Data Origin Authentication And The Classical Data Plane of A Quantum Network Link

J. S. Abrahams

February 2022

Technische Universiteit Delft

TU Delft

Data Origin Authentication And The Classical Data Plane of A Quantum Network Link

by

J. S. Abrahams

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday Feb 3, 2022 at 13:00.

Student number:	4443268	
Thesis committee:	Prof. dr. ir. S. Wehner,	TU Delft, supervisor
	Dr. ir. D. Elkouss,	TU Delft
	Dr. ir. Z. Erkin,	TU Delft, board of examiners
Daily Supervisor:	Ir. C. Delle Donne,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

ABSTRACT

Quantum networks allow multiple devices to exchange information encoded within quantum systems. Such quantum networks use classical control messages to coordinate entanglement between nodes. Third parties which can forge such control messages may interfere with the workings of quantum links, however: They may either perform fraudulent requests for entanglement, destroying local quantum memory of nodes as a result, or interfere with the inner workings of protocols within the quantum stack targeting the availability of the link(s). We, therefore, conclude that all link and physical layer control messages must be transmitted via authenticated channels. Typically one uses a Message Authentication Code (MAC) to do so, which takes as input a message and outputs a tag which is transmitted alongside the message. Additionally, it takes as input a unique number each time (nonce) to prevent replay-type attacks. In this work, we first investigate the use of information-theoretic MACs combined with quantum key distribution (QKD) to authenticate control messages. We find that the fastest QKD solutions provide key material at a rate that is sufficient to not become a bottleneck in current quantum links. Second, we survey multiple computationally secure MAC solutions and benchmarks to get an indication of their performance when authenticating short messages. While not information-theoretically secure, their latency is generally speaking greater than or equal to that of information-theoretic solutions. Finally, we augment the existing simulation of a single quantum link by Dahlberg et al. by inserting delays based on the performance of these MACs. The performance of the link is evaluated using the mean throughput: The rate at which successfully entangled pairs are delivered. We find that the introduction of transmission time overhead, without any authentication, causes a noticeable decrease in throughput of the link. When considering an authenticated channel that uses SipHash (a popular MAC) we find that throughput decreases even further, though less significantly. Therefore, the overall decrease in throughput appears to not be detrimental to the working of the quantum link, which remains functional even when the classical channel is authenticated.

ACKNOWLEDGEMENTS

Before you lies my thesis which was written to fulfil the requirements of the Computer Science master at the Delft University of Technology. It represents the culmination of 6 years of work.

Of course, I would like to extend some gratitude to those who helped make this project possible. Thank you Prof. Stephanie Wehner for introducing me to the world of quantum networks, for letting me do this project, and for providing valuable feedback for the duration of the project. I would like to thanks Dr. David Elkouss and Wojciech Kozlowski who further helped me refine my vague project idea into a more concrete proposal, as well as Matthew Skrzypczyk for helping me navigate the codebase where I proceeded to perform my own experiments. I also want to thank my daily supervisor Carlo Delle Donne for the weekly meetings, giving regular feedback, and keeping me on track for the duration of the project. I would also like to thank the rest of the committee, Dr. David Elkouss and Dr. Zeki Erkin, for evaluating my work.

Lastly, I would like to extend gratitude towards friends and family members. During the multiple lockdowns we've all had to endure, the evening walks, coffee breaks, and video calls were always a pleasure, and reminded me that there is more to life than computer science.

I hope you enjoy reading this thesis.

J. S. Abrahams
Delft, December 2021

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

Notation	Meaning
$\|0\rangle, \|1\rangle$	Bra-ket notation for the 0 and 1 state in the computational basis
$\|\psi\rangle$	State ψ
ρ	Density matrix
$\otimes$	Tensor product
$\{\cdots\}$	Set
tr	Trace
$H^\dagger$	Hermitian conjugate of operator H
M	Message space
K	Key space
$\{f_k\}_K$	Pseudo-random function family with K elements
f_k	Function from function family $\{f_k\}_K$, indexed using k
$f : \{0,1\}^n \rightarrow \{0,1\}^q$	Mapping from a n-bit input to a q-bit output
$g : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^q$	Mapping from a n- and m- bit input to a q-bit output
$f \circ h(x)$	The output of $h(x)$ as input for f
log	Logarithm base 2, unless specified otherwise
$\oplus$	Binary addition (addition in base 2)
$m\|\|n$	n appended to m
Pr[·]	Probability
$=$	Equality
$\leftarrow$	Assign
$\rightarrow$	Maps to
X	Random variable
x	Realization of random variable X
$\bar{x}$	Mean
σ	Standard deviation
Σ	Sum
$\lfloor\cdot\rfloor$	Floor
$\lceil\cdot\rceil$	Ceiling

ACRONYMS

AE Authenticated Encryption. 85
AES Advanced Encryption Standard. 34, 48, 50, 53, 56–59, 61, 85
ARIMA Autoregressive Integrated Moving Average. 60
ARX Addition-Rotation-XOR. 55, 61

CAN Controller Area Network. 16, 60
CCS Calderbank-Shor-Steane. 18
CK Create and Keep. 21, 23, 68
CNOT Controlled NOT. 10

DQP Distributed Queue Protocol. 5, 19–24, 33, 35, 63, 68, 73

HEG Heralded Entanglement Generation. 17, 23
HEP Heralded Entanglement Purification. 17, 44

