecy left, Alice and Bob apply privacy amplification: they compress the key material to a shorter key on which Mallory has negligible information. In the last step they confirm that they agree on a few key bits and use the rest for the last phase.

The last phase is called *key usage*. Assuming that the previous phase was not tampered with, the result is an ITS key. Besides using the key directly for data encryption (which may not be practical as it requires a lot of key material) there are other use-cases that benefit from having an ITS key. We consider the details of key usage out of scope in this work, but we do focus on the use case that an ITS key is required as output of the second phase.

An essential part of the second phase is the authentication of messages exchanged between Alice and Bob. Without this authentication Mallory can impersonate either party and compromise the security of the resulting key material. In order to achieve ITS authentication on a public channel, Alice and Bob must initially share some key material, which is why QKD is sometimes called quantum key *expansion*. Whenever a party authenticates one or more messages, they consume some of their key material (meaning that the key cannot be used for any other purpose) just like encrypting with a one-time pad consumes the key. In most QKD literature this is not considered a problem since a successful completion of the second phase results in a large key, part of which can be used for authenticating subsequent QKD sessions. What has not yet been considered sufficiently is what happens when the second phase does not successfully complete and no new key material comes from the second phase, yet some key material has been consumed. An adversary can exploit this and successfully execute a *key exhaustion* attack on any QKD protocol.

2.2 Message authentication codes

A message authentication code (MAC) asserts the authentication of a message by using a shared secret key. A MAC-*tag* is attached to a message, computed from both the key and the message, which proves that the received message/tag-pair originated from somebody that knows the corresponding secret key and was unaltered in transit.

A MAC scheme is a tuple (G, S, V) of algorithms. Given the security parameter λ, a key is generated as $k \xleftarrow{\$} G(1^\lambda)$. To authenticate a message m you compute a tag $t \xleftarrow{\$} S(k, m)$, which can then be verified by computing $V(k, m, t)$, which returns a bit indicating if the tag is valid. For correctness it has to hold that $V(k, m, S(k, m)) = 1$ with overwhelming probability. For most MAC-schemes G samples the keyspace uniformly at random, S is deterministic and V recomputes the tag on the received message and compares it with the received tag (note that this also implies perfect correctness). It is therefore common to leave G and V implicit and only describe S, which is then simply called the MAC *function* or just the MAC.

A MAC scheme is said to be secure if it an adversary has negligible (in the security parameter) probability of winning the existential unforgeability under chosen message attack (EUF-CMA) game, defined as follows.

Definition 2.1 (EUF-CMA game)**.** *Given security parameter λ, MAC scheme (G, S, V), and adversary A, run the following experiment:*

1. *Generate a key $k \xleftarrow{\$} G(1^\lambda)$.*
2. *Run $(m, t) \xleftarrow{\$} A^{S(k,\cdot),V(k,\cdot,\cdot)}$ and record each adversary query m' to $S(k, \cdot)$.*
3. *Output $V(k, m, t)$ if m is not recorded, otherwise output 0.*

Write $\mathit{EUF\text{-}CMA}_A^{(G,S,V)}(1^\lambda)$ for the output bit of the experiment.

In the EUF-CMA game, the adversary has oracle access to S and V under key k. By recording the queries, we exclude the attack where the adversary outputs a message/tag pair that it received directly from $S(k, \cdot)$. In other words, the adversary should find a valid tag for *a new message*. If the game allowed such an attack it would become trivial. It also means that MACs do not protect against *replay* attacks, where the adversary sends recorded message/tag-pairs.

Definition 2.2. *A MAC scheme (G, S, V) is (weak) EUF-CMA-secure if for all polynomial time adversaries A:*

$$\Pr[\mathit{EUF\text{-}CMA}_A^{(G,S,V)}(1^\lambda) = 1] \in \mathrm{negl}(\lambda). \tag{2.1}$$

The probability in Equation (2.1) is also called the *advantage* of the adversary. Oracle queries are assumed to have unit cost, so that the adversary can do only polynomially many queries.

A stronger variant of Definition 2.1 is called strong EUF-CMA (sEUF-CMA): the difference is that the adversary also wins if they output *a new tag* on any message, including messages on which S was queried.

Definition 2.3 (sEUF-CMA game)**.** *Given security parameter λ, MAC scheme (G, S, V), and adversary A, run the following experiment:*

1. *Generate a key $k \xleftarrow{\$} G(1^\lambda)$.*
2. *Run $(m, t) \xleftarrow{\$} A^{S(k,\cdot),V(k,\cdot,\cdot)}$ and for each query $t' \xleftarrow{\$} S(k, m')$, record (m', t').*
3. *Output $V(k, m, t)$ if (m, t) is not recorded, otherwise output 0.*

Write sEUF-CMA$_A^{(G,S,V)}(1^\lambda)$ for the output bit of the experiment.

Most definitions of MAC-security do not make the distinction between strong and weak security, since usually S is deterministic and V recomputes the tag. Then each message has a unique valid tag and both definitions are identical. I distinguish them here, because it can have implications in our construction later. An almost identical security definition applies to public key authentication in the form of signatures, with the difference that the adversary is given the public key instead of access to the verification oracle. The distinction between weak and strong existential unforgeability is more often relevant for signatures.

2.2.1 Wegman-Carter

Wegman-Carter MACs [WC81] are a scheme for generating an ITS message tag for a message. The Wegman-Carter construction uses a universal hash function: a function sampled uniformly from a strongly universal$_2$ family.

Definition 2.4 (strongly universal$_2$)**.** *Given a message space $\mathcal{M}$ and a security parameter λ, a family of hash functions $H = \{h : \mathcal{M} \to \{0,1\}^\lambda\}$ is* strongly universal$_2$ *if for all $x_1, x_2 \in \mathcal{M}$ with $x_1 \neq x_2$ and all $y_1, y_2 \in \{0,1\}^\lambda$ (not necessarily distinct):*

$$\Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{2^{2\lambda}}. \tag{2.2}$$

To send n messages, the construction samples a key $(h, (b_1, b_2, \ldots, b_n)) \xleftarrow{\$} H \times \{0,1\}^{\lambda n}$, and computes the MAC-tag on message m_i as

$$T_i = h(m_i) \oplus b_i. \tag{2.3}$$

Whereas h can be recycled for multiple messages, b_i act as a one-time pad and must be discarded after use. From here on we use notation $T_i := \text{MAC}(b_i, m_i)$ to focus on the part of the key that is consumed, it should be clear from context that this is relative to some shared secret hash function h.

In their seminal paper [WC81] Wegman and Carter prove that this scheme is information theoretically secure (ITS). Security is given as a variant of unforgeability: any (computationally unbounded) adversary given n message/tag pairs (m_i, T_i) (with m_i chosen by the adversary), has probability at most $2^{-\lambda}$ to output a valid (m, T) for any $m \notin \{m_i\}$.

They also prove that their scheme is asymptotically optimal in the number of bits consumed per message. They prove that a key with $\Omega(n\lambda)$ bits is required to authenticate n messages that are unforgeable with negligible probability. Their scheme requires $K + n\lambda$ bits, where K is the number of bits required to specify $h \in H$. K grows with the size of the messages that can be authenticated. Subsequent research on ITS MACs has been able to shrink K considerably and further research on QKD shows that not every post-processing message is required to be authenticated. However we assume that some upper bound on the post-processing transcript size is known, so that a reasonably small K suffices. More importantly, h is chosen once per pair of principals and it is never consumed.

Our mitigation authenticates QKD sessions with ITS MACs, and replenishes the authentication keys with some of the QKD output. Composing ITS authentication with QKD in this way has been proven secure by Portmann [Por14].

2.2.2 Alternate symmetric authentication

The protocols by Renner and Wolf [RW03; RW04] allow ITS QKD post-processing from the assumption that Alice and Bob have correlated bitstrings, on which the adversary has limited knowledge. That is are weaker assumption than those required for ITS MACs, where Alice and Bob must share an *identical* bitstring on which the adversary has *no* knowledge. They achieve this by replacing authentication tags with the interactive protocol AUTH. However that protocol also consumes the input bitstrings, both upon message acceptance and rejection, so that these protocols are similarly vulnerable to key exhaustion. Our mitigation could use the AUTH protocol for ITS authentication, but MACs are preferred since they require sending only a single message.

2.2.3 Authenticated channels

Authenticated channels are an abstraction that simplify modeling protocols, but some nuance is lost in the abstraction that turns out to be critical for security. The main difference on which we focus is the earlier mentioned one: ITS authentication keys can only be used once. Although the authentication *tag* can be sent multiple times (possibly over redundant channels), the used *key* must not be used for any other purpose.

Another difference between MAC tags and an authenticated channels is that tags give the choice of *when* to authenticate: do we attach a MAC tag to every message or do we send multiple messages, only to send a MAC tag computed over all messages at the end? As we will discuss in Section 3, this choice impacts the ease with which to execute the key exhaustion attack. Many models (if they consider such details at all) only consider the first option: attach a tag to each message.

We are not the first to observe this property of ITS authentication methods. In fact, the first QKD paper [BB84] states:

> Key bits are gradually used up in the Wegman-Carter scheme, and cannot be reused without compromising the system's provable security; however [...] these key bits can be replaced by fresh random bits successfully transmitted through the quantum channel.

In this chapter we concentrate on what happens if transferring those fresh random bits is *unsuccessful*. As it turns out, it opens up to protocol to a key exhaustion attack in which an attacker manages to deplete legitimate users of all their ITS key material.

2.3 Hash-based signatures

Because of the management of the authentication keys, QKD necessarily is a stateful protocol. Computational cryptography also has a stateful primitive in the form of stateful HBS. For a meaningful comparison later (see Section 7.2) a brief description is in order.

We briefly describe Merkle signatures [Mer89], a simple HBS system, see also Figure 2.2. Merkle signatures are based on one-time signatures (one signature per keypair, similar to how Wegman-Carter MACs must only send one tag per key). As the name implies, the one-time signature is no longer secure when used to sign two (different) messages. One-time signatures can be built from hash functions, the most basic example being a Lamport signature [Lam79]: given some security parameter λ and a hash function $h : \{0,1\}^* \to$

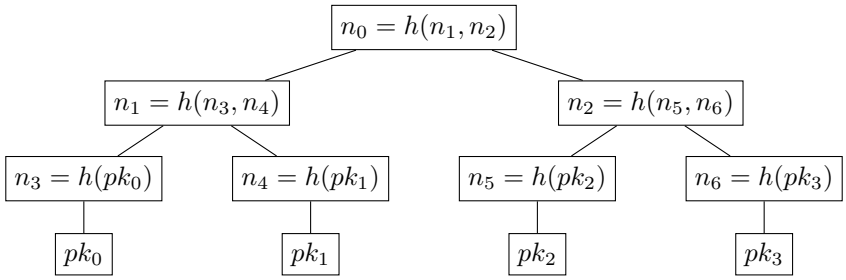


Figure 2.2: An example of a Merkle tree: the public key in a stateful HBS. Given one-time keypairs (sk_i, pk_i) for $0 \leq i < 4$, the private key is $(sk_0, \ldots, sk_3)$ and the public key is n_0, constructed as above with some hash function h. Take for example the third signature: let $\sigma = S(sk_2, m)$ be a one-time signature, then the signature $(\sigma, 2, pk_2, n_6, n_1)$ is valid if $V(pk_2, m, \sigma) = 1$ and $h(n_1, h(h(pk_2), n_6)) = n_0$.

$\{0,1\}^\lambda$, the private key consists of λ pairs $(x_0, x_1) \xleftarrow{\$} \{0,1\}^{2\lambda}$, and the public key consists of λ pairs $(h(x_0), h(x_1))$. To sign a message m, let m_i be the i-th bit of $h(m)$, then the signature consists of λ values x_{m_i}.

Key generation generates many one-time keypairs and stores the public keys in the leaves of a binary tree. Each intermediate node is a hash of its two children. The secret key consists of all one-time secret keys, the public key is the tree root. A signature consists of a one-time signature, the corresponding public key, its leaf index, and all direct siblings in the path from the leaf to the root. To verify a signature, verify the one-time signature with the given public key and then retrace the path to the root to ensure that a valid keypair was used. The signer has to locally maintain its state to ensure no one-time keypair is used more than once.

Another reason that HBS can be relevant for QKD is that in theory it allows a construction of signatures only from the assumption that one-way functions exist, a weaker assumption than required by other public-key authentication systems. This allows bootstrapping of QKD (without a shared secret key) from minimal computational assumptions.

State of the art work has brought many improvements over the basic scheme described above. Practical schemes with mature implementations exist as extended Merkle signature scheme (XMSS) [BDH11], which is standardized by the Crypto Forum Research Group (CFRG) [HBG+18], and SPHINCS$^+$ [BHK+19], which is a stateless variant and NIST round 3 alternate candidate [NIST17].

2.4 BB84-AES

The BB84-AES protocol by Price, Rarity and Erven [PRE20] prevents a different attack on availability in BB84: they reduce the time until tampering with the qubit encoding bases is detected. In BB84-AES, the sender computationally encrypts and authenticates each basis and sends this in parallel with the encoded qubit itself. Since both solutions provide protection against DoS attacks, we briefly compare their protocol against our mitigation.

BB84-AES is only computationally secure key against an active adversary, whereas our mitigation focusses on producing an ITS key, making their protocol incompatible with ours. Their protocol only replaces BB84 with the application of data encryption using the Advanced Encryption Standard (AES) in the key usage phase, whereas our mitigation is agnostic of both the specific QKD protocol and key usage. We recognize the importance of preventing the DoS attack they describe, but as engineering efforts keep increasing the key-rate of QKD, simply lowering the post-processing blocksize could limit the impact of this attack in practice.

The authors state that BB84-AES "inherently resists attempts to exhaust Alice and Bob's initial shared secret", but omit an important detail to assess the validity of the claim. In the final step of post-processing in BB84-AES "bits are taken from the final key and stored for use as the initial secret in the next round of QKD". The simplest interpretation is that the initial secret gets replaced, in which case the protocol is vulnerable to desynchronization (see Section 3). Even if Alice and Bob prevent desynchronization (for example using a variant of our mitigation) it is unclear why synchronization is required at all. Since the protocol uses computational primitives, the simpler solution would be to use a static shared key from which a session key can be derived using standard cryptographic techniques.

3 Key exhaustion

The post-processing phase of any QKD protocol can have two possible outcomes: either Alice and Bob conclude they share enough secret key material to generate new keys or they have to abort without generating any new key material. When the initial quantum communication contained too much noise (either from the environment or from an eavesdropper) Alice and Bob cannot distill a shared secret key from the quantum measurement outcomes and have to abort. We assume that the protocol is robust against environmental noise on the quantum channel: if the environmental noise is below a certain threshold, then the protocol will not abort and output a secure key. Another reason Alice or Bob might

have to abort is that the verification of message authentication on the classical channel fails, indicating interference from a third party.

When QKD attempts to generate ITS keys, the classical channel must have ITS authentication as well. For every ITS authentication tag Alice or Bob sends, they must discard the key they used. An adversary may exploit this by forcing Alice and Bob to send tags even though it will not result in them generating fresh key material, thereby exhausting the key. Storing N keys just means that the full key is exhausted after N successful attacks.

Technically, authentication failure could be a consequence of environmental noise. Classical communication can be encoded with sufficient redundancy to make this occur with low probability only. In practice any communications protocol will have to deal with message delivery failing. For our purposes we assume that an underlying message transport layer is responsible for robust message delivery, which may include message acknowledgements, resend requests and even sending messages over redundant channels. Given that the cost of losing messages in QKD is much higher than for many other protocols (as it may lead to key exhaustion even without an active attacker), the transport layer should be configured/implemented to minimize message delivery failure.

A possible objection to calling key exhaustion a legitimate attack, is that QKD generates exponentially many key-bits in the number of key-bits consumed for authentication. One may thus point out that once sufficient key material has been generated the key exhaustion attack becomes impractical for most adversaries. Our rebuttal is three-fold. First, the attack is still linear in the stored key-length, so that there is no significant gap between the cost for generating and exhausting the keys. Second, such an asymptotic statement neglects the constants that are important for real-world systems. A common critique on contemporary QKD systems is their low key-rate and extracting many bits to protect against key exhaustion would be a very impractical solution for the near-term future. Third, it requires that legitimate parties securely maintain an exponentially large state, including secure updating and deletion of keys. This third problem is exacerbated in network QKD systems, where the symmetric nature of ITS authentication means that secure keys must be stored for every pair of nodes in the network, making this problem quadratically larger.

We classify the methods available for the attacker to achieve key exhaustion. Depending on how ITS authentication is implemented, all QKD protocols could be vulnerable (and we believe they are) to one or more of these attacks. We provide some examples of protocols that are explicitly vulnerable, but we observe that most protocols simply assume an authenticated channel, thereby not providing the details to determine which vulnerabilities apply.

Noise on the quantum channel. When Alice and Bob attach an ITS tag to individual messages, the attacker achieves key exhaustion simply by sending sufficient noise on the quantum channel. The recipient of the noise will authenticate their first post-processing message and must discard the used stored key. Alice and Bob will only detect the noise after having consumed some key material this way. Since the number of messages until the first abort is constant for any specific protocol, the attack depletes N keys with $O(N)$ effort. Examples of explicitly vulnerable protocols are [DHHM99; GH00; PNM+05].

Tampering with messages on the classical channel. The attack becomes slightly less trivial when Alice and Bob authenticate *only* at the end of phase two. Sending noise will not exhaust keys, since Alice and Bob will abort if the post-processing messages indicate that there is insufficient fresh key material, which is before any ITS key material has been used for authentication. However a PITM attack or other tampering with post-processing messages will only be detected by the ITS tag at the end, at the cost of consuming an ITS key. An example of an explicitly vulnerable protocol is [Ina02].

Desynchronization. The solution to the above is to send all post-processing messages over a computationally authenticated channel, as further detailed in Section 7.1, followed by ITS authentication to prove integrity of the entire protocol transcript. We are not the first to suggest this, see for example [PRE20]. However, almost all implementations run into the problem of secure state synchronization: assume Alice updates her state first and then sends a message to Bob so he also knows to update his state: if that message is blocked the local states of the two parties are desynchronized. For example if both parties store a single authentication key that is replaced at the end of a protocol, then blocking one message desynchronizes the parties and effectively exhausts the shared key. Similarly, N keys could be exhausted with N successful desynchronizations.

Alice and Bob cannot solve this with simple delivery confirmation messages, because that runs into the two generals' problem: if Alice only updates her state after getting the confirmation that Bob did, then Mallory can desynchronize them by blocking the confirmation message itself. Confirming the confirmation message itself shifts the problem over again.

In theory only an adversary that can indefinitely block all messages can prevent the session from completing. However in practice it means that Alice and Bob have to securely maintain a large state until the protocol completes. Keeping that large state only in volatile memory means that a single error (such as a power failure) could lead to key consumption/exhaustion, while keeping such a large state synchronized with persistent

memory (storage) is non-trivial to do securely [MKF+16]. Many existing DoS attacks are based on exhausting resources by forcing devices to hold on to too many such states. Any mitigation to preventing key exhaustion that also wants to be robust against other DoS attacks should therefore be able to abort the protocol at any time while maintaining only a small state in persistent memory. We are not aware of any QKD protocol with countermeasures against desynchronization.

3.1 Aborts

We distinguish four causes for QKD aborts:

(i) post-processing failure: there was too much noise on the quantum channel;

(ii) time-out: the other party fails to reply in time;

(iii) system crash: for example due to a power outage; and

(iv) incorrect ITS tag.

If computational verification of an incoming message fails, including out-of-order or replayed messages, the protocol could either abort or the message could be ignored. Either way, a computationally bounded adversary that tampers with or blocks messages will eventually trigger a protocol abort explicitly or implicitly through a time-out. In case of cause (i) or (ii), the aborting party could optionally send a message indicating they aborted to allow quick initiation of the next session. Cause (iii) can happen at any time, which is why proper state management is so important.

In both our mitigations all ITS tags are computationally authenticated, so that an abort by cause (iv) indicates that the computational authentication was somehow broken. Aborting ensures that the security of any QKD output is not compromised, although the mitigation against key exhaustion may no longer be effective. We recommend that the protocol transcript is recorded as evidence of this breach of the computational security.[4] If indeed the key is exhausted, appropriate handling is required, as further specified in the next section.

[4]If the secrecy of this record cannot be guaranteed and the ITS key is used for future sessions, the *expected* ITS tag must not be logged since this could compromise the security of future QKD sessions.

3.2 Consequences beyond availability

Key exhaustion is primarily an attack on availability, so it can be classified as a DoS attack. In theory DoS cannot be prevented by any cryptographic or security measure: if we assume the attacker has full control over the channel they can simply block all messages. For this reason some of the QKD literature dismisses DoS attacks in general. However in practice, no adversary truly has indefinite full control of the channel and some attacks are worse than others. An additional reason to focus on key exhaustion is that it is likely to have consequences beyond reduced availability, unlike many other DoS attacks.

Once a key is exhausted, Alice and Bob first have to decide if they want to recover from it at all. If QKD has no way of recovering from key exhaustion, then key exhaustion blocks all secure communication between two nodes permanently. If a key exhaustion attack succeeds, it can provide a great payoff from the perspective of the adversary. Consider for example satellite-based QKD using ITS authentication on every message [CZC+21]: a malicious party can exhaust the keys and turn an expensive satellite into space-junk by shining a light on it. In almost every context it is unacceptable for users to have no secure communication channel at all. In practice, if the system will not fall back to some lower level of security, the users themselves will.

Recovery is thus required in all but a few rare cases. The next decision is whether to automate recovery or to require human interaction in the recovery process. We shall refer to automated methods as a recovery *protocols*, whereas we call a non-automated method a recovery *ceremony* [Ell07]. For example, the common suggestion [ABB+14] for a recovery protocol is to run QKD with only public key authentication, whereas a common suggestion for a recovery ceremony is to employ trusted couriers. We argue that *prevention* protocols (such as the ones given in this chapter) are preferable to recovery protocols.

Any recovery protocol must be authenticated, otherwise a trivial PITM or impersonation attack compromises the security of the new keys. Using ITS authentication for the recovery protocol does not help, because then the recovery protocol itself will be vulnerable to key exhaustion. Thus the recovery protocol must be computationally authenticated: let A_r be the authentication system that secures the recovery protocol. On the other hand, let A_p be the computational authentication system that secures the prevention protocol.

An immediate improvement of the above system is to combine both A_p and A_r to secure the prevention protocol. For example if A_p and A_r are both MACs, then two tags can be attached to every message. An adversary that can only break A_p can exhaust the keys in the first system, but not in the second. In other words, for every secure recovery protocol there exists a secure prevention protocol that puts more limitations on the adversary's capability to tamper with the communication.

That leaves us with ceremonies to recover from key exhaustion. Ceremonies require human interaction and therefore do not scale well, however our mitigation ensures that key exhaustion is a security failure, so that key exhaustion should occur sufficiently infrequently to make these demanding recovery methods realistic. For any real-world deployment of QKD, we recommend that implementers design a recovery *ceremony*, tailored to the context of the deployment. Preferably the ceremony includes an offline comparison of the protocol transcripts to detect security failures of the computational authentication.

One may ask why recovery is necessary when we already have prevention. First note that key exhaustion is only computationally prevented, so there could theoretically be a sufficiently powerful adversary that can exhaust the keys. Furthermore the real world contains edge cases where the shared keys must be replaced, such as key loss due to hardware failure or human error, or when suspicion exists that a key has been leaked.

When QKD has a method for recovery (whether protocol or ceremony), key exhaustion will not just be an attack on availability, but instead it is a stepping stone in an attack on the integrity (and thereby on the secrecy) of future QKD sessions.

3.3 Key exhaustion against computational cryptosystems

Computationally secure systems can take some security measures to mitigate the impact of DoS attacks. The key idea in these mitigations is to increase the cost of the attack beyond the point where executing and/or maintaining the attack is worth it.

The key exhaustion attack does not apply exclusively to QKD and even extends beyond ITS systems. Computational cryptography also has a limit on how often a key can be used until no more security can be guaranteed, even assuming the underlying computational assumption is correct. For example the probability of a nonce collision, which would undermine the security guarantees, can become unacceptably large if too much data is encrypted and/or authenticated with the same key. This limit is often high enough that it makes a key exhaustion attack impractical for sufficiently large keys. For most computational systems the cost for exhausting the key is exponential in the security parameter, compared with a linear cost for exhausting ITS systems. This makes the cost of losing a single signature or MAC tag so low that real-world cryptographic protocols simply abort incomplete sessions and try again during an active DoS attack, without much effort to resend the lost messages.

An interesting computational cryptosystem to consider here is a stateful HBS scheme, which will *explicitly* run out of key material, just like an ITS system. The key is exhausted when all leaves of the Merkle tree are consumed, thus the cost of exhausting the key is

linear in the time required for generating the tree. However when the private keys are derived from a small seed, for example as is done in XMSS [BDH11], then an honest party only has to maintain a small state.

3.4 Key exhaustion without computational cryptosystems

As a final thought experiment, consider QKD in a world without computational cryptography. The protocol will be vulnerable to key exhaustion, either by noise on the quantum channel or by tampering with the classical channel, depending on whether individual messages or the transcript is authenticated. I suspect that preventing key exhaustion is impossible in such a world.

4 A decoy-based mitigation

The decoy-based mitigation[5] prevents key exhaustion by making the cost for exhausting the key exponential in the number of stored keys. Phase one of QKD (quantum state transmission and measurement) and phase three (key usage) remain unaltered. All classical messages (including decoy and ITS tags) are sent over a computationally authenticated channel, as specified in Section 7.1. If any step of the protocol fails, then the protocol aborts and all unauthenticated key material from phase one is discarded.

4.1 Construction

A high-level overview of our mitigation is provided in Figures 2.3 and 2.4. The main idea of this protocol is to obscure *when* the authentication is being done. This is achieved by probabilistically hiding the message containing the ITS authentication tag among multiple *decoy* messages.[6] Whenever Alice and Bob detect any tampering with the decoy messages they can abort without having consumed an ITS key. Alice and Bob store multiple keys: the adversary can consume *some* keys by correctly guessing and blocking the real authentication message. However by increasing the number of decoy rounds as more keys are lost, Alice and Bob make the probability that the adversary consumes *all* keys negligible.

[5] The term *decoy* was coined by Brian Neill of evolutionQ.

[6] Decoy authentication messages should not be confused with decoy *state* QKD [Hwa03], which is a technique that allows secure QKD with practical multi-photon sources instead of theoretically perfect single-photon sources.

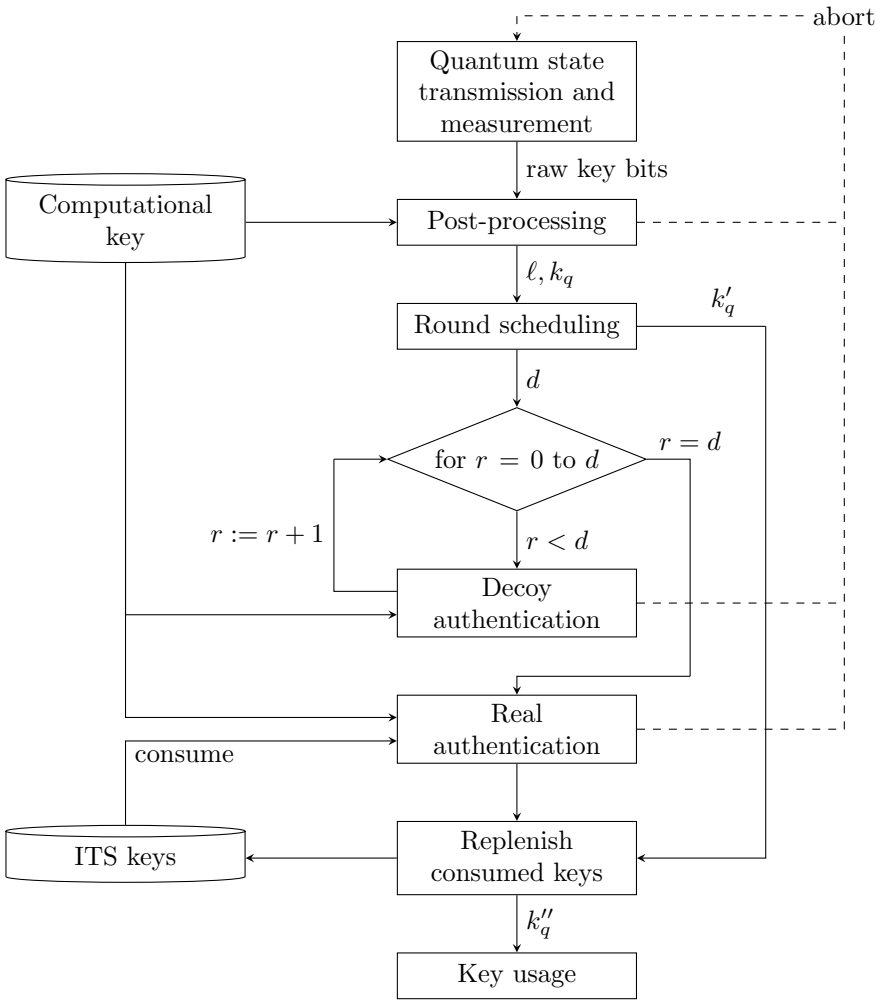


Figure 2.3: Flowchart of the decoy-based QKD key exhaustion mitigation. Successful post-processing of the raw QKD output results in shared key k_q, while ℓ indicates how many ITS keys were lost (for example due to earlier attacks). At most ℓ bits of k_q are used to schedule the amount of decoy authentication rounds (d), leaving bits k'_q. Real authentication consumes some ITS keys, which will be replenished from k'_q, leaving bits k''_q for key usage.

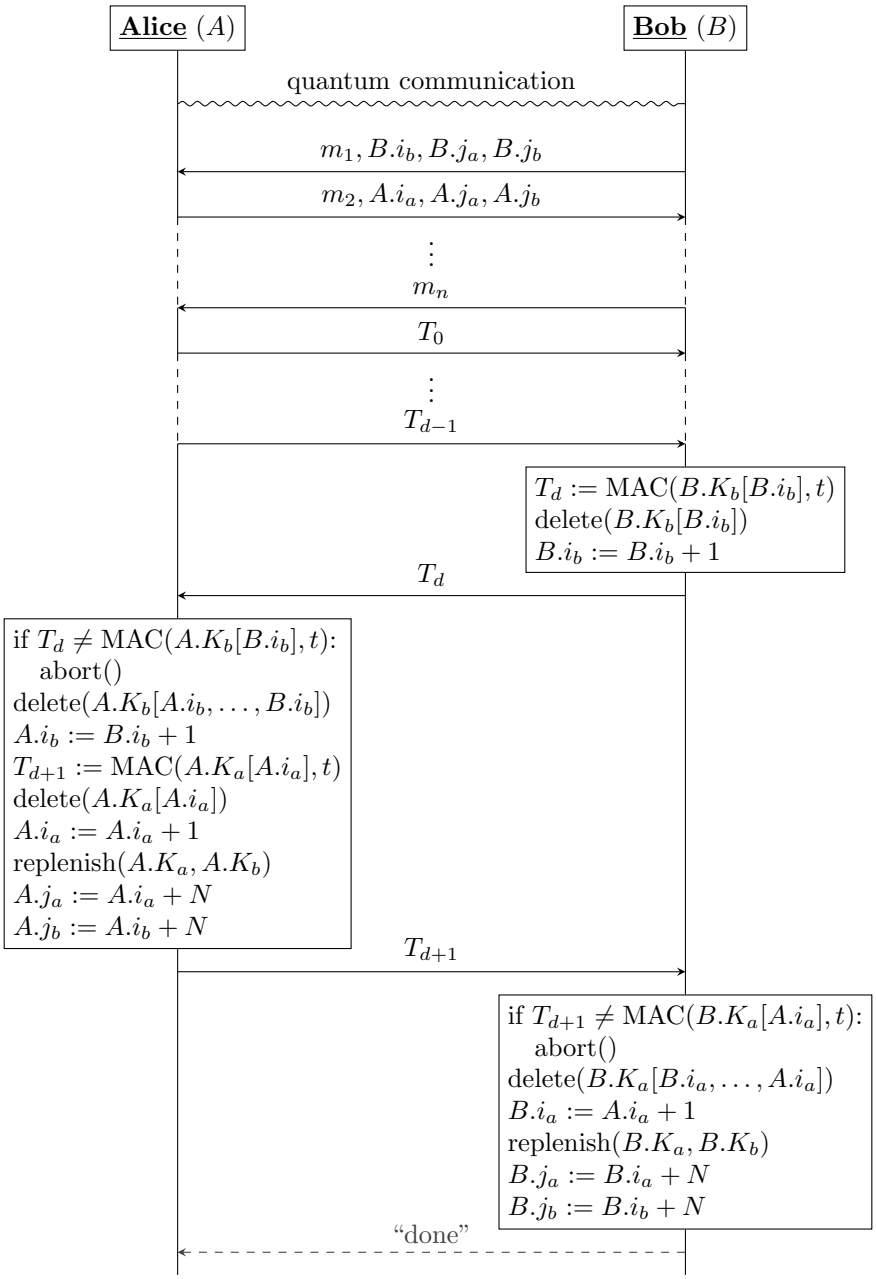


Figure 2.4: Sequence diagram of QKD with the decoy-based key exhaustion mitigation. All classical messages are sent over a computationally authenticated channel. The key indices i_a, i_b, j_a and j_b indicate which keys (K_a, K_b) the parties have stored. Tags T_d and T_{d+1} (with $d \geq 0$) provide ITS authentication over the protocol transcript (t). Decoy tags $(T_0, \ldots, T_{d-1})$ are locally generated random values, and are only required when the system is under attack.

Phase two can further be split in parts: the first part consists of the regular QKD post-processing. This part is modified to include indices of ITS keys in the messages to prevent desynchronization. The second part consists of the actual authentication: sending multiple decoy tags to hide the real ITS authentication tag, followed by the real authentication tag. The real tag ensures ITS authenticity of post-processing, as it is computed over the entire post-processing transcript. The key that was used to compute the real ITS tag must be securely discarded just before the tag is sent, to ensure it is never used again. Once both parties have received and verified the real tags, they have a guarantee that the fresh QKD key material is authentic and they can use it for replenishing consumed keys.

Under normal circumstances, Alice and Bob share the key material (K_A, K_B), which are lists of multiple ITS authentication keys for use in both directions: Alice authenticates using K_A and Bob uses K_B. In the current session Alice will use $K_A[A.i_a]$ (the first unused key in her list) and Bob uses $K_B[B.i_b]$.

Alice and Bob compute ℓ, a number indicating how many keys were lost, from the indices sent in the first messages and sample d, the number of decoy rounds, from the last ℓ bits of the QKD output k_q, as described in detail in Section 4.1.1. A few of the fresh bits of k_q will possibly be consumed by the decoy round scheduling, depending on how many keys are left: k'_q denotes the bits that are *not* used for computing d. In rounds d and $d+1$ Alice and Bob will send real ITS authentication tags, while for rounds r with $0 \leq r < d$ they will send *decoy* tags.

Real tags are computed over the entire protocol transcript[7]

$$t = (m_1, B.i_b, B.j_a, B.j_b, m_2, A.i_a, A.j_a, A.j_b, m_3, \ldots, m_n) \tag{2.4}$$

so that

$$T_d := \mathrm{MAC}(B.K_b[B.i_b], t) \tag{2.5}$$

and

$$T_{d+1} := \mathrm{MAC}(A.K_a[A.i_a], t). \tag{2.6}$$

Here we assume Bob sends the first real tag, otherwise Alice would send T_d and Bob would send T_{d+1}.

Decoy tags are generated such that they are indistinguishable from the real tags, to prevent the adversary from knowing which tag to block for successful key exhaustion. This

[7] If the protocol uses nonces to prevent replay attacks, the transcript must include the nonces N_a and N_b: $t = (m_1, B.i_b, B.j_a, B.j_b, N_b, m_2, A.i_a, A.j_a, A.j_b, N_a, m_3, \ldots, m_n)$.

is achieved by computing the decoy tag like the real ITS tag but with a random key. For $0 \leq r < d$:

$$T_r := \mathrm{MAC}(K_r, t), \tag{2.7}$$

where K_r is a locally generated random value. See Section 7.4 for further security considerations regarding decoy tag generation.

After the real authentication passed successfully, both parties replenish any consumed keys in their ITS key storage using key bits from k'_q.

This means that k_q must be sufficiently large to guarantee the ITS key storage can always be fully replenished, before initiating the second part of providing ITS authentication of the transcript. Either the post-processing blocksize should be chosen sufficiently large, or multiple QKD iterations (including computationally authenticated post-processing) could be executed until the concatenation of all outputs is sufficiently large.

Of course any message may be dropped by the adversary, with the result that $A.K_a \neq B.K_a$ and/or $A.K_b \neq B.K_b$. If message T_d was dropped in a previous session, its sender has already consumed a key but the recipient does not know yet. Therefore both Alice and Bob send their index i to indicate which sending shared key they have not consumed yet. On the other hand if message T_{d+1} was dropped in a previous session, the sender has already replenished both K_a and K_b, but the recipient was not yet able to do so. Therefore both Alice and Bob send the indices j_a and j_b to indicate how many keys they were able to replenish already. If it turns out one party replenished more keys than the other, these indices tell them those keys are not shared and they should be overwritten by future replenishments. Although Figure 2.4 shows that the indices are sent alongside m_1 and m_2, they could be sent alongside any post-processing message.

Finally we note that the recipient of the final message (T_{d+1}) should always be the one to initiate the key usage phase, so that the act of initiating this phase does not reveal that message T_d was real. Since not every context allows key usage to be initiated by either party and the honest parties do not know in advance if d is even or odd, we include an optional "done" message as the final message of phase two. This confirms message T_{d+1} was delivered and verified successfully, signalling that the key usage phase can start. An adversary can trivially distinguish the "done" message from authentication messages and block it, but at that point both honest parties have already replenished all ITS keys and blocking the "done" message has the same effect as blocking the key usage phase itself. The "done" message must be computationally authenticated: otherwise an adversary could hold on to any message and reply with "done" to learn if the held message was message T_{d+1} and only deliver it if it was not.

4.1.1 Round scheduling

The round scheduling computes the number of decoy rounds using a few bits of the (fresh) key output from post-processing. The maximum number of decoy rounds equals the number of keys that were lost, for example because of previous key exhaustion attacks. This also means that when the protocol is not under attack and no keys have been lost, zero decoy rounds are required and no fresh key bits are consumed.

First Alice and Bob determine ℓ: the number of lost keys, which can be derived from the indices sent in the first two messages. Let N be the size of key stores K_a and K_b when they are full, where N is a sufficiently large public parameter (see Section 4.2.3). Before every session Alice and Bob hold the states A and B respectively, with:

$$\begin{aligned} A.K_a &= [K_a[A.i_a], \ldots, K_a[A.j_a - 1]] \\ A.K_b &= [K_b[A.i_b], \ldots, K_a[A.j_a - 1]] \\ B.K_a &= [K_a[B.i_a], \ldots, K_a[B.j_a - 1]] \\ B.K_b &= [K_b[B.i_b], \ldots, K_b[B.j_b - 1]], \end{aligned}$$

where the indices themselves are also stored in the state. Initially $A.i_a = B.i_a = A.i_b = B.i_b = 0$ and $A.j_a = B.j_a = A.j_b = B.j_b = N$.

Alice and Bob compute $j_a = \min(A.j_a, B.j_a)$ and $j_b = \min(A.j_b, B.j_b)$, so they know the agree on the shared subset $A.K_a \cap B.K_a = [K_a[A.i_a], \ldots, K_a[j_a - 1]]$ and $A.K_b \cap B.K_b = [K_b[B.i_b], \ldots, K_b[j_b - 1]]$. Thus

$$\ell_a = N - |A.K_a \cap B.K_a| = N - j_a + A.i_a \tag{2.8}$$

and

$$\ell_b = N - |A.K_b \cap B.K_b| = N - j_b + B.i_b \tag{2.9}$$

and we simply set

$$\ell = \max(\ell_a, \ell_b). \tag{2.10}$$

If either $\ell_a = N$ or $\ell_b = N$, then Alice and Bob cannot complete the ITS authentication and the keys have been exhausted. Otherwise, Alice and Bob should replenish $\ell_a + \ell_b + 2$ keys at the end of the current session: the keys already lost plus the two keys for authentication of the current session.

Given the B-bit QKD output $k_q \in \{0,1\}^B$, Alice and Bob sample d, the number of decoy rounds, from the last ℓ bits of k_q as:

$$d = \ell - \min\left(\{x \mid k_q[B - x - 1] = 1\} \cup \{\ell\}\right), \tag{2.11}$$

where the array notation $k_q[x]$ means the x-th bit of k_q. This can be computed by scanning the last ℓ bits of k_q to find the last bit with value one.[8] Sampling d in this way accomplishes that the number of decoy rounds is likely to be close to ℓ.

The protocol leaks d by completing the session after d decoy rounds plus two real rounds, *after* having received and verified the last real message, but importantly this value must not leak *before* the end of the session. This leaks the last $\ell - d + 1$ bits (or ℓ bits when $d = 0$), since those must have been $10 \ldots 00$ (or all zeroes), so these suffix bits must be discarded. The remaining bits are k'_q, a prefix substring of k_q:

$$k'_q = k_q[0, \ldots, B - 1 - \min(\ell, \ell - d + 1)]. \tag{2.12}$$

The bits of k'_q do not need to be inspected by the scanning computation, therefore these bits do not leak and can be used for replenishing lost keys and for key usage.

Alternative round scheduling could be secure, but those are not covered by our proof in Section 4.2. It is important that such scheduling is biased to schedule many decoy rounds. Any distribution where $d \leq 1$ always has a non-negligible probability can be exhausted with polynomially many attacks by an adversary that always blocks T_1.

4.1.2 Initialization

Initialization is straightforward: if Alice and Bob are assumed to share sufficiently many key bits, they can set K_a and K_b directly from these shared bits, and set all indices to $i = 0$ and $j = N$. When bootstrapping from public key cryptography, they can indicate this by omitting or using some default values for the indices in the first messages and omitting the ITS tags altogether from the bootstrapping session.

4.2 Analysis

In this section we analyze the decoy-based mitigation. Assuming that the computational authentication is secure, we show that the mitigation prevents key exhaustion. We also answer how many keys should be stored in a real-world protocol, and analyze how many bits are consumed by the round scheduling.

[8] It should be computed differently to protect against side-channels, see Section 7.4.

4.2.1 Security analysis

A computationally unbounded adversary can break the computational authentication on every post-processing message. Even with our mitigation this adversary trivially achieves key exhaustion by setting up a PITM attack. Their presence will only be detected by the real authentication message at the cost of at least one ITS key, so that attacking at most $2N - 1$ sessions suffices to fully exhaust the ITS keys. Alternatively this adversary can tamper with the indices in order to make Alice and Bob believe that their keys are exhausted.

For the remainder of this section we consider an adversary that cannot break the computational authentication, so that tampering with any message (including real tags and decoy tags) will be detected by the honest parties. That adversary *can* block messages, so that the protocol aborts because of a message delivery time-out, possibly consuming one or two ITS keys. We prove that our mitigation prevents full key exhaustion by computationally bounded adversaries.

We model the QKD protocol itself as an *ideal* key distribution (κ), which can be formalized for example in the universal composability (UC) framework [BHL+05]. Summarized, when a party does not abort the protocol in this model, they are guaranteed that the output bitstring is shared with the other party and distributed uniformly random. A detail of that specific model is that Mallory can determine the length of the protocol output. However, as we require an output long enough to replenish keys, we simply assume that Alice and Bob also abort if their output is not long enough.

QKD realizes this functionality [BHL+05], but it requires an authenticated channel. We model that authenticated channel as an *ideal* authentication functionality (α), secure against probabilistic polynomial time (PPT) adversaries. When Alice wants to send a message to Bob (or vice versa), she sends it over the ideal channel: Mallory gets to see the message and then decide if the message is delivered. Importantly, Mallory is not able to alter messages or inject her own. In addition to this standard functionality, we also require that the authentication does not consume any key material, thereby excluding ITS authentication. See Section 7.1 for more details on how to securely realize this functionality with computational cryptography.

It is assumed that the adversary knows the maximum number of shared keys (N) and knows how many sessions they are able to attack (k). From inspection of the messages of the decoy-based protocol, the adversary also learns ℓ, the number of keys that were already lost. The following theorem does not consider side-channel attacks. See Section 7.4 for a further discussion of the side-channel protection that is required for this mitigation.

Theorem 2.5. *Consider any QKD protocol that realizes ideal key exchange (κ), using post-processing over a (computationally) authenticated channel (α) that does not consume any keys, with the mitigation of Section 4.1 applied. Let N be the total number of stored keys, and let k be an upper bound on the number of sessions that the adversary can attack.*

Then for any PPT adversary that can attack at most $k \in \mathrm{poly}(N)$ sessions:

$$\Pr[\textit{adversary consumes all } N \textit{ keys}] \in \mathrm{negl}(N). \tag{2.13}$$

Proof. By the assumption that communication occurs over an authenticated channel, all the adversary can do is eavesdrop on the messages and block a message. We distinguish four possible outcomes of an attack on a single QKD session:

1. the adversary does not block the real authentication and allows Alice and Bob to replenish all lost keys; or
2. the adversary blocks a decoy message, consuming zero ITS keys; or
3. the adversary blocks the first real message (T_d), consuming an ITS key from one party; or
4. the adversary blocks the second real message (T_{d+1}), consuming two ITS keys (one from each party).