IoT Internet of Things. 60
IP Internet Protocol. 2, 5, 15, 16, 67–69

LPWAN Low-Power Wide Area Network. 16

MAC Media Access Control. 16
MAC Message Authentication Code. iii, 3–6, 26, 28, 34–37, 41, 42, 45, 49–51, 53–62, 67, 68, 70–75, 77, 79–81, 84–86, 95, 96
MD Measure Directly. 23, 48, 49, 68, 78–81, 84
MHP Midpoint Heralding Protocol. 5, 6, 19–24, 33, 35, 39–41, 47, 58, 59, 62, 63, 67–69, 71–74, 77, 79, 80, 83, 90, 92

NL Network Layer. 68, 78–81, 84
NV Nitrogen-Vacancy. 64, 93

OSI Open Systems Interconnection. 2, 15, 16
OTP One Time Pad. 4, 37–39, 45, 46, 55, 56

PITM Person-In-The-Middle. 45, 46
PNS Photon Number Splitting. 30
POVM Positive Operator-Valued Measure. 65
PRF Pseudorandom Function. 4, 38, 50, 51, 54–58
PRNG Psuedo Random Number Generator. 55

QBER Quantum Bit Error Rate. 31
QEC Quantum Error Correction. 17
QEGP Quantum Entanglement Generation Protocol. 3, 5, 6, 15, 19–24, 33, 34, 39, 63, 68, 69, 72–74, 77, 80, 83, 90–92
QKD Quantum Key Distribution. 3–5, 19, 29, 33, 34, 39–42, 44, 46–50, 52, 83, 84, 86

RSA Rivest-Shamir-Adleman. 49

TCP Transmission Control Protocol. 2, 15, 16, 68
TLS Transport Layer Security. 85, 86

UDP User Datagram Protocol. 5, 16, 67–69

CHAPTER 1

INTRODUCTION

THE 20th century has seen the introduction of a new field within science: Quantum Information Theory. It investigates how one manipulates information encoded within the state of a quantum system [79]. This would allow one to perform certain calculations at speeds not possible on classical computation systems. We identify three key areas of research which laid the foundation for quantum information theory: Quantum mechanics, the study of the very small, computer science, the study of computation, and (classical) information theory, the study of the quantization of information.

The first, quantum theory, was conceived in 1900 when Max Planck introduced the notion of energy quanta [62]. While perhaps not deliberate, it did form the basis of a new field: Quantum Mechanics[I]. Quantum Mechanics aims to model and explain the buildings blocks of the universe [79]. It postulates that everything is quantized, meaning that the state of a system may be in one of a discrete set of values. At a more fundamental level, perhaps, it states that no process can be predicted with perfect certainty: We live in a stochastic universe, meaning that we may only calculate the probability of something happening.

The theory of computation emerged in 1937. This was the year in which Alan Turing published his paper "On Computable Numbers" to solve the decidability problem [103]. In it, he introduced the notion of a Universal Turing Machine: A mathematical model of a computer. The decidability problem, or *entscheidungsproblem* in German, was originally a challenge posed by David Hilbert and Wilhelm Ackermann in 1928 in "Grundzüge der theoretischen Logik" ("Basics of Theoretical Logic") [52]. It asked whether it was possible to use an algorithmic process to determine whether a mathematical theorem is provable, or decidable. Such a process would save mathematicians time as one would know in advance whether a theorem could be proven. The model Turing created showed that such a process cannot exist. Sometime later, during the second world war, Turing helped crack the Enigma code at Bletchley Park [104]. This forever intertwined the foundations of computer science with cryptography and cryptanalysis. Cryptography is the art of getting information from point A to point B without an outside party being able to pertain said information. Crypt-analysis is the art of pertaining said information (by breaking the system meant to transport it securely). Both cryptography and cryptanalysis have their roots in antiquity.

Sometime later, in 1948, Claude Shannon published his seminal paper "A Mathematical Theory of Communication" [93]. This was the start of the field of information theory. In it, he introduced the concept of entropy of an information source, which is in a sense a measure of uncertainty of an information source. The following year, in 1949, his work on "Communication

[I] Planck received a Nobel prize in 1918 for his efforts [82].

Theory of Secrecy Systems" introduced the notion of information-theoretic security, also referred to as unconditional security [94]. It is called unconditional because the security of such a system is provable. This is in contrast to computational secure systems, or schemes, whose security is conditioned on the presumed, but unproven, difficulty of certain problems. One such problem is the discrete logarithm problem [99].

The goal of quantum information processing is to manipulate, and extract, the information stored within quantum systems [79]. In turn, this forms the basis of the theory of quantum computing. Similar to the transition from classical to quantum physics, the transition from classical to quantum computation marked a major paradigm shift within the field of computation. In addition to presenting a paradigm shift, this also poses some considerable engineering challenges. How does one build a (scalable) quantum computer, and, how does one build a quantum network to connect such machines? One also has to contend with the relatively short lifetimes of *qubit* (short for *quantum bit*) in such systems[II]. This is in contrast to classical systems, where storing bits indefinitely is (more) technically feasible.

A known short-term application for quantum computation is to break current asymmetric cryptographic schemes we have in place to protect our data. To combat this, new classical cryptographic schemes, also called post-quantum cryptographic schemes, are being developed and tested. These are believed to be unbreakable within a "reasonable" time frame even for powerful quantum computers. Readers should note that it has been called into question how long it will take to construct the quantum hardware underpinning these machines such that they can break some cryptographic schemes currently in use [6]. Other cryptographic systems are likely to remain unbroken, such as hash- or symmetric-based cryptography [16].

Regardless of the technical feasibility of breaking current cryptographic schemes, quantum computers have many other, perhaps more interesting, applications. They could be used to simulate quantum processes, which was put forward in the 1981 lecture on "Simulating Physics with Computers" by Feynman [47]. This is exceedingly difficult using even the most powerful classical computers due to the complexity of such simulations scaling exponentially. Quantum computers therefore will likely become an invaluable asset within the researcher's tool-set.

Quantum information systems may be distributed across arbitrarily large distances, thus also opening the door to quantum networks. Such networks have many potential applications. This includes the extension of telescope baselines [48], synchronization of clocks [67], and secure distributed quantum computation [31]. Wehner et al. provide a general roadmap of quantum networks and the challenges they face [108]. Several proposals have been made as to how quantum networks should be structured [57, 75, 85, 92]. Kozlowski and Wehner further outline the challenges such networks face [64]. This includes tight timing constraints, the need for a quantum network stack, routing, and the security of control messages [64]. The security of control messages is not to be confused with quantum cryptography, which deals with the security information encoded within the quantum system itself.

1.1. Problem Statement

Several efforts exist to build quantum networks, but the technology is not yet widespread. Classical networks have the Open Systems Interconnection (OSI) model as a guide[III] [3, Ch. 2]. This model outlines a stack consisting of layers, which ensures the functioning of the network. Each layer then has its own set of responsibilities, ranging from ensuring the transmission of data along direct links to (reliable) end-to-end transmission of data (possibly along multiple links). Engineers design protocols which inhibit these different layers, each fulfilling a different

[II] Qubit lifetimes in the range of seconds have been measured in systems [2]. However, these lifetimes were not realized simultaneously in a complete system as far as Dahlberg et al. were able to identify when performing their simulations [38]. Therefore, in certain situations, one might have to contend with lifetimes in the range of merely a few milliseconds.

[III] The internet makes use of the TCP/IP model [3, Ch. 3], which is somewhat similar. While also a stack, its structure differs slightly from the OSI model. We shall later briefly highlight some of the differences.

purpose.

However, there is no such established (widespread) model as of yet for quantum networks. Physics experiments and quantum applications are in practice mostly ad-hoc. Quantum stacks, unlike the classical stack(s), are often still in the early stages of development and have not yet been proven.

QuTech is in the process of developing such a network stack, developing both an outline of the layers and separation of responsibilities, as well as protocols to populate these layers. Dahlberg et al. have designed a quantum link layer protocol called *Quantum Entanglement Generation Protocol (QEGP)*, which allows quantum nodes along a single link (possible with quantum repeater stations between them) in a network to perform entanglement to transport quantum information [38]. This presents a more system-level approach to building a quantum network: The particular application does not matter, only that such a network can reliably deliver end-to-end entanglement. One important design constraint is speed. This follows from the difficulty of storing quantum information reliably for extended periods of time. Therefore, classical control messages must be transported with as little delay as possible.

Satoh et al. mention forged classical messages as a general concern for quantum networks [91]. *Data origin authentication* of classical messages is necessary to prevent fraudulent requests for entanglement [38, 64]. Data origin authentication allows parties within the network to distinguish genuine and fraudulent (control) messages. This is not to be confused with mutual authentication, where two nodes verify one another identity before commencing further communication.

Such fraudulent requests for entanglement are an attack vector that may, for instance, destroy the local quantum memory of legitimate users of the network. Quantum memory, at least currently, is more fragile than classical memory. This problem is exacerbated by the fact that it is not possible to simply create multiple local copies of quantum memory due to the no cloning theorem, which states is impossible to create a perfect copy of an arbitrary quantum state [79]. Protecting said memory (more generally, information encoded within the system) should, therefore, be of high priority. More generally speaking such fraudulent control messages may interfere with the workings of the link itself. Forgery of error messages could, for instance, impact the availability of links, and by extension the network, if not handled properly. However, the mechanisms which prevent destruction by malicious parties should not introduce so much overhead that they are detrimental to the performance of the link (or network) as a whole. Limiting their impact on performance should therefore also be of high priority.

Data origin authentication is performed using a shared secret between two parties. One uses a Message Authentication Code (MAC) to produce a tag for each message (or set of messages) both at the sending and receiving end, to verify that (1) the message contents were not altered and (2) that they were produced by a party which owns the shared secret [99, Ch. 14]. Public key cryptography does offer an alternative; signatures, which are also useful for verifying the validity of individual messages. Using such asymmetric cryptography, messages are signed using a private key and verified using a public key, which together with the private key forms a public-private key pair. In such a system, only the holder of the private key *should* be able to produce a signature that will be recognized as valid using the public key counterpart. However, such asymmetric systems are typically much slower, and therefore most likely not suited for sessions within quantum networks.

Kozlowski and Wehner propose using Quantum Key Distribution (QKD) ([13]) to produce session keys combined with information-theoretic data origin authentication of classical control messages. Quantum Key Distribution (QKD) allows one to generate secret key material indefinitely (using a small amount of initial shared key material). This is necessary, as information-theoretic security secure data origin authentication requires a constant stream of fresh key material. The feasibility of QKD combined with information-theoretic security data origin authentication of control message of system-level protocols has not yet been investigated as far as we are able to identify. This is not to be confused with information-theoretic security data

origin authentication of QKD basis-announcements, which has been investigated previously.