To simplify the analysis we assume that the third outcome always increases ℓ (even though actually only ℓ_a *or* ℓ_b is increased), effectively treating the third outcome as identical to the fourth. We can do this because we only require an upper bound on the probability of key exhaustion.

Consider one QKD session. Let a be the index of the tag blocked by the adversary (with $0 \leq a \leq \ell + 1$). Let d be the number of decoy rounds, then the possible session attack outcomes are summarized in the following update function of ℓ:

$$\ell := \begin{cases} 0 & \text{if } a > d+1 \\ \ell & \text{if } a < d \\ \ell + 1 & \text{if } d \leq a \leq d+1. \end{cases} \tag{2.14}$$

We can make two simplifying observations. First we note that the adversary should always prefer $a = 1$ over $a = 0$: with both choices the keys are never replenished, but the former consumes a key both when $d = 0$ or $d = 1$, unlike the latter which requires $d = 0$.

Second we note that the adversary should always prefer $a = \ell$ over $a = \ell + 1$, at least under the assumption that both $a = d$ and $a = d + 1$ increment ℓ: both choices ensure that ℓ does not stay the same, but the former consumes a key both when $d = \ell - 1$ and $d = \ell$, unlike the latter which requires $d = \ell$. In the remainder we can therefore focus on adversaries that block $1 \leq a \leq \ell$ (for completeness we mention that $a = 1$ consumes a key with certainty when $\ell = 0$).

An adversary can see ℓ from inspection of the post-processing transcript, and we assume that they know k (how many more sessions they can attack). We define the adversary *strategy* as the probabilistic function $A : (k, \ell) \mapsto a$, which gives the value a for all $k \geq 1$ and $0 \leq \ell < N$. Note that both the real tags and decoy tags are distributed identically (uniformly random), so that there is no advantage for the adversary to base their strategy on the value of the tags themselves.

Let N be the number of keys and let A be the adversary strategy. We define $P_{N,A}(k, \ell)$ to be the probability that the adversary exhausts all N keys in k sessions or fewer, given that currently ℓ keys are lost. This allows us to rephrase the theorem as follows.

For all $k \in \text{poly}(N)$ and for all adversary strategies A:

$$P_{N,A}(k, 0) \in \text{negl}(N). \tag{2.15}$$

When N and A are clear from context, we simply write $P(k, \ell) = P_{N,A}(k, \ell)$ to simplify the notation. We have

$$P(k, N) = 1 \tag{2.16}$$

for all k, because the key is already fully exhausted, and for $0 \leq \ell < N$

$$P(0, \ell) = 0 \tag{2.17}$$

because there are no more sessions left to attack. For $k \geq 1$ and $0 \leq \ell < N$ we consider the outcome of attacking the current session. Since Equation (2.14) covers all possible outcomes, we have the following recurrence relation:

$$P(k, \ell) = \Pr[a > d+1]P(k-1, 0) + \Pr[a < d]P(k-1, \ell) + \Pr[d \leq a \leq d+1]P(k-1, \ell+1). \tag{2.18}$$

In the security model, the QKD output k_q is uniformly random. Since d is sampled according to Equation (2.11) from k_q, we have

$$\Pr[d = x] = \begin{cases} 2^{-\ell} & x = 0 \\ 2^{x-1-\ell} & 1 \leq x \leq \ell \end{cases} \tag{2.19}$$

For a fixed a we sum Equation (2.19) grouped by the different outcomes. First note that for $\ell \leq 1$ we have $\Pr[d \leq a \leq d+1] = 1$, so that $P(k+2,0) = P(k+1,1) = P(k,2)$ for all $k \geq 0$. For $2 \leq \ell \leq N$, this sums to

$$\Pr[a > d+1] = \begin{cases} 0 & a = 1 \\ (1/4)2^{a-\ell} & 2 \leq a \leq \ell, \end{cases} \tag{2.20}$$

and

$$\Pr[d \leq a \leq d+1] = \begin{cases} 2^{1-\ell} & a = 1 \\ (3/4)2^{a-\ell} & 2 \leq a \leq \ell, \end{cases} \tag{2.21}$$

and thus

$$\Pr[a < d] = 1 - 2^{a-\ell}. \tag{2.22}$$

We claim that $P(k,\ell)$ is bounded as follows.

Claim 2.6. *For $N \geq 3$, for any adversary strategy A, for all $k \geq 1$ and for all ℓ with $0 \leq \ell \leq N$:*

$$P_{N,A}(k,\ell) \leq \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} \tag{2.23}$$

Proof. We prove the claim by induction on k.

The base case with $k = 1$ has the bound $P(1,\ell) \leq (3/4)^{N-\ell}$. Note that for $\ell \neq N-1$ the bound is immediate, because $P(1,\ell) = 0$ for $\ell < N-1$ (there are not enough sessions left to exhaust all keys) and $P(1,N) = 1$. Otherwise we have

$$P(1,N-1) = \begin{cases} 2^{2-N} & \text{if } a = 1 \\ (3/4)2^{a-N+1} & \text{if } 2 \leq a \leq N-1 \end{cases} \tag{2.24}$$

Thus $P(1,N-1) \leq 3/4$, because $2^{2-N} < 3/4$ for $N \geq 3$ and because $(3/4)2^{a-N+1} \leq 3/4$ for $a \leq N-1$.

For $k \geq 2$ the induction hypothesis states that $P(k-1,\ell) \leq (3/4)^{N-\ell} + (k-2)(3/4)^N$ for all $0 \leq \ell \leq N$. We split the proof of the induction step in cases based on the value of ℓ. First we note that for $\ell = N$ the bound is greater than one, so it holds trivially.

Next consider $\ell \leq 1$, so that

$$\begin{aligned} P(k,\ell) &= P(k-1,\ell+1) \\ &\leq \left(\frac{3}{4}\right)^{N-\ell-1} + (k-2)\left(\frac{3}{4}\right)^{N} \\ &= \left(1+\frac{1}{3}\right)\left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} - \left(\frac{3}{4}\right)^{N} \\ &= \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} + \left(\frac{1}{3}\left(\frac{3}{4}\right)^{-\ell} - 1\right)\left(\frac{3}{4}\right)^{N} \\ &< \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} \end{aligned} \tag{2.25}$$

where the first inequality follows from the induction hypothesis and the last inequality follows from the fact that $(4/3)^{\ell} < 3$ for $\ell \leq 1$.

Next we prove the induction step for $2 \leq \ell < N$. First consider $a = 1$, so that we get the recurrence

$$\begin{aligned} P(k,\ell) &= 2^{1-\ell}P(k-1,\ell+1) + (1-2^{1-\ell})P(k-1,\ell) \\ &\leq 2^{1-\ell}\left(\left(\frac{3}{4}\right)^{N-\ell-1} + (k-2)\left(\frac{3}{4}\right)^{N}\right) + (1-2^{1-\ell})\left(\left(\frac{3}{4}\right)^{N-\ell} + (k-2)\left(\frac{3}{4}\right)^{N}\right) \\ &= \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} + \frac{2^{1-\ell}}{3}\left(\frac{3}{4}\right)^{N-\ell} - \left(\frac{3}{4}\right)^{N} \\ &= \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} + \left(\left(\frac{2}{3}\right)^{\ell+1} - 1\right)\left(\frac{3}{4}\right)^{N} \\ &< \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} \end{aligned} \tag{2.26}$$

where the last line holds since $(2/3)^{\ell+1} < 1$ for $\ell \geq 2$.

Finally consider the other strategies (with $2 \leq a \leq \ell$), giving the recurrence

$$\begin{aligned}
P(k,\ell) &= \frac{3}{4}2^{a-\ell}P(k-1,\ell+1) + \frac{1}{4}2^{a-\ell}P(k-1,0) + (1-2^{a-\ell})P(k-1,\ell) \\
&\leq \frac{3}{4}2^{a-\ell}\left(\left(\frac{3}{4}\right)^{N-\ell-1} + (k-2)\left(\frac{3}{4}\right)^{N}\right) + \frac{1}{4}2^{a-\ell}\left(\left(\frac{3}{4}\right)^{N} + (k-2)\left(\frac{3}{4}\right)^{N}\right) \\
&\qquad +(1-2^{a-\ell})\left(\left(\frac{3}{4}\right)^{N-\ell} + (k-2)\left(\frac{3}{4}\right)^{N}\right) \\
&= 2^{a-\ell}\left(\frac{3}{4}\right)^{N-\ell} + \frac{1}{4}2^{a-\ell}\left(\frac{3}{4}\right)^{N} + (1-2^{a-\ell})\left(\frac{3}{4}\right)^{N-\ell} + (k-2)\left(\frac{3}{4}\right)^{N} \\
&= \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N} + \left(\frac{2^{a-\ell}}{4} - 1\right)\left(\frac{3}{4}\right)^{N} \\
&< \left(\frac{3}{4}\right)^{N-\ell} + (k-1)\left(\frac{3}{4}\right)^{N}
\end{aligned} \tag{2.27}$$

where the last line holds since $2^{a-\ell} < 4$ for $a \leq \ell$ □

From the claim it immediately follows that for $N \geq 3$

$$P(k,0) \leq k\left(\frac{3}{4}\right)^{N} \tag{2.28}$$

which is indeed negligible in N for all $k \in \text{poly}(N)$. □

The proof also covers the improvement of Section 4.3, because it assumes that a key can always be consumed when $\ell \leq 1$.

The above theorem shows that the adversary is expected to attack superpolynomially many sessions in order to exhaust all keys between Alice and Bob. Small-scale simulations suggest that the optimal strategy (minimizing the number of sessions to attack) is to block the ℓ-th tag.[9] Consequently there is a significant chance that the adversary misses the real authentication tag, so that Alice and Bob replenish all keys and share key material for the key usage phase. As a result, they might choose to delay the initiation of the next QKD session until much later, when they need more shared ITS key material. Therefore it may be more effective in practice for the adversary to always block the second authentication

[9]When ℓ is small or k is small it is sometimes optimal for the adversary to block the second tag instead.

tag, ensuring that Alice and Bob are never able to fully complete the session. Attacking the first round also has the additional advantage (from the adversaries perspective) that each session aborts earlier and thus takes a shorter amount of time. However the expected number of sessions that need to attacked in this way is much more than when blocking the ℓ-th tag.[10]

4.2.2 Key store size

Whereas Theorem 2.5 shows key exhaustion is asymptotically negligible, in this section we provide parameters to prevent key exhaustion in the real world. Specifically we find appropriate values for N: the size of the key stores K_a and K_b.

If we instantiate the bound from Claim 2.6 at some value ε, we get the following lower bound on N, the size of the key stores:

$$\log_{3/4}\left(\frac{\varepsilon}{k}\right) < N. \tag{2.29}$$

For any deployment of QKD it should be possible to estimate an upper bound on k. For example $k \leq 2^{64}$ should suffice in almost all contexts, so that if we want probability $\varepsilon \leq 2^{-40}$ for key exhaustion, Equation (2.29) states that we should choose $N \geq 251$. Tighter bounds on $P_{N,A}(k, 0)$ can show that fewer keys are required to achieve the same protection against key exhaustion.

In order to sample d and be able to replenish all ITS keys, the QKD output key k_q should be at least $\ell + (\ell_a + \ell_b + 2)\lambda$ bits long, where λ is the size of the keys. Accordingly the block-size for post-processing could be chosen sufficiently large to be able to replenish all keys. For example if $\lambda = 128$, $N = 251$ and the system is estimated to have an expected key rate of 0.10 (ratio between the number of qubits exchanged and the length of k_q), then approximately $6.5 \cdot 10^5$ qubits sent in phase one will suffice. Alternatively, a system under attack can dynamically increase this block-size as ℓ increases (up to $N - 1$), and/or it may post-process multiple blocks before starting ITS authentication to ensure that k_q is sufficiently large. We remark that replenishing all ITS keys is required for *our* proof of Theorem 2.5, but that proof does not say anything about the security of partially replacing the ITS keys, which is left as future work.

[10] The adversary with strategy $a = 1$ is expected to attack $\Omega(2^N)$ sessions to achieve key exhaustion, while strategy $a = \ell$ is expected to attack $\Omega((4/3)^N)$ sessions. I omit the calculations of these claims for brevity.

4.2.3 Cost analysis

Note that the mitigation provides a low cost solution in both the number of fresh key bits that are consumed and in the number of extra messages that need to be sent.

Zero decoy messages are required when the system is not under attack and no keys have been lost ($\ell = 0$). In some QKD protocols the final post-processing messages can be merged with the first authentication messages (either real or decoy), for example when those messages are for key confirmation. For these protocols the mitigation has zero overhead when not under attack.

This also means that no fresh key bits are required for round scheduling when $\ell = 0$ and only enough bits for replenishing two ITS keys (one in each direction) are used. The following theorem shows that even if the system is under attack ($\ell > 0$) the number of bits consumed to sample d is low:

Theorem 2.7. *Let c be the number of fresh key bits consumed by the round scheduling of Section 4.1.1. Then $c \leq \ell$ (the number of lost keys) and $\mathrm{E}[c] < 4$.*

Proof. We can find the last 1 in the bitstring by inspecting at most ℓ bits, starting from the end. The upper bound $c \leq \ell$ is trivial from the fact that at most ℓ bits are inspected. Since the bits of k_q are independent and unbiased, so that we have expected value

$$\begin{aligned}
\mathrm{E}[c] &= \ell 2^{-\ell} + \sum_{d=1}^{\ell} (\ell - d + 1) 2^{d-\ell} \\
&= \sum_{d=0}^{\ell} (\ell - d) 2^{d-\ell} + \sum_{d=1}^{\ell} 2^{d-\ell} \\
&< \sum_{d'=0}^{\infty} d' \left(\frac{1}{2}\right)^{d'} + \sum_{d'=0}^{\infty} \left(\frac{1}{2}\right)^{d'} \\
&= \frac{1/2}{(1-1/2)^2} + \frac{1}{1-1/2} \\
&= 4,
\end{aligned} \tag{2.30}$$

where we substituted $d' = \ell - d$. □

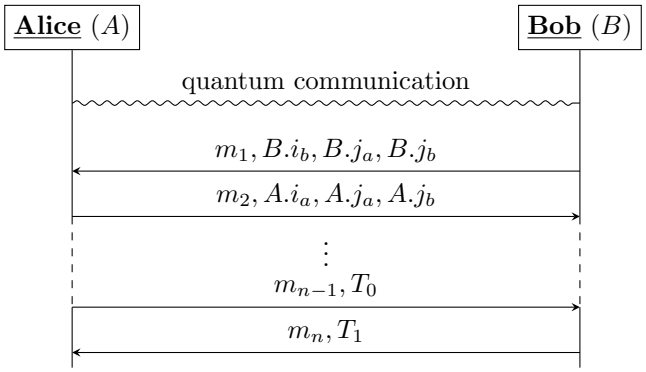


Figure 2.5: Sequence diagram of decoy-based key exhaustion mitigation, when $\ell \leq 1$. The difference is that ITS tags are sent alongside m_{n-1} and m_n instead of afterwards.

4.3 Improvement

An improvement is possible with the decoy-based mitigation when $\ell \leq 1$. Instead of sending the ITS tags *after* the last post-processing message, they can be attached to the last messages, as show in Figure 2.5. T_1 is computed over the full transcript, while T_0 is computed over the partial transcript that omits m_n. State updates remain the same as in the default decoy-based mitigation.

Since the sender of T_0 computes their ITS tag over all of their own messages in the transcript, this provides ITS authentication of the entire QKD session to the recipient. The honest parties might abort upon receiving m_{n-1} or m_n because post-processing failed, however this does not make the protocol more vulnerable to key exhaustion, since the proof of Theorem 2.5 already assumes for $\ell \leq 1$ a key can be consumed with certainty. However when $\ell > 1$, the proof assumes that protocol aborts are independent of the post-processing messages, so the ITS tags should be sent *after* m_n in that case. For specific QKD protocols, such an improvement could also apply when $\ell > 1$. For example, if the sender of m_n knows that it will never trigger a post-processing abort (for example in BB84 [BB84] m_n is just a message confirming that some revealed key bits were correct) then T_0 can always be attached to m_n, reducing the total number of messages by one in every session. The analysis of such improvements are left for future work.

5 A ratchet-based mitigation

Another mitigation is provided in Figure 2.6. To prevent the first two key exhaustion attacks described in Section 3, the entire protocol is executed over a computationally authenticated channel as specified in Section 7.1. The mitigation updates the keys in between every session, somewhat reminiscent of the way modern secure messaging protocols like Off-the-Record Messaging (OTR) [OTR] and Signal [Mar16a] update their keys using a ratchet. We combine that with careful state management of the ITS values to prevent desynchronization by messages being dropped. Our mitigation only alters the method for authenticating the post-processing phase, leaving the messages content of that phase and other phases unaltered, so that the mitigation applies to all QKD protocols.

5.1 Construction

We abstract away from the details of QKD post-processing: let $(m_1, m_2, \ldots, m_n)$ be the sequence of n post-processing messages when neither party aborts. In general we assume that either party may abort after receiving any of these messages, for example if they detect the quantum communication contained too much noise. However if the protocol accepts then it outputs shared secret key bits, denoted k_q, from which Alice and Bob can then take sufficiently many bits from that output to replenish two ITS authentication keys.

We require n to be even so that Bob always sends his ITS tag first, and $n \geq 4$ (which can be enforced by dummy messages if necessary). Bob *must* initialize post-processing by sending m_1. The Alice/Bob roles are fixed in this and all future QKD sessions.

Alice holds the variables (A, A'):

$$A = (K_a, K_b, T_a, T_b)$$
$$A' = (K'_a, K'_b, T'_a, T'_b) \text{ or } \emptyset.$$

A consists of two ITS authentication keys, her last sent tag and the last tag she received from Bob, while variable A' is either empty (denoted $\emptyset$) or holds unconfirmed replacement values for A. Bob only has the state variable B:

$$B = (K_a, K_b, T_a, T_b).$$

The idea of the protocol is that it maintains the invariant that $A = B$ or $A' = B$.

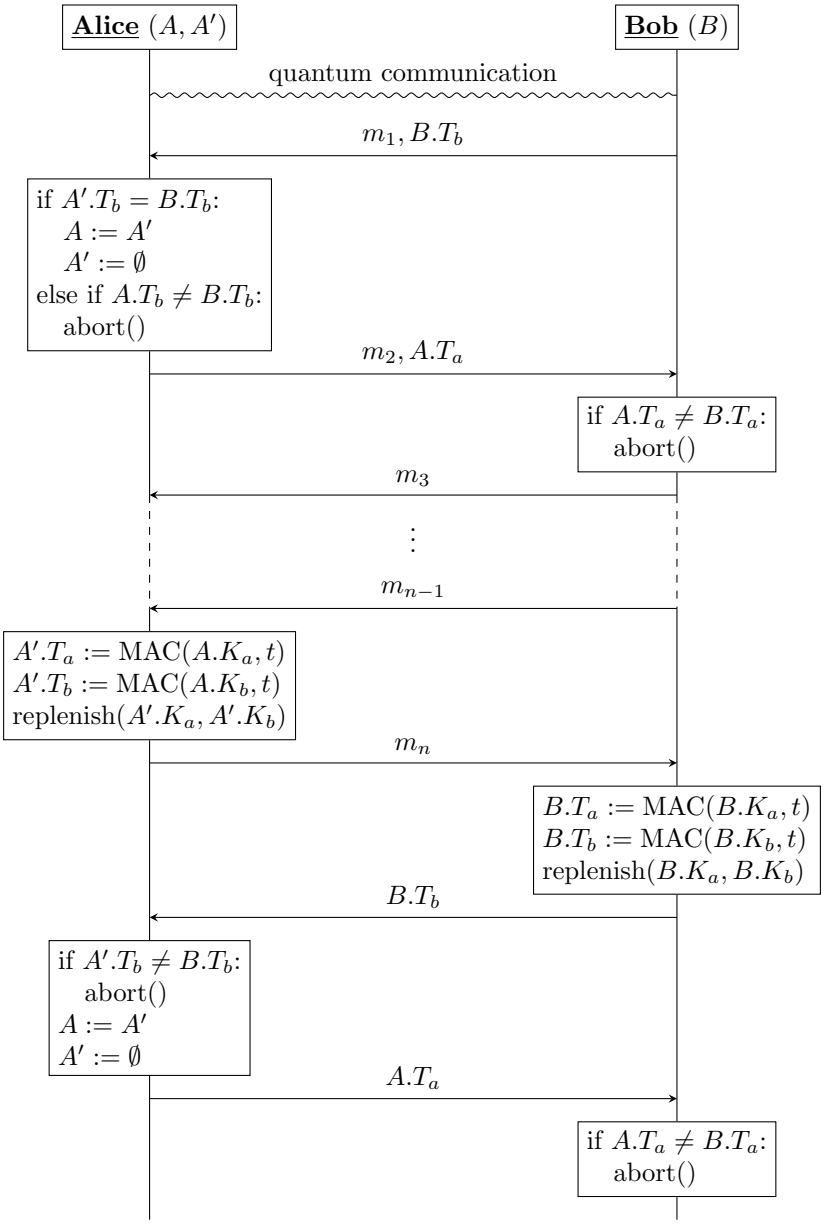


Figure 2.6: Sequence diagram of ratchet-based QKD key exhaustion mitigation. Classical messages are transmitted over a computationally authenticated channel. ITS tags T_b and T_a are computed over the full transcript $t = (m_1, B.T_b, m_2, A.T_a, \ldots, m_n)$.

First consider message m_{n-1}: upon receiving it, Alice can complete post-processing and compute m_n. She takes the full protocol transcript[11]

$$t = (m_1, B.T_b, m_2, A.T_a, m_3, \ldots m_n) \tag{2.31}$$

over which she computes the ITS tags

$$\begin{aligned} A'.T_a &:= \mathrm{MAC}(A.K_a, t) \\ A'.T_b &:= \mathrm{MAC}(A.K_b, t). \end{aligned} \tag{2.32}$$

She also takes some bits from the shared QKD output k_q, preparing the next ITS authentication keys $(A'.K_a, A'.K_b)$ for replenishing the current keys $(A.K_a, A.K_b)$. Let k'_q be the remaining key bits, which can be used in the next phase of key usage. She stores the values but *must not* overwrite any values in A, nor can she send any of the value in A': she just sends m_n. Upon receiving m_n Bob also completes post-processing and computes the new values of B: he *does* overwrite the old values. Despite having updated his state he *must not* accept yet. He sends $B.T_b$ as ITS authentication of the session and to indicate he has updated his state. Alice accepts only if $A'.T_b = B.T_b$, at which point she overwrites her old state $(A := A')$ and deletes the temporary state $(A' := \emptyset)$. Finally she sends her new tag (now stored in $A.T_a$), so that Bob can also accept when receiving $A.T_a = B.T_a$.

We also attach the ITS tags to the initial messages to ensure Alice and Bob can recover from any protocol abort. If initially Alice holds on to an incomplete update $(A' \neq \emptyset)$, then either m_n or $B.T_b$ must have dropped in the previous session. Bob sends $B.T_b$ along m_1, which will tell her which one was dropped. If he sends $A'.T_b$ then he must have received m_n and she can complete the previous session by updating $(A := A')$ and replying with the new $A.T_a$. Otherwise Bob sends the old $A.T_b$ which tells Alice that he has not updated his state, so either m_n must have been dropped or Bob aborted. Either way she will reply with $A.T_a$ alongside m_2, confirming to Bob that $A = B$: they share the required ITS key material to continue the current session.

If Alice holds $A' \neq \emptyset$ and Bob sends $A.T_b$ alongside m_1, she concludes he must have not received m_n in the previous session, yet she must not delete A'. This prevents key exhaustion by replay attacks. The computationally authenticated channel protects against replay attacks, as discussed in section Section 7.1, however in our suggested implementation it does so by including nonces to ensure freshness of the session. This can only guarantee that *replies* are fresh, but Bob's initial message could be replayed. If Bob holds $B.T_b = A'.T_b$ while Mallory replays a message $(A.T_b, m_1)$, then deleting A' would desynchronize Alice and Bob.

[11] If the protocol uses nonces to prevent replay attacks, the transcript should include the nonces N_a and N_b, so that $t = (N_b, B.T_b, m_1, N_a, A.T_a, m_2, \ldots, m_n)$.

5.1.1 Initialization

If QKD starts from the assumption that Alice and Bob already share sufficiently many random secret bits, they could initialize all values (K_a, K_b, T_a, T_b) with random shared bits. Similarly they could initialize two indices to sample the universal hash function in both directions and a shared computational authentication key in this way. Sharing T_a and T_b this way has no cryptographic significance, its purpose is to simplify the implementation so that no separate instructions or messages are required in the first session.

Instead if Alice and Bob start with each other's public key (verified, for example by a public key infrastructure (PKI)), they can bootstrap by using public key signatures to set up the computationally authenticated channel [SML09]. They initialize T_a and T_b to some default value to attach to m_1 and m_2 (this could simply be an empty string), all other messages remain the same. However initially Alice and Bob do not compute ITS MACs over the transcripts. Instead, before sending m_n, Alice sets A' (including T'_a and T'_b directly with bits from k_q. Similarly Bob sets B completely with bits from k_q. If the protocol aborts before Bob updated B, he resends the default tag alongside m_1 in the next session, signalling to Alice that they still need to bootstrap. Alice and Bob can also initialize a shared secret key for symmetric computational authentication of future sessions, which should be more efficient than using public key signatures.

5.2 A balanced variant

We can eliminate the requirement that Bob always has to initialize post-processing, by allowing either party to hold on to the unconfirmed updated value. This is depicted in Figure 2.7. The balanced variant requires $n \geq 3$.

This variant maintains the invariant that $A = B$, or $A' = B$, or $A = B'$, so that honest parties can always recover from aborted sessions. Values from A' and B' should never be sent, instead they must only be used to locally update the state ($A := A'$ or $B := B'$) *after* the other party confirms they have updated their state. After message m_2 is processed, the protocol ensures that $A = B$, so the protocol also maintains an invariant that either $A' = \emptyset$ or $B' = \emptyset$.

Assume Bob replies $B.T_b$ to message m_n, but his reply gets dropped and the parties abort, so that $A' = B$. If Alice initiates post-processing, she sends the old $A.T_a$. Bob cannot validate this value, but instead replies with $B.T_b$ to let Alice know she should update. After validating $A'.T_b = B.T_b$, Alice indeed overwrites $A := A'$ and sends the new $A.T_a$ to confirm she has done so. Bob can validate this value. In general, Alice and Bob

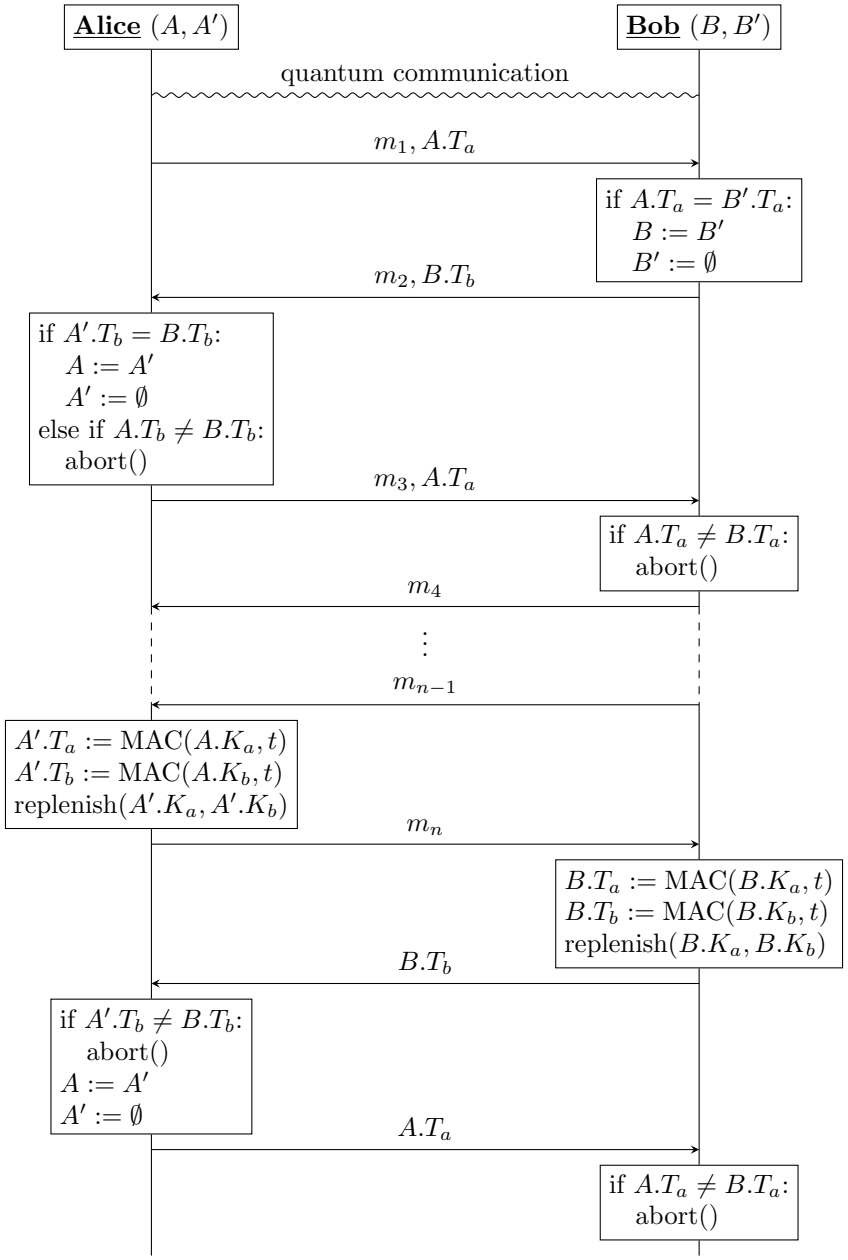


Figure 2.7: Sequence diagram of the balanced ratchet-based QKD key exhaustion mitigation. Even though the diagram shows that the number of messages (n) is odd and that Alice initializes, the parity of n is no longer important and either party can initialize.

use the tags attached to messages m_1 and m_2 to properly update their state, while they use the tags attached to messages m_2 and m_3 to validate that they are still synchronized.

6 Mitigation comparison

First we consider some minor advantages that the decoy-based mitigation has over the ratchet-based one. One advantage is that Alice and Bob (or at least one of them) are aware when their keys are exhausted, because they will have used their last ITS key. This means they know exactly when to initiate the recovery ceremony.

In the ratchet-based mitigation, an adversary that can break the computational authentication could get multiple guesses at the same ITS tag, possibly breaking the security of future sessions without having to compromise the recovery ceremony. That adversary can impersonate Bob and learns the keys that Alice stores in $(A'.K_a, A'.K_b)$. Alice cannot distinguish adversarial guesses for $A'.T_b$ from old replayed messages, so she will simply ignore these. However, choosing sufficiently large ITS tags ensures that the adversary's guesses are correct with negligible probability.

In the decoy-based mitigation each key is only *used* once, whereas the ratchet-based mitigation possibly creates multiple tags with the same key. In principle only one such tag is ever made public, but if multiple tags leak for whatever reason, this could lead to devastating loss of security.

Another advantage, at least with the improvement of Section 4.3, is that it requires fewer messages when the system is not under attack. The ratchet-based mitigation always adds a full round-trip to the post-processing protocol.

Despite these advantages of the decoy-based mitigation, I would recommend the ratchet-based mitigation in virtually all contexts. First, its state is much smaller so that local state management is easier, and second, the complexity of the side-channel protection required to secure the decoy-based mitigation can lead to vulnerabilities in any real-world deployment. This also means that an implementation of the ratchet-based mitigation will likely be somewhat faster.

Choosing between the standard ratchet-based mitigation and the balanced variant of Section 5.2 depends mostly on context. The former has smaller state and is recommended if possible, for example for prepare-and-measure QKD protocols that fix the roles anyway. The latter can be initiated by either party, so it could be more applicable to symmetric QKD protocol, such as entanglement based ones. It also has no parity requirement

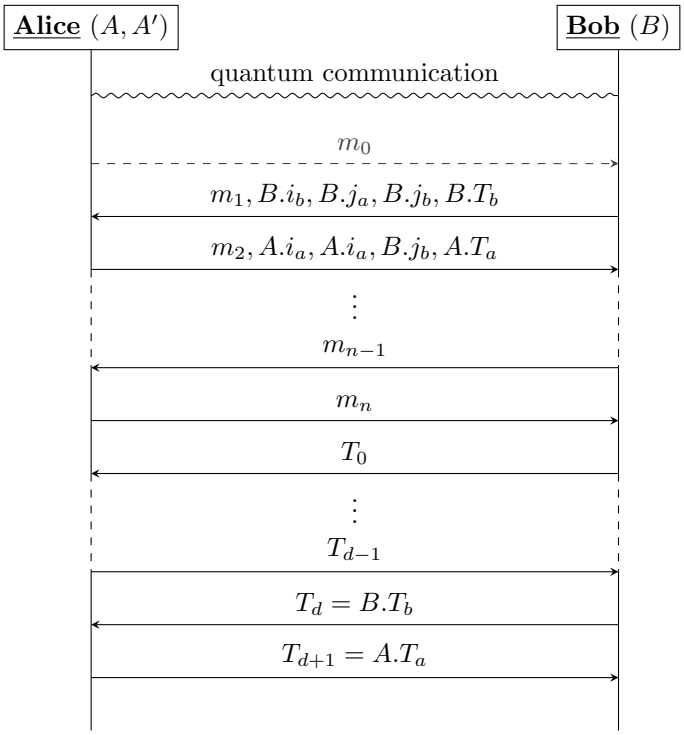


Figure 2.8: Sequence diagram of combined mitigations. State updates are omitted for simplicity.

on the number of post-processing messages, so it never needs dummy messages to make n even.

6.1 Combining the mitigations

The decoy-based mitigation has zero round-trip overhead when the system is *not* under attack. However the ratchet-based mitigation can turn out to be cheaper when the system *is* under attack, This suggests a method of combining them, where Alice and Bob always run the decoy-based mitigation but fall back to the ratchet-based mitigation just before the last key would be consumed. This is depicted in Figure 2.8.

Alice holds $A = (i_a, j_a, i_b, j_b, K_a, K_b, T_a, T_b)$, where K_a and K_b contain up to N ITS keys (as in the decoy-based mitigation), but she also holds A', while Bob only holds B (as in the ratchet-based mitigation). Alice and Bob run the decoy-based mitigation, possibly including the improvement of Section 4.3. However when $\ell = N - 1$ and only one key is

left, Alice prepares A' just before sending T_{d-1}: she computes the tags over the current transcript ($A'.T_a := \mathrm{MAC}(A.K_a[A.i_a], t)$ and $A'.T_b := \mathrm{MAC}(A.K_b[B.i_b], t)$), replenishes $A'.K_a$ and $A'.K_b$ and accordingly updates the indices in A'. She sends T_{d-1}, so Bob can overwrite B and then send $T_d = B.T_b$ to indicate he has updated his state. Upon receipt, Alice can overwrite $A := A'$.

If any message before T_d is dropped, no keys are consumed and Alice and Bob can try again in the next session. If T_d is dropped, then $A' = B$, so Alice can update her state upon receiving $B.T_b$ alongside m_1 in the next session. If T_{d+1} is dropped, then $A = B$ and the keys have already been replenished, which Bob will learn upon receiving $A.T_a$ alongside m_2 in the next session. Once Alice holds $A' \neq \emptyset$, Bob must send $B.T_b$ before she sends $A.T_a$ in the next session. If Alice initiates post-processing, she can send a dummy m_0 message to request m_1 from Bob (m_1 itself could also be a dummy message, but Bob's message must contain $B.T_b$). The role of Alice and Bob is fixed by the parity of d in the session where the last key was used, so if necessary the round scheduling of that session could be adjusted to fix the parity and thereby the roles.

The security of the combined mitigation follows directly from that of the ratchet-based one: key exhaustion will only occur when the last key is consumed, and the last key is protected by the ratchet-based mitigation. Therefore N can be chosen very small, so that the state that needs to be securely managed is only slightly larger than that of the ratchet-based mitigation. The idea could also be combined with the balanced variant of Section 5.2, in which case the dummy message m_0 is not required.

7 Security considerations

As with any security system, the protection it can provide in the real world also depends on the security of its implementation. In this section, we consider some details that are relevant to this particular problem. Section 7.1 discusses the computational channel, Section 7.2 considers local state management, Section 7.3 shows why parallelism must be avoided and Section 7.4 considers side-channels.

7.1 Computationally authenticated channel

We have described the protocol by assuming the existence of a computationally authenticated *channel*. Earlier we argued why such an abstraction might be harmful, but in case of stateless computational authentication this is acceptable.

The computational authentication itself can be achieved by either asymmetric cryptography (signatures)[12] or by symmetric cryptography (MAC tags). We require that the channel not only protects against message forgeries, but also prevents other tampering such as replay, reflection and typing attacks.

Providing only message authentication on individual messages is insufficient. The standard cryptographic engineering practice of setting up a secure channel with an ephemeral session key [FSK10] is not desirable as it requires another cryptographic primitive or protocol such as a KEX, introducing more computational assumptions.

We suggest the following techniques for realization of the channel: include randomly generated nonces at the beginning of each session to ensure the session is fresh, number and/or label individual messages to prevent out-of-order messages and typing attacks, and compute the message tags/signatures over the partial transcript instead of individual messages. If the authentication method is only EUF-CMA secure (instead of sEUF-CMA), then the computational authentication tags/signatures themselves should be omitted from the (partial) transcript. Note that nonces do not protect against replay attacks of the first message.

Computational keys can also be exhausted after having been used exponentially many times. We recommend to use a sufficiently large key instead of updating the computational keys, to avoid desynchronization issues while updating that key.

7.2 Local state management

QKD post-processing requires maintaining a large state. In this section we only consider how to manage the persistent state used for ITS authentication. We call other post-processing state the ephemeral state, which includes values such as qubit measurement outcomes. If the protocol aborts, then the ephemeral state can be discarded without leading to key exhaustion or loss of security.

The mitigations essentially describe how Alice and Bob keep their states synchronized. However this requires secure management of the *local* state by both Alice and Bob. The main problem is synchronization of volatile memory with persistent storage: the parties must ensure that updates to their state are written to persistent storage before sending the next protocol message. Another problem is that of state cloning, for example due to virtual machine cloning or system backups, which can lead to key reuse and devastating loss of security.

[12] Efficient stateless signatures from minimal assumptions exist [BHK+19].

Not much research has gone into secure synchronization in the context of stateful cryptographic protocols, however secure state management is also an issue with stateful HBS schemes [MKF+16]. Although HBS schemes only require synchronization of the private key state at one end, the problem is severe enough that the National Institute for Standards and Technology (NIST) recommends against HBS [CAD+20]:

> Stateful hash-based signature schemes [...] are not suitable for general use because their security depends on careful state management.

This indicates that the difficulty of secure state management should not be underestimated, although several factors make it easier to realize in the context of QKD than it is for HBS:

- QKD requires communication hardware such as single photon sources and detectors. Adding dedicated cryptographic hardware for state management should be relatively cheap, allowing safe implementations [MKF+16].
- HBS is a cryptographic *primitive* that could be deployed for different purposes and in many different hardware/software environments. QKD is a single *protocol* which puts secure state management in a more fixed context.
- QKD has a relatively low key-rate output, so the delay from secure state synchronization is unlikely to be a performance bottleneck.
- With our mitigation, QKD has to manage only a small state. Assuming ITS authentication uses the Wegman-Carter construction of Section 2.2 with $\lambda = 128$:
  - the decoy-based mitigation (with $N = 253$ and 128-bit indices i and j) has a local state of 8160 bytes.
  - the local state of the ratchet-based mitigation is 128 bytes for Alice and 64 bytes for Bob.

  For reference, a local state of 1664 bytes is estimated for HBS [MKF+16].

The updates to the state should be atomic, minimizing the probability that a crash during the update leaves the persistent storage in a corrupted state from which Alice and Bob cannot recover. If we extend the adversary model to include active side-channels, additional countermeasures must be taken to prevent desynchronization through side-channels such as fault attacks during state updates. We consider this outside the scope of our work.

Local state management in the decoy state solution must take extra care to not leak the sampled number of decoy rounds through a side-channel, as further discussed in Section 7.4.

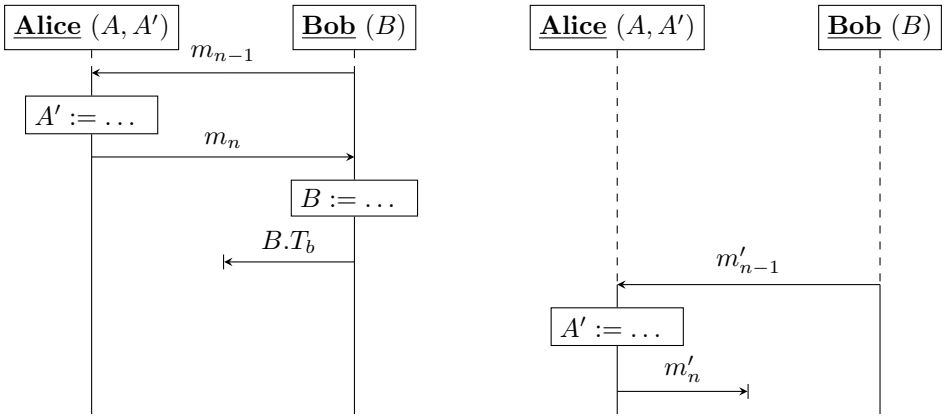


Figure 2.9: Desynchronization of the ratchet-based mitigation under parallel composition with the same keys, where messages $B.T_b$ and m'_n are dropped.

7.3 Parallelism

In both mitigations, Alice and Bob must prevent parallel composition under the same ITS keys.[13] Running multiple sessions concurrently runs into the problem of shared access to the secure state, which can lead to desynchronization. Preventing parallelism is not something that can be enforced by cryptography, but instead must be guaranteed by the executing system. While we indicate examples of security failures, we consider the mechanisms for preventing parallelism outside the scope of our work.

The decoy-based mitigation is insecure under parallel composition. If parallel sessions are allowed, an adversary attempting to learn d in one session can eavesdrop on other sessions to learn if any keys have been consumed or replenished. For example, assume Mallory holds a message with tag T_r sent in one session by Bob. If she can trick Bob into starting another concurrent session, she eavesdrops on his first message to learn $B.i_b$. If it is incremented compared to what she saw in the first session, she knows that $r = d$ or $r = d + 1$. Either way round r is a real authentication message and blocking it results in at least one key being consumed.

In the ratchet-based mitigation, parallel composition could directly lead to desynchronization. For example when message are timed as in Figure 2.9: Alice will updates A' twice while Bob only updates B once.

[13] QKD with parallel composition under *different* ITS keys is secure, as proven by Portmann and Renner [PR14].

7.4 Side-channel analysis

Standard side-channel protection should suffice in the ratchet-based mitigation. Side-channel protection of the decoy-based mitigation is non-trivial.

The analysis of the decoy-based mitigation assumes that an adversary has no way of distinguishing a decoy round from a real round, or learning d in any other way. An adversary that *can* accurately predict which message is the real authentication message can fully exhaust the ITS keys. This information might leak through side-channels in the round scheduling or in any of the (decoy) authentication rounds. Here we highlight a few pitfalls where an implementation must ensure that no side-channels exists that leak this information. We remark that this section only scratches the surface of a proper side-channel analysis, which should cover the entire implementation and not limit itself the following discussion. Our countermeasures are only suggestions: other mitigations for the mentioned side-channels likely exist and even be preferred in some contexts.

In Section 4.1.1 we analyzed the method of sampling of d by scanning the last ℓ bits in reverse order until finding the last one. However an implementation *must* sample d in constant time, for example by always scanning all last ℓ bits. The implementer could also consider to postpone discarding the used bits of k_q until after the real authentication has been completed, to prevent that the action of discarding itself leaks d.

The ITS tags T_r in decoy messages (when $r < d$) consist of random bits, so it might be tempting to sample local random bits directly for the value of T_r. However that would almost certainly leak that the round is a decoy round through a timing side-channel. This is the reason why we suggested the alternative implementation in which the sending party fully computes the ITS tag T_r using a random key K_r. In that implementation the sender should always sample a random key and select the real or random key in constant time.

Incoming decoy tags should always be validated, even if the recipient knows that the tag cannot be valid, because they do not know the key that was used. If instead the recipient ignores decoy tags and immediately sends a reply, then the message T_{d+1} will be delayed relative to the previous decoy messages and so reveal to the adversary that T_d was a real message. The adversary can then block T_{d+1} and successfully consume two keys (one in each direction).

In fact an eavesdropper should not be able to observe any difference between the processing of a real tag and a decoy tag. If the tag recipient only replenishes keys upon receiving T_d, this could leak that T_d was real through a side-channel. This is important even if replenishing keys happens locally *after* sending T_{d+1}: otherwise the following attack would consume a key: the adversary holds on to each message T_r before delivering it and

then learns through a side-channel if the sender either idles or replenishes keys. In the former case the adversary concludes that the sender just received a decoy message ($r-1 < d$) and otherwise they received a real message ($r - 1 = d$).