1.2. Research Objectives and Approach

We investigate the data origin authentication of classical messages exchanged at the physical and link layers of the quantum stack. In the remainder of this document, when we refer to "authentication", we also mean "data origin authentication" unless specified otherwise. Data origin authentication introduce some overhead to the classical counterpart of these networks. As such, we investigate the expected performance of these solutions, and lastly investigate their impact on a single quantum link.

RQ1 - Why, And How, Should Control Messages Be Authenticated?

As a precursor, it is first relevant to establish more fully why data origin authentication is required We also briefly explore why encryption is not strictly necessary, but could still prove useful in some cases.

RQ1.1 Why is data origin authentication necessary?
RQ1.2 When would encryption be necessary?

Kozlowski and Wehner note that either an information-theoretically secure data origin authentication solution using keys generated by QKD may be used, but do not investigate this further [64]. We investigate its feasibility, both when running QKD within the stack (at the application layer), or on an external "QKD box". We specifically investigate real-time data origin authentication, meaning that each message individually must be verified before it is used. Else, it would be possible for adversaries to (consistently) interfere with the protocol before being detected.

RQ1.3 Can a scheme which uses information-theoretic real-time data origin authentication be self-sustaining?

RQ2 - How fast are the fastest data origin authentication solutions?

Considering the tight timing constraints placed upon quantum networks, the delays incurred by secured classical channels should be kept to a minimum. Therefore, the MAC we use to authenticate the channel must be fast, in particular when producing tags for many small packets. The reason for investigating computationally secure solutions is two-fold:

1. Information theoretically secure data origin authentication solutions have stringent key requirements. In some situations, it might not be possible to supply key material at the rate required for these solutions to function themselves at an acceptable rate.
2. Computationally secure solutions are used more often in the industry. We, therefore, find that is easier to find benchmarks of these solutions. Furthermore, these solutions appear much slower than information-theoretic solutions by a considerable margin [50][IV]. Therefore, if we show that the link functions with these delays present, then we have also inadvertently shown that its performance would remain within acceptable bounds when using an information-theoretic security solution.

Data origin authentication is typically done using a tag generated by a Message Authentication Code (MAC). Transmitting large tags and nonces (to prevent replay attacks) is expensive,

[IV] Whether their performance is similar to information-theoretic security solutions in practical settings warrants its own investigation. For instance, computationally secure solutions make use of a PRF as opposed to a OTP, which conceptually is more complex. Handschuh and Preneel in fact note that computationally secure data origin authentication solutions are typically slower [50]. However, the OTP brings with it more stringent key requirements, requiring either a lot of memory or frequent loading and unloading of key material. This could also put a strain on the performance.

however. This is especially true when one uses classical channels with a low bit rate. We therefore also investigate ways optimizations that aim to shorten both tags and nonces without reducing (significantly) the security of the overall solution. In general, we take the security of MACs as-is: The consensus appears to be that if a MAC is secure in a classical setting, it remains secure in a setting where adversaries have quantum computing resources (but may still only submit classical queries to the system) [16].

RQ3 - How does the latency of data origin authentication on the classical data plane affect link throughput?

An increase in latency on the classical channel affects the overall performance of the link. This is indicated by for instance the results of Mehic et al. [74]: An increase in latency has a major effect on the performance of the Quantum Key Distribution (QKD) protocol under evaluation in their setup.

Using the findings from **RQ2**, we augment the simulation of QEGP and MHP by Dahlberg et al. [38] with extra classical delays. These delays are derived from our analysis of the performance of data origin authentication solutions. This should demonstrate how authentication of the accompanying classical channel would influence a quantum link in a realistic setting. The original simulation did not account for transmission delays. These delays are derived from the bit rate of the channel and the number of bits transmitted along the classical link. We also account for delays from the ethernet, IP, and UDP protocols.

Lastly, requests were scheduled within [38] using the Distributed Queue Protocol (DQP). However, this appears to be an ad-hoc scheduling solution. We, therefore, do not evaluate the latency of requests, which is the time between the submission of requests and the delivery of pairs.

RQ3.1 How does the introduction of transmission delays affect link throughput?
RQ3.2 How does the introduction of data origin authentication delays affect link throughput?

1.3. Contributions

The contributions of this work are as follows:

1. A demonstration of the required amount, and rate, of key material required by information-theoretic security data origin authentication at the physical level. Specifically, we investigate the key requirements at the physical layer and some hypothetical coordination mechanisms which may for instance reside at either the link or network layer.
2. A similar demonstration of the required amount of key material of a coordination mechanism, which could for instance reside at the link or network layer.
3. An analysis of how QEGP and MHP from [38] react to the overhead of an authenticated classical channel. This overhead is modelled using benchmarks we uncover analysing the performance of authentication solutions. Parameters are extracted from benchmarks, as well as our choice of MAC. Whether our results generalize to other system-level protocols remains to be seen (as there aren't others as far as we can identify).
4. By extension, an non-exhaustive overview of MACs which we believe are suited for the authentication of classical messages within quantum networks. Any MAC which shows similar performance characteristics to the ones which do not significantly affect the quantum link is likely suited.

1.4. Outline

Chapter 2 covers preliminary knowledge required to understand what quantum computation and networks are. This includes the basics of quantum information theory and a general illustration of what network stacks are. Readers already knowledgeable on these subjects may opt to skip this chapter.