8 Discussion

All QKD protocols that establish ITS key material require some post-processing protocol over an ITS authenticated channel. All known ITS authentication methods consume (some of) their key material upon use, which can be exploited by an adversary to exhaust all key material. This work classifies three ways for an adversary to do so: noise on the quantum channel, tampering with the classical channel and desynchronization. Up to my knowledge, all known QKD protocols and implementations are vulnerable to one or more key exhaustion attacks.

Key exhaustion can be the first step towards an attack on the security of future QKD sessions, especially if the recovery method is implemented as some automated protocol. This aspect of the attack sets it apart from other DoS attacks, making key exhaustion an attack on QKD that is often underestimated. Our mitigations provide protection against computationally bounded adversaries, without compromising the ITS properties of the QKD output.

This chapter provides two mitigations that only use ITS authentication after the post-processing session has been authenticated computationally, avoiding the first two classes of key exhaustion. To avoid desynchronization, the first mitigation hides *when* the state is updated. The second mitigation updates the state in stages before accepting the session, so that the shared state always has some usable shared key material. Both mitigations require secure management of only a small state, minimizing the impact of other attacks on availability that are based on device resource exhaustion.

Both mitigations only prevent key exhaustion by adversaries that cannot break the computational authentication. Our mitigation leaves open the possibility of key exhaustion through some side-channel, and real-world systems may break down in unexpected ways. Therefore some method for recovering from key exhaustion is required in almost every context. It is recommended that recovery is done with a dedicated ceremony involving humans, instead of an automated protocol.

Given the impact of key exhaustion as discussed in this chapter and the proposed mitigation based on computational authentication, I believe it is in order to re-evaluate

what the added value of QKD is to real-world cryptography. Mosca, Stebila and Lütkenhaus state [SML09]: "if authentication is unbroken during the first round of QKD, even if it is only computationally secure, then subsequent rounds of QKD will be information-theoretically secure." This statement is true for systems without recovery, if only for the trivial reason that there are no subsequent rounds after keys are exhausted. At least with our mitigation, this could only happen when computational authentication is broken. To reflect upon this dependency of computational cryptography, instead I prefer the following statement that considers QKD in the context of a key exhaustion attack:

> As long as computational authentication is unbroken, quantum key distribution can provide information theoretically secure key material.

Future research could focus on developing key exhaustion recovery ceremonies for QKD in different contexts and on assessing the security of the mitigations of this chapter in the context of side-channel attacks.

Chapter 3

Terrorist fraud in quantum distance bounding

1 Introduction

Distance bounding protocols convince a verifier of both the identity and physical proximity of a prover [BC93]. These protocols add protection when physical proximity is required, for example the verifier could be a car that should open as soon as the key fob (the prover) comes nearby. Distance bounding therefore closes a gap between cryptography and real-world security: no form of cryptographic authentication can detect relay attacks, where the adversary achieves meaningful results by simply forwarding messages between honest parties unaltered. Research on distance bounding has mostly been focussed on the context of radio-frequency identification (RFID), where undetected wireless relaying is relatively easy for an adversary. However, the applicability of distance bounding will increase as more of our physical lives are coming to depend on digital keys.

The protocols that are the most secure operate by timing the delay in a challenge-response protocol: if the response is quick enough somebody must be nearby, and if the response is correct that somebody must be the prover. Part of the technical challenge of realizing distance bounding is the short time the prover has to respond, because of the high speed at which information can travel. Protocols therefore consist of slow untimed phases in which cryptographic operations can be performed (such as deriving an ephemeral key from a shared long-term key and exchanged nonces), and a rapid timed phase for exchanging individual bits. Standard communication hardware introduces so much delay

that the bounds on the distance often becomes meaningless, so that dedicated hardware is required [RČ10].

Recent work proposed quantum communication in the rapid phase of distance bounding protocols. These protocols base their security on both computational assumptions in the slow phase and on quantum information properties such as the no-cloning theorem [WZ82] in the rapid phase. These protocols have different hardware requirements, which can have an effect on various properties such as the speed of the response, the cost of implementation, and the susceptibility to side channel attacks (although at this time, no experimental data exists).

The first quantum distance bounding protocol I will consider was designed by Abidin, Marin, Singelée and Preneel (I will refer to it as the AMSP protocol) [AMSP17]. Abidin later proposed a protocol that improved upon the AMSP protocol by eliminating a final slow phase [Abi19]. Jannati and Ardeshir-Larijani had earlier proposed the relay attack detection (RAD) protocol [JA16], but as Abidin observed in 2020 [Abi20] that protocol does not detect an adversary that relays the qubits without measuring them (for that reason I will not consider it further). In the same work Abidin proposed the improved RAD (IRAD) protocol to fix this.

These protocols claim to provide protection against terrorist fraud (TF): an attack where a far-away dishonest prover colludes with an accomplice that is close to the verifier in order to convince the verifier that the prover is nearby. The countermeasure uses a standard technique from classical distance bounding: the long-term key is encrypted using an ephemeral key. See Section 2.1 for a detailed explanation of this attack and its countermeasure. This countermeasure is effective in classical distance bounding, but it turns out to be detrimental to the quantum protocols.

In this chapter, I consider three variants of the three quantum distance bounding protocols, each variant differs in the implementation of the TF countermeasure. The first variant encrypts the long-term key using a one-time pad, the second variant encrypts the long-term key using a computational cipher (such as the Advanced Encryption Standard (AES)), and the third variant entirely omits the TF countermeasure (which can be relevant in some use cases). For all three protocols, I show that the long-term key is leaked with the first variant, I show that the second variant is ineffective, and for the third variant I improve upon the existing analysis: I either show that the suggested attacks are unphysical or I show that better attacks exist.

1.1 Outline

Section 2 gives the background to this chapter. Section 3 gives a brief description of the quantum distance bounding protocols, based on their similarities. Then we consider the three protocols in detail: AMSP in Section 4, Abidin's protocol in Section 5, and IRAD in Section 6. For each protocol we

(1) describe key-extraction against the proposed TF countermeasure with the one-time pad;

(2) describe TF against the proposed TF countermeasure with a computational cipher;

(3) improve the security analysis of the protocol without TF resistance.

Section 7 provides a countermeasure that prevents key leakage in the IRAD protocol, and compares the result with classical distance bounding protocols. Finally Section 8 summarizes the results and concludes by comparing the remaining security of quantum distance bounding protocols against those of existing classical distance bounding protocols.

2 Background

We consider the required background in this section. The field of distance bounding is summarized in Section 2.1, while the required theory of quantum information is discussed in Section 2.2.

2.1 Classical distance bounding

The security guarantees that cryptography can provide may be insufficient to achieve some desired properties in the real world. Take for example keyless car entry: the car should only unlock its doors when the corresponding key fob is nearby. If the fob authenticates itself using a standard cryptographic authentication protocol, then a pair of adversaries can set up a simple relay-attack: one adversary reads the challenge messages from the car and relays them unmodified to the other adversary, who presents the challenge to the key fob in the pocket of the unsuspecting car owner. The response messages by the fob are relayed back to the car, which is tricked into opening. Other scenarios requiring some verification of location are e-Passports, (contactless) payments and e-Voting. Some scenarios require user-interaction before the authentication protocol is initiated, such as the

car owner pressing a button on the key fob, or a credit card owner entering a PIN before a digital payment. In such cases a successful attack will require extra effort, for example through social engineering, but protection against relay attacks remains important. As society moves towards ubiquitous computing and more of our lives becomes digitally connected, I believe that proper distance bounding will become increasingly important.

Distance bounding protocols protect against such relay attacks (and against more complex attacks). In distance bounding protocols a prover attempts to convince a verifier of both their identity and their proximity to the verifier. These protocols are most commonly used in RFID settings, but the applications for establishing proximity apply in a much broader context.[1]

For consistency I follow the terminology and conventions from [ALM11] in this section. For example the corresponding parties are called the verifier and prover, although in the context of RFID they are often called the *reader* and the *tag*.

2.1.1 Security requirements

Distance bounding protocols are a variant of pure authentication protocols. These protocols provide the guarantee that if the verifier accepts and completes a protocol session, then they know that the *intended* prover has participated in the *same* session.

Take for example the MAP1.1 protocol [GJZ01], displayed in Figure 3.1 (this is a pure authentication protocol, not a distance bounding protocol). When Alice accepts, after receiving the second message containing the message authentication code (MAC), she knows that Bob has participated in a MAP1.1 session (up to that second message) that agrees on the nonces N_a and N_b.[2] Since different sessions are distinguished by having different nonces, the requirement that Alice and Bob participated in the same session means that nonces should be unique. When nonces are sampled uniformly from a sufficiently large space, the probability of collisions is indeed negligible. The requirement that the other party is the *intended* party follows from the assumption that the long-term shared key k is not leaked to a third party.

An attack that breaks this authentication requirement is called *impersonation* fraud. The scenario is displayed in Figure 3.2: the adversary (who may be arbitrarily close to

[1] Wireless/contactless communication protocols are more susceptible to relaying *in practice*, but in theory wired communication is just as vulnerable and may require similar countermeasures.

[2] The MAP1.1 protocol also achieves *mutual* authentication: when Bob accepts he is guaranteed that Alice completed a MAP1.1 session with nonces N_a and N_b. Mutual authentication is (usually) not required in distance bounding.

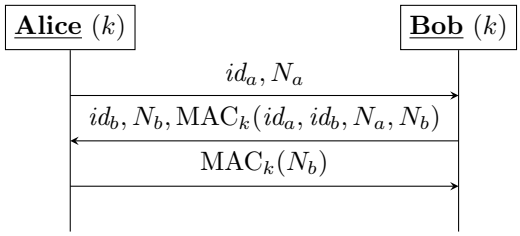


Figure 3.1: MAP1.1 pure authentication protocol [GJZ01]. Alice and Bob share a key k, they have identities id_a and id_b and they generate random nonces N_a, N_b.

Figure 3.2: Impersonation fraud

the verifier) attempts to make the verifier accept without having an honest prover participate in the same session. The adversary is allowed to eavesdrop on (and even tamper with) *other* sessions between honest parties. It should be noted that much of the existing literature leaves this security requirement implicit, and only evaluates the security of distance bounding protocols by considering its resistance against the other types of fraud (as explained below).

The other guarantee that the verifier wants from a distance bounding protocol is that the intended prover was *within proximity of the verifier* while participating in the same session.

Distance bounding based on time-of-flight is widely regarded as being the most secure (see Section 2.1.4 for alternative techniques). The main idea is that the verifier measures the time (Δt) between sending a challenge and receiving the response. Since information cannot travel faster than the speed of light (c), the verifier learns that the responder can not have been further than $(c \cdot \Delta t)/2$, where we divide by two because a message must go back and forth. Stated more practically: if the verifier requires the intended prover to be within distance D, they require the (correct) response to arrive within time $2D/c$. To prevent an *early-reply* strategy (where the prover sends the response before having received the challenge) the challenge must be unpredictable and the response must depend on the challenge.

Theoretically, distance bounding can be achieved by measuring time-of-flight on any pure authentication protocol. For example, the verifier could run MAP1.1 as Alice and measure the time until receiving Bob's first message. However this is much too slow in practice, because some *slack time* (t_s) must be allowed for a legitimate prover to compute

the correct response, so that the response must arrive in time $\tau = 2D/c + t_s$. For example, a single clock cycle on a typical smartcard (assuming a clock speed of 10 MHz) increases the distance by about 15 metres, due to the high speed of light ($c \approx 300{,}000\,\mathrm{km\,s^{-1}}$).

To prevent the slack time from dominating the total response time, distance bounding protocols are split into a slow setup phase for doing cryptographic operations, and a timed rapid phase consisting only of very simple operations for the prover. Sometimes the protocol is concluded by another slow phase, which is called the authentication phase. For example, in the rapid phase the verifier sends a challenge bit that selects one out of two prepared response bits. Of course this rapid phase should be repeated many times to prevent an adversary from always guessing the correct response. Even in such a simple challenge/response protocol, the latency introduced by signal processing (such as analog-to-digital conversion and error correction) can often introduce too much slack time.[3] In order to eliminate such overhead, some distance-bounding protocols have been designed that operate purely in the analog domain [RČ10]. A potential problem with those protocols is that they have to deal with analog attacks on the system, similar to how quantum hacking poses a threat to quantum cryptography. Indeed, subsequent work has demonstrated how a "double read-out" attack leads to key extraction, (as the name suggests, the attack reads out both response registers prepared by the prover) [RTŠ+12]. The same work also provides a countermeasure.

To assess whether a distance bounding protocol provides the proximity guarantee, one usually considers its resistance against the following four types of attacks (or frauds).

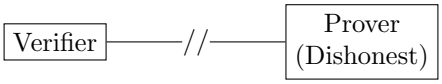


Figure 3.3: Distance fraud (DF)

Figure 3.3 depicts distance fraud (DF) [BC93], where the dishonest prover attempts to convince the verifier that they are close, while actually they are far away.

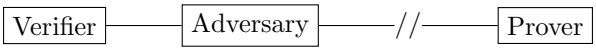


Figure 3.4: Mafia fraud (MF)

[3] Lack of error correction also means that real-world distance bounding protocols have to deal with noise in the rapid phase. This is usually dealt with by allowing some fixed number of errors (of any kind) in the rapid phase. This does not significantly alter the protocol, but complicates the analysis beyond what is necessary in this chapter.

Mafia fraud (MF) is depicted in Figure 3.4. In MF, the adversary is close to the verifier and the prover is far away. The adversary relays messages (possibly altering them or injecting their own) between the honest parties in order to convince the verifier that the prover is nearby. In most contexts, MF is the most important attack to protect against. The first description of a relay attack was given by Conway [Con76], where he described the *chess grandmaster problem*: an amateur player simultaneously challenges two grandmasters, playing one as black and the other as white. The amateur then plays the moves from one master against the other, effectively guaranteeing to win one match or draw both matches: either way an impressive result for an amateur.

The resistance against MF is often analyzed by considering two adversary strategies, depending on which party the adversary completes a rapid exchange first. In the *pre-ask* strategy the adversary completes a rapid phase with the prover first, while in the *post-ask* strategy the adversary first completes it with the verifier. The latter strategy usually only applies to protocols that have a final authentication phase.

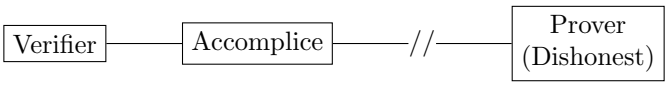


Figure 3.5: Terrorist fraud (TF)

In terrorist fraud (TF) [BBD+91] (shown in Figure 3.5) a far-away dishonest prover assists a nearby adversary (called the accomplice) in order to convince the verifier that the prover is actually nearby. This would be trivial if the prover could just give their long-term secret k to the accomplice, so that is not allowed in this fraud. TF resistance means that any successful attack reduces to the trivial attack.

Preventing TF is not always meaningful. The prover can simply leak the long-term key k to the accomplice if they trust the accomplice to discard the key afterwards or if they simply do not care about leaking the key. In the context of keyless car entry, being able to provide others one-time access to your car seems more like a feature than a bug, suggesting that TF should be allowed in that scenario. On the other hand if the protocol provides access to a secure location, preventing TF prevents the dishonest prover from setting up a business model for providing illegitimate access for a price, since giving access once leaks the key and allows indefinite access. Another relevant scenario where TF prevention is relevant is when the dishonest prover wants to create an alibi regarding their whereabouts while committing a crime elsewhere [DGB87]. Since some contexts do require TF resistance and others do not, it makes sense to analyze distance bounding protocols both with and without countermeasures against TF.

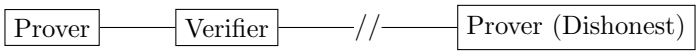


Figure 3.6: Distance hijacking

In *distance hijacking*, depicted in Figure 3.6, a remote adversary attempts to take over a session between honest participants [CRSČ12]. This attack is possible when the slow and rapid phases of a distance bounding protocol are not coupled securely. The remote dishonest prover first lets an honest prover complete the rapid phase, then hijacks the session to convince the verifier in the final slow phase that they just completed the rapid phase. I do not consider distance hijacking in this chapter since none of the original proposals claim security against it.

2.1.2 Swiss-Knife protocol

In order to explain the existing classical countermeasures against the above frauds, I briefly describe an existing classical distance bounding protocol: the Swiss-Knife protocol. I chose this protocol because it is structurally very similar to the existing quantum distance bounding protocols. The full Swiss-Knife protocol has other advantages such as mutual authentication, fault tolerance against noise, prover privacy, and an option for improved efficiency. I focus on a simplified version of the basic protocol to streamline the explanation.

Brands and Chaum introduced the first protocol based on time-of-flight [BC93] which is the foundation for modern distance bounding protocols. Hancke and Kuhn designed a protocol for the RFID setting, relying only on symmetric primitives [HK05]. Bussard and Bagga introduced the main idea for resisting TF [BB05], which was then realized using only symmetric primitives by Reid, González Nieto, Tang and Senadji [RGTS07]. That specific protocol turned out to be vulnerable to a key extraction attack, which was patched in the Swiss-Knife protocol [KAK+08]. See also the 2018 survey [ABB+18] for a comprehensive overview and comparison of existing distance bounding protocols.

In the Swiss-Knife protocol, shown in Figure 3.7, the verifier and prover share a secret k. The protocol consists of three phases. In the *setup phase* they exchange nonces N_v and N_p and establish two shared n-bit registers: $d := g_k(N_p)$, where g_k is a PRF, and $b := d \oplus k$. Denote $c = c_1 \dots c_n$ for the n-bit string, similarly for other variables. In the *rapid phase* the verifier sequentially sends challenge bits c_i. The prover has prepared two response registers: d_i and b_i. Depending on the received challenge the prover selects the correct response bit r_i. The verifier measures the time (Δt_i) between sending c_i and receiving r_i. I distinguish the sent values (c_i, r_i) from the received values (c'_i, r'_i), which helps in describing attacks where

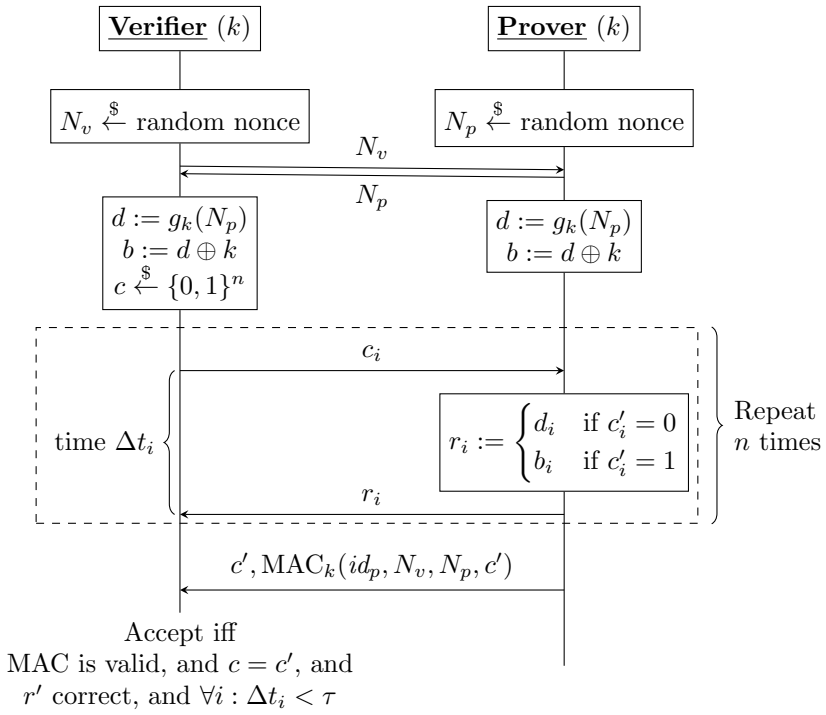


Figure 3.7: A simplified version of the basic Swiss-Knife distance bounding protocol [KAK+08]. The verifier and prover share a long-term key k. They first exchange nonces N_v and N_p, then compute ephemeral keys d (using a PRF g_k) and b. In the rapid phase the verifier sends n challenge bits c_i (received as c'_i) and the prover replies with r_i (received as r'_i). In the authentication phase the prover sends a MAC over their identity id_p, the nonces and the received challenges c'.

the adversary tampers with these values. In the *authentication phase* the prover sends all received challenges $c' = c'_1 \dots c'_n$ and an authentication tag $\mathrm{MAC}_k(id_p, N_v, N_p, c')$, where id_p is the prover identity.[4] The verifier accepts only if the MAC is valid, and all received challenges are correct ($c = c'$), and every response was correct ($r_i = (1 - c_i)d_i + c_i b_i$), and every response was given in time ($\Delta t_i < \tau$). The setup and authentication phases are untimed, providing the adversary with sufficient time to relay messages to far-away parties. However, the rapid phase is timed and must be executed locally.

The Swiss-Knife protocol essentially adds a rapid exchange phase to the MAP1.1 protocol [GJZ01], so the security against impersonation fraud follows directly from the security of MAP1.1.

The response r_i is independent of c'_i when $d_i = b_i$, and the correct response can be sent early, while the remaining responses have to be guessed. Distance fraud (DF) thus succeeds with probability $(1/2)^{\mathrm{HD}(d,b)} = (1/2)^{\mathrm{HW}(k)}$, where HD is Hamming distance and HW is Hamming weight. Since k is fixed and uniformly random by assumption, this is usually expressed as $(3/4)^n$, which takes the probability over both the early-reply guesses and the random bits of k.

Next consider MF: for a successful pre-ask strategy the adversary has to guess all challenges correctly in order to get the prover to send the correct MAC in the authentication phase (and then the adversary can also relay all responses to the verifier). With the post-ask strategy, the prover has to guess the correct responses to the verifier, (but can then relay the challenges to the prover to get the correct MAC). Either strategy succeeds with probability $(1/2)^n$.

The protocol protects against TF using the standard countermeasure: the response register b is an encryption (specifically: a one-time pad) of the long term key k using the other response register d as the key: $b := d \oplus k$. More generally, any symmetric encryption algorithm would suffice: $b := \mathrm{Enc}_d(k)$. The adversary can share either d or b without leaking k, so that in a post-ask strategy the accomplice has to guess approximately half the responses when interacting with the verifier. Alternatively the prover engages in DF while the accomplice is close to the verifier. Either strategy succeeds with probability $(3/4)^n$. Sharing both d and b with the accomplice (which would let them successfully complete the exchange phase) would reveal k.

The dishonest prover might decide to leak both bits d_i and b_i for a few values of i. This will increase the probability that TF succeeds, but it also leaks partial information on the

[4] Note that id_p is never sent in plaintext, adding privacy to the protocol. It is assumed that the verifier can search a database of all provers to find a pair (id_p, k) such that the MAC is valid, thereby learning which prover they just interacted with.

long term key k. The security argument becomes slightly more involved and is summarized as follows: if the dishonest prover leaks enough bits to make the success probability of TF unacceptably high, then they also leak enough information for the accomplice to extract k by a brute-force attack.

2.1.3 Provable security

The above listing of security requirements is rather informal. The lack of formal security definitions is unsatisfying for both the cryptanalyst that has no clear targets to aim for, but also for a system designer that is left with uncertainty about the guarantees the protocol provides. The main reason is that there is no consensus in the academic community about the proper formalization of the field, let alone the proper formalization in the context of quantum protocols and/or adversaries. With a lack of consensus, much research on distance bounding is conducted in the informal framework as described above, including the original proposals for quantum distance bounding and this chapter itself.

There are many existing formal models [DFKO11; FO13; Vau13; BV15; BMV15], which vary mainly in their security definitions that capture the various frauds and in their formalization of time-of-flight in communication. The 2018 survey paper [ABB+18] gives an overview of the debate for the interested reader.

The formalization of TF is the cause of much debate. The accepted informal interpretation is that any TF attack should reduce to the trivial attack where the dishonest prover gives the key to the accomplice (or more generally: any TF attack also allows the accomplice to fool the verifier in subsequent sessions) but formalizations of that statement vary significantly. In the SimTF definition of TF [DFKO11], the accomplice first interacts with the dishonest prover and gets the verifier to accept with probability p_A. Then a simulator is given the accomplice's *view* (protocol transcript and random coins) and has to authenticate without prover assistance, succeeding with probability p_S. To prove that a protocol is TF resistant, one has to prove that p_A is at most negligibly larger than p_S (which shows that the dishonest prover can only assist trivially). Other definitions, such as GameTF [FO13] and *collusion-fraud* [BMV15], instead allow the accomplice to act maliciously in order to extract information from the dishonest prover (without also having to make the verifier accept in the same session) and to increase the success probability of subsequent MF. Few protocols meet the strongest security definitions, although few of the demonstrated vulnerabilities in these strong models lead to practical attacks. Two interpretations of such a result are possible: either the protocol is insecure, but is currently unclear how the vulnerability can be exploited, or the security definition (or the model) is too strong.

With respect to quantum adversaries, it is not immediately clear which models (if any) properly capture the adversary's capabilities. A potential vulnerability comes from non-local strategies by adversaries that share entangled qubits. For example, results in the Boureanu, Mitrokotsa, Vaudenay model require that rapid responses are *locally* computed by nearby parties [BMV15, Lemma 1], which does not accurately model quantum adversaries.

Despite this lack of formalization, one note about the adversary model is in order: it is assumed that the adversary learns the accept/reject outcome bit of the protocol. To use the terminology of [ALM11], in this work we measure security against the result-adversary (RES-ADV). The blind-adversary (BD-ADV: an adversary that cannot see the protocol outcome) is considered too weak for most real-world scenarios: for example it is trivial to see if a door opens. On the other hand, the round-adversary (RD-ADV: an adversary that can see the accept/reject outcome of every rapid round) is considered too strong: we assume the protocol always concludes the full protocol before accepting/rejecting (no early aborts) and is implemented with sufficient side-channel protection to ensure information on individual rounds does not leak to the adversary.

2.1.4 Alternatives to time-of-flight distance bounding

Few real-world systems use time-of-flight distance bounding to establish the distance between communicating parties. Automatic vehicle location systems often rely on self-reported Global Navigation Satellite System (GNSS) coordinates, Wi-Fi positioning systems rely on measuring signal properties such as its strength. A notable exception is Wi-Fi Round Trip Time, which does measure time-of-flight, but uses self-reported timestamps to correct for slack time. Contactless payment systems offer no protection against MF [AT17], instead banks limit the maximum transfer amount and reimburse small thefts. Remote keyless systems often deploy *rolling codes* to prevent replay attacks, but usually have no protection against relay attacks.

Self-reported data is trivially spoofed to achieve DF, so it is insufficient in malicious settings. Signal strength (and other signal properties) can also spoofed, even by the adversary in the setting of MF. For example the Digital Key standard (on which the recent Apple feature *Car Keys* is built), lets you unlock and start your vehicle from your phone, by executing an authentication protocol over near-field communication (NFC). The standard claims: "The limited operational range of NFC prevents attackers from tricking the vehicle into thinking that your mobile device is nearby when it's not," [Con20] despite the fact that this is known to be false, as has been experimentally demonstrated [KCP07; Han11].

An often suggested countermeasure against relay attacks is to keep devices physically isolated, for example by using a Faraday cage or RFID blocking wallets. Such countermeasures, if at all practical, might be easily bypassed by some social engineering.

2.1.5 Position based cryptography

Although they are closely related, one should not confuse distance bounding with position based cryptography [CGMO09]. Both provide an interactive proof system based on the physical location of the prover and are based on measuring the time it takes for information to travel between two points, but they operate in different models. In distance bounding, the *identity* of the prover is a combination of their secret key and their physical distance to the verifier. Position based cryptography removes the secret key from the prover identity, so that *only* the physical location of the prover constitutes their identity. It also allows for multiple verifiers so that the physical location of the prover can be pinpointed through triangulation.

Position based cryptography can be better for modelling scenarios where collusion attacks are likely: the malicious prover may be colluding *arbitrarily* with their accomplices to try and convince the verifiers that there is somebody at the location in question, while in reality there is nobody there. Distance bounding protocols are insecure if deployed in this model: if the dishonest prover is allowed to share their secret key with their accomplice, there is no (cryptographic) way to distinguish the original prover from their accomplice. Note however that the scenario of Figure 3.5 does not constitute a legitimate attack against position based cryptography, because there is somebody (namely the accomplice) at the physical location that is being verified.

2.1.6 A note on terminology

The name mafia fraud was coined in response to a New York Times article about zero-knowledge proofs deployed on smartcards. In the article Shamir explains about an application where a credit card holder proves their identity in zero-knowledge instead of giving their card number to the merchant. He is quoted: "I can go to a Mafia-owned store a million successive times and they still will not be able to misrepresent themselves as me" [Gle87]. Desmedt, Goutier and Bengio responded [DGB87] by describing a scenario where an unsuspecting guest eats in a mafia-owned restaurant, while another mobster impersonates the guest to a jeweller. When the guest initiates the payment for the meal, the mobster initiates a payment for a diamond. The mobsters covertly relay the entire authentication

protocol between the jeweller and the guest, who has unknowingly just bought a diamond instead of a meal.

Sometimes the term *relay attack* is used when MF is meant, but a relay attacks generally only consider adversaries that relay messages *unaltered*, so MF is more general.

The name terrorist fraud was coined in 1991 [BBD+91]. The authors describe a scenario at a border crossing: the dishonest prover helps an accomplice to impersonate them to the customs officer (the verifier), in order to let the accomplice enter the country. The authors state: "We call this fraud the TF because its consequences could be disastrous if [the accomplice] were a terrorist." My criticism is that *every* attack could have very bad consequences if the attacker is a terrorist, but more importantly the scenario does not provide the prover with an incentive for assisting the accomplice, thereby missing the critical feature distinguishing the attack from MF.

Both the terms mafia fraud and terrorist fraud are non-descriptive and unnecessarily restrictive with regards to the contexts to which it applies. The terms stuck in the cryptographic literature, likely due to the high marketing value, so for that reason I will stick to the terminology as well.

2.2 Quantum information

In this section, I provide an overview of the framework for quantum information, but only just enough for understanding this chapter. There are excellent resources [KLM07; NC10; Wat18] giving far more complete descriptions.

2.2.1 Constructive

The constructions in this chapter only use pure states, which are commonly described with Dirac's bra-ket notation. The following summary is based on the book by Kaye, Laflamme and Mosca [KLM07].

States The state of a binary quantum system, a qubit, is described by a complex 2-dimensional unit vector $|\phi\rangle$, denoted as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \tag{3.1}$$

with respect to the computational basis $\{|0\rangle, |1\rangle\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$. The values α and β are called the amplitudes. Although generally amplitudes can be complex, in this chapter they are always real values.

Composed systems live in the tensor product space: the joint system of qubits $|\phi_0\rangle = (\alpha_0|0\rangle + \beta_0|1\rangle)$ and $|\phi_1\rangle = (\alpha_1|0\rangle + \beta_1|1\rangle)$ is

$$|\phi_0\rangle \otimes |\phi_1\rangle = \begin{pmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{pmatrix}, \tag{3.2}$$

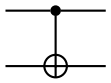
often abbreviated as $|\phi_0\rangle|\phi_1\rangle$ or even $|\phi_0\phi_1\rangle$. If a composite system cannot be expressed in product form it is called *entangled*.

Gates A quantum gate is a unitary operation on quantum states, I list a few relevant examples. The X gate flips qubits ($X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$) and is also called the NOT gate. The Z gate only flips the phase of the 1-qubit: $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. The Hadamard gate H operates as follows on the computational basis: $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2} = |+\rangle$ and $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2} = |-\rangle$, denoted as

$$H \equiv \quad \text{—}\boxed{H}\text{—} \quad \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.3}$$

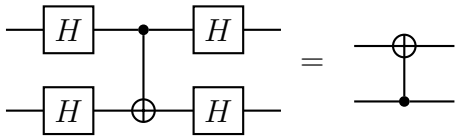
in circuit- and matrix representation, respectively. Horizontal lines in circuit diagrams denote qubits, with time flowing left-to-right, applying the gates in order. Double lines represent classical bits. Note that H is self-inverse: $H = H^{-1}$. The resulting basis $\{|+\rangle, |-\rangle\}$ is called the Hadamard basis.

An example of a two-qubit gate is the Controlled-NOT gate:

$$\text{CNOT} \equiv \quad \text{  } \quad \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.4}$$

If the first qubit has state $|0\rangle$, the gate does not affect the second qubit, but if the state is $|1\rangle$, the gate flips the second qubit. Therefore the first qubit is sometimes called the

control qubit and the second the target qubit. The following identity will be useful:



(3.5)

stating that the control and target are flipped in the Hadamard basis.

Measurements Amplitudes are not directly observable, but can instead be interpreted as a generalization of probabilities with respect to a measurement. A *Von Neumann measurement* with respect to orthonormal basis $\{|\psi_i\rangle\}$ on a state $|\phi\rangle = \sum_i \alpha_i |\psi_i\rangle$ outputs label i with probability $|\alpha_i|^2$ and leaves the system in a state $|\psi_i\rangle$. Let $\langle\psi|$ denote the dual vector of $|\psi\rangle$: defined as the action $\langle\psi| : |\phi\rangle \mapsto \langle\psi|\phi\rangle$, where $\langle\psi|\phi\rangle \in \mathbb{C}$ denotes the inner product. In matrix representation we write $\langle\psi| = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}$, where c^* denotes the complex conjugate. A Von Neumann measurement thus outputs i with probability $|\langle\psi_i|\phi\rangle|^2$. In this chapter, when I write $i := \text{measure}(|\phi\rangle)$ this should be understood as a measurement of qubit $|\phi\rangle$ in the computational basis with output label $i \in \{0, 1\}$.

It is also possible to do a partial measurement. Given a state $|\phi\rangle = \sum_i \alpha_i |\psi_i\rangle|\gamma_i\rangle$, where $\{|\psi_i\rangle\}$ is an orthonormal basis, we can measure the first subsystem to get outcome i, leaving the system in state $|\psi_i\rangle|\gamma_i\rangle$.

2.2.2 Analytic

In order to analyze the constructions in this chapter, a generalization of the above framework is required. The following summary is based on the book by Watrous [Wat18], with the exception that it uses the bra-ket notation to highlight the connection with the constructive notation.

Mathematical background Given complex Euclidean spaces $\mathcal{X}$ and $\mathcal{Y}$, the set of linear operators $A : \mathcal{X} \to \mathcal{Y}$ is denoted $\mathrm{L}(\mathcal{X}, \mathcal{Y})$. We use the shorthand notation $\mathrm{L}(\mathcal{X}) = \mathrm{L}(\mathcal{X}, \mathcal{X})$. The identity operator $I_\mathcal{X} \in \mathrm{L}(\mathcal{X})$ is the operator defined as $I_\mathcal{X} u = u$ for all $u \in \mathcal{X}$. An operator A has a unique adjoint operator $A^* \in \mathrm{L}(\mathcal{Y}, \mathcal{X})$, defined as the operator satisfying $\langle v, Au\rangle = \langle A^* v, u\rangle$ for all $u \in \mathcal{X}$ and $v \in \mathcal{Y}$. Operators $A \in \mathrm{L}(\mathcal{X})$ that are self-adjoint ($A = A^*$) are called Hermitian, denoted $\mathrm{Herm}(\mathcal{X})$. Hermitian operators with non-negative eigenvalues are called positive semidefinite, denoted $\mathrm{Pos}(\mathcal{X})$. Density operators are positive semidefinite operators with trace one, denoted $\mathrm{D}(\mathcal{X})$.

The trace of an operator is the sum of its diagonal entries, denoted $\mathrm{Tr}(A)$. This allows us to define an inner product on operators as $\langle A, B\rangle = \mathrm{Tr}(A^*B)$ for all $A, B \in \mathrm{L}(\mathcal{X}, \mathcal{Y})$.

An important norm in the context of quantum information is the *trace norm*, defined as

$$\|A\|_1 = \mathrm{Tr}\left(\sqrt{A^*A}\right). \tag{3.6}$$

The *trace distance* of operators $\rho_0, \rho_1 \in \mathrm{D}(\mathcal{X})$ is defined as $\|\rho_0 - \rho_1\|_1$ and provides a meaningful notion of how close the two operators are. One identity that is particularly useful in this chapter is

$$\|p|\phi\rangle\langle\phi| - q|\psi\rangle\langle\psi|\|_1 = \sqrt{(p+q)^2 - 4pq|\langle\phi|\psi\rangle|^2}, \tag{3.7}$$

where p, q are non-negative real numbers and $|\phi\rangle, |\psi\rangle$ are unit vectors.

States So far we only considered pure states, which can be described by a unit vector. *Density operators* allow us to model quantum states about which we have some uncertainty, as some kind of probability distribution over a finite set of quantum states. The density operator of a pure state $|\phi\rangle$ is defined as $\rho = |\phi\rangle\langle\phi|$.

Consider a mixture of pure states

$$\{(|\phi_1\rangle, p_1), \ldots, (|\phi_n\rangle, p_n)\}, \tag{3.8}$$

meaning that the quantum system is in state $|\phi_i\rangle$ with probability p_i. The density operator for this system is given by

$$\rho = \sum_{i=1}^{n} p_i |\phi_n\rangle\langle\phi_n| \tag{3.9}$$

and provides all observable properties of the quantum system. Note that this means that different mixtures can actually describe the same state.

The concept of mixed states is formalized as an *ensemble* of states. An ensemble is a function $\eta : \{1, \ldots, n\} \to \mathrm{Pos}(\mathcal{X})$, under the constraint that $\mathrm{Tr}\left(\sum_i \eta(i)\right) = 1$. The operator $\eta(i)$ encodes both the state and its probability: with probability $\mathrm{Tr}(\eta(i))$ the state is $\eta(i)/\mathrm{Tr}(\eta(i))$. An ensemble is not necessarily restricted to mixtures of *pure states*, although every density operator can be expressed as such.

Channels Quantum gates capture only unitary operators, but in general operations allowed on quantum states are given by quantum channels. which also captures the possibility to prepare quantum states and discard registers. We consider linear maps of the form $\Phi : \mathrm{L}(\mathcal{X}) \to \mathrm{L}(\mathcal{Y})$, sometimes called superoperators. The set of all such linear maps is denoted $\mathrm{T}(\mathcal{X}, \mathcal{Y})$. The identity map $I_{\mathrm{L}(\mathcal{X})} \in \mathrm{T}(\mathcal{X}, \mathcal{X})$ is the map defined as $I_{\mathrm{L}(\mathcal{X})}(X) = X$ for all $X \in \mathcal{X}$.

Quantum channels are the subset of completely positive and trace preserving maps. A map $\Phi \in \mathrm{T}(\mathcal{X}, \mathcal{Y})$ is

1. *positive* if $\Phi(P) \in \mathrm{Pos}(\mathcal{Y})$ for every $P \in \mathrm{Pos}(\mathcal{X})$;
2. *completely positive* if $\Phi \otimes I_{\mathrm{L}(\mathcal{Z})}$ is a positive map for every Euclidean space $\mathcal{Z}$; and
3. *trace-preserving* if $\mathrm{Tr}(\Phi(X)) = \mathrm{Tr}(X)$ for every $X \in \mathrm{L}(\mathcal{X})$.

The requirements on quantum channels essentially state that they must map density operators to density operators, even when applied to a subsystem of a larger system.

The act of discarding a register is given by the *partial* trace $\mathrm{Tr}_{\mathcal{X}} \in T(\mathcal{X} \otimes \mathcal{Y}, \mathcal{Y})$, which denotes the unique map satisfying $(\mathrm{Tr} \otimes I_{\mathrm{L}(\mathcal{Y})})(X \otimes Y) = \mathrm{Tr}(X)Y$ for all $X \in \mathcal{X}$, $Y \in \mathcal{Y}$.

Measurements A measurement is a function $\mu : \{1, \ldots, n\} \to \mathrm{Pos}(\mathcal{X})$ satisfying the constraint $\sum_i \mu(i) = I_{\mathcal{X}}$. The terms $\mu(i)$ are called the measurement *operators*. When a state $\rho \in \mathrm{D}(\mathcal{X})$ is measured with μ the probability of outcome i is $\langle \mu(i), \rho \rangle$ for every $1 \leq i \leq n$.

This description of a measurement is *destructive*, which is not directly compatible with the Von Neumann measurements given earlier. However any nondestructive measurement can be emulated by composing destructive measurements with quantum channels, which allows subsequent initialization of state $\mu(i)/\langle \mu(i), \rho \rangle$.

2.2.3 Optimal measurements

Consider the scenario where a challenger samples i according to some classical probability distribution, then prepares a quantum state ρ_i and gives it to the adversary. The goal of the adversary, who knows the classical probability function and all possible states ρ_i, is to guess i from the received ρ_i. In other words, the goal of the adversary is to find an optimal measurement to distinguish the quantum states of a known ensemble.

Bipartite systems When given a ensemble of two quantum states, the following theorem gives an exact quantity how well the states can be distinguished.

Theorem 3.1 (Holevo-Helstrom [Hel67; Hol72])**.** *Let $\rho_0, \rho_1 \in \mathrm{D}(\mathcal{X})$ be density operators, and $0 \leq p \leq 1$. For every measurement $\mu : \{0,1\} \to \mathrm{Pos}(\mathcal{X})$, it holds that*

$$p\langle \mu(0), \rho_0 \rangle + (1-p)\langle \mu(1), \rho_1 \rangle \leq \frac{1}{2} + \frac{1}{2}\|p\rho_0 - (1-p)\rho_1\|_1. \tag{3.10}$$

A measurement exists that achieves equality.

In the specific case with $p = 1/2$ and pure states $\rho_0 = |\phi\rangle\langle\phi|$ and $\rho_1 = |\psi\rangle\langle\psi|$, we can instantiate the theorem with Equation (3.7) to get optimal distinguishing value $1/2 + 1/2(\sqrt{1 - |\langle\phi|\psi\rangle|^2})$.

Semidefinite programs No closed-form expression is known for the optimal probability of distinguishing arbitrary ensembles with more than two quantum states (let alone finding the measurement achieving this optimum). However, the optimal measurement is naturally associated with a semidefinite program (SDP) [Wat18, Scenario 3.8].

Formally, an affine SDP is a triple (Φ, A, B), where $\Phi \in \mathrm{T}(\mathcal{X}, \mathcal{Y})$ is a Hermitian-preserving map, and where $A \in \mathrm{Herm}(\mathcal{X})$ and $B \in \mathrm{Herm}(\mathcal{Y})$ are Hermitian operators.[5] The associated (primal) problem is to maximize $\langle A, X \rangle$, subject to $\Phi(X) = B$ and $X \in \mathrm{Pos}(\mathcal{X})$.[6]

In order to distinguish an ensemble η we want to find $\max_\mu \sum_i \langle \mu(i), \eta(i) \rangle$, subject to the fact that μ must be a measurement. This is indeed an SDP, formally expressed as the triple $(\Phi, A, B) = (\mathrm{Tr}_{\mathcal{X}}, \sum_i |i\rangle\langle i| \otimes \eta(i), I_{\mathcal{Y}})$, where the feasible solutions $X \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$. Indeed, any measurement μ can be transformed into feasible solution $X = \sum_i |i\rangle\langle i| \otimes \mu(i)$, and in the other direction any feasible X gives the measurement operators $\mu(i) = (\langle i| \otimes I_{\mathcal{Y}}) X (|i\rangle \otimes I_{\mathcal{X}})$. One can check that these transformations do not affect the objective value $\langle A, X \rangle = \sum_i \langle \mu(i), \eta(i) \rangle$.

Product rule Given two SDPs (Φ_1, A_1, B_1) and (Φ_2, A_2, B_2), the *product instance* is defined as $(\Phi_1 \otimes \Phi_2, A_1 \otimes A_2, B_1 \otimes B_2)$. An affine SDP is said to obey the *product rule* if the optimal value of the product instance is the product of optimal values of each instance: $\max_X \langle A_1 \otimes A_2, X \rangle = (\max_{X_1} \langle A_1, X_1 \rangle)\,(\max_{X_2} \langle A_2, X_2 \rangle)$. In general this does not hold,

[5] More general notions of SDPs allow for inequality constraints.

[6] The associated *dual* problem is important, but not relevant for this brief description.

the optimal value of the product instance might be larger. However, Mittal and Szegedy prove [MS07, Theorem 1] that a sufficient condition for the product rule is that both A_1 and A_2 are positive semidefinite.

In case of distinguishing product states of an ensemble, the product rule does hold. We call the resulting measurement that measures each register separately a *product measurement*. Given two ensembles η_1 and η_2, the SDP operators are $A_1 = \sum_i |i\rangle\langle i| \otimes \eta_1(i)$ and $A_2 = \sum_j |j\rangle\langle j| \otimes \eta_2(j)$, which are both positive semidefinite since ensemble operators are positive semidefinite.

Since the tensor product $A_1 \otimes A_2$ is itself positive semidefinite, we can apply the product rule to the result, iteratively applying it many times. Say that we can distinguish the states from an ensemble η with optimum value $p = \max_\mu \sum_i \langle \mu(i), \eta(i) \rangle$. If we receive n independent samples from η, the best we can do is a product measurement with optimal value p^n.

3 Quantum distance bounding

In this chapter I consider three quantum distance bounding protocols: the AMSP protocol [AMSP17], Abidin's protocol [Abi19] and the IRAD protocol [Abi20]. Up to my knowledge these are the *only* quantum distance bounding protocols. Each protocol follows the same structure (compare Figures 3.8, 3.11 and 3.13), not unlike existing classical distance bounding protocols. I give a high-level overview before analysing each in detail.

The prover and verifier share a long-term key k. In an initial phase the verifier and prover exchange nonces N_v and N_p to establish shared bitstring a, using a keyed function f:

$$a = f_k(N_v, N_p), \tag{3.11}$$

where f is defined below. This is followed by a timed phase called the rapid bit exchange phase. For round i, with $1 \leq i \leq n$, the verifier generates a random classical bit c_i, derives a challenge qubit $|\phi_i\rangle$ from c_i and a, and then sends $|\phi_i\rangle$. We write $c = c_1 \dots c_n$ for the resulting n-bit string. The prover measures the challenge qubit with outcome c'_i, then derives the response qubit $|\psi_i\rangle$ from c'_i and a. The verifier accepts the response only if both the response (or more precisely: the response measurement outcome c''_i) is correct *and* the time Δt (the time between sending $|\phi_i\rangle$ and receiving $|\psi_i\rangle$) is below a threshold τ.[7] Both the challenge and response qubits are encodings of classical bits in either the

[7] Some protocols allow a few incorrect and/or slow responses to allow for noise on the channel. We do

computational or Hadamard basis. The three protocols differ mainly in the details of the encoding and decoding of the challenge/response qubits, as explained in the corresponding sections below. Only the AMSP protocol requires a final authentication message, whereas both Abidin's protocol and the IRAD protocol are done after the rapid phase.

Currently no experimental data exists to determine the slack time of these protocols, however standard quantum key distribution (QKD) equipment suffices for executing such experiments: no fault-tolerant quantum computer is required. We do remark that QKD hardware is unlikely to be optimized for minimizing slack time.

To prevent TF the protocols apply a standard technique from classical distance bounding: The proposed countermeasure for all three protocols is to instantiate f as

$$\begin{aligned} d &= g_k(N_v \| N_p) \\ b &= \mathrm{Enc}_d(k) \\ f_k(N_v, N_p) &= d \| b, \end{aligned} \tag{3.12}$$

where $\|$ denotes concatenation, g is a keyed PRF, and Enc_d is symmetric encryption using d as the key. The AMSP protocol explicitly states that the encryption should be a one-time pad:

$$\mathrm{Enc}_d(k) = d \oplus k, \tag{3.13}$$

whereas the other protocols do not specify what encryption algorithm to use.

In classical distance bounding, Equation (3.12) provides TF resistance, since knowledge of both d and b is required to provide correct responses in the timed phase. The same does not hold for the quantum protocols, since some information about the encoding basis leaks, so that an adversary can interact with the protocol in such a way that the leak reveals information about $b_i \oplus d_i$. If TF resistance uses a one-time pad, then this leaks k_i. Otherwise (when using a symmetric cipher for TF resistance) this leakage is insufficient to extract the key, however in that case TF is possible.

Some scenarios do not require TF resistance, therefore we also consider a version of each protocol without any countermeasure against TF. In that case, we instantiate f in Equation (3.11) directly with a PRF, and splitting the result to get $d \| b := a$. (A dishonest prover can achieve TF by sending a directly to the accomplice.) The resulting protocols still provide protection against DF and MF, so we analyze this to see how much security they offer.

not consider noise in this paper, but note that our attacks should generalize to noise-resistant adaptations of the analyzed protocols.

3.1 Unproven (in)security

In the absence of formal definitions, it is impossible to *prove* that a protocol is either secure or insecure, yet I claim the analysis in this chapter is meaningful. The attacks we describe against each protocol are based on real-world attack scenarios and lead to real loss of security: namely key extraction or TF. We avoid any debate surrounding the formalization of TF by appealing to the commonly accepted informal interpretation. In the TF attacks the verifier will certainly accept the accomplice, while the information leaked on the long term key is negligible in its length. Note that this is a significantly more powerful attack than one where the dishonest prover reveals both d_i and b_i for some indices i. In that case, to make TF succeed with non-negligible probability, the prover must reveal enough bits to allow efficient recovery of k through a brute-force search.

We also improve the security analyses and show that the protocols (without TF resistance) are likely secure. To do so, we provide realistic examples of DF and MF and analyze the success probabilities of the attacks. Thus we only provide an upper bound on the security of the analyzed protocol, but we do not claim that the described attacks are optimal. In case of the AMSP protocol we show the existing analysis is based on a physically impossible attack, while for the other two protocols we show that better attacks exist than was previously claimed.

4 AMSP protocol

The AMSP protocol [AMSP17], depicted in Figure 3.8, uses qubit encoding

$$|\phi_i\rangle = |\psi_i\rangle = H^{a_i}|c_i\rangle \tag{3.14}$$

in the timed phase, where a_i is the i-th bit of a and H is the Hadamard gate. In other words if $a_i = 0$ the verifier sends a bit in the computational basis and otherwise the verifier sends a bit in the Hadamard basis. The authentication phase consists of a single message by the prover:

$$\mathrm{MAC}_k(v, p, N_v, N_p, c'), \tag{3.15}$$

where v and p are some identifiers for the verifier and prover. This convinces the verifier that the prover was able to learn c by measuring $|\phi_i\rangle$ and re-encoding $|\psi_i\rangle$ in the correct basis, instead of merely reflecting $|\phi_i\rangle$.

Note that if f is instantiated with Equation (3.12), the first half of the rapid bit exchange uses d and the second half uses b, which implies that a protocol with n rapid rounds uses

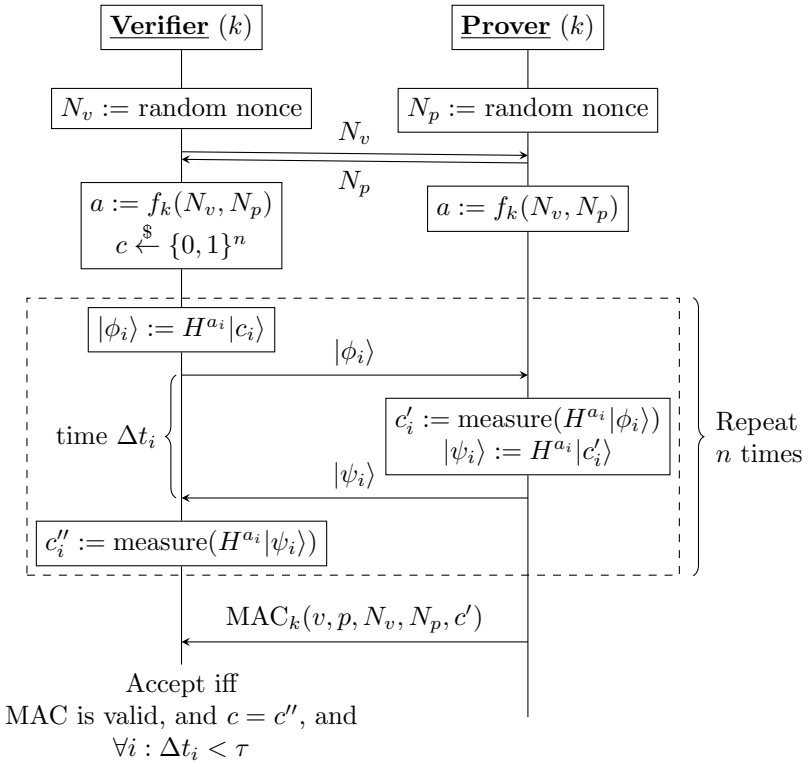


Figure 3.8: AMSP quantum distance bounding protocol [AMSP17]

an $(n/2)$-bit key k. If E is a one-time pad it holds that $a_i = d_i$ and $a_{i+n/2} = b_i = a_i \oplus k_i$ for $0 \leq i < n/2$.

4.1 Key-extraction under XOR encryption

The AMSP protocol leaks some information about the encoding bases. When applying the countermeasures of Equations (3.12) and (3.13), an adversary can impersonate a verifier and engage with a prover to extract the bits k_i. The adversary interacts with the prover only, so this attack can be executed at arbitrary distance from the verifier.

After exchanging some random initial nonces, the adversary guesses a'_i for the legitimate basis a_i and sends $|\phi_i\rangle = H^{a'_i}|c_i\rangle$ for some value of bit c_i. The prover measures $H^{a_i}|\phi_i\rangle$ with outcome c'_i, then replies with $|\psi_i\rangle = H^{a_i}|c'_i\rangle$. The adversary measures $H^{a'_i}|\psi_i\rangle$ with outcome c''_i. If the guess was correct ($a'_i = a_i$) then $c'_i = c''_i$, else both c'_i and c''_i are independent and uniform random bits. When $c''_i \neq c_i$, the adversary concludes that their guess was incorrect and thus the basis $a_i = a'_i \oplus 1$ has leaked completely. Else if $c''_i = c_i$, the adversary knows that their guess was *probably* correct and thus partial information on a_i has leaked. If both a_i and $a_{i+n/2}$ fully leak, the adversary has learned $k_i = a_i \oplus a_{i+n/2} = a'_i \oplus a'_{i+n/2}$ and otherwise they have gained some partial knowledge on k_i.

The adversary can attack every round in a single distance bounding session and then repeat the attack for multiple sessions until all $n/2$ bits of k have leaked. The PRF g ensures that the adversary can do no better than a random guess $a'_i = d'_i$ in the first half, leaking some information on d_i. In the second half the adversary can use the leaked information on d_i and k_i (leaked from earlier attacked sessions) to improve the guess b'_i. Here better means that it is more likely that $b'_i \neq b_i$ and b_i leaks completely. Instead of extracting the full key this way, the attacker can halt early and switch to an offline attack, brute-forcing f until having found a key k consistent with the already leaked bits of k and the observed N_v, N_p and leaked bits of a per session.

A full analysis is given below and shows that this leaks the complete key rapidly. For example when $n = 256$, a moderately powerful adversary (one that can compute 2^{40} hashes) is expected to extract the full 128-bit key after engaging in 16 sessions with the prover.

4.1.1 AMSP attack analysis

For $0 \leq i < n$, the attacker guesses $a'_i \neq a_i$ with probability $1/2$, so the prover replies with $|\psi_i\rangle \neq |\phi_i\rangle$. With probability $1/2$ the attackers measurement results in $c''_i \neq c_i$ and only in

this case does the attacker conclude that $a_i = a'_i \oplus 1$. Note that this is independent of the measurement outcome c'_i at the prover side. Thus the basis a_i leaks with probability 1/4.

From the attacker's perspective, initially the value a_{i+n} is independent from a_i (since nothing is known about k_i) and the same attack leaks basis $a_{i+n} = a'_{i+n} \oplus 1$ with probability 1/4. Only when both bases leak does the attacker conclude that $k_i = a_i \oplus a_{i+n} = a'_i \oplus a'_{i+n}$, so each bit of k leaks with probability 1/16.

The attacker initiates sessions with the prover until all the bits have leaked. Let ℓ denote the number of bits of k that have not leaked yet and let E_ℓ denote the expected number of sessions until ℓ bits have leaked. Let $E_0 = 0$ and for $\ell > 0$ define E_ℓ recursively: after each protocol execution $(\ell - j)$ out of ℓ bits leak, following a binomial distribution, and E_j additional sessions are necessary to extract the remaining j bits. By the linearity of the expectation value it then holds that:

$$E_\ell = 1 + \sum_{j=0}^{\ell} \binom{\ell}{j} \left(\frac{1}{16}\right)^{\ell-j} \left(\frac{15}{16}\right)^{j} E_j. \tag{3.16}$$

Extracting the $j = \ell$ term from the right-hand side sum reveals that:

$$E_\ell = \frac{1 + 16^{-\ell} \sum_{j=0}^{\ell-1} \binom{\ell}{j} 15^j E_j}{1 - (15/16)^\ell}. \tag{3.17}$$

Then E_n is the expected number of sessions to extract the full key. For example, the attack is expected to extract a 128-bit key after attacking $E_{128} \approx 84.7$ sessions.

4.1.2 Improved key extraction

It turns out that the above attack is not using everything that can be learned from the interaction with the prover: a few observations can reduce the number of protocol executions necessary to extract the key. These improvements have been incorporated in Algorithm 3.1.

First note that if $c_i = c''_i$ then it is more likely that $a_i = a'_i$, leaking partial information about the encoding basis a_i. This (partial) information about a_i and a_{i+n} leaks (partial) information about k_i. The adversary exploits this by keeping track of a variable k'_i that stores the probability that the real secret bit k_i is one, initialized at $k'_i = 1/2$. After each session the attacker updates the value to reflect what was learned in that session.

Algorithm 3.1 Improved key extraction in the AMSP protocol.

for $i := 1, \ldots, n$ **do**
  $k'_i := 0.5$
while not enough information has leaked **do**
  initialize session ▷ store N_v and N_p
  $a' \xleftarrow{\$} \{0,1\}^n$
  $c \xleftarrow{\$} \{0,1\}^{2n}$
  for $i := 1, \ldots, n$ **do**
    send $|\phi_i\rangle := H^{a'_i}|c_i\rangle$
    $c''_i := \text{measure}(H^{a'_i}|\psi_i\rangle)$
  for $i := 1, \ldots, n$ **do**
    $a'_{i+n} := \lfloor 1 - k'_i \rceil \oplus a'_i \oplus c_i \oplus c''_i$ ▷ guess $a'_{i+n} \neq a_{i+n}$
    send $|\phi_{i+n}\rangle := H^{a'_{i+n}}|c_{i+n}\rangle$
    $c''_{i+n} := \text{measure}(H^{a'_{i+n}}|\psi_{i+n}\rangle)$
    if $c_i \neq c''_i$ and $c_{i+n} \neq c''_{i+n}$ **then**
      $k'_i := a'_i \oplus a'_{i+n}$
    else if $c_i = c''_i$ and $c_{i+n} = c''_{i+n}$ **then**
      if $a'_i = a'_{i+n}$ **then**
        $k'_i := 4k'_i/(5 - k'_i)$
      else
        $k'_i := 5k'_i/(4 + k'_i)$
    else if $a'_i = a'_{i+n}$ **then**
      $k'_i := 2k'_i/(1 + k'_i)$
    else
      $k'_i := k'_i/(2 - k'_i)$

Let K_i denote the probability that $k_i = 1$ after the session and k'_i the probability before the session. The adversary computes

$$\begin{aligned} K_i &= \Pr[k_i = 1 | c''_i \wedge c''_{i+n}] \\ &= \Pr[a_i = 0 \wedge a_{i+n} = 1 | c''_i \wedge c''_{i+n}] + \Pr[a_i = 1 \wedge a_{i+n} = 0 | c''_i \wedge c''_{i+n}], \end{aligned} \tag{3.18}$$

using Bayes' theorem for both terms. Four cases need to be distinguished based on whether $c''_i = c_i$ and $c''_{i+n} = c_{i+n}$. When $c''_i \neq c_i$ and $c''_{i+n} \neq c_{i+n}$ the bit k_i has completely leaked, otherwise some more calculation is required. It is straightforward that $\Pr[c''_i = c_i | a_i = a'_i] = 1$, $\Pr[c''_i = c_i | a_i \neq a'_i] = 1/2$ (similar when replacing i with $(i+n)$) and $\Pr[a_i = a'_i] = 1/2$. Given the knowledge k'_i and under the assumption that the attacker has guessed $a'_{i+n} = a'_i$, it holds that $\Pr[a_{i+n} = a_i] = 1 - k'_i$. By combining all these observations the necessary probabilities can be calculated, revealing that when $c''_i = c_i$ and $c''_{i+n} = c_{i+n}$:

$$K_i = \frac{4k'_i}{5 - k'_i} \tag{3.19}$$

and for the other cases:

$$K_i = \frac{2k'_i}{1 + k'_i}. \tag{3.20}$$

Observe that if $a'_{i+n} \neq a'_i$, then Equations (3.19) and (3.20) express the updated value of $(1-K_i)$ in terms of $(1-k'_i)$. Substituting these terms reveal the form given in Algorithm 3.1.

The attacker then uses this partial knowledge on k_i in subsequent sessions to improve their guess a'_{i+n}, based on k'_i and what was learned about a_i in the same session. By attempting to guess wrong ($a'_{i+n} \neq a_{i+n}$) the attacker maximizes the probability that the basis a_{i+n} leaks.

When the attacker stores the nonces N_v and N_p that were sent over the public channel, they have the input to the PRF g_k. Some output bits leak to the adversary as well, either directly through the leaked bits of d or indirectly through the leaked bits of b and the known bits of k. After enough protocol executions (see Section 4.1.3) the attacker has learned enough about k and knows enough output bits of f_k to find a collision in f and thus extract k. This splits up the attack in an online phase followed by an offline phase.

4.1.3 Improved key extraction analysis

The required number of sessions is estimated by simulating the attack of Algorithm 3.1 several times. After each protocol execution store the number of bits k_i that have not

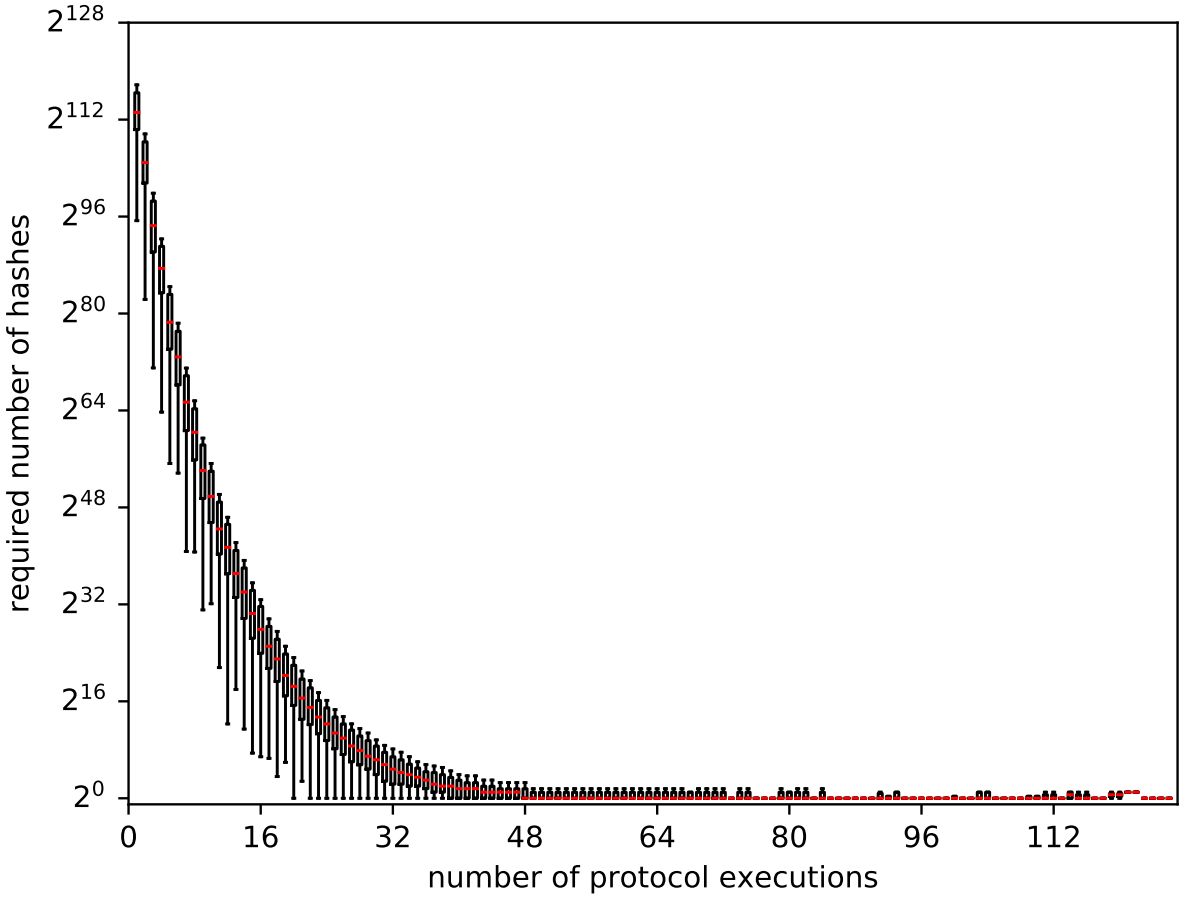


Figure 3.9: Expected number of offline hashes required for extracting a 128-bit key with Algorithm 3.1, assuming the adversary initiates the offline part of the attack after the given number of protocol executions. The results are based on 1000 full attack simulations.

fully leaked yet (ℓ) and compute the Hamming distance $D = \mathrm{HD}(k, k')$ between the actual key and the attackers best guess $\lfloor k' \rceil$, obtained by rounding all values of k'. Assuming the attacker iterates key-candidates in increasing Hamming distance to $\lfloor k' \rceil$, the expected number of hashes until a collision is found is given by

$$\sum_{d=0}^{D-1} \binom{\ell}{d} + \frac{1}{2}\binom{\ell}{D}.$$

This number is computed after each session until the entire key has been extracted without an offline phase. The results of repeating this process several times is given in Figure 3.9. This strategy may have some false positives caused by hash collisions in the leaked bits, but after attacking a few sessions this occurs with only negligible probability.

Consider an adversary that can compute approximately 2^{40} hashes. If that adversary overestimates the required number of protocol executions given in Figure 3.9, to prevent false positives, they conclude that 16 sessions should suffice to extract the key with overwhelming probability.

Python code executing the attack against simulated protocol executions is provided online [Ver18], allowing the user to obtain similar results using different key sizes and number of simulations.

4.2 Terrorist fraud

When Enc encrypts using a computational cipher, the partial leakage of d and b does not provide the adversary with sufficient information to extract k. Instead a far-away malicious prover can assist an accomplice close to the verifier to convince the verifier that the prover is nearby, without revealing k to the accomplice. They achieve this by letting the accomplice clone the challenge qubit onto one out of two qubits sent by the prover, without the accomplice ever learning which qubit holds the clone. The no-cloning theorem does not apply because the prover is assisting, and the prover knows the basis of the state to be cloned. The challenge qubit is sent back (unchanged) to the verifier, and the two qubits containing the clone are sent back to the prover. The prover selects the clone and measures in the correct basis, learning c so they can send the MAC to the verifier in the final authentication phase. Note that this final MAC is not part of the timed phase, and thus there is time for the prover to perform this step

Figure 3.10 shows the details of the attack. The accomplice relays the slow initial phase to the prover so the prover can compute a. The prover replies by sending two qubits per

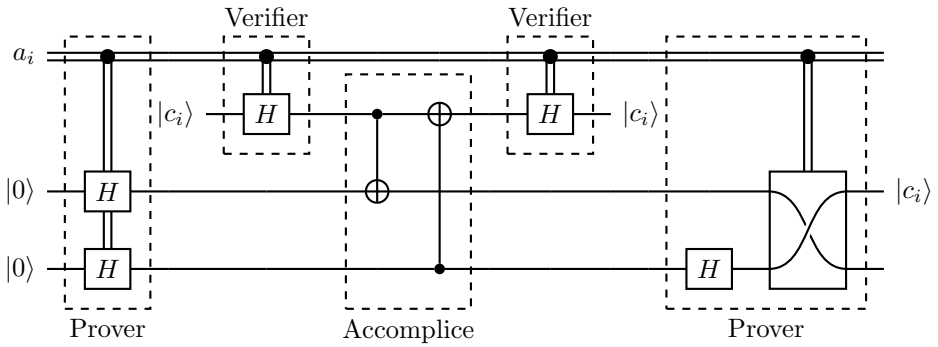


Figure 3.10: Terrorist fraud against the AMSP protocol [AMSP17] with computational encryption.

round to the accomplice: either $|00\rangle$ if $a_i = 0$ or $|{+}{+}\rangle$ otherwise. The accomplice applies two CNOT gates: between qubits one and two and between qubits three and one. Note that the role of the control- and target-qubit is swapped in the Hadamard basis. If $a_i = 0$ the first CNOT clones $|\phi_i\rangle$ and the second gate does not alter the state, resulting in the state $|\phi_i\rangle|\phi_i\rangle|0\rangle$, where $|\phi_i\rangle = |c_i\rangle$. If $a_i = 1$ the second CNOT clones $|\phi_i\rangle$ and the first gate does nothing, resulting in the state $|\phi_i\rangle|0\rangle|\phi_i\rangle$ where $|\phi_i\rangle = H|c_i\rangle$.

It remains to show that the provided information does not leak k to the accomplice. For the accomplice to learn k, they must distinguish the cases $a_i = 0$ from $a_i = 1$ based on the messages of both verifier and prover. These received states have trace distance bounded by

$$\begin{aligned}
&\left\| \frac{1}{2}\left(|0\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes |00\rangle\langle 00| - \frac{1}{2}\left(|+\rangle\langle +| + |-\rangle\langle -|\right) \otimes |{+}{+}\rangle\langle {+}{+}| \right\|_1 \\
&\leq \left\| \frac{1}{2}I, \frac{1}{2}I \right\|_1 + \left\| |00\rangle\langle 00| - |{+}{+}\rangle\langle {+}{+}| \right\|_1 \qquad (3.21) \\
&= 0 + \sqrt{1 - |\langle 00| + +\rangle|^2} = \frac{\sqrt{3}}{2},
\end{aligned}$$

so the Holevo-Helstrom theorem provides the upper bound $\frac{2+\sqrt{3}}{4}$ on the probability that the accomplice successfully distinguishes these states. To extract the key k, the accomplice cannot do better than a product measurement on the n qubits, succeeding with probability $\left(\frac{2+\sqrt{3}}{4}\right)^n \approx 0.93^n$, which is negligible in n and thus the protocol is vulnerable to TF.

4.2.1 Non-asymptotic analysis

Although the above argument shows that asymptotically TF is always possible, that may not be true when instantiated with actual parameters. With the above construction the dishonest prover may leak too much information to the accomplice. The accomplice can measure the prover-provided state of Figure 3.10 and learn approximately 93% of the bits of a. The accomplice then brute-force searches for a k so that the output of f_k matches the observed output, where we note that the accomplice can extract output bits of f_k using the strategies described in Section 4.1. The accomplice can iterate the guesses for k in order of increasing Hamming distance to the measurement outcome. This requires an expected $\sum_{i=0}^{0.07n} \binom{n}{i}$ decryptions and PRF calls. For example if encryption is 128-bit AES and thus $n = 256$, this evaluates to approximately 2^{87} calls. If the encryption or PRF has any weaknesses when partial information about the key is known, faster attacks than such bruteforcing might be possible. The dishonest prover might conclude that this is not enough work to prevent the accomplice from learning the key, and therefore they do not want to assist in TF.

The above could be interpreted as an argument that the protocol does protect protect against TF. However we remark that interpretation is not without risk, as it puts an unusual burden on the implementer that has to choose security parameters. For almost all cryptographic systems, choosing larger parameters increases security. If the instantiation of this protocol should be TF resistant, a larger key is not always better: the key should be small enough to allow a sufficiently powerful accomplice to extract the key from the dishonest prover, but at the same time it should be large enough to prevent brute-force attacks on g. Choosing the correct parameter relies on an accurate estimation of both current and future adversary computing power. This makes the protocol undesirable compared to classical distance bounding protocols, whose security relies only on upper bound estimates of adversary computing power.

4.3 Improved analysis

We consider the protocol without countermeasures against TF: directly use a PRF for f, since those countermeasures either leak the key or do not prevent TF.

For DF, the protocol authors give a success probability of $(1/2)^n$ [AMSP17]. This is correct and easy to see, since both prover and verifier have complete knowledge over the encoding bases the protocol can be considered classical (not quantum) and an optimal strategy for the prover is to guess every challenge bit c_i.

For MF the original analysis [AMSP17] suggests the following attack: "the mafia fraud attack succeeds with probability $(3/4)^n$, i.e. an adversary can pre-ask the prover for responses—if he guesses the pre-asked challenge correctly he always wins the round, otherwise he needs to guess the response with probability $(1/2)$." However this attack does not work: it does not address how the adversary distinguishes between incorrect and correct challenges and does not consider the computation of the correct MAC value. Perhaps surprisingly, the success probability of the best known attack is also $(3/4)^n$, despite the fact that it requires a completely different strategy, as I will discuss next.

The security of the protocol follows from the no-cloning theorem [WZ82]: after all an adversary that can perfectly clone $|\phi_i\rangle$ can reflect the original to the verifier and send the clone to the prover and thus successfully complete a relaying attack. The no-cloning theorem only rules out perfect cloning devices but does not rule out imperfect devices. These devices have been the subject of much research [BH96; Wer98; BCMM00; MVW13], providing both constructions of the devices and proofs of their optimality in different metrics.

The setting for MF is similar to that of a 1-to-2 cloning device of Wiesner's quantum money [Wie83]. In that scenario the bank provides a banknote that contains some qubits encoded in non-orthogonal states. A customer that wants to verify the validity of the note gives it to the bank so they can measure the qubits and check if they are in the correct state. A 1-to-2 cloning device attempts to create two copies from a single bank note so when the bank measures each note individually they both pass the verification. The task for the distance bounding adversary is similar, where both the prover and verifier as representing the bank as they will both do an independent measurement on the qubits that the adversary sends them.

More formally the challenge qubit $|\phi_i\rangle$ is described by one of four states, each with probability $1/4$. Consider the quantum channel Φ, representing the cloning device that takes a single qubit $|\phi_i\rangle$ as input and returns a two-qubit state, one shall be sent to either party. The success probability of the attack on a single qubit is then given by the average

$$\frac{1}{4}(\langle 00|\Phi(|0\rangle\langle 0|)|00\rangle + \langle 11|\Phi(|1\rangle\langle 1|)|11\rangle + \langle ++|\Phi(|+\rangle\langle +|)|++\rangle + \langle --|\Phi(|-\rangle\langle -|)|--\rangle), \quad (3.22)$$

which is optimal at the value $3/4$. An explicit expression for Φ and a proof of its optimality is given by Molina, Vidick and Watrous [MVW13]. Furthermore their work uses the product rule to prove that n of these qubits (or similar: n rapid rounds) can be cloned successfully with probability $(3/4)^n$.

There could exist an attack that improves on this result by exploiting the differences between the quantum money scenario and distance bounding, such as the pseudo-random choice of bases or some other interaction with the prover and/or verifier that is not available in the quantum money scenario.

5 Abidin's protocol

The 2019 protocol by Abidin [Abi19], shown in Figure 3.11, removes the final authentication phase by encoding the challenge and response in different bases. Specifically the verifier sends challenge

$$|\phi_i\rangle = H^{d_i}|c_i\rangle, \tag{3.23}$$

which the prover measures with outcome c_i'. The prover responds with

$$|\psi_i\rangle = H^{b_i}|c_i'\rangle. \tag{3.24}$$

The initialization phase remains the same: the verifier and prover exchange nonces; and there is no final authentication phase.

5.1 Key-extraction under XOR encryption

When instantiating Abidin's protocol with XOR encryption (Equation (3.13)), we can extract the key bits. The attack described in Section 6.1 is available to the adversary, but here we describe a more efficient attack. The adversary impersonates the verifier to the prover to gain information about k_i. The adversary interacts with the prover only, so this attack can be executed at arbitrary distance from the verifier.

The adversary sends

$$|\xi\rangle = \cos\frac{3\pi}{8}|0\rangle + \sin\frac{3\pi}{8}|1\rangle \tag{3.25}$$

to the prover, which is the state "between" $|1\rangle$ and $|+\rangle$, as visualized in Figure 3.12. Specifically $|\langle 1|\xi\rangle|^2 = |\langle +|\xi\rangle|^2 = \frac{2+\sqrt{2}}{4}$. Let $|\xi^\perp\rangle = \cos\frac{-\pi}{8}|0\rangle + \sin\frac{-\pi}{8}|1\rangle$, so the adversary measures the reply $|\psi\rangle$ in the orthonormal basis $\{|\xi\rangle, |\xi^\perp\rangle\}$ with outcomes $c_i'' = 0$ or $c_i'' = 1$, respectively. The prover measurement likely collapses $|\xi\rangle$ to $|1\rangle$ (resp. $|+\rangle$) if $d_i = 0$ (resp. $d_i = 1$). When $k_i = 0$ and thus $b_i = d_i$, the prover replies with that collapsed state, so the adversary likely measures $c_i'' = 0$. Otherwise if $k_i = 1$, then upon measuring $|1\rangle$ (resp. $|+\rangle$) the prover replies with $|-\rangle$ (resp. $|0\rangle$), which is close to $|\xi^\perp\rangle$ and the adversary likely

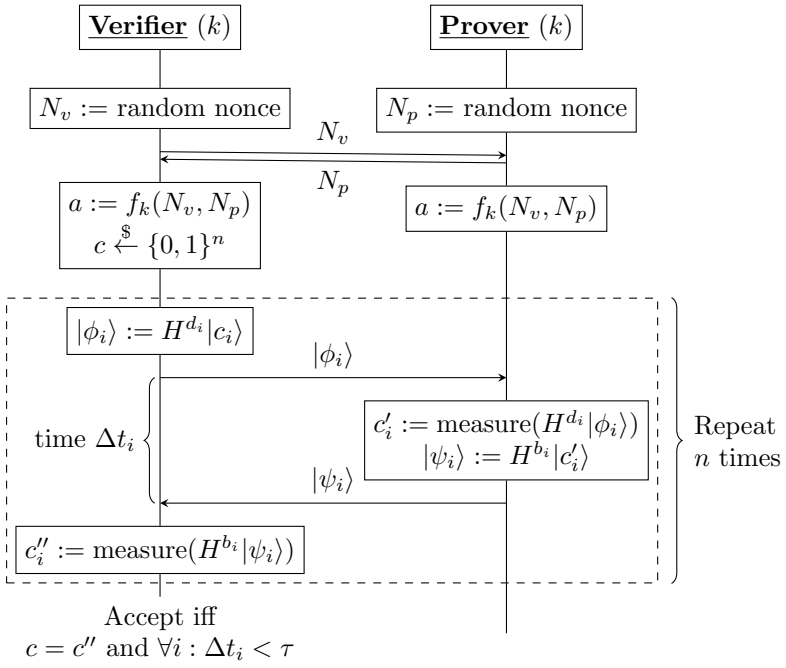


Figure 3.11: Abidin's quantum distance bounding protocol [Abi19]. The relation between a and (d, b) depends on the countermeasure against TF.

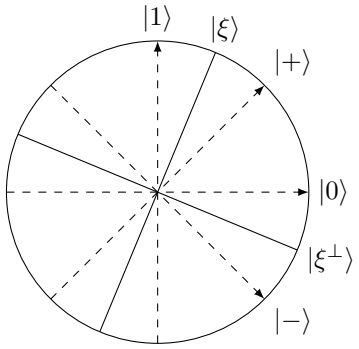


Figure 3.12: Non-orthogonal bases.

measures $c_i'' = 1$. The adversary can attack every round in a single distance bounding session and then repeat the attack for multiple sessions. If $c_i'' = 1$ occurs more often than $c_i'' = 0$, they guess $k_i = 1$, otherwise they guess $k_i = 0$. Note that each k_i is independent of each other. Optionally when the adversary is sufficiently confident that all guesses k_i are correct, they can confirm their guess for k by trying to impersonate the prover to the verifier.

The following analysis shows that the attack is effective and efficient. We have

$$\begin{aligned}\Pr[c_i'' = 0 \mid k_i = 0] &= |\langle\xi|1\rangle|^2|\langle 1|\xi\rangle|^2 + |\langle\xi|0\rangle|^2|\langle 0|\xi\rangle|^2 \\ &= \left(\frac{2+\sqrt{2}}{4}\right)^2 + \left(\frac{2-\sqrt{2}}{4}\right)^2 = \frac{3}{4} \\ &= |\langle\xi|+\rangle|^2|\langle +|\xi\rangle|^2 + |\langle\xi|-\rangle|^2|\langle -|\xi\rangle|^2,\end{aligned} \tag{3.26}$$

where the last line considers the case $d_i = 1$ to highlight the independence on d_i. Similarly we have

$$\begin{aligned}\Pr[c_i'' = 0 \mid k_i = 1] &= |\langle\xi|+\rangle|^2|\langle 0|\xi\rangle|^2 + |\langle\xi|-\rangle|^2|\langle 1|\xi\rangle|^2 \\ &= 2\left(\frac{2+\sqrt{2}}{4}\right)\left(\frac{2-\sqrt{2}}{4}\right) = \frac{1}{4} \\ &= |\langle\xi|0\rangle|^2|\langle +|\xi\rangle|^2 + |\langle\xi|1\rangle|^2|\langle -|\xi\rangle|^2,\end{aligned} \tag{3.27}$$

showing that this attack is a Bernoulli trial with success event $c_i'' = k_i$ and probability of success 3/4. The number of success events thus follows a binomial distribution and the strategy of guessing k_i as the majority of outcomes c_i'' fails if more than half the Bernoulli trials fail. After attacking R rounds we have $\Pr[\#(c_i'' = k_i) \leq R/2] \leq \exp(-2(1/4)^2 R) = \exp(-R/8)$ by Hoeffding's tail bound on the binomial distribution, so the error for the guess k_i becomes negligible in R.

Write k_i' for the majority of outcomes c_i'', then the probability p that the full guess k' is correct is

$$\begin{aligned}p = \Pr[k' = k] &= \prod_{i=1}^{n} \Pr[k_i' = k_i] \\ &= (1 - \Pr[\#(c_i'' = k_i) \leq R/2])^n \\ &\geq (1 - e^{-R/8})^n.\end{aligned} \tag{3.28}$$

So attacking $R = -8\ln(1 - \sqrt[n]{p})$ sessions extracts the full key with probability p. For example if $n = 128$, then attacking $R = 57$ rounds extracts the key with probability $p \geq 0.9$.

5.2 Terrorist fraud

Instead of measuring $|\phi_i\rangle$ and re-encoding $|\psi_i\rangle$ (as stated in [Abi19]) the prover also succeeds by replying

$$|\psi_i\rangle = H^{d_i \oplus b_i} |\phi_i\rangle, \tag{3.29}$$

which does not measure anything and applies a Hadamard to the challenge qubit conditional on the value of $d_i \oplus b_i$. Therefore this protocol does not prove to the verifier that the prover knows both d and b, only that they know $d \oplus b$. This also means that a malicious prover can give $d \oplus b$ to the accomplice for achieving TF.

Assuming g is a secure PRF and Enc is a secure encryption function, the value $d \oplus b$ does not leak k to the computationally bounded accomplice. To see that the accomplice learns nothing about d from interaction with the verifier, compare the density operators of the challenge state given the different states of d_i:

$$\frac{1}{2}\left(|0\rangle\langle 0| + |1\rangle\langle 1|\right) = \frac{1}{2}\left(|+\rangle\langle +| + |-\rangle\langle -|\right), \tag{3.30}$$

which means that $|\phi_i\rangle$ does not provide any information about d_i, so the accomplice cannot use the verifier to learn k. Note that the attack described in Section 6.1 may reveal partial information of d and b, but this is insufficient to reveal k.

5.3 Improved analysis

Consider the protocol without any countermeasure to TF, so the key is just split in two parts as $d \| b := g_k(N_v, N_p)$. The original analysis [Abi19] computes the success probability of both DF and MF as a function of $\mathrm{HD}(d, b)$: the Hamming distance between d and b. In some protocols this makes sense:[8] sometimes DF is easier depending on $d_i \oplus b_i$, so that a dishonest prover can lower $\mathrm{HD}(d, b)$ by evaluating g_k on many N_p. For this protocol however, it does not help with DF. In MF the prover is honest, so that analysis should assume $\Pr[d_i = b_i] = 1/2$.

The original analysis [Abi19] claims that if $d_i = b_i$ the prover can reflect the challenge qubit instead of measuring and re-encoding it, thereby reducing the required processing time and increasing the distance slightly. I do not consider this to be DF, since the prover must still be within the distance imposed by the verifier threshold τ for this to work. The verifier should assume zero computation time when calculating the upper bound of the

[8] For example, this is relevant for the protocol I analyze in Section 6.3.

prover distance. Instead, the adversary can send $|\psi_i\rangle = |0\rangle$, succeeding with probability 1/2. The protocol with n rapid rounds thus achieves a $(1/2)^n$ success probability for DF, instead of the claimed $(1/2)^{\mathrm{HD}(d,b)}$.

I summarize the original analysis for MF: it considers three strategies, pre-ask, post-asking or simply reflecting the challenges,[9] but limits the adversary to the computational and Hadamard basis. Pre-asking $|\phi_i\rangle \xleftarrow{\$} \{|0\rangle, |1\rangle, |-\rangle, |+\rangle\}$ gets the correct response $|\phi_i\rangle$ from the prover with probability 1/2. In the suggested post-ask strategy, the adversary measures and replies in classically guessed bases. That succeeds when the adversary guesses both d_i and b_i correct, otherwise $\Pr[c_i = c_i''] = 1/2$, so this succeeds with probability 5/8. Reflecting succeeds if $d_i = b_i$, otherwise $\Pr[c_i = c_i''] = 1/2$, so this has success probability of 3/4.

Instead we combine the ideas described earlier to show a better pre-ask strategy exists (without claiming this attack is optimal). The adversary first relays the nonces so the prover and verifier will compute the same d and b. The adversary then executes the key-extraction attack of Section 5.1 on the prover, resulting in a guess k_i' for $d_i \oplus b_i$ which is correct with probability 3/4. The adversary now executes the TF attack of Section 5.2 with the verifier, sending $H^{k_i'}|\phi\rangle$ as response. If k_i' was indeed correct the verifier will accept, otherwise their measurement outcome is only correct with probability 1/2. Thus each rapid round is successfully attacked with probability $3/4 + (1/4)(1/2) = 7/8$ and attacking n rapid rounds succeeds with probability $(7/8)^n$.

6 Improved RAD protocol

The relay attack detection (RAD) protocol [JA16] has neither a timed phase nor a randomized challenge, so despite the name it does not detect relaying attacks, as was observed by Abidin [Abi20]. In that same work Abidin proposes the improved RAD (IRAD) protocol as a fix, shown in Figure 3.13.

After the initial phase of exchanging nonces, the verifier sends a classical challenge bit c_i and the prover replies with qubit

$$|\psi_i\rangle = \begin{cases} H^{d_i}|0\rangle & \text{if } c_i' = 0 \\ H^{b_i}|1\rangle & \text{if } c_i' = 1, \end{cases} \tag{3.31}$$

there is no final authentication phase.

[9] Without a authentication phase, the post-ask and reflection strategy do not require the prover, so that technically these describe impersonation attacks.

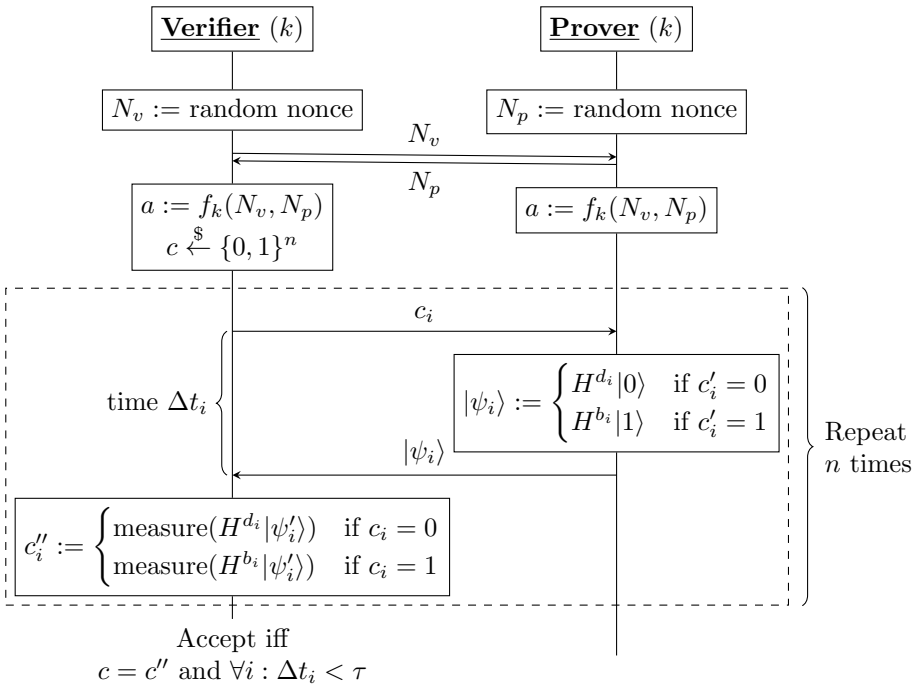


Figure 3.13: The IRAD quantum distance bounding protocol [Abi20]

6.1 Key-extraction under XOR encryption

When Enc is implemented as a one-time pad, an adversary can extract the key k. The attack we propose is an adaptation of the attack on the Tu and Piramuthu protocol [TP07] described in [KAK+08]. The adversary tampers with only a single rapid round, relays all other rounds between the verifier and prover, and then observes if the verifier accepts. In the classical setting this accept/reject bit always leaks one bit of the key k_i, but in the quantum setting it introduces a one-sided error, which will eventually leak the bit k_i. Since the attack requires tampering with a legitimate session, it is assumed that the adversary is able to set up a person-in-the-middle (PITM) attack between the verifier and the nearby prover, which might be harder to execute in practice than the previously discussed key-extraction attacks.