Chapter 3 covers some preliminary knowledge pertaining to the QuTech quantum network stack itself[V]. Chapter 4 covers classical security notions relevant to our discussion about mutual authentication. This is also where we explain what information-theoretic security means, and its relevance to **RQ1**. Chapter 5 first performs a case study of the quantum network stack and how it may be attacked, by considering an un-authenticated version QEGP and Midpoint Heralding Protocol (MHP). This is followed by an exploration of the amount of key material required by an information-theoretic security data origin authentication solution, thus addressing **RQ1**. We discuss why it may not be necessary to use information-theoretic security solutions for control messages, and why computationally secure are likely sufficient.

Chapter 6 gives an overview of MACs we identified as being suited for our application: optimized for sending many short messages. This addresses **RQ2** and provides input to our simulation.

Chapter 7 delves into the prior simulation setup, previously used in [38]. Chapter 8 builds upon this by outlining our extensions to this simulation. This is followed by a presentation of the results in chapter 9. We also discuss the implications of the results, addressing **RQ3**. Finally, chapter 10 contains a general discussion of the implications of our findings and what we believe to be future directions of research.

The general outline of the document is depicted in figure 1.1.

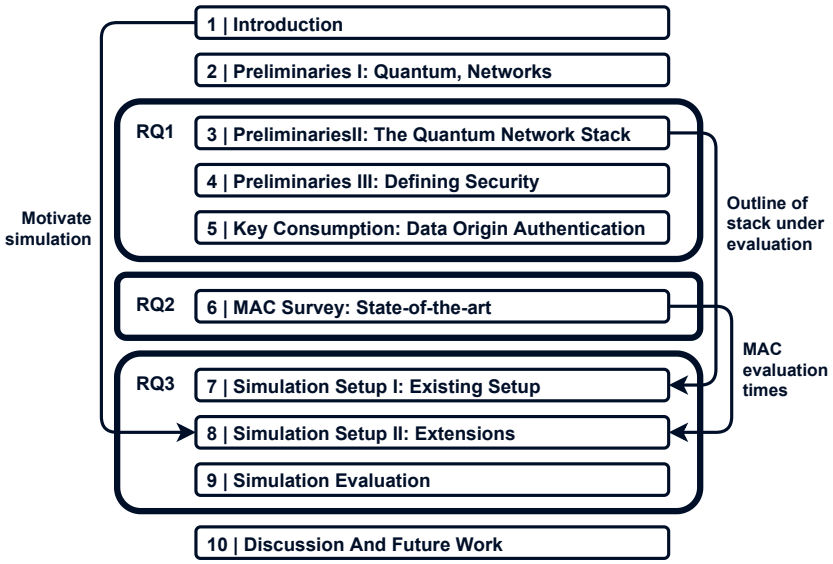


Figure 1.1: Structure of this document. Implicit underlying structure shown by the arrows.

[V] A more complete overview is given in [38], [64], and [65]. The link-layer protocol presented by [38] is the focus of this work.

CHAPTER 2

PRELIMINARIES I: QUANTUM, NETWORKS

> The purpose of computation is insight, not numbers.
>
> RICHARD HAMMING

The fields of quantum computation and (classical and quantum) networks are quite broad, with intersections with the fields of physics, mathematics, electronics, and computer science, to name a few. This chapter briefly delves into the basics of quantum information theory. We discuss what a qubit is, including the concept of *superposition*, and why it is useful. We cover entanglement, and how this forms the basis of quantum networks. We also cover the concepts of mixed states and fidelity, which may also be used to measure the quality of entanglement. We do assume the reader has a basic understanding of linear algebra.

Next, we discuss some networking principles in section 2.2. We first cover the classical stack, which has inspired the quantum stack which we discuss in chapter 3. Lastly, we cover quantum repeaters. These, much like in classical networks, are vital for information to be able to travel large distances.

2.1. Quantum Information Theory, A Prelude

Quantum computers can perform certain calculations much faster than traditional classical computers. They derive this computation power from quantum bits: Qubit in short. For a complete overview of quantum computation and quantum information theory, we recommend "quantum computation and quantum information" by Nielsen and Chuang [79].

2.1.1. A Single Qubit

Much like how bits form the basis of classical systems, qubits form the basis of quantum computing systems. A qubit is defined as follows:

Definition 2.1.1 (qubit)**.** A two-state quantum-mechanical system.

In the classical world, a bit (a binary digit), may have either value or state 0 or 1. These "classical" bits, as we shall refer to them for the remainder of this document, may be implemented

using any two-state device or system. Qubits, unlike their classical counterpart, can be in both in the 0 and 1 state at the same time. More specifically, a qubit may be in any linear combination of the two: It may be in *superposition* (see definition below).

Definition 2.1.2 (linear combination)**.** Given vectors $v_1, ..., v_n$ in the space V, and $a_1, ...a_n$ scalars, a linear combination of the given vectors with the given scalars is

$$a_1 v_1 + a_2 v_n + ... + a_n v_n \tag{2.1}$$

For qubits, we have the following: A qubit state may be described using two complex numbers α_0 and α_1, which are probability amplitudes, and vectors $|0\rangle$ and $|1\rangle$, which represent the 0 and 1 states in the computational basis:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle, \ \alpha_0, \alpha_1 \in \mathbb{C} \tag{2.2}$$

The notation of $|0\rangle$ and $|1\rangle$ for the 0 and 1 states is referred to the Dirac[I], or bra-ket, notation. They are shorthand for the following:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \text{and} \qquad \langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

Superposition, Probabilities, And Unit Vectors This combination of different states ($|0\rangle$ and $|1\rangle$ here) that a qubit may be in is also referred to as *superposition*. These $|0\rangle$ and $|1\rangle$ themselves are also valid quantum states. Therefore, this combination of states is defined as such:

Definition 2.1.3 (Superposition)**.** A combination of valid quantum states which itself is also a valid quantum state.