Specifically the adversary can target any round i and flips the challenge bit c_i. The adversary then relays the prover response $|\psi_i\rangle$ to the verifier. If $d_i = b_i$, then the verifier measures $|\psi_i\rangle$ in the correct basis but with outcome $c_i'' \neq c_i$ and the verifier rejects. Otherwise if $d_i \neq b_i$ then the verifier gets a uniform random measurement outcome c_i'' and may accept. If the verifier accepts the adversary concludes that $d_i \neq b_i$ and $k_i = 1$, but if the verifier rejects the adversary is not sure about the value of k_i (although it is more likely to be zero).

A similar strategy using the accept/reject bit can, with 50% probability, leak that $d_i = b_i$ and in that event the adversary knows $k_i = 0$ with certainty. This time the adversary both flips the challenge bit c_i and applies a Hadamard gate to the prover reply $|\psi_i\rangle$. The verifier measures $H|\psi_i\rangle$ in the correct basis if $d_i \neq b_i$ and rejects, otherwise they accept with probability 1/2.

A strategy for extracting the entire key k is to alternate the above two attacks on a single round i until k_i leaks and repeating it on all n bits. Note that if the verifier rejects after the first attack, relaying $|\psi_i\rangle$, the adversary has gained partial information about k_i: $\Pr[k_i = 0 \,|\, \text{verifier rejects}] = (1/2)(1/2)/(3/4) = 1/3$ by Bayes' theorem. If the verifier also rejects after the second attack, relaying $H|\psi_i\rangle$, the adversary considers k_i to be a uniform random bit again. To extract a single bit k_i has expected runtime

$$\begin{aligned}
&\sum_{i=0}^{\infty}(2i+1)\left(\frac{3}{4}\right)^i\left(\frac{2}{3}\right)^i\left(\frac{1}{4}\right)+\sum_{i=1}^{\infty}(2i)\left(\frac{3}{4}\right)^i\left(\frac{2}{3}\right)^{i-1}\left(\frac{1}{3}\right)\\
&=\frac{2}{4}\sum_{i=0}^{\infty}i\left(\frac{2}{4}\right)^i+\frac{1}{4}\sum_{i=0}^{\infty}\left(\frac{2}{4}\right)^i+\left(\frac{2}{3}\right)\left(\frac{3}{2}\right)\sum_{i=1}^{\infty}i\left(\frac{2}{4}\right)^i \qquad (3.32)\\
&=1+\frac{1}{2}+2=\frac{7}{2}
\end{aligned}$$

and the attack extracts the full key after an expected $7n/2$ sessions. The adversary can extract some bits of d_i by sending $c_i = 0$, providing a target for bruteforcing f to extract k when it has only partially leaked through the above attack.

Alternatively the adversary can (in addition to flipping the challenge bit that gets passed to the prover) apply the gate XZ (or XZH) to the prover response $|\psi_i\rangle$ so if the verifier *rejects* it leaks the bit to the adversary. This does not change the expected runtime, but may be harder to detect by the legitimate users as it results in fewer overall rejections.

6.2 Terrorist fraud

When Enc is implemented as a computational cipher it does not protect against TF. The malicious prover can send two qubits to the accomplice: $H^{d_i}|0\rangle$ and $H^{b_i}|1\rangle$, and the accomplice selects the correct reply depending on the challenge bit c_i.

The prover does not reveal d and b to the accomplice by sending those qubits. The Holevo-Helstrom theorem provides the bound $\frac{1}{2} + \frac{1}{2}\sqrt{1 - |\langle 0|+\rangle|^2} = \frac{2+\sqrt{2}}{4}$ on what the accomplice can learn per bit d_i (and the same value for b_i). This bound is achieved by the optimal measurement in the basis $\{|\xi\rangle, |\xi^\perp\rangle\}$ as defined in Equation (3.25). Assuming that g and Enc are secure functions, the accomplice can do no better than a product measurement to guess all bits of d and b. The overall success probability is $\left(\frac{2+\sqrt{2}}{4}\right)^{2n} = \left(\frac{3+2\sqrt{2}}{8}\right)^{n} \approx 0.73^n$, meaning that the accomplice has only negligible probability of extracting k and thus the protocol is vulnerable to TF.

Again the argument is asymptotically sufficient, but here we show that a brute-force attack by the accomplice is not possible either in the real world. The accomplice learns approximately 85% of both n-bit strings d and b and can extract k with $\sum_{i=0}^{0.3n} \binom{2n}{i}$ expected decryptions and PRF calls. However, for 128-bit AES and $n = 128$ this evaluates to approximately 2^{151} calls, so the dishonest prover can send the two qubits without leaking too much and TF is possible.

6.3 Improved analysis

The original analysis claims DF success probability $(1/2)^n$, because it only considers an adversary limited to the computational and Hadamard bases.

For DF we distinguish three cases. When $d_i = b_i$, the correct replies (depending on c_i) are orthogonal and thus the prover cannot do better than a random guess for c_i. However

when $d_i = 1$ and $b_i = 0$ the correct response is $|+\rangle$ or $|1\rangle$, so sending $|\xi\rangle$ provides a $\frac{2+\sqrt{2}}{4}$ probability that the verifier measurement outcome is correct, independent of the actual challenge bit that was sent. Similarly when $d_i = 0$ and $b_i = 1$ the verifier will likely accept the reply $|\xi^{\perp}\rangle$. The success probability for DF is thus given by

$$\left(\frac{1}{2}\right)^{n-\mathrm{HD}(d,b)} \left(\frac{2+\sqrt{2}}{4}\right)^{\mathrm{HD}(d,b)} . \tag{3.33}$$

For an unbiased PRF f it holds that $\Pr[d_i = b_i] = 1/2$, so when trying a single input N_p the success probability for DF is

$$\left(\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{2+\sqrt{2}}{4}\right)\right)^n = \left(\frac{4+\sqrt{2}}{8}\right)^n \approx 0.68^n. \tag{3.34}$$

In order to maximize the success probability of DF, the dishonest prover should maximize $\mathrm{HD}(d, b)$. The prover can compute $a := f_k(N_v, N_p)$ for many nonces N_p and then send the one with the largest $\mathrm{HD}(d, b)$. The exact number of tries the dishonest prover gets depends on their computing power and the maximum delay the verifier allows for the reply N_p in the slow phase. Asymptotically, $\max_{d,b}\{\mathrm{HD}(d, b)\}$ approaches n as the number of tries is increased, so that the success probability for DF asymptotically approaches

$$\left(\frac{2+\sqrt{2}}{4}\right)^n \approx 0.85^n. \tag{3.35}$$

The original MF-analysis considered a straightforward pre-ask strategy: the adversary guesses the challenge when pre-asking the prover, and if it turns out correct then just forward the prover's reply otherwise encode the verifier's challenge in a random basis (computational or Hadamard). Note that this strategy succeeds if the guessed challenge, or the guessed encoding basis, or the verifier's measurement is correct, so that this succeeds with probability 7/8 (and not 3/4 as calculated in the original analysis).

The adversary can also do better, using a similar pre-ask strategy. Instead of guessing a random basis when the guessed challenge was incorrect, the adversary can send a state that will very likely be accepted: the state "between" $|0\rangle$ and $|+\rangle$ (or between $|1\rangle$ and $|-\rangle$). Specifically, if $c_i = 0$ the adversary replies with $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$ and if $c_i = 1$ the adversary replies with $\cos(5\pi/8)|0\rangle + \sin(5\pi/8)|1\rangle$, which in either case the verifier accepts with probability $\frac{2+\sqrt{2}}{4}$. Thus each round is attacked successfully with probability $\frac{1}{2} + \left(\frac{1}{2}\right)\left(\frac{2+\sqrt{2}}{4}\right) = \frac{6+\sqrt{2}}{8}$ and overall probability $\left(\frac{6+\sqrt{2}}{8}\right)^n \approx 0.93^n$ for MF.

7 Fixing the IRAD protocol

Since the key extraction attack is similar to the attack described by Kim, Avoine, Koeune, Standaert and Pereira [KAK+08], we could apply the same countermeasure that they suggest. In this section I show that the protocol will indeed be secure and prevents TF, but a comparison with the Swiss-Knife protocol reveals that the classical protocol is *more* secure than the quantum protocol.

The fix is to conclude the IRAD protocol with a final authentication phase where the prover sends $c' = (c'_1, \cdots c'_n)$ and an authentication tag

$$\mathrm{MAC}_k(c', id_p, N_v, N_p) \tag{3.36}$$

to the verifier, where id_p is some identifier for the prover. The verifier accepts only if $c = c' = c''$ and the tag is valid. Any attempt to learn a bit k_i by flipping a challenge bit will be detected since $c_i \neq c'_i$ and in that case the protocol rejects without leaking a key bit. The encryption must be a one-time pad, otherwise TF would be possible using the attack of Section 6.2. The fixed protocol is depicted in Figure 3.14.

7.1 Analysis

The analysis for DF in the fixed protocol does not change, so that the success probability is given by Equation (3.33).

The MF attack of Section 6.3 will not work. Any pre-ask strategy will have to guess all challenges c_i correct in order to retrieve the correct MAC tag. Instead a post-ask strategy will perform better: the adversary tries to give the correct responses to the challenges, then forwards the challenges to the prover in order to get the correct MAC tag. The response is the same as described in Section 6.3, so each round is attacked successful with probability $\frac{2+\sqrt{2}}{4}$ and overall probability $\left(\frac{2+\sqrt{2}}{4}\right)^n \approx 0.85^n$.

TF is prevented by the fix. If the dishonest prover provides $H^{d_i}|0\rangle$ and $H^{b_i}|1\rangle$ to the accomplice, this leaks information about the key bit k_i: by the analysis of Section 6.2 the values leak $\frac{2+\sqrt{2}}{4}$ bits about d_i and b_i, respectively. Since $k = d \oplus b$, sending both values leaks $\frac{3+2\sqrt{2}}{8} \approx 0.73$ bits of information about k_i to the accomplice. Although sending these values for every round does not leak the entire key k, it likely provides sufficient information to start a brute-force to extract the rest of the key, for example against the MAC sent in the final phase. Furthermore, if the prover engages in TF just a few times in this manner, the leaked information per bit k_i rapidly increases to one.

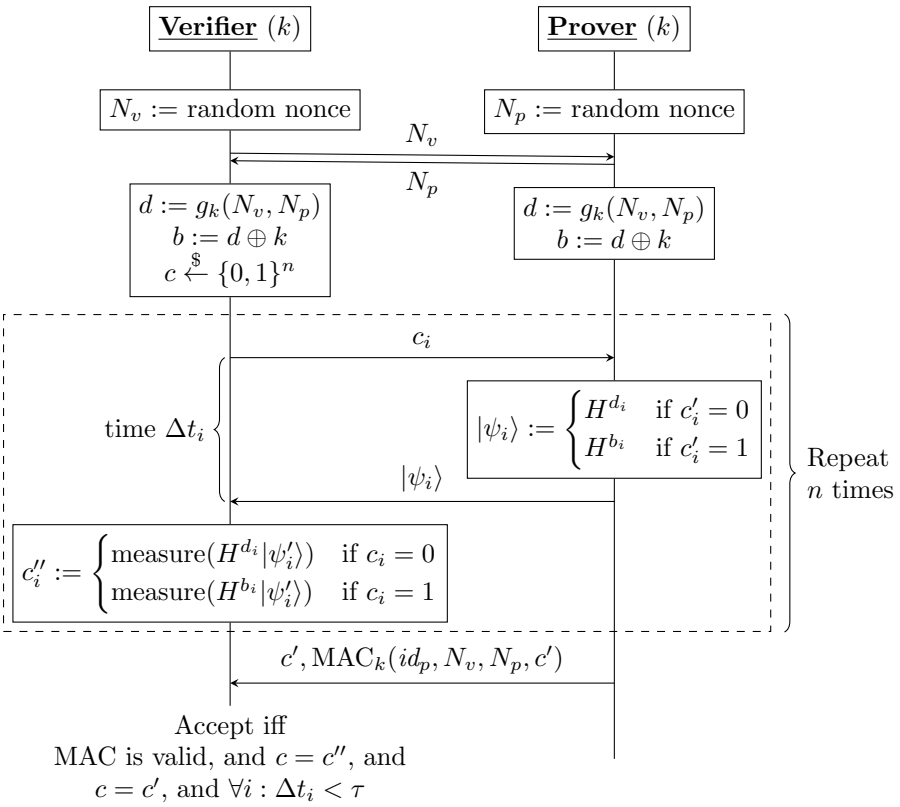


Figure 3.14: The IRAD protocol with a countermeasure against key extraction.

However, the adversary can also help the accomplice by sending only $H^{d_i}|0\rangle$. This leaks partial information about d_i, but by the nature of the one-time pad it leaks nothing about k_i. Similarly the prover could send only $H^{b_i}|1\rangle$, but never both possible responses for the same round. Effectively the dishonest prover allows the pre-ask strategy for MF described in Section 6.3: giving a TF success probability of $\left(\frac{6+\sqrt{2}}{8}\right)^n \approx 0.93^n$.

7.2 Comparison to Swiss-Knife

In order to fix the IRAD protocol, I had to make it almost identical to the basic version of the Swiss-Knife protocol [KAK+08] discussed in Section 2.1, with the only difference that the prover responds with $|\psi_i\rangle$ instead of

$$r_i = \begin{cases} d_i & \text{if } c'_i = 0 \\ b_i & \text{if } c'_i = 1. \end{cases} \tag{3.37}$$

The best known attacks on the Swiss-Knife protocol achieve DF with probability $(3/4)^n$, MF with probability $(1/2)^n$ and TF with probability $(3/4)^n$, as also summarized Table 3.1. Introducing qubits in the rapid phase thus only diminishes the security.

8 Discussion

This paper has demonstrated that none of the (three) existing quantum distance bounding protocols [AMSP17; Abi19; Abi20] protect against TF. In case the TF countermeasure is implemented with a one-time pad, the long-term key leaks. The problem in the AMSP protocol and Abidin's protocol is that some information about the encoding bases leaks, which reveals sufficient information about the long term key to fully extract it. Instead in the IRAD protocol one key bit may leak through the accept/reject output bit of the protocol itself.

When the TF countermeasure is instead implemented with computational cryptography, it no longer provides meaningful protection against TF. Against the AMSP protocol the dishonest prover can assist the accomplice in order to perfectly clone a qubit, without knowing where it was cloned to. A case could be made that this protocol does provide *some* resistance against TF, since the provided state *might* allow a brute-force attack on the key. Nevertheless this protection seems much weaker than what can be achieved in classical distance bounding. Against Abidin's protocol and the IRAD protocol, the prover

Table 3.1: Success probability of best-known attacks on distance bounding protocols with n rapid rounds. The quantum distance bounding protocols are implemented without any TF resistance. See Section 4.2.1 for further discussion of TF resistance of the AMSP protocol.

	DF	MF	TF
AMSP [AMSP17]	$(1/2)^n$	$(3/4)^n$	1^*
Abidin [Abi19]	$(1/2)^n$	$(7/8)^n$	1
IRAD [Abi20]	$\lesssim 0.85^n$	$\approx 0.93^n$	1
fixed IRAD	$\lesssim 0.85^n$	$\approx 0.85^n$	$\approx 0.93^n$
Swiss-Knife [KAK+08]	$(3/4)^n$	$(1/2)^n$	$(3/4)^n$
Brands-Chaum [BC93]	$(1/2)^n$	$(1/2)^n$	1

can provide the accomplice with information to complete the protocol himself, without leaking too much information about the long-term key.

Some scenarios for distance bounding do not require TF resistance so that the attacks of this paper do not apply. I summarize the result of the improved analysis in Table 3.1, where the quantum distance bounding protocols are assumed to be implemented without any TF countermeasure. However in that case, both Abidin's protocol and the IRAD protocol provide less security than originally claimed (Sections 5.3 and 6.3), and more importantly it provides less security than existing classical protocols [ABB+18]. Even when fixing TF resistance in the IRAD protocol, the result performs worse than the Swiss-Knife protocol. The AMSP protocol has better DF resistance than most existing RFID protocols, at the cost of having worse MF resistance. However, it requires both a single-photon source and detector, so that the practicality in RFID settings is questionable. In that case a comparison against something like the original Brands-Chaum protocol seems more appropriate, and no security is gained by the AMSP protocol.

Unlike most other proposals within the field of quantum cryptography the proposed protocols rely on computational primitives and do not achieve information theoretic security. It might be possible that the different hardware requirements somehow favor these quantum protocols, although my (admittedly limited) knowledge in that area tells me that is unlikely. One issue not mentioned so far is that sending qubits encoded in photon polarisation makes the communication directional, unlike the broadcast signal of RFID communication. Without a clear benefit provided by using quantum communication, it is unlikely that sending qubits in the rapid phase will ever be preferred over sending classical

bits.

Both the attacks and improved analyses on the protocols highlight the importance of unambiguous security definitions and clear security models, both for distance bounding protocols and security protocols in general. At the moment it remains an open question whether the existing models are meaningful against quantum adversaries. I highlight that this remark extends beyond distance bounding protocols that employ quantum information. Obviously the computational primitives used in any distance bounding protocol should be secure against such adversaries, but it should protect against certain quantum strategies. For example, future research could investigate whether a quantum adversary can benefit from nonlocal strategies [CHTW04].

Chapter 4

Key authentication from post-quantum key encapsulation mechanisms and signatures

1 Introduction

Secure messaging has become increasingly popular, with the most notable recent development being the Signal protocol [Mar16a; Mar16b], which brought end-to-end encryption to billions of users.[1] Since its inception, Signal has been studied by many academics, resulting in a formal security analysis of the abstract protocol in a custom security model [CCD+20] and post-quantum secure variants [ACD19; BFG+21a]. Despite the popularity of the Signal protocol, in this chapter I want to focus on the Off-the-Record Messaging (OTR) protocol, a predecessor that introduced a unique solution to the key authentication problem.

The OTR protocol [BGB04] is an instant messaging protocol for secure and deniable conversations. The initial handshake is performed by an authenticated key exchange (AKE), which establishes a shared secret between two public keys. However (as with any secure messaging solution) the AKE is only the first part of *trust establishment*, and additional *key authentication* is required [UDB+15]. The communicating parties should confirm that the public keys that were used in the handshake actually belong to the other

[1] WhatsApp uses the Signal protocol:
https://blog.whatsapp.com/two-billion-users-connecting-the-world-privately.

person. Without key authentication, a person-in-the-middle (PITM) could inject their own key to eavesdrop on and tamper with all communication undetected. Key authentication is therefore required for *binding* the users to their public key.

A widespread method that achieves this is provided by certificates. These are documents containing the user identity and the public key (in addition to some other data), which is then signed by a trusted third party. Centralized solution can be provided by a certificate authority (CA), which is usually part of a larger public key infrastructure (PKI). This solution is most widely known, since it is required in HTTPS [Res00] and it is typically used in other applications over Transport Layer Security (TLS) [Res18]. A decentralized alternative is the web of trust (WOT) as used in Pretty Good Privacy (PGP) [Zim95], where users sign "keys" of other users.[2] In both solutions the signer does not have to be a directly trusted party, but they should be indirectly trusted via a chain of signed certificates/keys. Each "trust hop" comes with an indication of how much the next party can be trusted to certify other parties, where the WOT usually allows for more fine-grained indications. That also means that some form of initial trust establishment is still required. For example, root certificates come preinstalled with many operating systems, which pushes trust further down the chain to device vendors and manufacturers, while WOT solutions require ceremonies for entering the web.

Secure messaging systems generally avoid the above mentioned issues, instead they require direct key authentication. For example, most Signal-based applications operate in the trust on first use (TOFU) model with optional verification: the public keys used in the initial handshake are accepted and an error or warning is presented to the user only if the public key ever changes in the future. Verifying the owner of the key is left as the responsibility of the users, who are provided with a fingerprint (the hash of the public key)[3] for manual out-of-band comparison.

OTR offers another solution [AG07] by assuming that the users share a low-entropy secret, such as a password. After establishing a secure channel with the AKE, the users execute the socialist millionaire protocol (SMP) over that channel: a zero-knowledge protocol for checking the equality of two values. For the input to the SMP the parties hash the user identities, the public keys, the session identifier, and the low-entropy shared secret. This binds the users not just to their public keys, but to the entire session. The SMP is advantageous from a usability perspective: using shared information is natural for humans, it avoids having to expose users to cryptographic concepts such as keys and fingerprints,

[2] In the context of PGP, the "key" being signed is a document containing both the user's long-term public key and identity, making it more like a certificate than a key.

[3] Signal replaced the fingerprint with a safety number: a hash of the user telephone number and public key [Mar17].

and in-band key authentication avoids user errors that occur in setting up an out-of-band channel.

The problem with the SMP is that it is based on the Diffie-Hellman (DH) primitive, and can therefore be broken by a quantum adversary using Shor's algorithm [Sho94]. In this chapter I present a post-quantum solution to allow in-band key authentication as a necessary step towards fully post-quantum OTR. The solution I propose builds (a weakened version of) a private equality test (PET) out of oblivious transfers (OTs). The OT construction is built from ephemeral key exchange (KEX) protocols and a group operation on its public keys. The implemented KEX combines elliptic curve DH (ECDH) with a post-quantum key encapsulation mechanism (KEM) such that it is secure as long as one of them is secure. The PET protocol is then executed over a "pseudo-authenticated" channel, realized from post-quantum signatures. Pseudo-authenticated roughly means that the honest user is guaranteed that they are communicating authentically with someone, but they have no guarantee who is on the other side. KEMs and signatures have been the focus of much of the research in post-quantum cryptography, partially due to the ongoing National Institute for Standards and Technology (NIST) post-quantum cryptography standardization effort [NIST17]. The result is a protocol with somewhat large messages, but built from components that have already been under scrutiny of many cryptographers. Compared to custom solutions, which might be more efficient, this provides a higher level of confidence in its security.

1.1 Contributions

This chapter provides a post-quantum solution to the key authentication problem in secure messaging. The solution first sets up a pseudo-authenticated channel, using post-quantum signatures with ephemeral keys. I provide a weakened version of the PET functionality, which is then executed over that channel. I realize this functionality by adapting an existing protocol [RR17] so that it can provide output to both users. I prove security of the protocol in the universal composability (UC) framework [Can01], using a simpler variant of the framework [CCL15] where possible. The protocol is based on the OT functionality, so that the protocol is quantum secure as long as the used OT is quantum secure.

For the OT I use an existing construction [MR21]. Although the authors claim quantum security of their construction, I identify a few problems that undermine the validity of that claim and provide solutions to some of them. An implementation is available online [Ver21].

1.2 Notation

The notation $[\![p]\!]$ represents the indicator function, which has value 1 if property p is true and 0 if it is false. The notation $a[\cdot]$ represents the entire array a: so when $a[i]$ is defined for $1 \leq i \leq m$, then $a[\cdot] = (a[1], \ldots, a[m])$. When specifying protocols and functionalities, I enumerate steps that must be taken in order, steps listed with bullet points can be taken in arbitrary order, unless explicitly stated otherwise.

2 Background

I briefly discuss OTR in Section 2.1. Section 2.2 considers the usability aspects of key authentication ceremonies. I describe the current solution for key authentication in Section 2.3. In Section 2.4 I outline the idea of a PET, which forms the foundation of the protocol proposed in this chapter. I go into the details of the security model in Sections 2.5 and 2.6. Sections 2.7 and 2.8 provide two constructions (OT and split functionalities), both of which are used in the proposed protocol. Finally, section Section 2.9 discusses an alternative solution for key authentication based on password authenticated key exchanges (PAKEs).

2.1 Off-the-Record Messaging

A historical step towards making encryption available to the masses was PGP [Zim95]. As its popularity increased, the usability problems became increasingly apparent [WT99]. The secure messaging protocol OTR was developed in response [BGB04], which also improved upon security by adding deniability and forward secrecy.

Currently OTR version 4 (OTRv4) is being developed [OTRv4]. The protocol establishes a shared secret between two participants (Alice and Bob) using a deniable authenticated key exchange (DAKE) [UG18], which will act as a seed for the double ratchet algorithm [Mar16a] to authenticate and encrypt messages. The DAKE is based on public-key cryptography: it uses both ECDH and (ring) signatures. The DAKE assumes that Alice and Bob have a way to verify the public key of the other party, as is the norm for AKEs.

Unger and Goldberg suggest mixing the output of a post-quantum KEM with the ECDH output using a key derivation function (KDF) to achieve quantum transitional security. The developers have instead opted to ignore transitional security and went with

a "brace key" that mixes the result of a regular DH key-exchange instead.[4] This means that transitional security can already be achieved with available cryptographic primitives.

For key authentication, OTR assumes no trusted third parties in the form of a PKI or a WOT, but instead assumes that parties verify the keys themselves. They can do this either out-of-band (for example by exchanging key fingerprints in person) or in-band over the encrypted channel, by executing the SMP [BST01], which is a protocol for comparing if two values are equal in zero-knowledge. The SMP is historically tied to the OTR protocol, but there is no technical reason that limits its applicability: the SMP and likewise solutions can be used for key authentication in any secure messaging application.

2.2 Authentication ceremonies

A systematization of knowledge study from 2015 [UDB+15] identifies three key challenges in the design of of secure messaging: trust establishment, conversation security and transport privacy. We focus on the trust establishment, which is the process of users verifying that they are communicating with the intended party. It consists of a KEX using long-term keys (the initial cryptographic handshake), plus key authentication that confirms that the used keys are associated with the intended party. When users are required to actively interact in key authentication it is called an authentication ceremony. A key feature from that study was a comparison of the many existing authentication methods that focussed not only on the security features, but to also compared their usability and adoption properties. While the above study focussed on the properties allowed by the methods, many follow-up studies have assessed the perceived usability by humans (see for example [HLS+21] for a high-level overview of results). These studies indicate that usability is essential, because security is reduced if users do not authenticate keys or make mistakes. I focus on key fingerprint verification methods, because they allow the most security features, especially in situations that do not provide a trusted third party.

Some of the observed usability problems are general, and mostly orthogonal to the method of comparison. These problems must be solved in all authentication ceremonies. For example, many users do not understand the need to authenticate keys at all [HL16; WGH+19], cannot find how to authenticate keys [VWO+18], do not understand the (often limited) feedback received when key authentication succeeds/fails [SHWR16], or do not react adequately to authentication failures [WGH+19].

Currently, the most popular method for key authentication is manual out-of-band fingerprint comparison. Sometimes a QR-code containing the fingerprints is available, which

[4] See https://github.com/otrv4/otrv4/issues/206 for further discussion on this.

can be scanned by the device of the other party if they are in close proximity. For manual fingerprint comparison, it is important that the out-of-band channel is authenticated. Letting users figure this out for themselves often leads to ceremony failures. Examples of observed errors are that users send the fingerprint in-band [VWO+17], users only compare part of the fingerprint or in one direction [VWO+17; NRS18], or users simply toggle the "mark as verified" switch without any check [VWO+18].

I want to highlight one study from 2017 [VWO+17], for the reason that it tested with pairs of friends instead of pairing the participant with a researcher, thereby presenting a more accurate reflection of secure messaging in the real world. The researchers observed that many pairs attempted to authenticate each other based on shared knowledge, indicating that this is natural human behaviour. They also found that the success rate of the authentication ceremony was significantly higher in the Viber application compared to WhatsApp or Facebook Messenger. The difference was that Viber provided the users with a channel (a Viber phone call) to compare the fingerprints, while hiding the fingerprints (and the ability to manually toggle verification) outside the duration of the ceremony. This indicates that such guided in-band ceremonies have better usability, but it should be noted that voice-based authentication is insecure [SSM18] and similarly the security of video-based authentication is threatened by deepfakes.

The SMP allows in-band comparison as key authentication, but requires a shared secret as input from both users. It solves the specific problems of manual out-of-band fingerprint verification: it allows in-band authentication, compares values in both directions,[5] and the verification switch can be toggled according to the protocol outcome instead of manually. Users can authenticate without ever having to be exposed to technical terms such as "key" or "fingerprint", and do not have to solve the problem of finding a properly authenticated channel themselves. Instead this solution leans into the observed natural human behaviour by allowing users to authenticate in a way that many would have done anyway.

Usability issues have been observed with SMP ceremonies [SYG08; AHIC15]. Besides general issues, observed problems specific to the SMP were that users entered secrets in the question field instead of the answer field, and that inputs were case sensitive. A study comparing the usability of authentication ceremonies would be insightful to assess the correctness of my conjecture that the SMP leads to improved authentication ceremonies.[6]

[5] An alternative user interface to the SMP lets one user input a question and expected answer, and the other user will answer the question. Such questions might be more intuitive, but it should be noted that they often only authenticate one person to the other, not in both directions. Indeed the OTR software provides this interface, marking users as verified in one direction only.

[6] The study [AHIC15] attempts this, but study participants were instructed to complete both ceremonies subsequently, making it unclear if the reported success rates per ceremony are independent.

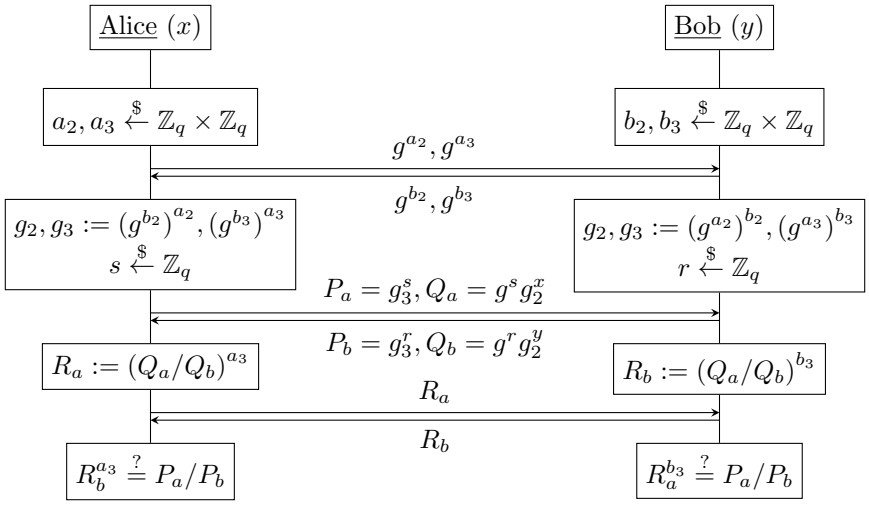


Figure 4.1: The socialist millionaire protocol (SMP) for testing $x = y$ in a group of prime order q. Each message is accompanied by a zero-knowledge proof (not shown) establishing that the message was generated honestly.

2.3 Socialist millionaire protocol

The SMP realizes key authentication with a variant of Yao's millionaire problem. In that problem, two millionaires want to know which of them is richer, without telling the other how much money they have. The *socialist millionaire problem* is a variant, where the millionaires only care if they are equally rich. A solution is given by the SMP [BST01], shown in Figure 4.1. The input to the protocol (x for Alice, y for Bob) is the hash output of the long-term keys used in the DAKE, the user identities, the session id (an output of the DAKE), and a low-entropy secret shared between the users. The zero-knowledge property of the protocol ensures an incorrect guess reveals nothing besides $x \neq y$, while the zero-knowledge proofs within the protocol ensure that any deviations are detected.

Each message sent in the SMP is accompanied by a zero-knowledge proof that it was generated correctly. For the discrete-log messages this is done using the non-interactive version [FS86] of Schnorr's protocol [Sch89]: Alice proves she knows a (belonging to message $m = g^a$). She randomly chooses $r \in \mathbb{Z}_q$, then computes $c = h(g^r)$ (where h is a random oracle) and $d = r - ac \mod q$, and sends (c, d) to Bob, he accepts if $c = h(g^d(g^a)^c)$. Similar constructions, based on the Okamoto's protocol [Oka92] and the Chaum-Pedersen protocol [CP92], prove that the messages (P_a, Q_a) and R_a were formed correctly. Bob accompanies his messages with similar zero-knowledge proofs.

Both the protocol itself and the zero-knowledge proofs rely on the hardness of computing discrete logs. However, quantum adversaries can easily compute this using Shor's algorithm [Sho94]. None of the existing post-quantum primitives provide the necessary operations to *directly* replace the operations in the SMP, so instead of finding a post-quantum variant of the SMP, we provide a post-quantum solution to the underlying problem.

Although the original protocol provides a fair solution [BST01], an unfair variant of the SMP is implemented in OTR: Bob can get his output and abort, never sending the last message to Alice. This drastically lowers the number of messages sent between the parties.

2.4 Private equality test

A PET lets the users privately check if their inputs are equal, and can be expressed as the secure function evaluation (SFE) $(x, y) \mapsto ([\![x = y]\!], [\![x = y]\!])$.

A basic protocol for a PET in the semi-honest model comes from [FNW96]. The basic idea is explained intuitively as a physical interaction. To secretly compare Alice's bitstring $x = x_1 \dots x_n$ with Bob's $y = y_1 \dots y_n$, Alice writes down $2n$ random values $((A_1[0], A_1[1]), \dots, (A_n[0], A_n[1]))$, computes $\bigoplus_{i=1}^{n} A_i[x_i]$, and puts the values in sealed envelopes laid out in pairs. While Alice is not watching, Bob selects envelope y_i from each pair, so that he can compute $\bigoplus_i A_i[y_i]$ when opening the envelopes. Alice destroys the remaining n envelopes. Then the roles reverse: Bob generates $2n$ envelopes with values $((B_1[0], B_1[1]), \dots, (B_n[0], B_n[1]))$ and computes $\bigoplus_i B_i[y_i]$, then Alice selects his envelopes so she can compute $\bigoplus_i B_i[x_i]$. Now Alice writes down $\bigoplus_i A_i[x_i] \oplus B_i[x_i]$ and simultaneously Bob writes down $\bigoplus_i A_i[y_i] \oplus B_i[y_i]$, which they then hand to each other. If they receive the value they just wrote down, they conclude $x = y$ and otherwise $x \neq y$.

The envelopes are an analogy for OT, which allows a digital implementation of the protocol. A version of this protocol has been shown to be UC-secure [RR17] against active adversaries, with the caveat that only Bob receives output. Since sending messages simultaneously is unrealistic in almost all models, we assume Bob sends his message last. The above protocol then becomes insecure, because Bob can simply reflect Alice's message.

2.5 Universal composability

The UC framework provides a strong guarantee of security. In this section I recall the basics of modelling protocols in the UC framework, and go over some details that are

relevant to this chapter.[7] To keep the below description comprehensible, I omit many technical details in the description. See the most recent paper defining the framework for a full description [Can01].

2.5.1 Plain universal composability

At a high level, the UC definition for security is based on the simulation paradigm, see [Lin17] for a well-written overview of the ideas behind and the techniques used in simulation. The framework defines a real model, which contains the protocol execution between parties in the presence of an *adversary*.[8] The framework then defines the ideal model, which contains an *ideal functionality* that specifies what the protocol needs to achieve, but also how much leakage and other adversarial influence on the functionality is allowed. An ideal functionality can be thought of as being executed by a trusted third party. The context in which the protocol is executed is captured by an entity called the *environment*. Through interaction and collaboration with the adversary, the environment tries to distinguish the real model from the ideal model. To prevent this task from being trivial, the ideal model also has an adversary (called the *simulator*) as an interface between the ideal functionality and the environment. A successful simulation in the ideal model ensures that anything the environment observes is indistinguishable from what it observes in the real model, for any instructions it may give the adversary and any input/output of the protocol parties.

Security proofs take the form of constructing a simulator S relative to an adversary A. The simulator then *internally* simulates the interaction between the adversary and the parties while it *externally* interacts with some ideal functionality, which then (indirectly) communicates with the environment Z. Any messages between the adversary and environment are usually forwarded unaltered by the simulator. The simulator is then correct if the *joint* distribution over party inputs, party outputs, protocol messages and messages between the adversary and environment is indistinguishable in the real model and the ideal model. Since the environment acts as an interactive distinguisher, S must be a *straight-line*

[7] I will use the version of the UC framework uploaded in 2020. I also use some UC results from before that time, which are therefore based on older versions of the framework and therefore require minor adjustments to fit in the new framework. While there is no reason to suspect that the differences between versions invalidate those older results or undermine the results of this chapter, the slightly uncomfortable truth is that a formal argument towards this statement is lacking.

[8] This is sometimes referred to as the real *world*, but I reserve that term to refer to implementations of the protocol, which may differ significantly from the model. Talking about models communicates much more directly that UC results, no matter how powerful, are at best results about mathematical abstractions of the real world.

simulator, meaning that partial protocol executions must also be indistinguishable for Z. Most importantly this means that S can not use the simulation technique of rewinding A, since the external environment Z would trivially detect this.[9]

Computation The basic computation unit in the UC framework is an instance of an interactive Turing machine (ITM). Its most important tapes are the (read-only) *identity tape*, the *outgoing message tape*, and three incoming message tapes: for *input*, (subroutine) *output* and *backdoor*. An ITM instance (ITI) M of ITM μ is defined as $M = (\mu, id)$, where $id = (sid, PID)$ is the *identity*, consisting of a *session identifier* and *party identifier*. The contents of the (read-only) identity tape of an ITI M is also referred to as its *extended identity* and equals $(\bar{\mu}, id)$, where $\bar{\mu}$ is the program code of μ. A *protocol instance* or *session* is defined as the set of all ITIs with the same code and session identifier, and each ITI in the session is called a *(main) party*. An ITI receiving input from M or sending subroutine output to M is called a *subroutine* of M.

Execution ITIs can write to the input tapes of other ITIs via an external-write request, which are interpreted and "delivered" by a control function. An external-write request is a message with format (f, M, t, r, M', m). Here M and M' are the extended identity of target and source ITI, respectively. The tape $t \in \{\mathsf{input}, \mathsf{subroutine\text{-}output}, \mathsf{backdoor}\}$ specifies which tape of the target ITI the message m should be written. If the reveal-sender flag r is set, then M' is revealed to M, and if the forced-write flag f is set, then a new ITI with extended identity M is *invoked* if it does not exist yet. If the control function *allows* an external-write request, it writes m to the specified tape and activates the target ITI next. If the external-write request is *disallowed* (or if an ITI halts without an external-write request) then the control function activates the initial ITI next. If an ITI *ignores* a message, they revert their state to their state before receiving the message and halt (and thus the initial ITI is activated next).

Execution of a protocol π in the UC framework is a sequence of activations in the context of an environment Z (with input z) and adversary A. Given ITMs π and A, let $C^{\pi,A}_{\text{EXEC}}$ be the standard UC control function. It activates Z as the initial ITI with identity 0 and z on its input tape. The environment is allowed to invoke other parties, limited to a single session. The control function overwrites the code of environment-invoked ITIs to be $\bar{\pi}$. A ξ-identity-bounded environment can choose any extended identity M' as the source of messages to main parties, as long as it satisfies some predicate ξ (for example, ξ may disallow Z from setting M' to that of any invoked machine). The environment may also

[9] Technically A is internal and *can* be rewound, but Z is external and cannot be rewound.

invoke the adversary, by setting a fixed target identity $\diamond$, in that case the control function overwrites the code of the invoked ITI with $\bar{A}$. This adversary is allowed to send other ITIs messages with f unset and $t = \mathsf{backdoor}$. Other ITIs can send backdoor messages to A with r unset, and they can send input/output messages to other ITIs, which the control function allows if r is set and if the source M' equals the extended identity of the sender. Protocol execution halts when the initial ITI halts with some (binary) output.

Runtime Each message furthermore contains a natural number called the *import*, to bound the ITI runtime: the number of computation steps performed by an instance must at all times be at most some polynomial in the difference between the total received and total sent import. An execution can furthermore be parameterized by a (security) parameter λ, then each ITI only starts when having received an import of at least λ. For security proofs we only consider *balanced* environments, where at any point the total import given to the adversary is at least the sum of imports given to other ITIs. These restrictions ensure that in the dynamic context of the UC each ITI and the overall system run in probabilistic polynomial time (PPT), while preventing the environment from trivially distinguishing the real and ideal model (without any correspondence to a security failure of the protocol).

Emulation Denote $\text{EXEC}_{\pi,A,Z}(\lambda, z)$ for the random variable describing the output of the above described execution, where the probability is taken over the (uniformly chosen) bits on the random tapes of all involved ITIs. This lets us state when a protocol emulates another.

Definition 4.1 (UC-emulation)**.** *Let π and ϕ be PPT protocols, let ξ be a predicate on extended identities, and let $\lambda \in \mathbb{N}$ be a security parameter. We say that π* ξ-UC-emulates *ϕ if for any PPT adversary A there exists a PPT adversary S such that for any balanced, PPT, ξ-identity-bounded environment Z, on all inputs z with $|z| \in \text{poly}(\lambda)$:*

$$|\Pr[\text{EXEC}_{\pi,A,Z}(\lambda, z) = 1] - \Pr[\text{EXEC}_{\phi,S,Z}(\lambda, z) = 1]| \in \text{negl}(\lambda). \tag{4.1}$$

If ξ allows all identities we simply write that π UC-emulates ϕ. The adversary S is called the *simulator*. Note that the definition implies that any attack on π can be translated (by the simulator) into an equivalent attack on ϕ, but not necessarily the other way around, which is often summarized as "π is at least as secure as ϕ".

Structured protocols To be able to model security properties in a meaningful manner, protocols are augmented with a *shell* mechanism. A structured protocol (or ITM) is split

into a shell and a body, where the shell has read/write access to the tapes of the body, but the body does not have access to the shell tapes (it is oblivious to the existence of the shell). Upon activation the shell processes incoming messages, possibly altering them, then write it on the incoming tape of the body. After the body completes its activation, the shell continues its activation, where it may (optionally) alter and forward any outgoing messages from the body. The body itself may be structured, thus multiple shells could be nested. The innermost body represents the actual protocol code (when describing a protocol in the UC framework one usually only specifies this innermost code), while shells takes care of code dealing with details of the security model.

Composition The fact that emulation is defined relative to arbitrary environments allows for strong guarantees when protocols are composed. Composition essentially replaces one subroutine with another. Its main purpose is to design protocols with some simple abstract building blocks and then instantiate these with a concrete protocol while preserving security. A few conditions must be fulfilled for a meaningful and secure definition of composition.

A protocol π is *subroutine respecting* if only the main parties communicate with existing ITIs outside the extended session. (The extended session is defined recursively: it contains all main parties and all parties invoked by a party from the extended session.) This ensures that we can cleanly replace the protocol with another protocol that may have a different internal structure. A protocol π is *subroutine exposing* if it lets the adversary know the extended identity of all subroutines. The purpose is to ensure that a protocol cannot "hide" subroutines by giving them a randomized identity (in which case the adversary would not be able to send backdoor messages to them). This property can be realized via some central directory ITI that is accessed via shell code. Finally, a protocol ρ is (π, ϕ, ξ)-*compliant* if for all executions with a ξ-identity-bounded environment and for all ITIs in the extended session of ρ: all external-writes with $t = \mathsf{input}$ have the forced-write flag set; and incoming messages on the output tape must have r set; and no external-write targeting π has the same target session identifier as any external-write targeting ϕ; and all ITIs providing input to either π or ϕ satisfy the predicate ξ.

Composition takes as input the protocols ρ, π and ϕ, where ρ interacts with subroutine ϕ. The composed protocol $\rho^{\phi \to \pi}$ is the protocol with all subroutine calls to ϕ replaced with calls to π. This is achieved by wrapping ρ in a shell that takes all input and output message to/from ϕ and replaces the code $\bar{\phi}$ with $\bar{\pi}$ (backdoor messages are left unaltered).

Theorem 4.2 (UC theorem [Can01])**.** *Let ρ, π, ϕ be PPT protocols, and let ξ be a predicate on extended identities. If ρ is (π, ϕ, ξ)-compliant, π and ϕ are subroutine exposing and*

subroutine respecting, and π ξ-UC-emulates ϕ, then $\rho^{\phi \to \pi}$ UC-emulates ρ.

Ideal functionalities Ideal functionalities represent the process that is aimed to be achieved, and can often be thought of as being executed by a trusted third party. The goal then is to show that some protocol (preferably one that can be realized in the real world) emulates this functionality. Formally, an ideal functionality $\mathcal{F}$ is just an ITI. To prevent the calling ITI from interacting with $\mathcal{F}$ directly (which would expose its code), it is executed in a protocol $\text{IDEAL}_{\mathcal{F}}$ which defines dummy parties (with only shell code) that forward input/output between the caller and $\mathcal{F}$, such that $\mathcal{F}$ sees the external identity. If π ξ-UC-emulates $\text{IDEAL}_{\mathcal{F}}$, we say that a protocol π *ξ-UC-realizes* $\mathcal{F}$. The protocol executions corresponding with $\text{EXEC}_{\pi,A,Z}$ and $\text{EXEC}_{\text{IDEAL}_F,S,Z}$ are often referred to as the real model and ideal model, respectively. If π itself uses some ideal functionality $\mathcal{G}$ as a subroutine, then π is said to operate in the $\mathcal{G}$-hybrid model.

The UC theorem is especially important when ρ UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and π UC-realizes $\mathcal{G}$ in the $\mathcal{H}$-hybrid (or real) model, because then $\rho^{\mathcal{G} \to \pi}$ UC-realizes $\mathcal{F}$ in the $\mathcal{H}$-hybrid (or real) model.

Most ideal functionalities cannot simply provide output to protocol parties, because in the realizing protocol the adversary can decide to abort early. This is usually modelled through *delayed* output: before providing a party with output, the functionality sends a (backdoor) message to the adversary, who can then decide if the output is delivered or not. We distinguish *private* and *public* delayed output, with the following conventional meaning: when we write that functionality $\mathcal{F}$ sends private delayed output (Output, *sid*, y) to party P, it means that $\mathcal{F}$ sends backdoor message (Output, *sid*, P) to the adversary, and only if the adversary replies with (Deliver, *sid*, P), will $\mathcal{F}$ actually send the output to P. Public delayed output is handled the same, but now the adversary receives (Output, *sid*, P, y). Input can similarly be delayed. When not specified, (delayed) output is private.

Corruption The adversary can corrupt protocol parties in the real model by sending them a backdoor Corrupt message, which is interpreted by the corruption shell of the party. In the standard PID-revealing corruption model, the shell first reports its ITI party identity to a dedicated corruption aggregation ITI, that records a list of corrupted PIDs and reveals this lists on request. We focus on *malicious*, *static* corruption. Malicious means that if corrupted, the shell of a party M forwards any message sent by the adversary as if coming from M, and any received message is forwarded to the adversary, while the body of M is never activated. Static means that upon invocation, the shell sends a notification

message to the adversary and only accepts a Corrupt message if it is delivered in the next activation, later Corrupt messages are ignored.

Ideal functionalities accept (Corrupt, P) messages, where P is the PID of a dummy party. The functionality marks P as corrupted, and input for p is now accepted directly from the adversary and any output for p is sent to the adversary. The ideal functionality also takes on the role of the corruption aggregation ITI, by responding to requests from the environment. Note that an ideal functionality is allowed to change its behaviour depending on who is corrupted, this should be explicitly mentioned in its description. An ideal functionality may also accept other backdoor messages that influence its behaviour or leakage. Dummy parties ignore Corrupt messages.

Communication The plain UC framework does not model communication between protocol parties, but instead requires protocol designers to specify this. Two commonplace models, both relevant to this work, are plain communication and authenticated communication.

Plain communication means that all communication goes unprotected via the adversary, who can read, alter, inject or drop arbitrary messages. In the plain UC framework this can be formalized with a shell mechanism: if the body completes its activation with output (Network, m), then the shell takes any (Network, m) output from the body and writes it to the backdoor tape of the adversary. When the shell is activated with backdoor input (Network, m), it activates the body with (Network, m) on its input tape.

Authenticated communication can be modelled with a functionality $\mathcal{F}_{\text{AUTH}}$ [Can01], which takes a single message input from the sender and then lets the adversary (who can read the full message) decide if and when it is delivered to the receiver. Instead I will use the closely related functionality $\mathcal{F}_{\text{MAUTH}}$, which allows multiple messages to be sent. I use the formalization by Barak, Canetti, Lindell, Pass and Rabin [BCL+05], also given in Figure 4.2. The formalization assumes that the participating parties are known beforehand.[10]

2.5.2 Simple universal composability

The simple UC (SUC) framework [CCL15] is called simple because it removes some of the complications that arise from the generality of the UC framework. The result is a frame-

[10]Looking forward, I note that this does not conflict with the post-specified peers as allowed by the OTR AKE: we assume that when key authentication is initiated, the parties know whose key they are trying to authenticate.

Figure 4.2: Ideal functionality $\mathcal{F}_{\text{MAUTH}}$: multi-message authentication [BCL+05].

1. On input (Init, sid): interpret $sid = (\mathcal{P}, sid')$, where $\mathcal{P}$ is a set of *PID*s, record $\mathcal{P}$ and forward it to the adversary. Initialize an empty list W of waiting messages.
2. - On input $(\mathsf{Send}, sid, P, P', m)$ from P: verify that $P, P' \in \mathcal{P}$, send (P, P', m) to the adversary and add it to W (triples may appear multiple times in W).
   - On backdoor message $(\mathsf{Deliver}, sid, P, P', m)$: if $P \in \mathcal{P}$ and P is corrupted, then output $(\mathsf{Received}, sid, P, P', m)$ to P', else if $(P, P', m) \in W$, then remove one such triple from W and output $(\mathsf{Received}, sid, P, P', m)$ to P'. Otherwise do nothing.

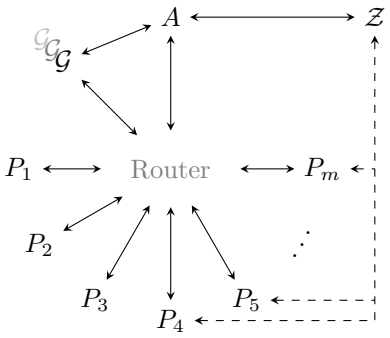


Figure 4.3: SUC $\mathcal{G}$-hybrid model parties P, adversary A, and environment Z. (Image adapted from [CCL15].)

work that is less expressive, but as secure as the UC framework: it cannot model all UC functionalities, but all SUC-secure protocols are UC-secure (after a transformation). The simplicity comes mainly from the fact that parties are fixed in advance and all communication goes via a central router. In this section I lay out the basics of the SUC, focussing on the relevant differences between UC and SUC.

Figure 4.3 provides a visual overview of the hybrid model, representing the actual protocol execution. The real model looks identical but omits ideal functionality subroutine $\mathcal{G}$.

The parties $P_1, \ldots, P_m$ are fixed ahead of time and are known to all. Parties send all messages via the router. The router forwards all messages between parties to the adversary A, and only delivers messages upon instruction by A: A has read access and full control over

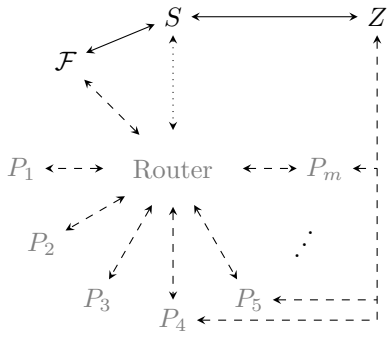


Figure 4.4: SUC ideal model, with functionality $\mathcal{F}$, dummy parties $P_1 \dots P_m$, adversary/simulator S, and environment Z. (Image adapted from [CCL15].)

message scheduling, but A cannot alter any message. The environment Z communicates directly with A, and Z provides protocol input to and receives output from the protocol parties. Messages between the ideal functionality $\mathcal{G}$ (or any of its copies, distinguished by unique session identifiers) and parties are partially opaque to the adversary.[11] Since the input/output of ideal functionalities always goes via the router, input/output is always delayed. Backdoor messages are sent directly between the functionality and the adversary. The parties are fixed ahead of time and are known to all. Execution is identical to that of the plain UC: it starts with Z and ends when Z halts.

An adversary corrupts a party by notifying the router. That party no longer participates in the protocol, instead the router lets the adversary send/receive messages on behalf of all corrupted parties. The router also reveals all corrupted parties to the environment when requested.

Figure 4.4 shows the corresponding setup in the ideal model. Besides the fact that the parties are dummy parties, the setup is identical to that of the hybrid model. In order for the functionality to learn who is corrupted (like it knows in the plain UC framework), the functionality can request the corruption list from the router as well.

Security is defined the same as in the plain UC model. A protocol π, executed in the SUC hybrid model, SUC-realizes a functionality $\mathcal{F}$ if for every environment Z an adversary A there exists a simulator S that ensures the environment cannot distinguish the two models.

[11] Although technically only a single ideal functionality $\mathcal{G}$ is allowed, multiple different ones can be combined in one larger functionality, where labels ensure that messages are handled by the correct inner functionality.

The SUC framework can express fewer functionalities than the UC framework. For example it is limited to a fixed set of parties (so SUC cannot model signatures), all communication is via the adversary-controlled router (including output to parties, so SUC cannot model fairness), and corruption is always total (a party reveals its entire state upon corruption, so SUC cannot model forward secrecy). However if a protocol SUC-realizes an ideal functionality, then its transformation of the protocol UC-realizes the transformation of the functionality. Briefly summarized, the protocol transformation encodes inputs differently, forces all communication between parties over an authenticated channel and lets parties communicate directly with ideal functionalities. In the UC framework, the adversary cannot see or influence the communication with the ideal functionality, so an additional transformation of the functionality is required that extends the capabilities of the adversary accordingly. For example, the UC functionality sends the public header of the message to the adversary and waits for the adversary to "approve" the message before continuing. Besides the requirement for authenticated communication, these transformations do not affect a real-world implementation of the protocol.

We note that the SUC framework specification allows multiple message to be sent via a single $\mathcal{F}_{\text{AUTH}}$ instance, while technically $\mathcal{F}_{\text{AUTH}}$ only allows a single message to be sent. This can trivially be solved by giving the UC to SUC transformation access to $\mathcal{F}_{\text{MAUTH}}$, a functionality that allows multiple messages to be sent (see also Figure 4.2).

Aborts Dealing with protocol aborts is partially built into the SUC framework, in the sense that all protocols are inherently *unfair*. That means that the adversary can always abort early, ensuring that some parties receive an output while others do not. This notion of security is called *security-with-abort*. Aborts are always visible to the environment.

However such aborts may never lead to security failures. The protocol itself has to decide what to do upon invalid inputs or subroutine failures: abort or continue with some default value. The simulator must be able to show indistinguishable behaviour, which is not always possible, as demonstrated in this chapter. I will be explicit considering abort handling throughout this chapter.

2.6 Random oracle model

This chapter operates in the local random oracle model (ROM) [BR93], which models hash functions by giving parties access to an oracle that provides uniformly random output, while giving identical output on identical queries. In the UC framework this is realized by ideal functionality $\mathcal{F}_{\text{RO}}$ (Figure 4.5). The simulator can implement the oracle itself, replacing

Figure 4.5: Ideal functionality $\mathcal{F}_{\text{RO}}$: the local random oracle model.

Parameters: input set X and output set Y.

- On (Init, *sid*), initialize an empty dictionary D_{sid}. Ignore if already initialized.
- On (Query, *sid*, $i \in X$): if $D_{sid}[i]$ is not set: sample $D_{sid}[i] \xleftarrow{\$} Y$; and output $D_{sid}[i]$. Ignore query if not initialized.

the oracle outputs with outputs of their own choice (this is called *programming* the oracle) as long as the responses are distributed indistinguishably from the real output.

The ROM is heuristic, in the sense that no real hash function can ever achieve the strong security guarantees given by the ROM [CGH04]. Despite this theoretical shortcoming of the ROM, no real-world protocol failures have resulted from using random oracles and attempts to replace ROM-based protocols have even introduced potential security weaknesses [KM15].

Note that $\mathcal{F}_{\text{RO}}$ constructs independent dictionaries per *sid*, which is called the *local* ROM. To realize this with a hash function, a method for *oracle cloning* is required [BDG20]. The simplest method, *domain separation*, prefixes a fixed-length encoding of *sid* before all hash inputs.

The ROM is inherently classical, so a valid objection to using it in the presence of a quantum adversary is that they have the capability to evaluate a hash function on a superposition state. If we reflect this capability in the model we get the quantum ROM (QROM) [BDF+11]: which allows queries of the form $\sum \alpha_i |i\rangle$ which receive a reply $\sum \alpha_i |D[i]\rangle$ (appropriately encoded to be unitary).

2.7 Oblivious transfer

OT [Rab81] is an important building block for secure multi-party computation (MPC), and in fact has been shown that any MPC functionality can be built from it [Kil88]. In 1-out-of-2 OT, Alice inputs $(s[0], s[1])$ and Bob inputs a bit j. The protocol outputs $s[j]$ to Bob, such that Alice does not learn j and Bob does not learn $s[1-j]$. Expressed as a SFE, we write $((s[0], s[1]), j) \mapsto (\emptyset, s[j])$, where $\emptyset$ indicates that a party has no output. A common variant of OT has the functionality itself generate the secrets: $(\emptyset, j) \mapsto ((s[0], s[1]), s[j])$, where $s[0]$ and $s[1]$ are distributed uniformly randomly, although possibly a malicious party

can bias their own output. All variants can be generalized to 1-out-of-m OT where Alice has m inputs.

2.8 Split functionalities

Most MPC functionalities are modelled as computations over authenticated channels. The justification is that in most contexts the MPC parties can initialize such a channel with standard authentication protocols, based on assumptions such as the existence of an ideal PKI. Since the aim of this work is to implement such an authentication protocol, authenticated channels are not available and instead we operate in the plain communication model.

There exists a general transformation that turns a functionality that is secure when computed using authenticated channels into a *split* functionality [BCL+05].[12] For any UC functionality $\mathcal{F}$, we can define the split functionality $s\mathcal{F}$, as formally specified in Figure 4.6. The split functionality $s\mathcal{F}$ is initialized with an (ordered) set U of participating parties. The adversary now has the capability to *split* U into subsets of parties, where the intersection between different subsets can only contain corrupt parties. For each subset H, the adversary can run a separate protocol execution $\mathcal{F}_H$. This separation is enforced by initializing each execution with a unique session identifier. The only dependency that can exist between these executions is that the input of one execution may depend on the output of another execution. Note that initialization steps must occur in order, and all computation messages are ignored if initialization is not completed.

I applied a minor technical correction to the original specification of $s\mathcal{F}$: in that definition each instance $\mathcal{F}_H$ is initialized with a *locally* unique session identifier sid_H, while the UC framework formally requires *globally* unique identifiers. Instead I initialize $\mathcal{F}_H$ with session identifier (sid, sid_H), which uses the fact that sid is globally unique by assumption. The unimportance of this detail is highlighted by the fact that the protocol for realizing split functionalities does ensure that sid_H is globally unique with overwhelming probability.

From here on I focus on the two-party case with $U = \{\text{Alice}, \text{Bob}\}$. The environment can choose U (and thereby fix the role of participants), but since it is part of the session identifier, it has to be fixed for the entire session. The split functionality ensures that the adversary is limited to two strategies: either let Alice and Bob communicate with each other authentically ($H = U$), or set up separate sessions with Alice ($H_1 = \{\text{Alice}\}$) and one with Bob ($H_2 = \{\text{Bob}\}$). To simplify the description I omit the identities in calls to $(s)\mathcal{F}_{\text{MAUTH}}$, since the message source and destination are never ambiguous.

[12] This should not be confused with split KEMs [BFG+21b], which are unrelated.

Figure 4.6: Split functionality $s\mathcal{F}$, given some functionality $\mathcal{F}$, assuming malicious static corruption [BCL+05].

1. Initialization

1. On input (Init, sid) from P: interpret $sid = (U, sid')$, where U is an ordered set of identities, and check that $P \in U$. Send backdoor message (Init, sid, P).
2. On backdoor message $(\mathsf{Init}, sid, P', H, sid_H)$: verify that $H \subseteq U$, $P' \in H$, and for all recorded H' either ($H \cap H'$ contains only corrupted parties and $sid_H \neq sid_{H'}$) or ($H = H'$ and $sid_H = sid_{H'}$). If all checks pass, record (H, sid_H), send $(\mathsf{Init}, sid, sid_H)$ to P', and invoke $\mathcal{F}_H$: a new instance of $\mathcal{F}$ with session identifier (sid, sid_H). Corrupt $U \setminus H$ in $\mathcal{F}_H$.

2. Computation

- On input (Input, sid, x) from P: find H such that $P \in H$, then provide x as input from P to $\mathcal{F}_H$, or do nothing if such H is not found.
- On backdoor message $(\mathsf{Input}, sid, H, P, x)$: if $\mathcal{F}_H$ is initialized and $P \in U \setminus H$, then provide x as input from P to $\mathcal{F}_H$, otherwise do nothing.
- On output y for P from $\mathcal{F}_H$: if $P \in H$, output y to P, otherwise forward it to the adversary.
- Forward backdoor messages between the adversary and any initialized $\mathcal{F}_H$.

Figure 4.7: Protocol $\Pi_{\mathcal{F}}$: UC-realizing $s\mathcal{F}$ in the $s\mathcal{F}_{\text{MAUTH}}$-hybrid model, given a protocol π that UC-realizes functionality $\mathcal{F}$ [BCL+05]. Specified here for two-party protocols.

1. Initialization

1. On input (Init, sid):
   interpret $sid = (s, \{\text{Alice}, \text{Bob}\})$, let $sid' = ((s, 1), \{\text{Alice}, \text{Bob}\})$, send (Init, sid') to $s\mathcal{F}_{\text{MAUTH}}$.[13]
2. On output $(\mathsf{Init}, sid', sid_H)$ from $s\mathcal{F}_{\text{MAUTH}}$:
   initialize $\pi_{\mathcal{F}}$ with session identifier (sid, sid_H) and output $(\mathsf{Init}, sid, sid_H)$.

2. Computation

- On input (Input, sid, x):
  send the input x to $\pi_{\mathcal{F}}$.
- On output y from $\pi_{\mathcal{F}}$: output y.
- When $\pi_{\mathcal{F}}$ would send $(\mathsf{Send}, (sid, sid_H), m)$ via $\mathcal{F}_{\text{MAUTH}}$:
  instead send (Send, sid', m) to $s\mathcal{F}_{\text{MAUTH}}$.
- On output $(\mathsf{Received}, sid', m)$ from $s\mathcal{F}_{\text{MAUTH}}$:
  send $(\mathsf{Received}, (sid, sid_A), m)$ to $\pi_{\mathcal{F}}$ (as if coming from $\mathcal{F}_{\text{MAUTH}}$).

A general construction $\Pi_{\mathcal{F}}$ was given in 2005 [BCL+05]. Given a protocol $\pi_{\mathcal{F}}$ that UC-realizes $\mathcal{F}$ in the $(\mathcal{G}, \mathcal{F}_{\text{MAUTH}})$-hybrid model, $\Pi_{\mathcal{F}}$ realizes a split functionality $s\mathcal{F}$ in the $\mathcal{G}$-hybrid model [BCL+05, Lemma 4.1]. The construction simply has the parties set up a pseudo-authenticated channel $s\mathcal{F}_{\text{MAUTH}}$ and then they run the protocol over that channel, see Figure 4.7.

Split authentication can be implemented with the protocol π_{SA} of Figure 4.8. Protocol π_{SA} UC-realizes $s\mathcal{F}_{\text{MAUTH}}$ [BCL+10, Theorem 11]. Note that the computed sid_H is used internally to sign/verify messages, and it is also given as output when concluding the intialization phase. Since sid_H contains their own key that was generated randomly, each party is guaranteed that it is globally unique with overwhelming probability, assuming the space of verification keys is sufficiently large.

[13] The sid is changed to ensure $s\mathcal{F}_{\text{MAUTH}}$ is not a main party.

Figure 4.8: Protocol π_{SA}, UC-realizing $s\mathcal{F}_{\mathrm{MAUTH}}$ [BCL+05], adjusted for two-party protocols.

Let $(\mathrm{KeyGen}_S, \mathrm{Sign}, \mathrm{Verify})$ be a signature scheme.

1. Initialization

1. On input (Init, sid):
   interpret $sid = (U, sid')$, generate signature keypair (sk, vk) and send message $(\mathsf{PubKey}, sid, vk)$.
2. On message $(\mathsf{PubKey}, sid, vk')$:
   compute $sid_H = (vk, vk')$ (or $sid_H = (vk', vk)$, where the ordering is determined by the order of parties in U), compute signature $\sigma = \mathrm{Sign}(sk, sid_H)$ and send message $(\mathsf{Sig}, sid, \sigma)$.
3. On message $(\mathsf{Sig}, sid, \sigma')$:
   if $\mathrm{Verify}(vk', sid_H, \sigma')$ then accept, initialize a counter $c := 0$, and output $(\mathsf{Init}, sid, sid_H)$. Ignore the message if the signature is invalid.

2. Computation

- On input (Send, sid, m):
  Compute signature $\sigma = \mathrm{Sign}(sk, (sid_H, m, c))$, increment c, and send message $(\mathsf{Msg}, sid, (m, c, \sigma))$.
- On message $(\mathsf{Msg}, sid, (m, c', \sigma))$:
  if c' did not occur in a message received before, and $\mathrm{Verify}(vk', (sid_H, m, c'), \sigma)$ holds, record c' and output $(\mathsf{Received}, sid, m)$. Ignore invalid messages.

2.9 Password authenticated key exchange

Password based key authentication is closely related to PAKE, which both take a low-entropy shared secret as input, Whereas the former functionality just checks if the inputs are equal, the latter also provides the users with a shared secret as output.

PAKEs are usually modelled with implicit authentication [BPR00; CHK+05]: both users get a key as output from the protocol, but only if the authentication succeeded will the keys be equal for both parties. Such a protocol can be transformed into a protocol with mutual authentication (often called explicit authentication in the context of UC) by doing key confirmation using a pseudo-random function (PRF). After getting output s from the PAKE, Alice sends $\mathrm{PRF}(s, 1)$, Bob sends $\mathrm{PRF}(s, 2)$ and both output $\mathrm{PRF}(s, 0)$ if they receive the expected value, or abort otherwise.

This immediately suggests we can use a PAKE with mutual authentication to realize password based key authentication. Provide the public keys used in the AKE and shared secret as input to the PAKE with mutual authentication. Reject the authentication if the PAKE aborts, otherwise accept (the PAKE output $\mathrm{PRF}(s, 0)$ can simply be discarded).

While a PAKE is conceptually more complex than a equality test, many constructions for them are known, including some based on post-quantum primitives [KV09; DAL+17; TSJL21]. PAKEs are usually more efficient than the protocol we design in this chapter. However, the goal was to construct a post-quantum solution based on the primitives from the NIST standardization effort: KEMs and signatures. This means that we can work with primitives and implementations that have been the focus of much of the post-quantum cryptographic research in recent years.

3 Protocol

The protocol for key authentication is (a variant of) a PET, built in the OT hybrid model, executed over a pseudo-authenticated channel. To realize the OT functionality with existing post-quantum key-agreement protocols, I take the construction of Masny and Rindal. In Section 3.1 I discuss how I remove some of the sharp edges of their construction so that it is simpler to implement securely.

The main contribution is given in Section 3.2: I loosen the functionality of a PET so that corrupt parties can introduce a one-sided error, but only such that the output is still meaningful for key authentication. I realize this by taking the protocol of [RR17], and modifying it so it securely provides output to both parties.

Section 3.3 concludes this section by discussing how to execute the protocol over a pseudo-authenticated channel to achieve a split equality test that is secure when computed over unauthenticated channels.

3.1 Oblivious transfer

For the OT I use an existing construction by Masny and Rindal, based on KEX protocols (and which can therefore be implemented with KEMs).[14] There are significant differences between their proceedings version [MR19] and their preprint version [MR21]: I will use the preprint version (revised last at July 13, 2021) as the main reference, in which the mistakes in the proofs that were found by others and myself have been corrected.

The formalization of the OT ($\mathcal{F}_{\text{OT}}$) is given in Figure 4.9. This describes a variant of OT with outputs generated by the functionality, but where a malicious sender can choose their own output. Note that $\mathcal{F}_{\text{OT}}$ deviates from the Sender Chosen Message OT ($\mathcal{F}_{\text{OT}}^{S}$) [MR21, Definition 2.4], which instead formalizes SFE $(s[\cdot], j) \mapsto (\emptyset, s[j])$. Instead it is closer to Endemic OT ($\mathcal{F}_{\text{OT}}^{E}$) [MR21, Definition 2.4]: a variant of SFE $(\emptyset, j) \mapsto (s[\cdot], s[j])$, where both parties can bias their own output when malicious. Although these formalizations are conceptually the same, technically they are incompatible, so Figure 4.9 can be seen as a minor correction. For example, the lemma that $\mathcal{F}_{\text{OT}}^{S}$ (S)UC-emulates $\mathcal{F}_{\text{OT}}^{E}$ [MR21, Lemma 3.1] does not hold since no simulator exists when both parties are honest. The proof of that lemma does however imply that $\mathcal{F}_{\text{OT}}$ (S)UC-emulates $\mathcal{F}_{\text{OT}}^{E}$.

3.1.1 Existing protocol

The constructions require a 1-RTT uniform KEX protocol, shown in Figure 4.10 (see [MR21] for definitions). The term 1-RTT means one message in each direction, while uniform means that pk is indistinguishable from a random group element. In Figure 4.10 I used suggestive names for the values and subroutines to correspond with a KEM, but other KEX protocols such as DH are also allowed: then $sk = x$, $pk = g^x$, $ct = g^y$, and $k_B = g^{xy} = g^{yx} = k_A$. For correctness, the probability that $k_A = k_B$ must be overwhelming. Furthermore, the public keys need to form a group with respect to some operator, which I will denote additively:

[14] The authors call this a key agreement protocol, but I will use key exchange (KEX) or key establishment according to the terminology of [BMS20]. The term key *agreement* is used for protocols in which the output shared key is a function of inputs of both participants (as opposed to key *transport* protocols, where the key is generated by one party).

Figure 4.9: Ideal functionality $\mathcal{F}_{\text{OT}}$: 1-out-of-m OT between sender S and receiver R.

Parameters: the OT secrets have $s[i] \in \{0,1\}^\lambda$ for $1 \leq i \leq m$.
Each message (per *sid*) is only accepted once.

1. On input (InputR, *sid*, j) from R: verify that $1 \leq j \leq m$, then record (*sid*, j).
2. On backdoor message (DeliverS, *sid*, $s'[\cdot]$):
   if S is corrupt: validate that $s'[i] \in \{0,1\}^\lambda$ for $1 \leq i \leq m$, then set $s[\cdot] := s'[\cdot]$, otherwise sample $s[\cdot] \xleftarrow{\$} \{0,1\}^{\lambda m}$. Record (*sid*, $s[j]$) and output (OutputS, *sid*, $s[\cdot]$) to S.
3. On backdoor message (DeliverR, *sid*): output (OutputR, *sid*, $s[j]$) to R.

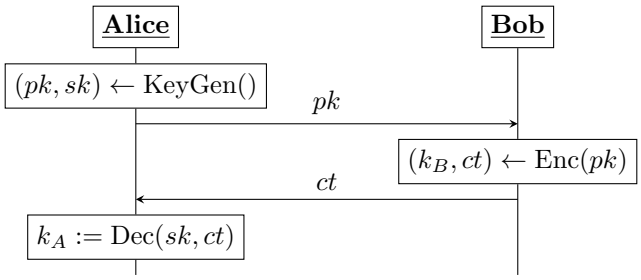


Figure 4.10: 1-round trip time (RTT) KEX.

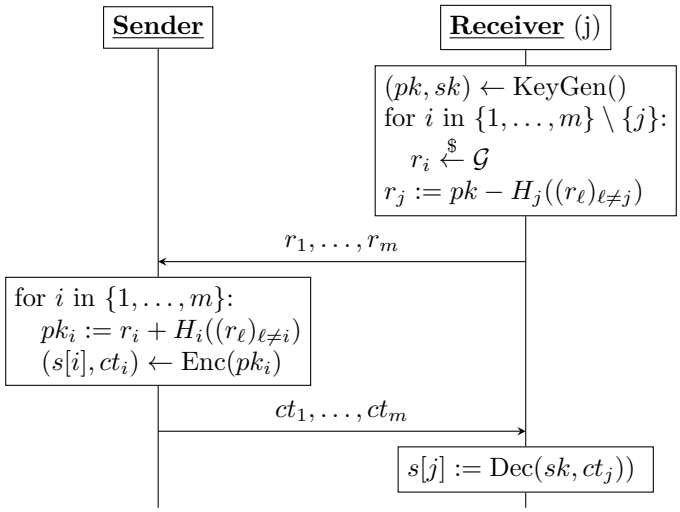


Figure 4.11: π_{OT}: OT construction by Masny and Rindal [MR21].

$(\mathcal{G}, +)$.[15] The authors provide an implementation of Figure 4.10 based on the round 1 version of Kyber.CPAPKE [ABD+17].

For 1-out-of-m OT, the protocol uses m random oracles $H_i : \mathcal{G}^{m-1} \to \mathcal{G}$. The protocol ($\pi_{\text{OT}}$) is shown in Figure 4.11: the receiver outputs $s[j]$ and the sender outputs $s[\cdot]$.

Repairing the proof The protocol is secure if it SUC-emulates $\mathcal{F}_{\text{OT}}$. Instead the authors give an incomplete definition for security: they require security only against a malicious sender and against a malicious receiver [MR21, Definition 2.6]. For completeness the passive adversary should also be considered.[16] For security against the passive receiver the protocol must be correct with overwhelming probability, otherwise the environment can distinguish the real from ideal model via input/output of the honest parties. Against the passive adversary, correctness of the OT protocol follows directly from the correctness of the KEX protocol.

The authors recognize this necessity for correctness, but applied it in the wrong place.[17]

[15] Formally, a quasigroup (a group without associativity or identity) suffices. Furthermore, the public keys can be a subset of the group $\mathcal{G}$, as long as public keys cannot be efficiently distinguished from random group elements. I will loosely use the term group as is done in the literature [BDD+17; MR19].

[16] Technically we should also consider the adversary corrupting both parties, but then simulation is trivial as S can just run A who acts as both parties.

[17] This is corrected in the latest version.

Summarizing their work, they give the following definition for security against the malicious sender:[18] for all adversaries A there exists a simulator S such that for all environments Z on auxiliary input z, the quantity

$$\left|\Pr[Z(z,(A,R)_\pi)=1]-\Pr[Z(z,(S,\mathcal{F}_{\mathrm{OT}}^E)=1]\right| \tag{4.2}$$

is negligible. The authors state that $(A,R)_\pi$ is the joint output of A and honest receiver R when running the protocol π, while leaving $(S,\mathcal{F}_{\mathrm{OT}}^E)$ unspecified. In the SUC framework A has no output, so both probabilities are ill-defined. However the proof [MR19, Claim 4.2] used to suggest that Z is an *interactive* distinguisher and has all capabilities of the environment in the SUC framework, so the above can be interpreted as some abuse of notation. The reason I elaborate on this, is that the environment is made just a bit too strong: it also *directly* sees the output of the ideal functionality, possibly as a result of the above definition. The SUC environment does not see this: once a party is maliciously corrupted it is deactivated,[19] which is simulated by *not* delivering the output to the corrupted dummy party. Their claim suggests a malicious sender can induce a security loss of $(1-\delta)$ in case of a δ-correct KEX protocol, but this is not true.

The problem with requiring correctness in the presence of maliciously corrupted parties, is that it cannot be guaranteed by many KEX protocols. For many post-quantum public key encryption (PKE) schemes a malicious sender can send a maliciously crafted ciphertext ct' upon which decryption will either explicitly fail or decrypt to an incorrect value.

To summarize the above: the OT construction of Figure 4.11 *is secure* with the Kyber.CPAPKE instantiation of Figure 4.10. In fact it is marginally *more* secure, because no $(1-\delta)$ security loss can be induced by malicious parties.

There is another place where their protocol is actually stronger than their proof suggests. The fixed UC proof for the malicious receiver [MR21, Appendix E.1] claims that the KEM instantiation only achieves *endemic* security for the OT, while in fact the receiver has no influence over the value $s[j]$ and thus the protocol SUC-realizes $\mathcal{F}_{\mathrm{OT}}$. More concretely, note that the *simulator* of the proof of Claim E.1 samples a random value $s[j]$ and sends that as input to the ideal functionality. I suspect the authors were aware of this fact, since they listed the KEM instantiation as achieving $\mathcal{F}_{\mathrm{OT}}^S$ in their benchmarks table [MR21, Figure 10], despite never asserting this anywhere in the paper.[20] This somewhat simplifies

[18] Security against a malicious receiver is defined similarly and has the same issues described in this section.

[19] This deactivation is why the simulator must extract the effective input from the adversary in case of static malicious corruption.

[20] Technically, the instantiation does not achieve $\mathcal{F}_{\mathrm{OT}}^S$, but it does achieve $\mathcal{F}_{\mathrm{OT}}$, as explained before.

my proof, although it should be noted that the endemic OT would be sufficient for this work.

Selective failure attack The current specification implicitly assumes that subroutines (specifically KeyGen, Enc, and Dec in Figure 4.10) never abort. Many cryptosystems do abort upon incorrect input, which in the context of OT can lead to a *selective failure* attack. If a malicious sender can send values ct_i such that $\mathrm{Dec}(\cdot, ct_i)$ fails only on *some* of them, then the sender can learn (partial information on) the receiver input j by observing if the receiver aborts or not.

Take for example the instantiation based on DH, where the honest sender sends $ct_i = g^{y_i}$. In reality the receiver does not receive a group element but receives a bytestring which *should* represent a group element. The receiver must parse and validate the correctness of the incoming bytes, before they can generate the shared key. The seemingly straightforward implementation, where the receiver selects the bytes of index j, parses (possibly failing) and runs Dec, is therefore vulnerable. In this case there is a solution, because validation operates on public information: the receiver could validate *all* incoming bytestrings, before selecting the j-th one and running Dec. If any validation fails, the receiver can abort securely, but note that some protection against side-channels is required in the implementation.

The authors claim their construction results in endemic OT from many cryptosystems, including McEliece [McE78], but in consideration of selective failure attacks this is not immediate. Decrypting/decoding in the McEliece PKE cryptosystem can lead to a decoding failure. The above countermeasure of validating all incoming bytestrings is no longer available, since decoding depends on the secret key. A possible fix is to output some other value in case of a decoding failure. Any value will do, as long as the party does not abort. Note that the KEM cryptosystem Classic McEliece [ABC+20] already does this by wrapping the Fujisaki-Okamoto transform with implicit rejection ($\mathsf{FO}^{\not\perp}$) [HHK17] around the PKE system. The transform applies to most PKE schemes and is in fact the construction used by many post-quantum KEM schemes, therefore I prefer this countermeasure to prevent selective failure attacks.

In principle a malicious receiver can send invalid values pk, so that $\mathrm{Enc}(pk)$ aborts. However, the sender in π_{OT} runs Enc on all received pk_i and has no secret input to leak, so sender aborts will not lead to loss of security.

Implementation errors The DH-based construction of the main text [MR19] is not implemented, instead an optimized version [MR21, Appendix D.2] is implemented [Rin21],

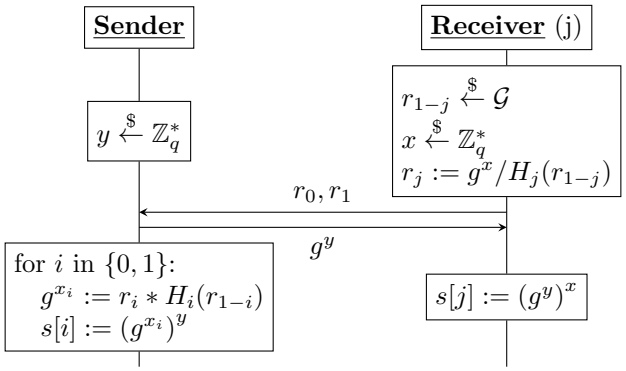


Figure 4.12: Optimization of DH based 1-out-of-2 OT construction by Masny and Rindal [MR21]. (The group operations are written multiplicatively, but they fulfill the same role as the operations of Figure 4.11.)

which is never mentioned in the proceedings version. The implementation contained a bug that made that version insecure. The optimized version is shown in Figure 4.12, it allows the sender to send only a single group element. Inspection of the source code reveals that the implementation contained a bug:[21] the random oracles were implemented as a hash function *without domain separation*, so that effectively $H_0 = H_1 = \cdots = H_{n-1}$. Without proper oracle cloning [BDG20] the receiver can send $r_0 = r_1$ and thereby ensure that $s[0] = s[1]$. The paper only provides a proof sketch under a custom variant of the decisional DH (DDH) assumption for this optimized version. The post-quantum implementation of π_{OT} in libOTe does not properly clone the oracles either, which does mean the security proof no longer applies, but in that case I have not been able to find an attack.

3.1.2 Quantum security

In the paper the authors claim to have built "the first implementation of a quantum resistant OT" [MR21]. However, simply instantiating the protocol with post-quantum primitives is not sufficient (see also Section 3.2.1) for achieving quantum resistance. One obstacle towards proving quantum security is to consider security in the QROM. Without going into too much detail, I describe why I believe such a proof to be non-trivial.

Adapting the proof for security against a maliciously corrupt sender should be possible. In the proof, the simulator generates m keypairs (pk_i, sk_i) and m random group elements

[21] I have reported this to the authors, who fixed the issue straightaway: https://github.com/osu-crypto/libOTe/commit/23106591573a478f3fa039ac938995a7a2ee3b2a.

r_i, then programs the random oracle such that $pk_i = r_i + H_i((r_\ell)_{\ell \neq i})$ for all i, so that it knows *all* secret keys and can extract the input. Since the oracle answers are independent of earlier queries, this construction is *history-free* [BDF+11] and security in the QROM should follow.

However the main difficulty lies in proving security against a corrupt receiver. The simulator does not program the random oracle at all, instead it inspects the queries and finds the one compatible with the received $(r_1, \ldots, r_m)$ to extract the input index j. In the QROM, queries can be in superposition, so that in general inspection of the queries is not possible without being detected by the environment. A QROM proof would thus have to work around this difficulty, up to my knowledge there are no standard techniques known for doing so.

3.1.3 Combined instantiation

At the moment of writing there exists ongoing debate about the concrete security of the relatively young post-quantum cryptosystems. A benefit of basing the OT on KEX is that there are straightforward methods for combining them: simply concatenate the KEX outputs, and put them in a KDF. The resulting key will be secure as long as one of its components are secure. Concretely, I will combine ECDH on the Goldilocks curve with the lattice-based KEM Kyber [ABD+21]. The former allows us to use the Decaf library [Ham15] already present in the OTRv4 codebase, while the latter uses liboqs [SM16]. In this combined solution, using the DH optimization of would not gain much, but it would come at the price of a stronger cryptographic assumption, therefore I will not use it. Both cryptosystems naturally induce group operations on the public keys, I further discuss these when talking about the implementation.

3.2 Private equality confirmation

For key authentication, Alice and Bob hash their user identities, public keys, the OTR session identity (not to be confused with the UC session identity) and their low-entropy secret to get local inputs x and y, respectively. To confirm they are in the same session (and thus there is no PITM) they could run a PET, expressed as the SFE $(x, y) \mapsto (\llbracket x = y \rrbracket, \llbracket x = y \rrbracket)$. Both parties learn a single bit that indicates if their inputs are equal.

A weaker functionality suffices, as they only want to *confirm* that their values are equal: they require "(the protocol output $= 1$) $\implies$ $x = y$", but not necessarily "(the protocol output $= 1$) $\iff$ $x = y$". To a degree, this is already encoded in the SUC framework with

Figure 4.13: Ideal functionality $\mathcal{F}_{\text{PEC}}$ for private equality confirmation

Parameters: n is the fixed length of the honest inputs (x for Alice and y for Bob).

- On input (InputA, *sid*, x) from Alice: verify that $x \in \{0,1\}^n$, or if Alice is corrupt verify $x \in \{\emptyset\} \cup \{0,1\}^n$, then record x.
- On input (InputB, *sid*, y) from Bob: verify that $y \in \{0,1\}^n$, then record y.
- When having received both InputA and InputB:
  send public output (OutputB, *sid*, $[\![x = y]\!]$) to Bob.
- On backdoor message (DeliverA, *sid*, b):
  Ignore if no OutputB was given before.
  If Bob is corrupt, send private output (OutputA, *sid*, $b[\![x = y]\!]$) to Alice,
  otherwise send private output (OutputA, *sid*, $[\![x = y]\!]$) to Alice.

unfairness: one party can always abort to prevent the other party from receiving output. We can allow the adversary slightly more power: both corrupt Alice and corrupt Bob are allowed to force the output of the other party to be zero. A corrupt Alice can input a value $\emptyset$, with $\emptyset \neq y$ for all $y \in \{0,1\}^n$ (this corresponds to the fact that Alice can input subsets in the private set intersection (PSI) protocol on which this protocol is based [RR17]). Bob is given slightly more power: he can change Alice's output to zero *after* having received output himself. I call the resulting functionality a private equality confirmation (PEC): a formal definition $\mathcal{F}_{\text{PEC}}$ is given in Figure 4.13.

We base our solution on OT, but also require a function G that is one-way and pseudorandom:

Definition 4.3. *A function* $g : \{0,1\}^* \to \{0,1\}^*$ *is* one-way *if*

1. *there exists a deterministic polynomial time algorithm* G*, that on input* x *outputs* $g(x)$*.*
2. *for every PPT adversary* A *and polynomial* p*, there exists a* $\lambda_0 \in \mathbb{N}$ *such that for all* $\lambda > \lambda_0$ *and for all* $z \in \{0,1\}^*$ *with* $|z| \in \text{poly}(\lambda)$*:*

$$\Pr[A(g(U_\lambda), z) \in g^{-1}g(U_\lambda)] < \frac{1}{p(\lambda)}, \tag{4.3}$$

where U_λ *is a random variable uniformly distributed over* $\{0,1\}^\lambda$*.*

I will leave the first property implicit and directly write G from here on.

Definition 4.4. *A deterministic polynomial time algorithm* $G : \{0,1\}^\lambda \to \{0,1\}^\lambda$ *is* pseudorandom *if for all PPT algorithms* D*, and every polynomial* p*, there exists a* $\lambda_0 \in \mathbb{N}$ *such that for all* $\lambda > \lambda_0$ *and all* $z \in \{0,1\}^*$ *with* $|z| \in \text{poly}(\lambda)$*:*

$$\left| \Pr_{r \in \{0,1\}^\lambda}[D(G(r), z) = 1] - \Pr_{r \in \{0,1\}^\lambda}[D(r, z) = 1] \right| < \frac{1}{p(\lambda)}. \tag{4.4}$$

Which essentially states that G is a pseudorandom generator, but without any expansion.

Formally, the SUC protocol description of π_{PEC} requires additional information about the state management to ensure the correct ordering of messages. For this specific protocol, both parties should accept input just once, accept output from the OT once per index i, and must execute the enumerated steps in order: Alice must ignore any MsgBob message until having sent MsgAlice, Bob must ignore any MsgAlice message until having computed both $\alpha(y)$ and $\beta(y)$. Technically, each party should halt activation directly after sending a message (this includes input to $\mathcal{F}_{\text{OT}}$ and output to the environment), and they can only resume computation upon explicit activation by the next message. Parties can enforce this by ignoring all activations other than receiving an empty continuation message from the adversary.

Protocol π_{PEC} of Figure 4.14 realizes $\mathcal{F}_{\text{PEC}}$ by running $2n$ executions of $\mathcal{F}_{\text{OT}}$. It is essentially the protocol of Section 2.4, with a one-way function G. We use the shorthand notation $\alpha(x) = \bigoplus_{i=1}^n A_i[x]$ (and similarly $\beta(x) = \bigoplus_{i=1}^n B[x]$). Intuitively the protocol is secure: in order to construct the message m_A that Bob accepts, Alice has to know $\beta(y)$. If $x \neq y$, there is at least one $x_i \neq y_i$, so that $B_i[y_i]$ is random to her and so is $\beta(y)$. Similarly Bob must know $\alpha(x)$ to construct m_B. The one way function G ensures that it does not leak to him through m_A. This intuitive security argument is formalized in Theorem 4.5. The proof is structured like that of the dual-execution protocol by Rindal and Rosulek [RR17].

Theorem 4.5. *Protocol* π_{PEC} *SUC-realizes* $\mathcal{F}_{PEC}$ *in the* $\mathcal{F}_{OT}$*-hybrid model.*

Proof. We consider three cases: both parties are honest, Alice is corrupt, or Bob is corrupt. Technically there is a fourth case where both parties are corrupt, but simulation is trivial in that case.

Figure 4.14: Protocol π_{PEC}, SUC-realizing $\mathcal{F}_{\text{PEC}}$ in the $\mathcal{F}_{\text{OT}}$-hybrid model. Formal state management to ensure correct ordering of messages is omitted from the description, see text for further details.

Parameters: n is the length of the inputs in base m.
Functionality $\mathcal{F}_{\text{OT}}$ is the 1-out-of-m OT functionality with outputs in $\{0,1\}^\lambda$.
Function $G : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is pseudorandom and one-way.

1. Interaction with the OT functionality:

   - On input (InputA, *sid*, x) to Alice, let $x = x_1 x_2 \dots x_n$ in base m.
     For $i = 1$ to n:
     send (InputR, (*sid*, 0, i), x_i) to $\mathcal{F}_{\text{OT}}$.
   - On input (InputB, *sid*, y) to Bob, let $y = y_1 y_2 \dots y_n$ in base m.
     For $i = 1$ to n:
     send (InputR, (*sid*, 1, i), y_i) to $\mathcal{F}_{\text{OT}}$.
   - On output (OutputS, (*sid*, 1, i), $A_i[\cdot]$) to Alice:
     record $A_i[x_i]$. If n outputs are received, compute $\alpha(x) = \bigoplus_{i=1}^n A_i[x_i]$.
   - On output (OutputR, (*sid*, 0, i), $B_i[x_i]$) to Alice:
     record $B_i[x_i]$. If n outputs are received, compute $\beta(x) = \bigoplus_{i=1}^n B_i[x_i]$.
   - On output (OutputR, (*sid*, 1, i), $A_i[y_i]$) to Bob:
     record $A_i[y_i]$. If n outputs are received, compute $\alpha(y) = \bigoplus_{i=1}^n A_i[y_i]$.
   - On output (OutputS, (*sid*, 0, i), $B_i[\cdot]$) to Bob:
     record $B_i[y_i]$. If n outputs are received, compute $\beta(y) = \bigoplus_{i=1}^n B_i[y_i]$.
   - When Alice has computed both $\alpha(x)$ and $\beta(x)$:
     send message (MsgAlice, *sid*, $m_A = G(\alpha(x)) \oplus \beta(x)$) to Bob.