This is a uniquely quantum property: The system may be in both states at the same time, and will only collapse to a single state upon measurement. The terms $|\alpha_0|^2$ and $|\alpha_1|^2$ above are the probability of the qubit collapsing to either $|0\rangle$ or $|1\rangle$ upon measurement. As probabilities must add up to 1, we have the following:

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \tag{2.3}$$

2.1.2. Product States: Multiple Qubits

A single qubit, while displaying interesting properties, does not provide much in the way of extra computational power, nor does it allow one to build a quantum (inter)net. Adding multiple qubits to the mix does change things considerably, however. A system built using qubits can essentially encode information using the above coefficients α_0 and α_1. In a sense, a single qubit allows one to store two units of information, as opposed to a single unit of information as would be the case in a classical system. The power of quantum algorithms stems from how the amount of these coefficients scales with the number of qubits. For instance, in a two-qubit system, we have four such coefficients. We represent this below using the computational basis ($|0\rangle$ and $|1\rangle$):

$$|\psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle, \quad |\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$$
$$\alpha_{xy} \in \mathbb{C}, \quad x, y \in \{0, 1\}$$

Where $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ shorthand for the tensor products of basis states $|0\rangle$ and $|1\rangle$:

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0\begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \text{, meaning that} \qquad |\psi\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$$

[I] Paul Dirac (8 August 1902 - 20 October 1984) is regarded as one of the great physicists of the 20th century, and made fundamental contributions quantum mechanics.

Doing the same for a three-qubit system, one would see that it requires eight coefficients. In general, a n qubit system requires 2^n such coefficients. Table 2.1 showcases what the internal state of a multi-qubit quantum looks like. While a classical system scales linearly in terms of computation power with the number of bits, a quantum system scales exponentially. In a classical system, we manipulate individual bits using classical gates. In a quantum system, we manipulate the probability amplitudes (which scales exponentially with the number of qubits) using quantum gates.

Table 2.1: A representation of the internal state of a system with n qubits, with 2^n rows (coefficients) for n qubits. The percentages on the left represent the probability of the quantum system collapsing to said state upon measurement. Quantum algorithms, through quantum gates, manipulate these percentages. Note how adding a single qubit doubles the amount of rows, and, in theory, the computational power of the system.

Amplitude	**Example Prob.**	b_1	b_2	...	b_{n-1}	b_n
α_1	$\lvert\alpha_1\rvert^2 = 4.1\%$	0	0	...	0	0
α_2	$\lvert\alpha_2\rvert^2 = 2.2\%$	0	0	...	0	1
α_3	$\lvert\alpha_3\rvert^2 = 7.5\%$	0	0	...	1	0
	$\vdots$					
α_{2^n}	$\lvert\alpha_{2^n}\rvert^2 = 1.1\%$	1	1	...	1	1

To put into context how quickly such machines grow in terms of computation power, consider a quantum system of 275 qubits. To describe such a system, one requires 2^{275} coefficients. This number is greater than the number of atoms we believe to be in observable universe[II]. It would be impossible to simulate such a system on a classical computer.

There is one caveat, however. There is no way for us to observe these probability amplitudes directly. They simply represent the probability the system will collapse to a certain state upon measurement. The challenge for quantum algorithms (and their designers) is to manipulate these probability amplitudes such that the output of the algorithm is a state encoding the solution to a given problem. Lastly, unlike their classical counterparts, this process is stochastic: We cannot for certain say what the system will collapse to, even if the hypothetical "correct" answer has a high probability associated with it. This may be overcome by running an algorithm multiple times to build confidence in the pertained answer.

2.1.3. Quantum Gates

We mentioned above quantum gates: Quantum gates are the building blocks of quantum algorithms. We only cover quantum gates so far as to understand how they facilitate entanglement generation. Quantum algorithms and their building blocks are not the focus of this work.

Gates And Their Properties First, a quantum gate is much like a gate in a classical circuit, but with one important constraint: The gate must be reversible: A reversible gate has a one-to-one mapping between its output and input, given an output, there exists only one input that could give that result. In other words, given the output of a gate, it should be possible to determine the input. This is not necessarily the case for all classical gates:

Example 2.1.1. If a classical AND gate gives as output true, it is not possible to using that alone to determine which inputs were true. Either the first, or the second, or both, may have been true.

Two important quantum gates are the X and H gates. The X gate is the quantum equivalent of the NOT gate. It is also a Pauli operation. The latter is the "Hadamard" gate and allows one

[II]There are estimated to be approximately 10^{80} $(\approx 2^{266})$ atoms in the observable universe [51].

to bring a qubit into *superposition*. They are both, as is the case with any valid quantum gate, denotable using a matrix.

$$\mathrm{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Both of these are unitary matrices. Any such unitary matrix represents a valid quantum operation (which may be implemented using a gate), and is defined as follows:

Definition 2.1.4 (Unitary Matrix)**.** A matrix U is unitary if it is square and its conjugate transpose U^* is its inverse, meaning

$$U^*U = UU^* = I \tag{2.4}$$

Where I is the identity matrix. In quantum mechanics, the conjugate transpose is typically referred to as the Hermitian adjoint denoted using $U^\dagger$.

The conjugate transpose of a matrix is obtained by taking the transpose of the matrix and taking the complex conjugate of each entry.