2. On message (MsgAlice, *sid*, m_A) to Bob:
   Let $z = [\![m_A = G(\alpha(y) \oplus \beta(y))]\!]$, give public delayed output (OutputB, *sid*, z).
   Let $m_B = \begin{cases} \text{REJECT} & \text{if } z = 0 \\ \alpha(y) \oplus \beta(y) & \text{if } z = 1 \end{cases}$
   Send (MsgBob, *sid*, m_B) to Alice. Halt.

3. On message (MsgBob, *sid*, m_B) to Alice:
   give private delayed output (OutputA, *sid*, $[\![m_B = \alpha(x) \oplus \beta(x)]\!]$). Halt.

Two honest parties When both parties are honest, the simulator S operates as follows for an adversary A.

Instruct the router to deliver the input from dummy Alice and Bob to $\mathcal{F}_{\text{PEC}}$. Internally simulate the interaction of real-model Alice and Bob with the OTs by sending the private delayed input/output messages that A would see (devoid of any content). Set $\alpha, \beta \xleftarrow{\$} \{0,1\}^\lambda$. Internally simulate Alice sending (MsgAlice, *sid*, $G(\alpha) \oplus \beta$) to Bob. If A would instruct the router to deliver this message, inspect *public* output (OutputB, *sid*, $[\![x = y]\!]$), and instruct the router to deliver this output to (dummy) Bob. If $x \neq y$, set $m_B :=$ REJECT, otherwise set $m_B \xleftarrow{\$} \{0,1\}^\lambda$, then simulate Bob sending (MsgBob, *sid*, m_B) to Alice. If A would instruct the router to deliver this message, send backdoor message (DeliverA, *sid*, 1) to $\mathcal{F}_{\text{PEC}}$ and instruct the router to deliver the output to Alice.

The simulator is valid, for it provides an identical view to the adversary/environment, except for the event that real-model Bob erroneously accepts, but that only occurs with negligible probability. In more detail, by the properties of the OT, $\alpha(x)$ and $\beta(x)$ in the real-model execution are both uniformly random and are thus distributed identically to α and β in the ideal-model and therefore MsgAlice is distributed identically in both models. If $x = y$ then both in the real model Alice and Bob accept, and MsgBob is again identical in both protocols. Otherwise if $x \neq y$ then ideal-model Bob will reject, while real-model Bob might accept. However, if $x \neq y$, there is at least one $x_i \neq y_i$, so that for real-model Bob: $\Pr[z = 1] = \Pr[B_i[x_i] = B_i[y_i]] = 2^{-\lambda}$. Thus he erroneously accepts only with negligible probability, and since Alice accepts only if Bob does, the probability that her output is wrong is also negligible.

Corrupt Alice Next we consider static malicious corruption of Alice: for any real-model adversary, define the ideal-model simulator S_A as follows:

1. Extract x', and $\alpha(x')$ from the adversary's interaction with $\mathcal{F}_{\text{OT}}$. For $1 \leq i \leq n$, get (InputR, (*sid*, 0, *i*), x'_i), and (DeliverS, (*sid*, 1, *i*), $A_i[\cdot]$) from the adversary. Let $B_i \xleftarrow{\$} \{0,1\}^\lambda$, and send (OutputR, (*sid*, 0, *i*), B_i) and (OutputS, (*sid*, 1 *i*), $A_i[\cdot]$) to the adversary. Let $\alpha(x') = \bigoplus_{i=1}^n A_i[x'_i]$ and $\beta = \bigoplus_{i=1}^n B_i$. Abort if any $\mathcal{F}_{\text{OT}}$ aborts.
2. On adversary message (MsgAlice, *sid*, m_A): compute $z' = [\![m_A = G(\alpha(x')) \oplus \beta]\!]$. If $z' = 0$ set $x = \emptyset$, otherwise set $x = x'$. Send (InputA, *sid*, x) and (DeliverA, *sid*, 1) to $\mathcal{F}_{\text{PEC}}$.
3. On (OutputA, *sid*, $[\![x = y]\!]$) from $\mathcal{F}_{\text{PEC}}$: if $x \neq y$, send (MsgBob, *sid*, REJECT) to the adversary, otherwise send (MsgBob, *sid*, $\alpha(x') \oplus \beta$).

To show this is a valid simulator, we consider a sequence of hybrids. Each hybrid is like an ideal-model simulator (relative to a fixed real-model adversary), but with the additional capabilities to read the input and set the output of the honest party directly. For each hybrid I describe the modification to the previous hybrid, and then prove that the modification is indistinguishable for the environment (by proving that the inputs to the environment are indistinguishable). The first hybrid ensures the inputs to the environment are identical to those in the real-model, while the last hybrid is identical to the ideal-model simulator S_A.

Hybrid 0. The hybrid semi-honestly runs $\mathcal{F}_{\text{OT}}$ and runs the code of honest Bob in π_{PEC}, getting the input y from the dummy party and setting its output to the output of honest Bob's code. Note that the hybrid can observe the adversary messages x', and $A_i[\cdot]$. This hybrid trivially provides identical inputs to the environment as the real-model interaction.

Hybrid 1. This hybrid removes Bob's interaction with the OTs. Compute $\alpha(y)$ directly from the observed $A_i[\cdot]$. Let $B_i[\cdot] \xleftarrow{\$} \{0,1\}^{\lambda m}$ and send $(\mathsf{OutputR}, (sid, 0, i), B_i[x'_i])$ to the adversary. By the guarantees of the OT functionality, the output to the adversary remains identical, thus hybrids 0 and 1 are indistinguishable.

Hybrid 2. Extract Alice's effective input x directly from x' and m_A: Let $z' = [\![m_A = G(\alpha(x')) \oplus \beta(x')]\!]$: if $z' = 1$ set $x = x'$, otherwise set $x = \emptyset$. If $x \neq y$, set $m_B = \text{REJECT}$, otherwise set $m_B = \alpha(x') \oplus \beta(x')$. Send m_B to the adversary and set Bob's output to $[\![x = y]\!]$.

If $x' = y$, then $G(\alpha(x')) \oplus \beta(x') = G(\alpha(y)) \oplus \beta(y)$, so that z' (in hybrid 2) is equal to z (in hybrid 1). If $z' = 0$, then Bob rejects (with output zero and message REJECT) in both hybrids, otherwise $z' = 1$ and $x = x'$, so that Bob accepts in both hybrids (with output one and message $\alpha(x') \oplus \beta(x') = \alpha(y) \oplus \beta(y)$).

If $x' \neq y$, then $x \neq y$ (independent of the value of z') and Bob rejects in this hybrid. However in hybrid 1 Bob might have erroneously accepted. Since at least one $B_i[y_i]$ is random for the adversary, the probability that the adversary sent a value m_A that Bob accepted is $2^{-\lambda}$ and thus the hybrids are statistically indistinguishable.

Hybrid 3. Hybrid 2 only uses $B_i[x_i]$ out of the array $B_i[\cdot]$, so it may as well generate only that value. Let $B_i \xleftarrow{\$} \{0,1\}^{\lambda}$ and use B_i where hybrid 2 used $B_i[x'_i]$. Write $\beta = \beta(x')$ to highlight that its value is independent of x'. This change is only cosmetic so this hybrid is identical to hybrid 2.

Hybrid 4. Instead of setting the output of dummy Bob directly, do so indirectly by sending $(\mathsf{InputA}, sid, x)$ to $\mathcal{F}_{\text{PEC}}$. This also ensures that the hybrid learns $[\![x = y]\!]$ from the $\mathsf{OutputA}$ message, to be able to construct Bob's final message. The hybrid no longer requires access to the input of dummy Bob, while the correctness of $\mathcal{F}_{\text{PEC}}$ guarantees the output of dummy

Bob is identical to that of hybrid 3. Therefore this is a valid ideal-model simulator, and in fact it is identical to S_A.

Corrupt Bob Finally we consider static malicious corruption of Bob: For any real-model adversary, define the ideal-model simulator S_B as follows:

1. Extract y, and $\beta(y)$ from the adversary's messages to $\mathcal{F}_{\text{OT}}$. For $1 \leq i \leq n$, get (DeliverS, (*sid*, 0, *i*), $B_i[\cdot]$), and (InputR, (*sid*, 1, *i*), y_i) from the adversary. Let $A_i \xleftarrow{\$} \{0,1\}^\lambda$, and send (OutputR, (*sid*, 1, *i*), A_i) and (OutputS, (*sid*, 0, *i*), $B_i[\cdot]$) to the adversary. Let $\beta(y) = \bigoplus_{i=1}^n B_i[y_i]$ and $\alpha = \bigoplus_{i=1}^n A_i$. Abort if any $\mathcal{F}_{\text{OT}}$ aborts.
2. Send (InputB, *sid*, y) to $\mathcal{F}_{\text{PEC}}$ and receive (OutputB, *sid*, $[\![x = y]\!]$). If $x = y$, set $m_A = G(\alpha) \oplus \beta(y)$, otherwise let $m_A \xleftarrow{\$} \{0,1\}^\lambda$. Send (MsgAlice, *sid*, m_A) to the adversary.
3. On (MsgBob, *sid*, m_B) from the adversary, let $b = [\![m_B = \alpha \oplus \beta(y)]\!]$. Send (DeliverA, *sid*, b) to $\mathcal{F}_{\text{PEC}}$.

Again we define a sequence of hybrids to prove indistinguishability.

Hybrid 0. The hybrid runs the code of honest Alice with the input x of dummy Alice, and semi-honestly runs $\mathcal{F}_{\text{OT}}$. This hybrid is trivially indistinguishable by the adversary.

Hybrid 1. This hybrid removes Alice's interaction with the OTs. Compute $\beta(x)$ directly from the observed $B_i[\cdot]$, and send (OutputR, (*sid*, 1, *i*), $A_i[x_i]$) to the adversary. The output to the adversary remains identical, thus hybrids 0 and 1 are indistinguishable.

Hybrid 2. Output (OutputA, *sid*, $[\![m_B = \alpha(y) \oplus \beta(y)]\!][\![x = y]\!]$) for Alice.

If $x = y$ the output remains identical. Otherwise, if $x \neq y$, hybrid 2 outputs zero while hybrid 1 outputs $[\![m_B = \alpha(x) \oplus \beta(x)]\!]$, which might be one. By the difference lemma [Sho04], the distinguishability of the hybrids is thus bounded by $\Pr[m_B = \alpha(x) \oplus \beta(x) \mid x \neq y]$ in hybrid 1.

Assume to the contrary there exists an environment Z and input $(z, 1^\lambda)$ so that Z replies with such an m_B with high probability: there exists a polynomial p such that for infinitely many λ:

$$\Pr[m_B = \alpha(x) \oplus \beta(x) \mid x \neq y] \geq 1/p(\lambda). \tag{4.5}$$

Then there exists an algorithm Z' that finds a preimage of G. On input $X = G(U_\lambda)$ and z, algorithm Z' invokes $Z(z, 1^\lambda)$ and runs hybrid 1, but sends $m_A = X \oplus \beta(x)$ instead of

$G(\alpha(x)) \oplus \beta(x)$. Let m_B be the reply by Z, then Z' outputs $m_B \oplus \beta(x)$. Since $x \neq y$, there is at least one $x_i \neq y_i$ so that $\bigoplus_i A_i[x_i]$ contains at least one uniform term and is therefore $\alpha(x)$ is a uniform random value. Thus X and $G(\alpha(x))$ are distributed identically and so is m_A. Then

$$\Pr[Z'(G(U_\lambda), z) \in G^{-1}G(U_\lambda)] \geq \Pr[Z'(G(U_\lambda), z) = U_\lambda] \geq 1/p(\lambda), \tag{4.6}$$

contrary to the assumption that G is one-way. Thus such a Z cannot exist, proving that hybrid 1 and 2 are indistinguishable.

Hybrid 3. If $x = y$, send $m_A = G(\alpha(y)) \oplus \beta(y)$, otherwise (if $x \neq y$) send $m_A \xleftarrow{\$} \{0,1\}^\lambda$.

If $x = y$, then m_A remains identical. Otherwise, if $x \neq y$, then the environment cannot distinguish the hybrids by the pseudorandom property of G.

Assume to the contrary that there exists an environment Z that on input $(z, 1^\lambda)$ sets $x \neq y$ and distinguishes hybrids 2 and 3. Since $x \neq y$, $\alpha(y)$ is uniformly random. Thus by assumption there exists a polynomial p such that for infinitely many λ:

$$|\Pr[Z(G(U_\lambda) \oplus \beta(x), z, 1^\lambda) = 1] - \Pr[Z(U_\lambda, z, 1^\lambda) = 1]| \geq \frac{1}{p(\lambda)}. \tag{4.7}$$

Then we can build a distinguisher D. D has oracle access to $\mathcal{O}$, which either samples from $G(U_\lambda)$ (denoted $\mathcal{O}_G$), or samples from U_λ (denoted $\mathcal{O}_U$). Let $r \xleftarrow{\$} \mathcal{O}$, then D runs hybrid 2, but it outputs $m_A = r \oplus \beta(x)$ and it outputs whatever Z outputs. Then $D^{\mathcal{O}_G}$ is distributed identical to hybrid 2 and $D^{\mathcal{O}_U}$ is distributed identical to hybrid 3. Thus

$$\begin{aligned} &|\Pr[D^{\mathcal{O}_G}(z, 1^\lambda) = 1] - \Pr[D^{\mathcal{O}_U}(z, 1^\lambda) = 1]| \\ &= |\Pr[Z(G(U_\lambda) \oplus \beta(x), z, 1^\lambda) = 1] - \Pr[Z(U_\lambda \oplus \beta(x), z, 1^\lambda) = 1]| \geq \frac{1}{p(\lambda)}. \end{aligned} \tag{4.8}$$

contrary to the assumption that G is pseudorandom. Thus the distinguisher between hybrid 2 and 3 cannot exist.

Hybrid 4. Generate only $A_i = A_i[y_i]$ for all i. This is the same change as hybrid 3 for corrupt Alice.

Hybrid 5. Replace direct interaction with dummy Alice with interaction via $\mathcal{F}_{\text{PEC}}$: send (InputB, *sid*, y) to $\mathcal{F}_{\text{PEC}}$ to get (OutputB, *sid*, $[\![x = y]\!]$), and construct MsgAlice accordingly. On (MsgBob, *sid*, m_B), send (DeliverA, *sid*, $[\![m_B = \alpha \oplus \beta(y)]\!]$) to $\mathcal{F}_{\text{PEC}}$ to ensure dummy Alice gets output. This is the valid simulator S^B described earlier. □

Note that the above proof could be adjusted to work for a protocol where Bob sends $\alpha(y)$ as MsgBob if he accepts. The reason he sends $\alpha(y) \oplus \beta(y)$ instead is that an implementation might contain a bug and sends this value *when rejecting*. If $x \neq y$, the message $\alpha(y)$ leaks y to a corrupted Alice, while $\alpha(y) \oplus \beta(y)$ does not. The impact of such an implementation error is therefore less severe.

Let π'_{PEC} be the protocol π_{PEC}, as transformed by the SUC protocol transformation (messages are sent via $\mathcal{F}_{\text{MAUTH}}$ instead of via the router). Let $\mathcal{F}'_{\text{PEC}}$ be the functionality $\mathcal{F}_{\text{PEC}}$ as transformed by the SUC functionality transformation (the same functionality with explicitly delayed input/output). Then Theorem 4.5 implies that π'_{PEC} UC-realizes $\mathcal{F}'_{\text{PEC}}$ in the $\mathcal{F}_{\text{OT}}$-hybrid model. And by the UC theorem we have the following corollary:

Corollary 4.6. *Protocol π'_{PEC} UC-realizes $\mathcal{F}'_{PEC}$ in the $(\mathcal{F}_{RO}, \mathcal{F}_{MAUTH})$-hybrid model.*

3.2.1 Quantum security

Simply instantiating the used primitives with post-quantum primitives does not automatically make the protocol secure against quantum adversaries. We follow [HSS11] and define quantum-UC-emulation as normal UC-emulation with two changes: the PPT environment and adversary are replaced with a polynomial time quantum adversary and the classical advice z is replaced with quantum advice: polynomially many qubits with arbitrary state. It should be noted that the quantum variant of the UC-theorem still holds.

The security proof for honest parties and for corrupt Alice does not depend on computational arguments, but instead gives statistical indistinguishability between the real model and ideal model, so that computational indistinguishability against quantum adversaries is immediate.

For the security proof for corrupt Bob, we strengthen the assumptions on G by making the same changes: we assume the one-way property and pseudorandomness hold against all quantum polynomial time adversaries on any quantum advice. Although we made two changes to be explicit, all we did was change the adversary computational power, from P/poly to BQP/qpoly. Stated in terms of [Son14]: this is the game-preserving case.

Security for the quantum case then follows by the quantum lifting theorem for game-preserving reductions [Son14]. To show it applies we simply remark that both reductions are black-box and straight-line.

3.3 Split private equality confirmation

The implemented protocol $\Pi_{\mathcal{F}'_{\text{PEC}}}$ applies the general split transformation (see Figure 4.7) in order to execute π'_{PEC} over a pseudo-authenticated channel. This realizes $s\mathcal{F}'_{\text{PEC}}$: the split functionality (see Figure 4.6) of $\mathcal{F}'_{\text{PEC}}$.

Recall that in the transformation $\Pi_{\mathcal{F}'_{\text{PEC}}}$, relies on split authentication ($s\mathcal{F}_{\text{MAUTH}}$). I realize this with the signature based protocol π_{SA} (see Figure 4.8). In order to achieve post-quantum security, I instantiate those signatures with post-quantum signatures. Note that the security proof of this construction is a simple hybrid argument that reduces its security to the existential unforgeability under chosen message attack (EUF-CMA) property of the signature scheme [BCL+10, Lemma 4.1]. Post-quantum security thus follows directly from the quantum lifting theorem [Son14].

The resulting functionality $s\mathcal{F}'_{\text{PEC}}$ is more complex than a simple PET, but it still suffices in the context of key authentication as done in OTR. Using a PEC instead of a PET does not change the conclusion of honest parties: if a party receives output 0, then the input was unequal because of independent inputs or the other party was corrupt and was tampering with the PEC itself, but either way the party should not accept the OTR session.[22] Having delayed input/output just means that the adversary can always make honest parties abort and reject. Finally the split functionality means that the adversary gets a (single) attempt to impersonate the other party (per honest party), but even then they cannot make the other party accept without providing the correct shared secret input.

Note that one of the motivating use cases for developing split functionalities was to simplify the design of PAKEs [BCL+05]. The authors suggest a functionality $\mathcal{F}_{\text{PW}}$ that takes low-entropy inputs and outputs a high-entropy shared key to both parties if the inputs were equal, or $\perp$ otherwise, so that the split variant $s\mathcal{F}_{\text{PW}}$ is essentially equal to the standard UC PAKE functionality [CHK+05]. The most important distinctive feature of that construction is that it provides the resulting session key as output, instead of only providing the authentication itself. The advantage of the functionality of this chapter is that it neatly separates AKE from key authentication. Specifically in the use case of OTR, instead of having to design an AKE that is both deniable and based on passwords, the AKE design can focus on deniability (and other required properties), while the key authentication protocol can be based on passwords, as long as it does not directly break deniability itself. Offline deniability of $\Pi_{\mathcal{F}'_{PEC}}$ follows directly from the fact that it uses

[22] Unequal inputs could be the result of an honest *user* providing the wrong input. Cryptography cannot protect against this user error, but it does limit the number of adversarial guesses to the number of times the honest party is willing to retry the entire protocol.

only ephemeral keys, while online deniability holds by the fact that authentication is based on a *shared* secret value.

3.3.1 Split authentication

It should be noted that using protocol π_{SA} for realizing $s\mathcal{F}_{\text{MAUTH}}$ could be superfluous in the context of OTR. Generally it would seem that an AKE with unauthenticated keys already provides the parties with a pseudo-authenticated channel: the AKE output can be used to setup a secure channel using symmetric cryptography. Honest parties using this channel know they communicate with the owner of the static key that was used in the AKE. Intuitively that makes sense, but there are a few choices that were made in the current model of the OTR AKE that make proving such a statement non-trivial.

The OTRv4 handshakes are modelled in the UC framework as custom AKE functionalities (with additional properties such as deniability) [Ung21]. The functionalities and their realizing protocols are implemented relative to a PKI functionality. Since the goal of key authentication is to replace such a PKI, we cannot assume its existence. Instead we have to run the protocol with a public key provided by the adversary, in which case some adjustments to the current model (both the functionalities and the protocol) are required.

Another issue is that both AKE functionalities use a standard trick from UC KEX models [CK02]: a corrupt party is allowed to choose the resulting output of the key exchange. In many KEX protocols, the responding party has some small influence over the output: they can compute a few candidates as their share of the KEX and then select the one that gives an output with a desirable property. The ideal functionality gives the corrupt party significantly more influence over the output (recall that to UC-realize a functionality, a protocol must be *at least* as secure the functionality). The problem when trying to realize $s\mathcal{F}_{\text{MAUTH}}$ with such a functionality is that the adversary can split the parties in disjoint subsets and run two AKEs with equal outputs, which means that the subsets are not cleanly separated and messages from one subset might be accepted in another subset. This appears to be an artifact of the model, not a problem in the protocol itself: a functionality that guarantees that KEX outputs do not collide if at least one party is honest would be sufficient for this specific problem, but it is not immediately clear how to formalize such a requirement in the UC framework.

We leave the realization of split authentication directly with the AKE as future work. Note that as a result, the current key authentication protocol is not required to be executed in-band. An in-band implementation may provide an improved user experience (as discussed in Section 2.2).

4 Implementation

The available implementation of $\Pi_{\mathcal{F}'_{\text{PEC}}}$ collapses as many protocol steps as possible into one step. Practically that means that we get a 3-RTT protocol, summarized here.

1. Alice to Bob: ($\mathsf{Msg1}$, vk_A),
   where vk_A is the public key of freshly generated signature keypair (sk_A, vk_A).

2. Bob to Alice: ($\mathsf{Msg2}$, vk_B, σ_2),
   where vk_B is the public key of freshly generated signature keypair (sk_B, vk_B). Let $sid = (vk_A, vk_B)$, and $\sigma_2 = \text{Sign}(sk_B, (\mathsf{Msg2}, sid))$.

3. Alice to Bob: ($\mathsf{Msg3}$, m_3, σ_3),
   Let $sid = (vk_A, vk_B)$. Initialize n OTs as receiver, resulting in message m_3. Compute $\sigma_3 = \text{Sign}(sk_A, (\mathsf{Msg3}, sid, m_3))$.

4. Bob to Alice: ($\mathsf{Msg4}$, m_4, σ_4).
   Execute n OTs as sender (with output $\beta(y)$), and initialize n OTs as receiver, resulting in message m_4. Compute $\sigma_4 = \text{Sign}(sk_B, (\mathsf{Msg4}, sid, m_4))$.

5. Alice to Bob: ($\mathsf{Msg5}$, m_5, m_B, σ_5).
   Complete n OTs as receiver (with output $\beta(x)$), execute n OTs as sender (with output $\alpha(x)$, resulting in message m_5. Let $m_A = G(\alpha(x)) \oplus \beta(x))$. Compute $\sigma_5 = \text{Sign}(sk_A, (\mathsf{Msg5}, sid, m_5, m_A))$.

6. Bob to Alice: ($\mathsf{Msg6}$, m_6, σ_6).
   Complete n OTs as receiver (with output $\alpha(y)$), resulting in message m_6. Accept iff $m_A = G(\alpha(y)) \oplus \beta(y)$, and compute m_B accordingly (see π_{PEC}, step 2). Compute $\sigma_6 = \text{Sign}(sk_B, \mathsf{Msg6}, sid, m_6, m_B))$.

7. Alice.
   Accept iff $m_B = \alpha(x) \oplus \beta(x)$

Each message is accepted only once and in order, messages with invalid signatures are ignored. The local session identifier *sid* is computed in steps 2 and 3. It is used for executing π'_{PEC}, where it is used to ensure that each random oracle (m oracles per OT) is domain-separated with a globally unique identifier. The message counters $\mathsf{Msg1}\ldots\mathsf{Msg6}$ are single byte counters.

A functional C99 prototype of $\Pi_{\mathcal{F}'_{\text{PEC}}}$ is available as the *libkop* library [Ver21], where KOP stands for "(KEM-based OT)-based PEC". The implemented KEM is the combined

instantiation of Section 3.1.3, π_{OT} (Figure 4.11) realizes the OT functionality, and the PEC is realized by π_{PEC} (Figure 4.14). Since we rely on existing libraries for implementing the primitives, the code is simple and small: approximately 3500 lines of code which includes tests and benchmarks.

DH fits the formulation of a KEM with implicit rejection as follows. Given an encoding function $\mathcal{E} : \mathcal{G} \to \mathcal{B}^*$ and a parser $\mathcal{P} : \mathcal{B}^* \to \mathcal{G} \cup \{\perp\}$ that can explicitly fail, define

$$\mathcal{P}'(b) := \begin{cases} \mathcal{P}(b) & \text{if } \mathcal{P}(b) \neq \perp \\ g & \text{if } \mathcal{P}(b) = \perp, \end{cases} \tag{4.9}$$

where $g \in \mathcal{G}$ is the standard generator. For a given keypair $(\mathcal{E}(g^x), x)$, encapsulation generates $c = \mathcal{E}(g^y)$ and shared secret g^{xy}. Decapsulation then outputs $(\mathcal{P}'(c))^x$, which equals g^x when decapsulation (implicitly) fails.

Using KEMs lets us use the standard interface as exposed by available libraries, as long as we take care to use a KEM where decapsulation never explicitly fails. Using a publicly available library has the advantage that including upstream updates is cheap: this covers minor specification updates, security patches and performance improvements. The prototype implements ECDH via the Decaf library [Ham15] with Kyber [BDK+18; ABD+21] via the liboqs library [SM16]. Symmetric cryptography uses the Keccak primitives [BDPA13] via the XKCP library [XKCP]. The signatures use the Dilithium primitive as implemented in liboqs [SM16]. I chose Decaf and Keccak for compatibility with the existing OTRv4 specification and Kyber and Dilithium for post-quantum security with an OT implementation already available.

Decaf public keys form a prime-order group of points on the Ed448 (Goldilocks) elliptic curve with fast complete addition laws. The Elligator map hashes bytestrings ($\mathcal{B}^{112}$) to Decaf points, which allows both sampling random group elements (apply Elligator to a random bytestring) and realization of the random oracle (hash the group elements to a bytestring and then apply Elligator).

The group of Kyber public keys is that of [MR21], with minor efficiency improvements. A Kyber public key is a pair $(t, \rho) \in R_q^k \times \mathcal{B}^{32}$, where R_q is the ring $\mathbb{Z}_q[X]/(X^n + 1)$, $q = 3329$, and k is 2, 3, or 4 (depending on the security level). The group operation is then defined as $(t_0, \rho) + (t_1, \rho) = (t_0 + t_1, \rho)$, with the usual addition in R_q^k. As a minor improvement over libOTe, I avoid packing and immediately unpacking public keys for these group operations. Although we could include addition of ρ in the group operation (for example using the $\oplus$ operation), this leads to a potential vulnerability when considering timing side-channels (discussed later). Instead ρ is kept constant per OT, generated by the Kyber key-generation.

In Kyber the seed ρ is expanded with an extendable-output function (XOF), then using the rejection-sampling subroutine `Parse`, a random element R_q is generated. This is repeated k^2 times to generate a matrix $A \in R_q^{k \times k}$. Using the XOF and `Parse` subroutines we can generate a random public key and implement the random oracles. The libOTe implementation generates the entire matrix and discards all but the first row, while my implementation only generates one vector.

The default parameter set for protocol π_{OT} sets $m = 2$: implementing 1-out-of-2 OT. The Kyber KEM has relatively large public keys and ciphertexts, so I focus on minimizing the size of messages. Note that the message size is identical for *two* 1-out-of-2 OTs and *one* 1-out-of-4 OT: four public keys and four ciphertext per two bits of the input. The measured performance for $m = 2$ is better, despite the fact that $m = 4$ requires fewer public-key operations for the receiver.

We apply domain separation by prefixing ("KOP-RO", *sid*, i_{role}, i_{OT}, i_{RO}) to each hash call: "KOP-RO" is a system constant, *sid* is the concatenation of the ephemeral verification keys, the index $i_{\text{role}} \in \{0, 1\}$ indicates who is the oblivious transfer sender, the index $0 \leq i_{\text{OT}} < \sigma$ indicates the OT, and index $0 \leq i_{\text{RO}} < n$ indicates the random oracle in π_{OT}.

The default parameter set assumes 80-bit inputs, so that $n = 80$ with the default 1-out-of-2 OT. The KEM keys are combined with a KDF, instantiated as SHAKE256("KOP-KDF"$\|k_{ec}\|k_{pq}$), so that $\lambda = 256$. The function $G(x)$ is instantiated as SHA3-256("KOP-PEC-G"$\|x$).

A limitation of the analysis in Theorem 4.5 is that it says nothing about the required size of the security parameter: it is non-tight due to the quantum lifting and requires stronger primitives than are realistically necessary due to the non-uniformity [KM12]. All security parameters have instead been chosen to be sufficiently large to withstand cryptanalysis by *uniform* (quantum) adversaries *on the primitives*.

4.1 Side-channel protection

In the OT, the receiver input j must not leak in any way. As a minimal protection against side-channel attacks, the implementation does not branch upon secret data and does not use secret memory indices. This protection is built into the libraries we use, but here I highlight the additional protection provided by the implementation.

- Implicit rejection in DH decoding uses a constant time select function built into the Decaf library.

- In generating the first OT message, the receiver always computes on the memory layout $[r_j, r_0, \ldots, r_{j-1}, r_{j+1}, \ldots, r_{n-1}]$, then puts element r_j in its correct place with $(n-1)$ constant time conditional swaps.
- Upon receiving the last OT message, the receiver selects the j-th ciphertext and only parses/decapsulates that one. The selection uses conditional moves and runs in time independent of the value of j.
- The OT sender receives m secrets, but requires only one. The required is selected in constant time by a series of conditional moves.
- As mentioned earlier, the Kyber public key contains a seed ρ, that is then (deterministically) expanded with a XOF and then turned into an element of R_q with the (non-constant time) rejection sampling subroutine `Parse`. If we would vary ρ for different group elements in the OT, a corrupt sender could check which received value of ρ matches the time taken to construct the first message. The OT receiver only expands one seed (during key generation) and does not expand the seed for randomly generated group elements. Different OTs can have different values of ρ, so we take the value ρ generated by the Kyber key generation and then ensure that the random sampling, random oracle, and group operations keep that value of ρ constant.

The signature implementation operates on bytes and requires no additional side-channel protection (beyond what is already provided by the library).

4.2 Measurements

With Decaf both KEX messages are 56 bytes. Kyber and Dilithium both come with different parameter sets, targeting NIST PQC security levels 1, 3, and 5 [NIST17]. Kyber has public key sizes of 800, 1184, and 1568 bytes, and ciphertexts of 768, 1088, and 1568 bytes, respectively. Dilithium has public key sizes of 1312, 1952, and 2592 bytes, and signatures of 2420, 3293, and 4595 bytes, respectively. Conservatively choosing the highest security level and 32-byte oblivious encodings, the six messages thus have sizes 2593, 7188, $1624mn + 4596$, $3248mn + 4596$, $1624mn + 4628$, and 4628 bytes, Thus with 1-out-of-2 OT and 80-bit inputs, just over a megabyte of messages is transferred in total. In OTR, some small overhead in both message size and runtime will be introduced if the messages are sent over the established secure channel.

The results are reported in Tables 4.1 and 4.2. Measurements were performed on an Intel Core i5-8265U (Whiskey Lake) clocked at 1.6 GHz, with TurboBoost (overclocking),

Table 4.1: Performance of 1-out-of-m OT: mean runtime and standard deviation in µs.

m	2	4	8	16
`ot_recv_init`	358 ± 2	588 ± 3	1059 ± 5	1980 ± 8
`ot_send`	1159 ± 2	2383 ± 3	5047 ± 3	$11\,450 \pm 50$
`ot_recv_out`	366 ± 1	366 ± 1	367 ± 1	368 ± 1

Table 4.2: Performance of $\Pi_{\mathcal{F}'_{\text{PEC}}}$ on various input sizes: mean runtime and standard deviation in ms. Implemented with 1-out-of-2 OT. The input size $|x|$ is the number of bits in x.

$\lvert x\rvert$	40	80	128	256
`alice_m1`	0.164 ± 0.002	0.164 ± 0.002	0.165 ± 0.002	0.17 ± 0.02
`bob_m2`	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1	0.5 ± 0.1
`alice_m3`	15.6 ± 0.1	31.5 ± 0.4	48.4 ± 0.2	96.3 ± 0.2
`bob_m4`	63.7 ± 0.2	128.4 ± 0.2	202.3 ± 0.3	404 ± 1
`alice_m5`	64.0 ± 0.2	128.2 ± 0.1	203.7 ± 0.2	406.9 ± 0.2
`bob_m6`	15.9 ± 0.2	31.2 ± 0.4	49.6 ± 0.2	98.7 ± 0.2

hyperthreading, and underclocking disabled. For a reference point, we measured the combined KEM on the same system at: keygen 104 µs, encaps 438 µs, and decaps 364 µs. On this specific system, TurboBoost increases the clockspeed to 3.9 GHz, which means that the reported runtimes can trivially be cut in half. Although I expect that the relative infrequent use of the protocol (at least in the context of OTR) means that a real-world deployment can take full advantage of TurboBoost, I decided to report the results without TurboBoost for increased reproducability.

5 Discussion

This work gives a solution to the key authentication problem in secure messaging protocols based on post-quantum primitives. Most popular secure messaging applications solve this problem out-of-band, but OTR provides an in-band solution in the form of the SMP, a solution to the socialist millionaire problem, but which is not secure against quantum adversaries.

I provide a solution in the form of a PET that allows for a one-sided error in its output when either of the parties is malicious. The protocol uses OTs as a building block. Although any post-quantum OT will suffice, we chose one that can be constructed out of KEX protocols. This means it can be built out of post-quantum KEMs. The entire protocol is executed over a pseudo-authenticated channel, which is realized from post-quantum signatures. The underlying cryptography and implementations of these post-quantum primitives have been under scrutiny of many cryptographers. The KEM construction is realized with a hybrid solution: the output of a post-quantum KEM and a pre-quantum KEM are hashed together, so that the overall protocol is secure as long as one of the KEMs is secure.

A mathematical argument towards the security of the protocol has been provided, by proving that the PEC protocol is secure in the SUC framework, specifically in the OT-hybrid model. The protocol is then executed over a pseudo-authenticated channel, which is realized with a general signature construction. The proof structure (in the OT-hybrid model) is simple, so that it can be lifted to the quantum setting and therefore it is quantum secure when instantiated with post-quantum primitives. The mathematical argument relies on the existence of post-quantum signatures and a quantum secure protocol for OT. A downside of the chosen OT protocol is that its proof structure is not simple, so despite claims by the author that the protocol is quantum resistant, the post-quantum security of the OT (and therefore the security of the entire protocol) is technically still a

conjecture. If this is considered unsatisfying, the OT protocol can be swapped out for any other post-quantum UC-secure one.

An online implementation is available [Ver21]. It implements a combined ECDH/Kyber KEX protocol, uses Dilithium signatures, and uses Keccak for symmetric cryptography, where each primitive is instantiated with conservatively chosen parameters. It should be noted that the choice of parameters was not driven by the mathematical proof, but by the availability of cryptographic implementations, choosing the largest security parameters available. The result is a fast-enough protocol with messages that are about the size of sending a large photo. Tighter cryptanalysis could reveal that the parameters can be decreased, increasing the efficiency of the construction.

Future research could focus on whether more efficient protocols exist, which seems likely. Possibly these can be built directly from some of the properties provided by (for example) code-, lattice-, or isogeny-based cryptosystems. Whatever the solution, it may benefit from targeting our weaker $s\mathcal{F}'_{\text{PEC}}$ functionality instead of aiming to construct a PET. An improvement that possibly eliminates a full RTT would be to use the AKE itself for realizing the split authentication. Within the current model, it is still unknown if such a construction is secure or can be made secure with some minor adjustments to either the functionality and/or the protocol.

Another research area that might prove fruitful is that of PAKEs, since these protocols are already tasked with establishing strong cryptographic guarantees from low-entropy shared secrets. Indeed PAKEs can be used for realizing key authentication. With ongoing development and research into the security of post-quantum PAKEs, the confidence in these systems may increase so that a PAKE based implementations may become the preferable solution. Some may feel confident enough already to prefer a PAKE.

Chapter 5

Conclusion

A formal approach to cryptography is fundamental for security. By constructing an abstract model and then applying rigorous mathematical reasoning, we are able to capture a broad class of adversarial behaviour and provide strong guarantees of security against such adversaries. To assess if this leads to real-world security, we should ask two questions: can an attacker realistically bypass the adversary model to break security, and are the security guarantees meaningful and complete? A negative answer to either of those questions indicates a gap between theoretic and practical security. When possible, that gap should be narrowed with stronger models and increasingly powerful mathematics, but otherwise sound engineering practices are required.

Quantum cryptography, despite eliminating the need for computational complexity assumptions in its models, is no exception. The key exhaustion attack on quantum key distribution (QKD) demonstrates this. The attack does not indicate a mistake in any of the existing security proofs, nor does it put into question any of the assumptions made by those proofs: theoretically QKD operates exactly as specified. Practical deployments however require some method to recover from key exhaustion, either in the form of a ceremony (between users) or a protocol (between devices), both of which have a significant chance of lowering the security of future communications. Preventing key exhaustion is therefore essential.

I proposed two mitigations against key exhaustion in QKD. The mitigations combine computationally secure authentication and information theoretically secure (ITS) authentication. The resulting protocol provides computational protection against key exhaustion, without compromising any of the ITS guarantees that QKD provides over its output. The current arguments for the security of the mitigations are not as rigorous as the arguments

for other guarantees provided by cryptographic protocols. Currently the biggest obstacle in that regard is the non-existence of a mathematical definition of availability, which can be the subject of future research.

Mathematical reasoning should be used wherever possible. This is especially true in the context of quantum information, which is notorious for being unintuitive. The intuitions we have built up in reasoning about classical systems do not always carry over to quantum systems, as demonstrated by the attacks on quantum distance bounding protocols. Each of the protocols without any terrorist fraud (TF) resistance is secure, but when combined with a classical countermeasure (that is effective and secure in classical protocols), the result is completely insecure.

Much of the existing work on distance bounding operates in informal frameworks. Currently there is lack of consensus on the right formal model for time-of-flight in communication and on the security definition of TF. The lack of a formal adversary model means that it is unclear what the adversary can and cannot do by employing quantum strategies. Future research could study whether quantum strategies affect classical distance bounding protocols more generally than demonstrated in this thesis.

The last chapter highlights another aspect of real-world security that is not covered by mathematics: the usability of security software. Usability studies indicate that the currently popular solution of manually comparing fingerprints for key authentication is problematic for many users and results in diminished security. The socialist millionaire protocol (SMP) has the potential to remove some of these problems, and I believe that a post-quantum replacement is essential for security in the near future.

The split private equality confirmation (PEC) protocol as presented in this thesis provides such a post-quantum replacement. Much of the focus of the research community lately has been on key encapsulation mechanisms (KEMs) and signatures: the categories in the NIST PQC project. KEMs (with some assumptions on the structure of the public keys) can be used to construct oblivious transfer (OT), a powerful cryptographic tool. Indeed in this thesis I use OT to construct the PEC, and a prototype implementation (also using post-quantum signatures) demonstrates that the construction is relatively efficient. Building the protocol from KEMs and signatures has several advantages: it is built from primitives that have received much scrutiny by the cryptographic community, it allows for a simple and robust implementation that uses available cryptographic libraries, and it allows combining multiple KEMs such that the protocol is secure as long as one KEM is secure. The current analysis of the protocol is not without limitations. Specifically, future research could strengthen the security argument by analyzing the OT construction in the QROM and by considering concrete analysis against uniform adversaries.

References

[ABB+14] R. Alléaume, C. Branciard, J. Bouda, T. Debuisschert, M. Dianati, N. Gisin, M. Godfrey, P. Grangier, T. Länger, N. Lütkenhaus, C. Monyk, P. Painchault, M. Peev, A. Poppe, T. Pornin, J. Rarity, R. Renner, G. Ribordy, M. Riguidel, L. Salvail, A. Shields, H. Weinfurter, and A. Zeilinger. "Using quantum key distribution for cryptographic purposes: A survey". In: *Theoretical Computer Science* 560.P1 (Dec. 2014), pp. 62–81. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.09.018 (cit. on p. 24).

[ABB+15] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, Tanja Lange, Mohamed Saied Emam Mohamed, Christian Rechberger, Peter Schwabe, Nicolas Sendrier, Frederik Vercauteren, and Bo-Yin Yang. *Initial recommendations of long-term secure post-quantum systems*. Tech. rep. Technische Universiteit Eindhoven, Sept. 2015. URL: https://pqcrypto.eu.org/docs/initial-recommendations.pdf (visited on 08/07/2021) (cit. on p. 3).

[ABB+18] Gildas Avoine, Muhammed Ali Bingöl, Ioana Boureanu, Srdjan Čapkun, Gerhard P. Hancke, Süleyman Kardas, Chong Hee Kim, Cédric Lauradoux, Benjamin Martin, Jorge Munilla, Alberto Peinado, Kasper Bonne Rasmussen, Dave Singelée, Aslan Tchamkerten, Rolando Trujillo-Rasua, and Serge Vaudenay. "Security of Distance-Bounding: A Survey". In: *ACM Computing Surveys* 51.5 (Sept. 2018), pp. 1–33. DOI: 10.1145/3264628 (cit. on pp. 63, 66, 100).

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. *Classic McEliece: conservative code-based cryptogra-*

phy. Post-Quantum Cryptography – Round 3. National Institute of Standards and Technology, Oct. 2020. URL: https://classic.mceliece.org/nist/mceliece-20201010.pdf (visited on 08/07/2021) (cit. on p. 129).

[ABD+17] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber, Algorithm Specifications and Supporting Documentation*. Post-Quantum Cryptography – Round 1. National Institute of Standards and Technology, Nov. 2017. URL: https://pq-crystals.org/kyber/data/kyber-specification.pdf (visited on 08/07/2021) (cit. on p. 127).

[ABD+21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber, Algorithm Specifications and Supporting Documentation (version 3.01)*. Post-Quantum Cryptography – Round 3. National Institute of Standards and Technology, Jan. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf (visited on 08/07/2021) (cit. on pp. 131, 143).

[Abi19] Aysajan Abidin. "Quantum Distance Bounding". In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '19. Miami, Florida: Association for Computing Machinery, May 2019, pp. 233–238. ISBN: 9781450367264. DOI: 10.1145/3317549.3323414 (cit. on pp. 6, 57, 75, 88, 89, 91, 99, 100).

[Abi20] Aysajan Abidin. "On Detecting Relay Attacks on RFID Systems Using Qubits". In: *Cryptography* 4.2 (May 2020), p. 14. DOI: 10.3390/cryptography4020014 (cit. on pp. 6, 57, 75, 92, 93, 99, 100).

[ACD19] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 129–158. ISBN: 978-3-030-17653-2. DOI: 10.1007/978-3-030-17653-2_5 (cit. on p. 102).

[AG07] Chris Alexander and Ian Goldberg. "Improved User Authentication in Off-the-Record Messaging". In: *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. WPES '07. Alexandria, Virginia, USA: Association for Computing Machinery, Oct. 2007, pp. 41–47. ISBN: 9781595938831. DOI: 10.1145/1314333.1314340 (cit. on p. 103).

[AHIC15] Hala Assal, Stephanie Hurtado, Ahsan Imran, and Sonia Chiasson. "What's the deal with privacy apps?: a comprehensive exploration of user perception and usability". In: *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. Ed. by Clemens Holzmann and René Mayrhofer. ACM, Nov. 2015, pp. 25–36. DOI: 10.1145/2836041.2836044 (cit. on p. 107).

[ALM11] Gildas Avoine, Cédric Lauradoux, and Benjamin Martin. "How secret-sharing can defeat terrorist fraud". In: *Proceedings of the Fourth ACM Conference on Wireless Network Security*. Ed. by Dieter Gollmann, Dirk Westhoff, Gene Tsudik, and N. Asokan. ACM, June 2011, pp. 145–156. DOI: 10.1145/1998412.1998437 (cit. on pp. 59, 67).

[AMSP17] Aysajan Abidin, Eduard Marin, Dave Singelée, and Bart Preneel. "Towards Quantum Distance Bounding Protocols". In: *Radio Frequency Identification and IoT Security 2016*. Ed. by Gerhard P. Hancke and Konstantinos Markantonakis. Cham: Springer International Publishing, 2017, pp. 151–162. ISBN: 978-3-319-62024-4. DOI: 10.1007/978-3-319-62024-4_11 (cit. on pp. 6, 57, 75, 77, 78, 85–87, 99, 100).

[AT17] Nicholas Akinyokun and Vanessa Teague. "Security and Privacy Implications of NFC-enabled Contactless Payment Systems". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, Sept. 2017, pp. 1–10. ISBN: 9781450352574. DOI: 10.1145/3098954.3103161 (cit. on p. 67).

[BB05] Laurent Bussard and Walid Bagga. "Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks". In: *Security and Privacy in the Age of Ubiquitous Computing, SEC 2005*. Ed. by Ryôichi Sasaki, Sihan Qing, Eiji Okamoto, and Hiroshi Yoshiura. Vol. 181. IFIP. Springer, 2005, pp. 223–238. DOI: 10.1007/0-387-25660-1_15 (cit. on p. 63).

[BB84] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: public key distribution and coin tossing". In: *International Conference on Computers, Systems and Signal Processing*. Vol. 1. Bangalore, India, Dec. 1984, pp. 175–179. arXiv: 2003.06557 [quant-ph] (cit. on pp. 3, 13, 18, 41).

[BBD+91] Samy Bengio, Gilles Brassard, Yvo Desmedt, Claude Goutier, and Jean-Jacques Quisquater. "Secure Implementations of Identification Systems". In: *Journal of Cryptology* 4 (May 1991), pp. 175–183. DOI: 10.1007/BF00196726 (cit. on pp. 62, 69).

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*. 1st ed. Information Security and Cryptography. Springer-Verlag Berlin, 2009. ISBN: 978-3-540-88701-0. DOI: 10.1007/978-3-540-88702-7 (cit. on p. 3).

[BC93] Stefan Brands and David Chaum. "Distance-Bounding Protocols". In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 344–359. ISBN: 978-3-540-48285-7. DOI: 10.1007/3-540-48285-7_30 (cit. on pp. 56, 61, 63, 100).

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. "Secure Computation Without Authentication". In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer, 2005, pp. 361–377. ISBN: 978-3-540-31870-5. DOI: 10.1007/11535218_22 (cit. on pp. 115, 116, 120–123, 140, 154).

[BCL+10] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. *Secure Computation Without Authentication*. Revision of 2010-08-20, this is the full version of [BCL+05]. Aug. 2010. Cryptology ePrint Archive: 2007/464 (cit. on pp. 122, 140).

[BCMM00] Dagmar Bruß, Mirko Cinchetti, G. Mauro D'Ariano, and Chiara Macchiavello. "Phase-covariant quantum cloning". In: *Physical Review A* 62.012302 (June 2000), p. 7. DOI: 10.1103/PhysRevA.62.012302 (cit. on p. 87).

[BDCZ98] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. "Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication". In: *Physical Review Letters* 81.26 (Dec. 1998), pp. 5932–5935. DOI: 10.1103/PhysRevLett.81.5932 (cit. on p. 13).

[BDD+17] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. *A Framework for Efficient Adaptively Secure Composable Oblivious Transfer in the ROM*. Oct. 2017. Cryptology ePrint Archive: 2017/993 (cit. on p. 127).

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 41–69. DOI: 10.1007/978-3-642-25385-0_3 (cit. on pp. 119, 131).

[BDG20] Mihir Bellare, Hannah Davis, and Felix Günther. "Separate Your Domains: NIST PQC KEMs, Oracle Cloning and Read-Only Indifferentiability". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 3–32. DOI: 10.1007/978-3-030-45724-2_1 (cit. on pp. 119, 130).

[BDH11] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. "XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions". In: *Post-Quantum Cryptography*. Ed. by Bo-Yin Yang. Berlin, Heidelberg: Springer, 2011, pp. 117–129. ISBN: 978-3-642-25405-5. DOI: 10.1007/978-3-642-25405-5_8 (cit. on pp. 19, 26).

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *European Symposium on Security and Privacy*. IEEE, Apr. 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032 (cit. on p. 143).

[BDPA13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Keccak". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 313–314. DOI: 10.1007/978-3-642-38348-9_19 (cit. on p. 143).

[Ber18] Daniel J. Bernstein. *Is the security of quantum cryptography guaranteed by the laws of physics?* Mar. 2018. arXiv: 1803.04520 [quant-ph] (cit. on p. 12).

[BFG+21a] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. *Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake*. June 2021. Cryptology ePrint Archive: 2021/769 (cit. on p. 102).

[BFG+21b] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. "Towards Post-Quantum Security for Signal's X3DH Handshake". In: *Selected Areas in Cryptography*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Cham: Springer, 2021, pp. 404–430. ISBN: 978-3-030-81652-0. DOI: 10.1007/978-3-030-81652-0_16 (cit. on p. 120).

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. "Off-the-Record Communication, or, Why Not to Use PGP". In: *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. WPES '04. Washington DC, USA: Association for Computing Machinery, 2004, pp. 77–84. ISBN: 1581139683. DOI: 10.1145/1029179.1029200 (cit. on pp. 102, 105).

[BH96] V. Bužek and M. Hillery. "Quantum copying: Beyond the no-cloning theorem". In: *Physical Review A* 54.3 (Sept. 1996), pp. 1844–1852. DOI: 10.1103/PhysRevA.54.1844 (cit. on p. 87).

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. "The SPHINCS$^+$ Signature Framework". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM, 2019, pp. 2129–2146. DOI: 10.1145/3319535.3363229 (cit. on pp. 19, 50).

[BHL+05] Michael Ben-Or, Michał Horodecki, Debbie W. Leung, Dominic Mayers, and Jonathan Oppenheim. "The Universal Composable Security of Quantum Key Distribution". In: *Theory of Cryptography*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer, 2005, pp. 386–406. ISBN: 978-3-540-30576-7. DOI: 10.1007/978-3-540-30576-7_21 (cit. on p. 33).

[BL17] Daniel J. Bernstein and Tanja Lange. "Post-quantum cryptography". In: *Nature* 549 (Sept. 2017), pp. 188–194. DOI: 10.1038/nature23461 (cit. on p. 3).

[BMS20] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment*. 2nd ed. Information Security and Cryptography. Springer, 2020. ISBN: 978-3-662-58145-2. DOI: 10.1007/978-3-662-58146-9 (cit. on p. 125).

[BMV15] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. "Practical and provably secure distance-bounding". In: *Journal of Computer Security* 23.2 (2015), pp. 229–257. DOI: 10.3233/JCS-140518 (cit. on pp. 66, 67).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer, 2000, pp. 139–155. ISBN: 978-3-540-45539-4. DOI: 10.1007/3-540-45539-6_11 (cit. on p. 124).

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM, 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on p. 118).

[BST01] Fabrice Boudot, Berry Schoenmakers, and Jacques Traoré. "A fair and efficient solution to the socialist millionaires' problem". In: *Discrete Applied Mathematics* 111.1 (2001). Coding and Cryptology, pp. 23–36. ISSN: 0166-218X. DOI: 10.1016/S0166-218X(00)00342-5 (cit. on pp. 7, 106, 108, 109).

[BV15] Ioana Boureanu and Serge Vaudenay. "Optimal Proximity Proofs". In: *Information Security and Cryptology*. Ed. by Dongdai Lin, Moti Yung, and Jianying Zhou. Cham: Springer International Publishing, 2015, pp. 170–190. ISBN: 978-3-319-16745-9. DOI: 10.1007/978-3-319-16745-9_10 (cit. on p. 66).

[CAD+20] David A. Cooper, Daniel C. Apon, Quynh H. Dang, Michael S. Davidson, Morris J. Dworkin, and Carl A. Miller. *Recommendation for Stateful Hash-Based Signature Schemes*. Tech. rep. Special Publication 800-208. National Institute of Standards and Technology, Oct. 2020. DOI: 10.6028/NIST.SP.800-208 (cit. on p. 51).

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*. At the moment of writing, an additional 16 versions of this work exist on ePrint (https://eprint.iacr.org/2000/067, latest 2020-02-12) and 2 versions in the Journal of the ACM (10.1145/3402457). IEEE Computer Society, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888 (cit. on pp. 7, 104, 110, 113, 115).

[CCD+20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. "A Formal Security Analysis of the Signal Messaging Protocol". In: *Journal of Cryptology* 33 (2020), pp. 1914–1983. DOI: 10.1007/s00145-020-09360-1 (cit. on p. 102).

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. "A Simpler Variant of Universally Composable Security for Standard Multiparty Computation". In: *Advances in Cryptology – CRYPTO 2015*. Ed. by Rosario Gennaro and

Matthew Robshaw. Berlin, Heidelberg: Springer, 2015, pp. 3–22. ISBN: 978-3-662-48000-7. DOI: 10.1007/978-3-662-48000-7_1 (cit. on pp. 104, 115–117).

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. "The Random Oracle Methodology, Revisited". In: *Journal of the ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411. DOI: 10.1145/1008731.1008734 (cit. on p. 119).

[CGMO09] Nishanth Chandran, Vipul Goyal, Ryan Moriarty, and Rafail Ostrovsky. "Position Based Cryptography". In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 391–407. DOI: 10.1007/978-3-642-03356-8_23 (cit. on p. 68).

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Phil MacKenzie. "Universally Composable Password-Based Key Exchange". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Berlin, Heidelberg: Springer, 2005, pp. 404–421. ISBN: 978-3-540-32055-5. DOI: 10.1007/11426639_24 (cit. on pp. 124, 140).

[CHTW04] Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. "Consequences and Limits of Nonlocal Strategies". In: *Proceedings. 19th IEEE Annual Conference on Computational Complexity (CCC'04)*. June 2004, pp. 236–249. DOI: 10.1109/CCC.2004.1313847 (cit. on p. 101).

[CK02] Ran Canetti and Hugo Krawczyk. "Universally Composable Notions of Key Exchange and Secure Channels". In: *Advances in Cryptology — EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Berlin, Heidelberg: Springer, 2002, pp. 337–351. ISBN: 978-3-540-46035-0. DOI: 10.1007/3-540-46035-7_22 (cit. on p. 141).

[Con20] Car Connectivity Consortium. *Digital Key - The Future of Vehicle Access (Whitepaper)*. Tech. rep. Release 2.0. Apr. 2020. URL: https://global-carconnectivity.org/wp-content/uploads/2020/04/CCC_Digital_Key_2.0.pdf (visited on 08/07/2021) (cit. on p. 67).

[Con76] John H. Conway. *On numbers and games*. Ed. by P. M. Cohn and G. E. H. Reuter. Vol. 6. London Mathematical Society Monographs. Academic Press, 1976. ISBN: 0-12-186350-6 (cit. on p. 62).

[CP92] David Chaum and Torben P. Pedersen. “Wallet Databases with Observers”. In: *Advances in Cryptology - CRYPTO ’92, 12th Annual International Cryptology Conference*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 89–105. DOI: 10.1007/3-540-48071-4_7 (cit. on p. 108).

[CRSČ12] Cas J. F. Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Čapkun. “Distance Hijacking Attacks on Distance Bounding Protocols”. In: *IEEE Symposium on Security and Privacy, SP 2012*. IEEE Computer Society, 2012, pp. 113–127. DOI: 10.1109/SP.2012.17 (cit. on p. 63).

[CZC+21] Yu-Ao Chen, Qiang Zhang, Teng-Yun Chen, Wen-Qi Cai, Sheng-Kai Liao, Jun Zhang, Kai Chen, Juan Yin, Ji-Gang Ren, Zhu Chen, Sheng-Long Han, Qing Yu, Ken Liang, Fei Zhou, Xiao Yuan, Mei-Sheng Zhao, Tian-Yin Wang, Xiao Jiang, Liang Zhang, Wei-Yue Liu, Yang Li, Qi Shen, Yuan Cao, Chao-Yang Lu, Rong Shu, Jian-Yu Wang, Li Li, Nai-Le Liu, Feihu Xu, Xiang-Bin Wang, Cheng-Zhi Peng, and Jian-Wei Pan. “An integrated space-to-ground quantum communication network over 4,600 kilometres”. In: *Nature* 589 (Jan. 2021), pp. 214–219. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03093-8 (cit. on p. 24).

[DAL+17] Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. “Provably Secure Password Authenticated Key Exchange Based on RLWE for the Post-Quantum World”. In: *Topics in Cryptology – CT-RSA 2017*. Ed. by Helena Handschuh. Cham: Springer International Publishing, 2017, pp. 183–204. ISBN: 978-3-319-52153-4. DOI: 10.1007/978-3-319-52153-4_11 (cit. on p. 124).

[DFKO11] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. “A Formal Approach to Distance-Bounding RFID Protocols”. In: *Information Security*. Ed. by Xuejia Lai, Jianying Zhou, and Hui Li. Berlin, Heidelberg: Springer, 2011, pp. 47–62. ISBN: 978-3-642-24861-0. DOI: 10.1007/978-3-642-24861-0_4 (cit. on p. 66).

[DGB87] Yvo Desmedt, Claude Goutier, and Samy Bengio. “Special Uses and Abuses of the Fiat-Shamir Passport Protocol”. In: *Advances in Cryptology - CRYPTO ’87, A Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Carl Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 21–39. DOI: 10.1007/3-540-48184-2_3 (cit. on pp. 62, 68).

[DHHM99] Miloslav Dušek, Ondřej Haderka, Martin Hendrych, and Robert Myška. "Quantum identification system". In: *Physical Review A* 60.1 (July 1999), pp. 149–156. DOI: 10.1103/PhysRevA.60.149 (cit. on p. 22).

[Eke91] Artur K. Ekert. "Quantum cryptography based on Bell's theorem". In: *Physical Review Letters* 67 (Aug. 1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661 (cit. on p. 13).

[Ell07] Carl M. Ellison. *Ceremony Design and Analysis*. Oct. 2007. Cryptology ePrint Archive: 2007/399 (cit. on pp. 4, 24).

[ETSI] ETSI QSC working group. *Quantum Safe Cryptography (QSC)*. URL: https://www.etsi.org/technologies/quantum-safe-cryptography (visited on 08/07/2021) (cit. on p. 3).

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. "Comparing Information Without Leaking It". In: *Communications of the ACM* 39.5 (1996), pp. 77–85. DOI: 10.1145/229459.229469 (cit. on p. 109).

[FO13] Marc Fischlin and Cristina Onete. "Terrorism in Distance Bounding: Modeling Terrorist-Fraud Resistance". In: *Applied Cryptography and Network Security*. Ed. by Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer, 2013, pp. 414–431. ISBN: 978-3-642-38980-1. DOI: 10.1007/978-3-642-38980-1_26 (cit. on p. 66).

[FS86] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology - CRYPTO '86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12 (cit. on p. 108).

[FSK10] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering - Design Principles and Practical Applications*. Wiley, 2010. ISBN: 978-0-470-47424-2. URL: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470474246.html (visited on 08/07/2021) (cit. on p. 50).

[FV16] Jennifer Katherine Fernick and Sebastian R. Verschoor. *Private patent*. Dec. 2016. URL: https://github.com/sebastianv89/private-patent (visited on 08/07/2021) (cit. on p. 174).

[FV17] Jennifer Katherine Fernick and Sebastian Verschoor. "Privacy-Preserving Patent Search: Additively Homomorphic Encryption Techniques for Private Text Mining over Public Datasets". In: *ECML PKDD 2017, Workshop Data Mining with Secure Computation*. Sept. 2017. URL: https://drive.google.com/file/d/0B1v5Ij69wutGZEVtRFBJTEVtN0tqd3Nrb2FDcVRhUlo4QWJr/view (visited on 08/07/2021) (cit. on p. 174).

[GH00] Gerald Gilbert and Michael Hamrick. *Practical Quantum Cryptography: A Comprehensive Analysis (Part One)*. Sept. 2000. arXiv: quant-ph/0009027 (cit. on p. 22).

[GJZ01] Joshua D. Guttman, F. Javier Thayer, and Lenore D. Zuck. "The faithfulness of abstract protocol analysis: message authentication". In: *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security*. Ed. by Michael K. Reiter and Pierangela Samarati. ACM, 2001, pp. 186–195. DOI: 10.1145/501983.502009 (cit. on pp. 59, 60, 65).

[Gle87] James Gleick. *A new approach to protecting secrets is discovered*. Feb. 17, 1987. URL: https://www.nytimes.com/1987/02/17/science/a-new-approach-to-protecting-secrets-is-discovered.html (visited on 08/07/2021) (cit. on p. 68).

[Gol04] Oded Goldreich. *The Foundations of Cryptography – Volume 2: Basic Applications*. Cambridge University Press, 2004. ISBN: 0-521-83084-2. DOI: 10.1017/CBO9780511721656 (cit. on p. 4).

[Gro96] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*. July 1996, pp. 212–219. DOI: 10.1145/237814.237866 (cit. on p. 3).

[Ham15] Mike Hamburg. "Decaf: Eliminating Cofactors Through Point Compression". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. Lecture Notes in Computer Science. Springer, 2015, pp. 705–723. DOI: 10.1007/978-3-662-47989-6_34 (cit. on pp. 131, 143).

[Han11] Gerhard P. Hancke. "Practical eavesdropping and skimming attacks on high-frequency RFID tokens". In: *Journal of Computer Security* 19.2 (2011), pp. 259–288. DOI: 10.3233/JCS-2010-0407 (cit. on p. 67).

[HBG+18] A. Heulsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen. *XMSS: eXtended Merkle Signature Scheme*. RFC 8391. RFC Editor, May 2018. DOI: 10.17487/RFC8391 (cit. on p. 19).

[Hel67] Carl W. Helstrom. "Detection theory and quantum mechanics". In: *Information and Control* 10.3 (Mar. 1967), pp. 254–291. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(67)90302-6 (cit. on p. 74).

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *Theory of Cryptography - 15th International Conference, TCC 2017*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. Lecture Notes in Computer Science. Springer, 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12 (cit. on p. 129).

[HK05] Gerhard P. Hancke and Markus G. Kuhn. "An RFID Distance Bounding Protocol". In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. Sept. 2005, pp. 67–73. DOI: 10.1109/SECURECOMM.2005.56 (cit. on p. 63).

[HL16] Amir Herzberg and Hemi Leibowitz. "Can Johnny finally encrypt?: evaluating E2E-encryption in popular IM applications". In: *Proceedings of the 6th Workshop on Socio-Technical Aspects in Security and Trust, STAST 2016*. Ed. by Gabriele Lenzini, Giampaolo Bella, Zinaida Benenson, and Carrie E. Gates. ACM, 2016, pp. 17–28. DOI: 10.1145/3046055.3046059 (cit. on p. 106).

[HLS+21] Amir Herzberg, Hemi Leibowitz, Kent E. Seamons, Elham Vaziripour, Justin Wu, and Daniel Zappala. "Secure Messaging Authentication Ceremonies Are Broken". In: *IEEE Security & Privacy* 19.2 (2021), pp. 29–37. DOI: 10.1109/MSEC.2020.3039727 (cit. on p. 106).

[HO17] Cormac Herley and Paul C. van Oorschot. "SoK: Science, Security and the Elusive Goal of Security as a Scientific Pursuit". In: *IEEE Symposium on Security and Privacy SP*. IEEE Computer Society, 2017, pp. 99–120. DOI: 10.1109/SP.2017.38 (cit. on p. 1).

[Hol72] Alexander S. Holevo. "An analogue of statistical decision theory and noncommutative probability theory". In: *Trudy Moskovskogo Matematicheskogo Obshchestva* 26 (1972), pp. 133–149 (cit. on p. 74).

[HSS11] Sean Hallgren, Adam Smith, and Fang Song. "Classical Cryptographic Protocols in a Quantum World". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Berlin, Heidelberg: Springer, 2011, pp. 411–428. ISBN: 978-3-642-22792-9. DOI: 10.1007/978-3-642-22792-9_23 (cit. on p. 139).

[Hwa03] Won-Young Hwang. "Quantum Key Distribution with High Loss: Toward Global Secure Communication". In: *Physical Review Letters* 91.057901 (Aug. 2003). DOI: 10.1103/PhysRevLett.91.057901 (cit. on p. 26).

[Ina02] Hitoshi Inamori. "Security of Practical Time-Reversed EPR Quantum Key Distribution". In: *Algorithmica* 34 (Nov. 2002), pp. 340–366. ISSN: 1432-0541. DOI: 10.1007/s00453-002-0983-4 (cit. on p. 22).

[ISO18] *Information technology - Security techniques - Information security management systems - Overview and vocabulary*. Standard. ISO/IEC 27000:2018(E). Geneva, CH: International Organization for Standardization, Feb. 2018. URL: https://www.iso.org/standard/73906.html (visited on 08/07/2021) (cit. on p. 1).

[JA16] Hoda Jannati and Ebrahim Ardeshir-Larijani. "Detecting relay attacks on RFID communication systems using quantum bits". In: *Quantum Information Processing* 15.11 (Aug. 2016), pp. 4759–4771. DOI: 10.1007/s11128-016-1418-5 (cit. on pp. 57, 92).

[JSK+16] Nitin Jain, Birgit Stiller, Imran Khan, Dominique Elser, Christoph Marquardt, and Gerd Leuchs. "Attacks on practical quantum key distribution systems (and how to prevent them)". In: *Contemporary Physics* 57.3 (2016), pp. 366–387. DOI: 10.1080/00107514.2016.1148333 (cit. on p. 12).

[KAK+08] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. "The Swiss-Knife RFID Distance Bounding Protocol". In: *Information Security and Cryptology - ICISC 2008, 11th International Conference*. Ed. by Pil Joong Lee and Jung Hee Cheon. Vol. 5461. Lecture Notes in Computer Science. Springer, 2008, pp. 98–115. DOI: 10.1007/978-3-642-00730-9_7 (cit. on pp. 63, 64, 94, 97, 99, 100).

[KCP07] Timo Kasper, Dario Carluccio, and Christof Paar. "An Embedded System for Practical Security Analysis of Contactless Smartcards". In: *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*. Ed. by Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater. Vol. 4462. Lecture Notes in Computer Science. Springer, 2007, pp. 150–160. DOI: 10.1007/978-3-540-72354-7_13 (cit. on p. 67).

[Kil88] Joe Kilian. "Founding Cryptography on Oblivious Transfer". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*. Ed. by Janos Simon. ACM, Jan. 1988, pp. 20–31. DOI: 10.1145/62212.62215 (cit. on p. 119).

[KLM07] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. New York: Oxford University Press, 2007. ISBN: 978-0-19-857049-3 (cit. on p. 69).

[KM07] Neal Koblitz and Alfred Menezes. "Another Look at "Provable Security"". In: *Journal of Cryptology* 20.1 (Nov. 2007), pp. 3–37. DOI: 10.1007/s00145-005-0432-z (cit. on p. 7).

[KM12] Neal Koblitz and Alfred Menezes. *Another look at non-uniformity*. June 2012. Cryptology ePrint Archive: 2012/359 (cit. on p. 144).

[KM15] Neal Koblitz and Alfred J. Menezes. "The random oracle model: a twenty-year retrospective". In: *Designs, Codes and Cryptography* 77.2-3 (2015), pp. 587–610. DOI: 10.1007/s10623-015-0094-2 (cit. on p. 119).

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*. Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Springer, 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9 (cit. on p. 2).

[KV09] Jonathan Katz and Vinod Vaikuntanathan. "Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices". In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Berlin, Heidelberg: Springer, 2009, pp. 636–652. ISBN: 978-3-642-10366-7. DOI: 10.1007/978-3-642-10366-7_37 (cit. on p. 124).

[Lam79] Leslie Lamport. *Constructing Digital Signatures from a One Way Function*. Tech. rep. CSL-98. This paper was published by IEEE in the Proceedings of HICSS-43 in January, 2010. Oct. 1979. URL: https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/ (cit. on p. 18).

[Lin17] Yehuda Lindell. "How to Simulate It - A Tutorial on the Simulation Proof Technique". In: *Tutorials on the Foundations of Cryptography*. Ed. by Yehuda Lindell. Springer International Publishing, 2017, pp. 277–346. DOI: 10.1007/978-3-319-57048-8_6 (cit. on p. 110).

[Mar16a] Moxie Marlinspike. *The Double Ratchet Algorithm*. Ed. by Trevor Perrin. Version 1. Nov. 2016. URL: https://signal.org/docs/specifications/doubleratchet/ (visited on 08/07/2021) (cit. on pp. 42, 102, 105).

[Mar16b] Moxie Marlinspike. *The X3DH Key Agreement Protocol*. Ed. by Trevor Perrin. Version 1. Nov. 2016. URL: https://signal.org/docs/specifications/x3dh/ (visited on 08/07/2021) (cit. on p. 102).

[Mar17] Moxie Marlinspike. *Safety number updates*. June 2017. URL: https://signal.org/blog/verified-safety-number-updates/ (visited on 08/07/2021) (cit. on p. 103).

[May01] Dominic Mayers. "Unconditional Security in Quantum Cryptography". In: *Journal of the ACM* 48.3 (May 2001), pp. 351–406. ISSN: 0004-5411. DOI: 10.1145/382780.382781 (cit. on p. 12).

[McE78] Robert J. McEliece. "A Public-Key Cryptosystem Based On Algebraic Coding Theory". In: *Deep Space Network Progress Report* 44 (Jan. 1978), pp. 114–116. URL: https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF (visited on 08/07/2021) (cit. on p. 129).

[Mer89] Ralph C. Merkle. "A Certified Digital Signature". In: *Advances in Cryptology - CRYPTO '89 Proceedings*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 218–238. DOI: 10.1007/0-387-34805-0_21 (cit. on p. 18).

[MKF+16] David A. McGrew, Panos Kampanakis, Scott R. Fluhrer, Stefan-Lukas Gazdag, Denis Butin, and Johannes A. Buchmann. "State Management for Hash-Based Signatures". In: *Security Standardisation Research - Third International Conference, SSR 2016*. Ed. by Lidong Chen, David A. McGrew, and Chris J. Mitchell. Vol. 10074. Lecture Notes in Computer Science. Springer, 2016, pp. 244–260. DOI: 10.1007/978-3-319-49100-4_11 (cit. on pp. 23, 51).

[MR19] Daniel Masny and Peter Rindal. "Endemic Oblivious Transfer". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM, 2019, pp. 309–326. DOI: 10.1145/3319535.3354210 (cit. on pp. 125, 127–129).

[MR21] Daniel Masny and Peter Rindal. *Endemic Oblivious Transfer*. July 2021. Cryptology ePrint Archive: 2019/706 (cit. on pp. 7, 104, 125, 127–130, 143).

[MS07] Rajat Mittal and Mario Szegedy. "Product Rules in Semidefinite Programming". In: *Fundamentals of Computation Theory, 16th International Symposium*. Ed. by Erzsébet Csuhaj-Varjú and Zoltán Ésik. Vol. 4639. Lecture Notes in Computer Science. Springer, 2007, pp. 435–445. DOI: 10.1007/978-3-540-74240-1_38 (cit. on p. 75).

[MV19] Michele Mosca and Sebastian R. Verschoor. *Factoring semi-primes with (quantum) SAT-solvers*. Feb. 2019. arXiv: 1902.01448 [cs.CR] (cit. on p. 174).

[MVV20] Michele Mosca, João Marcos Vensi Basso, and Sebastian R Verschoor. "On speeding up factoring with quantum SAT solvers". In: *Scientific Reports* 10.15022 (Sept. 2020), pp. 1–8. DOI: 10.1038/s41598-020-71654-y (cit. on p. 175).

[MVW13] Abel Molina, Thomas Vidick, and John Watrous. "Optimal Counterfeiting Attacks and Generalizations for Wiesner's Quantum Money". In: *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by Kazuo Iwama, Yasuhito Kawano, and Mio Murao. Berlin, Heidelberg: Springer, 2013, pp. 45–64. ISBN: 978-3-642-35656-8. DOI: 10.1007/978-3-642-35656-8_4 (cit. on p. 87).

[NC10] Michael A. Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2010. ISBN: 978-1-107-00217-3 (cit. on p. 69).

[NIST17] *Post-Quantum Cryptography*. National Institute of Standards and Technology, 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography (visited on 08/07/2021) (cit. on pp. 3, 19, 104, 145).

[NRS18] Moni Naor, Lior Rotem, and Gil Segev. "The Security of Lazy Users in Out-of-Band Authentication". In: *Theory of Cryptography - 16th International Conference*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11240. Lecture Notes in Computer Science. Springer, 2018, pp. 575–599. DOI: 10.1007/978-3-030-03810-6_21 (cit. on p. 107).

[NSA20] NSA. *Quantum Key Distribution (QKD) and Quantum Cryptography (QC)*. 2020. URL: https://www.nsa.gov/what-we-do/cybersecurity/quantum-key-distribution-qkd-and-quantum-cryptography-qc/ (visited on 08/07/2021) (cit. on p. 9).

[Oka92] Tatsuaki Okamoto. "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes". In: *Advances in Cryptology - CRYPTO '92*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 31–53. DOI: 10.1007/3-540-48071-4_3 (cit. on p. 108).

[OTR] Ian Goldberg and the OTR Development Team. *Off-the-Record Messaging*. 2016. URL: https://otr.cypherpunks.ca/ (visited on 08/07/2021) (cit. on p. 42).

[OTRv4] The OTR team. *Off-the-Record Messaging Protocol version 4*. 2020. URL: https://bugs.otr.im/otrv4/otrv4/ (visited on 08/07/2021) (cit. on p. 105).

[PAB+07] Stefano Pironio, Antonio Acín, Nicolas Brunner, Nicolas Gisin, Serge Massar, and Valerio Scarani. "Device-Independent Security of Quantum Cryptography against Collective Attacks". In: *Physical Review Letters* 98.230501 (June 2007). DOI: 10.1103/PhysRevLett.98.230501 (cit. on p. 12).

[PNM+05] M. Peev, M. Nölle, O. Maurhardt, T. Lorünser, M. Suda, A. Poppe, R. Ursin, A. Fedrizzi, and A. Zeilinger. "A Novel Protocol-Authentication Algorithm Ruling Out a Man-in-the-Middle Attack in Quantum Cryptography". In: *International Journal of Quantum Information* 3.1 (2005), pp. 225–231. DOI: 10.1142/S0219749905000797 (cit. on p. 22).

[Por14] Christopher Portmann. "Key Recycling in Authentication". In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 4383–4396. DOI: 10.1109/TIT.2014.2317312 (cit. on p. 17).

[PPS07] Kenneth G Paterson, Fred Piper, and Rüdiger Schack. "Quantum cryptography: a practical information security perspective". In: *NATO Science for Peace and Security Series – D: Information and Communication Security* 11 (2007), p. 175. arXiv: quant-ph/0406147 (cit. on p. 12).

[PR14] Christopher Portmann and Renato Renner. *Cryptographic security of quantum key distribution*. Sept. 2014. arXiv: 1409.3525 [quant-ph] (cit. on p. 52).

[PRE20] Alasdair B. Price, John G. Rarity, and Chris Erven. "A quantum key distribution protocol for rapid denial of service detection". In: *EPJ Quantum Technology* 7.8 (May 2020). DOI: 10.1140/epjqt/s40507-020-00084-6 (cit. on pp. 20, 22).

[Rab81] Michael O. Rabin. *How to exchange secrets with oblivious transfer*. Tech. rep. TR-81. Aiken Computation Lab, Harvard University, 1981. Cryptology ePrint Archive: 2005/187 (cit. on p. 119).

[RČ10] Kasper Bonne Rasmussen and Srdjan Čapkun. "Realization of RF Distance Bounding". In: *19th USENIX Security Symposium*. USENIX Association, Aug. 2010. URL: https://www.usenix.org/conference/usenixsecurity10/realization-rf-distance-bounding (visited on 08/07/2021) (cit. on pp. 57, 61).

[Res00] E. Rescorla. *HTTP Over TLS*. RFC 2818. RFC Editor, May 2000. DOI: 10.17487/RFC2818 (cit. on p. 103).

[Res18] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor, Aug. 2018. DOI: 10.17487/RFC8446 (cit. on p. 103).

[RGTS07] Jason Reid, Juan M. González Nieto, Tee Tang, and Bouchra Senadji. "Detecting Relay Attacks with Timing-based Protocols". In: *Proceedings of the 2nd ACM Symposium on Information*. ASIACCS '07. ACM, 2007, pp. 204–213. ISBN: 1-59593-574-6. DOI: 10.1145/1229285.1229314 (cit. on p. 63).

[Rin21] Peter Rindal. *libOTe: an efficient, portable, and easy to use Oblivious Transfer Library*. 2021. URL: https://github.com/osu-crypto/libOTe (cit. on p. 129).

[RR17] Peter Rindal and Mike Rosulek. "Malicious-Secure Private Set Intersection via Dual Execution". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 1229–1242. DOI: 10.1145/3133956.3134044 (cit. on pp. 7, 104, 109, 124, 132, 133).

[RR20] Joseph M. Renes and Renato Renner. *Are quantum cryptographic security claims vacuous?* Oct. 2020. arXiv: 2010.11961 [quant-ph] (cit. on p. 12).

[RTŠ+12] Aanjhan Ranganathan, Nils Ole Tippenhauer, Boris Škorić, Dave Singelée, and Srdjan Čapkun. "Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System". In: *Computer Security – ESORICS 2012*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Berlin, Heidelberg: Springer, 2012, pp. 415–432. ISBN: 978-3-642-33167-1. DOI: 10.1007/978-3-642-33167-1_24 (cit. on p. 61).

[Rud02] Terry Rudolph. *The Laws of Physics and Cryptographic Security*. Feb. 2002. arXiv: quant-ph/0202143 (cit. on p. 12).

[RW03] Renato Renner and Stefan Wolf. "Unconditional Authenticity and Privacy from an Arbitrarily Weak Secret". In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 78–95. DOI: 10.1007/978-3-540-45146-4_5 (cit. on p. 17).

[RW04] Renato Renner and Stefan Wolf. "The Exact Price for Unconditionally Secure Asymmetric Cryptography". In: *Advances in Cryptology - EUROCRYPT*. Ed. by Christian Cachin and Jan L. Camenisch. Berlin, Heidelberg: Springer, May 2004, pp. 109–125. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_7 (cit. on p. 17).

[Sch89] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology - CRYPTO '89*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 239–252. DOI: 10.1007/0-387-34805-0_22 (cit. on p. 108).

[Sho04] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Nov. 2004. Cryptology ePrint Archive: 2004/332 (cit. on p. 137).

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on pp. 2, 104, 109).

[SHWR16] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermanner. "When Signal hits the fan: On the usability and security of state-of-the-art secure mobile messaging". In: *European Workshop on Usable Security. IEEE*. 2016, pp. 1–7. DOI: 10.14722/eurousec.2016.23012 (cit. on p. 106).

[SM16] Douglas Stebila and Michele Mosca. "Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project". In: *Selected Areas in Cryptography - SAC 2016*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. Lecture Notes in Computer Science. Springer, 2016, pp. 14–37. DOI: 10.1007/978-3-319-69453-5_2 (cit. on pp. 131, 143).

[SML09] Douglas Stebila, Michele Mosca, and Norbert Lütkenhaus. "The Case for Quantum Key Distribution". In: *Quantum Communication and Quantum Networking*. Ed. by Alexander V. Sergienko, Saverio Pascazio, and Paolo Villoresi. Vol. 36. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, Oct. 2009, pp. 283–296. DOI: 10.1007/978-3-642-11731-2_35 (cit. on pp. 14, 45, 55).

[Son14] Fang Song. "A Note on Quantum Security for Post-Quantum Cryptography". In: *Post-Quantum Cryptography*. Ed. by Michele Mosca. Cham: Springer International Publishing, 2014, pp. 246–265. ISBN: 978-3-319-11659-4. DOI: 10.1007/978-3-319-11659-4_15 (cit. on pp. 139, 140).

[SPD+10] Louis Salvail, Momtchil Peev, Eleni Diamanti, Romain Alléaume, Norbert Lütkenhaus, and Thomas Länger. "Security of trusted repeater quantum key distribution networks". In: *Journal of Computer Security* 18.1 (Jan. 2010), pp. 61–87. ISSN: 0926-227X. DOI: 10.3233/JCS-2010-0373 (cit. on p. 12).

[SSM18] Maliheh Shirvanian, Nitesh Saxena, and Dibya Mukhopadhyay. "Short voice imitation man-in-the-middle attacks on Crypto Phones: Defeating humans and machines". In: *Journal of Computer Security* 26.3 (2018), pp. 311–333. DOI: 10.3233/JCS-17970 (cit. on p. 107).

[SYG08] Ryan Stedman, Kayo Yoshida, and Ian Goldberg. "A user study of off-the-record messaging". In: *SOUPS*. ACM International Conference Proceeding Series. ACM, 2008, pp. 95–104. DOI: 10.1145/1408664.1408678 (cit. on p. 107).

[TP07] Yu-Ju Tu and Selwyn Piramuthu. "RFID distance bounding protocols". In: *First International EURASIP Workshop on RFID Technology*. Vienna, Austria, 2007, pp. 67–68. URL: https://www.eurasip.org/Proceedings/Ext/RFID2007/pdf/s5p2.pdf (visited on 08/07/2021) (cit. on p. 94).

[TSJL21] Oleg Taraskin, Vladimir Soukharev, David Jao, and Jason T. LeGrow. "Towards Isogeny-Based Password-Authenticated Key Establishment". In: *Journal of Mathematical Cryptology* 15.1 (2021), pp. 18–30. DOI: doi:10.1515/jmc-2020-0071 (cit. on p. 124).

[UDB+15] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. "SoK: Secure Messaging". In: *2015 IEEE Symposium on Security and Privacy, SP 2015*. IEEE Computer Society, 2015, pp. 232–249. DOI: 10.1109/SP.2015.22 (cit. on pp. 102, 106).

[UG18] Nik Unger and Ian Goldberg. "Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging". In: *Proceedings on Privacy Enhancing Technologies* 2018.1 (2018), pp. 21–66. DOI: 10.1515/popets-2018-0003 (cit. on p. 105).

[Ung21] Unger, Nik. "End-to-End Encrypted Group Messaging with Insider Security". PhD thesis. 2021. URL: http://hdl.handle.net/10012/17196 (cit. on p. 141).

[Unr13] Dominique Unruh. "Everlasting Multi-party Computation". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Berlin, Heidelberg: Springer, 2013, pp. 380–397. ISBN: 978-3-642-40084-1. DOI: 10.1007/978-3-642-40084-1_22 (cit. on pp. 9, 13).

[Vau13] Serge Vaudenay. "On Modeling Terrorist Frauds". In: *Provable Security*. Ed. by Willy Susilo and Reza Reyhanitabar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–20. ISBN: 978-3-642-41227-1. DOI: 10.1007/978-3-642-41227-1_1 (cit. on p. 66).

[Ver15] Sebastian R. Verschoor. *SCimp Proverif models*. Dec. 2015. URL: https://github.com/sebastianv89/scimp-proverif (visited on 08/07/2021) (cit. on p. 174).

[Ver18] Sebastian R. Verschoor. *qbdb*. Sept. 2018. URL: https://github.com/sebastianv89/qbdb (visited on 08/07/2021) (cit. on p. 84).

[Ver19] Sebastian R. Verschoor. *factoring-sat (GitHub repository)*. Jan. 2019. URL: https://github.com/sebastianv89/factoring-sat (visited on 08/07/2021) (cit. on p. 174).

[Ver21] Sebastian R. Verschoor. *libkop*. July 2021. URL: https://github.com/sebastianv89/libkop (cit. on pp. 104, 142, 148).

[VL16] Sebastian R. Verschoor and Tanja Lange. *(In-)Secure messaging with the Silent Circle instant messaging protocol*. July 2016. Cryptology ePrint Archive: 2016/703 (cit. on p. 174).

[VV19] João Marcos Vensi Basso and Sebastian R. Verschoor. *NFS-SAT*. Oct. 2019. URL: https://github.com/sebastianv89/NFS-SAT (visited on 08/07/2021) (cit. on p. 175).

[VWO+17] Elham Vaziripour, Justin Wu, Mark O'Neill, Jordan Whitehead, Scott Heidbrink, Kent E. Seamons, and Daniel Zappala. "Is that you, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications". In: *Thirteenth Symposium on Usable Privacy and Security, SOUPS 2017*. USENIX Association, 2017, pp. 29–47. URL: https://www.usenix.org/conference/soups2017/technical-sessions/presentation/vaziripour (visited on 08/07/2021) (cit. on p. 107).

[VWO+18] Elham Vaziripour, Justin Wu, Mark O'Neill, Daniel Metro, Josh Cockrell, Timothy Moffett, Jordan Whitehead, Nick Bonner, Kent E. Seamons, and Daniel Zappala. "Action Needed! Helping Users Find and Complete the Authentication Ceremony in Signal". In: *Fourteenth Symposium on Usable Privacy and Security, SOUPS 2018*. Ed. by Mary Ellen Zurko and Heather Richter Lipford. USENIX Association, 2018, pp. 47–62. URL: https://www.usenix.org/conference/soups2018/presentation/vaziripour (visited on 08/07/2021) (cit. on pp. 106, 107).

[Wat18] John Watrous. *The theory of quantum information*. Cambridge University Press, Apr. 2018. ISBN: 9781107180567 (cit. on pp. 69, 71, 74).

[WC81] Mark N. Wegman and Larry Carter. "New Hash Functions and Their Use in Authentication and Set Equality". In: *Journal of Computer and System Sciences* 22.3 (June 1981), pp. 265–279. DOI: 10.1016/0022-0000(81)90033-7 (cit. on pp. 10, 16, 17).

[Wer98] R. F. Werner. "Optimal cloning of pure states". In: *Physical Review A* 58.3 (Sept. 1998), pp. 1827–1832. DOI: 10.1103/PhysRevA.58.1827 (cit. on p. 87).

[WGH+19] Justin Wu, Cyrus Gattrell, Devon Howard, Jake Tyler, Elham Vaziripour, Daniel Zappala, and Kent E. Seamons. ""Something isn't secure, but I'm not sure how that translates into a problem": Promoting autonomy by designing for understanding in Signal". In: *Fifteenth Symposium on Usable Privacy and Security, SOUPS 2019*. Ed. by Heather Richter Lipford. USENIX Association, 2019. URL: https://www.usenix.org/conference/soups2019/presentation/wu (visited on 08/07/2021) (cit. on p. 106).

[Wie83] Stephen Wiesner. "Conjugate Coding". In: *SIGACT News* 15.1 (Jan. 1983), pp. 78–88. ISSN: 0163-5700. DOI: 10.1145/1008908.1008920 (cit. on p. 87).

[WT99] Alma Whitten and J. D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0". In: *8th USENIX Security Symposium*. Aug. 1999. URL: https://www.usenix.org/conference/8th-usenix-security-symposium/why-johnny-cant-encrypt-usability-evaluation-pgp-50 (visited on 08/07/2021) (cit. on p. 105).

[WZ82] W.K Wootters and W. H. Zurek. "A single quantum cannot be cloned". In: *Nature* 299 (Oct. 1982). DOI: 10.1038/299802a0 (cit. on pp. 3, 57, 87).

[XKCP] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. *eXtended Keccak Code Package*. URL: https://github.com/XKCP/XKCP (visited on 08/07/2021) (cit. on p. 143).

[Zim95] Philip R. Zimmermann. *The official PGP user's guide*. MIT press, 1995. ISBN: 0-262-74017-6 (cit. on pp. 103, 105).

Appendix

Other work by the author

- SRV and Tanja Lange [VL16]. "(In-)Secure messaging with the Silent Circle instant messaging protocol."
  - *Proverif models of SCimp [Ver15].*

  This work analyzes the Silent Circle instance messaging protocol (SCimp), a secure messaging protocol that historically sits between Off-the-Record Messaging (OTR) and Signal. The work comprises a formal analysis in Proverif and reports vulnerabilities that were found, and includes an analysis of the source code. While the bulk of this work was done at Eindhoven University of Technology as part of my Master's thesis, this version of the work was prepared at the start of my PhD program at the University of Waterloo.

- Jennifer Katherine Fernick and SRV [FV17]. "Privacy-Preserving Patent Search: Addictively Homomorphic Encryption Techniques for Private Text Mining over Public Datasets."
  - *Experiment source code [FV16].*

  This work considers the applicability of somewhat homomorphic encryption to secure text mining.

- Michele Mosca and SRV [MV19]. "Factoring semi-primes with (quantum) SAT-solvers."
  - *Circuits, experiment source code and results [Ver19].*

Experimental results in quantum annealing report factorizations of increasingly large semi-primes. The underlying algorithms effectively reduce factoring to an NP-hard problem. In this paper we study and question the practical effectiveness of this approach. We find no evidence that this is a viable path towards factoring large numbers.

- Michele Mosca, João Marcos Vensi Basso and SRV [MVV20]. "On speeding up factoring with quantum SAT solvers".

  – *Circuits, experiment source code and results [VV19].*

  In an extension of the above work, we attempt to factor numbers by applying (quantum) SAT solving to finding smooth numbers as a step in the number field sieve. The result has the same asymptotic runtime as the classical number field sieve and can theoretically benefit from quadratic quantum speedup to enable faster factoring. Benchmarks indicate there is massive overhead to this approach that needs to be overcome.