Definition 2.1.5 (Complex Conjugate)**.** The complex conjugate of complex number $z = a + ib$ is $z = a - ib$.

Any $2^n \times 2^n, \forall n > 0$ unitary matrix represents a valid quantum operation (which may be implemented using a gate) that operates on n qubits.

To get a feel of how these gates function (in a theoretical sense, we do not discuss physical implementations here), we consider the X gate first. The following is an example of how the X (NOT) gate operates:

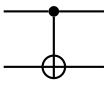
$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \to \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix} \quad \text{so,} \quad \mathrm{X}|0\rangle \to |1\rangle, \mathrm{X}|1\rangle \to |0\rangle$$

The following is an example of how the H gate operates. We demonstrate how it brings the basis states ($|0\rangle$ and $|1\rangle$ in the computational basis) into superposition.

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \to \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha_0 + \alpha_1 \\ \alpha_0 - \alpha_1 \end{pmatrix} \quad \text{so,} \quad \mathrm{H}|0\rangle \to \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle, \mathrm{H}|1\rangle \to \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \tag{2.5}$$

Two Qubit Gates A gate may also act upon multiple qubits at once. We demonstrate the Controlled NOT (CNOT) quantum gate. This gate takes two qubits as input, with one being the control. If the first qubit is $|0\rangle$, then the second qubit is left unchanged. If the first qubit is $|1\rangle$, the second qubit is flipped. One would write the gate out as follows:

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} =$$



We now have the following:

$$\mathrm{CNOT}|00\rangle \to |00\rangle \qquad \mathrm{CNOT}|01\rangle \to |01\rangle$$
$$\mathrm{CNOT}|10\rangle \to |11\rangle \qquad \mathrm{CNOT}|11\rangle \to |10\rangle$$

2.1.4. Entanglement

The states of two or more qubits may be interlinked such that manipulation (which includes measurement) of one affects the other. Two qubits are entangled if the state of the system as a whole cannot be written as the product of two separate states as above.

Consider the following joint state, which is a generalization of the aforementioned example of a two-qubit states ($|00\rangle, |01\rangle, |10\rangle, |11\rangle$):

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \tag{2.6}$$

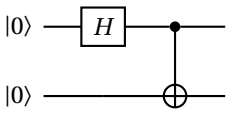
This is a separable state, as $|\psi\rangle$ may be separated into its constituents components $|\psi_1\rangle$ and $|\psi_2\rangle$. If $|\psi\rangle$ is not separable, then it is entangled, meaning that is is not possible to write $|\psi\rangle$ as

$$|\psi\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle \tag{2.7}$$

for all possible states $|\psi_1\rangle$ and $|\psi_2\rangle$.

Four well-known such states which are maximally entangled are the Bell states.

- $|\psi_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This state is obtained by passing $|00\rangle$ to a Hadamard and then a CNOT gate in the following fashion:



- $|\psi_{01}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$, produced by giving as input $|01\rangle$ to the above circuit.
- $|\psi_{10}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$, produced by giving as input $|10\rangle$ to the above circuit.
- $|\psi_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$, produced by giving as input $|11\rangle$ to the above circuit.

Figure 2.1: Depiction of entanglement. The arrows are graphical representations of the quantum state of these particles, which may be the *spin* of the particles for instance. The first pair is not entangled, which means that their spin is independent. The second is entangled such that the particles have opposite spin.

In a network setting, one would then send on half of this entangled pair to a receiving party. This pair may then in turn be used to transmit information encoded within a quantum system.

Quantum Teleportation In a classical system, if we want to transfer information from point A to B, it is trivial to simply copy the information and then transmit it using any form of encoding. However, it is not so trivial to do this when information is encoded in a quantum system due to the no cloning theorem [109].

Definition 2.1.6 (No Cloning Theorem)**.** The no-cloning theorem states that it is impossible to create a perfect copy of an arbitrary quantum state.

Instead, one may make use of a Bell state to transfer an arbitrary qubit, which encodes quantum information, from one point to another. Say that Alice and Bob share a Bell state, and Alice wants to send the state $|\psi\rangle$ to Bob. They perform teleportation as follows:

1. Alice sends the $|\psi\rangle$ state through a CNOT gate, using her own state as a control qubit and her half of the Bell pair as the other half. Her own qubit is then sent through a H gate.
2. Alice then measures both the qubits on her end. This projects her qubit and half of the Bell pair onto the Bell basis. After this, she sends the measurement results to Bob.
3. Bob then performs corrections on his half of the Bell pair. Bob passes his half of the Bell state through the X gate if Alice's measurement of her half of the Bell pair was 1, and additionally through the Z gate if Alice's measurement of the first qubit was 1.

The circuit used to perform these operations is depicted in figure 2.2. The top two qubits are stored at Alice's end and the bottom qubit on Bob's end.

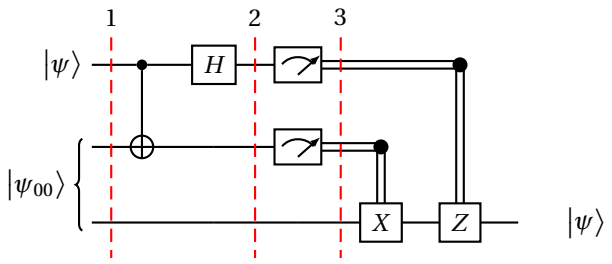


Figure 2.2: Circuit for quantum teleportation. Classical communication is depicted using the double lines.

Without classical information, Alice cannot transmit the information encoded within the qubit to Bob. The above scheme allows transmission of arbitrary states, including ones where Alice does not know what information is encoded within the qubit. One should note that this system cannot transmit information faster than the speed of light even though qubits may be entangled over arbitrarily large distances. Without classical communication and the accompanying corrections, no actual *information* is transmitted.

Entanglement Swapping It may be that Alice and Bob are too far removed to send qubits to one another directly. This is not too dissimilar to classical networks where signals may only propagate so far before they are too weak to be read. At that point, they must be repeated. In quantum networks, one way to achieve this is by performing an entanglement swap. To do this, Alice and Bob both make use of a locally entangled state $|\psi_{00}\rangle$. For clarity, we shall refer to the two qubits of this pair as A_1, A_2 on Alice's side, and B_1, B_2 on Bob's side. The process is then as follows:

1. Alice and Bob both prepare this state, giving the joined state

$$|\psi\rangle = |\psi_{00}\rangle_{A_1,A_2} |\psi_{00}\rangle_{B_1,B_2}$$

   Note that Alice and Bob's local states are still separable from one another in this global state.
2. Both send their 1st qubit (A_1, B_1) to the midpoint station.
3. The midpoint station performs a Bell projection onto one of the $\{|\psi_{00}\rangle, |\psi_{01}\rangle, |\psi_{10}\rangle, |\psi_{11}\rangle\}_{A_2,B_2}$ states.
4. The midpoint station send the measurement outcome to he end-nodes such that they may perform local corrections.
5. The final pair A_2, B_2 now forms a $|\psi_{00}\rangle_{A_2,B_2}$ pair.

The above process is depicted in figure 2.3. This entanglement swapping is an important component of quantum networks. These allow, just as is the case with its classical counterpart, for quantum computers to be connected over arbitrarily large distances.

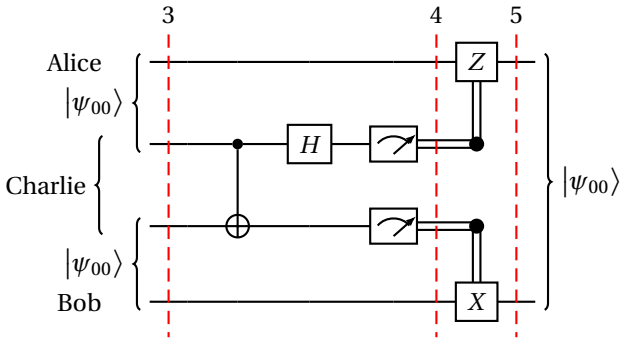


Figure 2.3: Circuit for of entanglement swapping. Charlie is the midpoint station located between Alice and Bob, with which Alice and Bob share a Bell pair. The goal is to ensure that Alice and Bob themselves share an entangled pair between them.

2.1.5. Density Matrices, Mixed States, And Quantum Fidelity

Up until this point we have discussed only pure states. However, sometimes we might not know exactly what state a system is in due to, for instance, environmental noise. In that case, "mixed states", which is a classical mixture of states, offers an elegant solution for representing such uncertainty about the state.

Mixed states are not to be confused with the notion of superposition. Any state may be in superposition, i.e. $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. We might not know for certain what the system will collapse to upon measurement, but we know what the probability that it will collapse to any state is. With a mixed state, one only has partial or no knowledge of these probabilities: $|\psi\rangle = ??|0\rangle + ??|1\rangle$. It is not true that the coefficients have somehow "vanished". They certainly still "exist"[III] in some sense. We simply cannot with certainty say what they are anymore.

To work with such mixtures of pure states, we require a mathematical formalism called density matrices. Say we have an arbitrary state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, then the density matrix would be written as follows:

$$|\psi_1\rangle\langle\psi_1| = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix}\begin{pmatrix}\alpha_0 & \alpha_1\end{pmatrix} = \begin{pmatrix}\alpha_0^2 & \alpha_0\alpha_1\\ \alpha_0\alpha_1 & \alpha_1^2\end{pmatrix}, \quad \text{so for instance,} \quad |0\rangle\langle 0| = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}$$

For an arbitrary collection of pure states, the density matrix is as follows:

$$\rho = \Sigma_x p_x |\psi_x\rangle\langle\psi_x| \tag{2.8}$$

We thus have three fundamental manners in which to describe a quantum system.

Pure A quantum system that is a linear combination of multiple states.

Entangled A quantum system of two or more qubits that cannot be described as two or more separate quantum systems.

Mixed A (classical) mixture of quantum states. Note that the mixture itself is not in superposition. The mixture only quantifies our ignorance about the *true* quantum state.

Fidelity A mixture may be used to quantify the presence of environmental noise. It would then be useful to quantify the presence of this noise using some metric. One such metric is called fidelity. This metric allows one to determine the distance between two states, for instance between a noisy state and a perfect system with no noise present. A fidelity of 0 means the states are nothing alike, and 1 means that they are indistinguishable.

[III] In a loose sense, as they are nevertheless still only a mathematical description of the underlying system.

The formula for fidelity between two states ρ_1 and ρ_2 is as follows, with tr being the trace of the resulting density matrix.

$$F(\rho_1, \rho_2) = \text{tr}\left[\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}\right]^2 \tag{2.9}$$

Definition 2.1.7 (Trace)**.** The sum of the diagonals of a square matrix. The shorthand for trace is tr.

We demonstrate an example relating to our Bell states from earlier.

Example 2.1.2. Consider that the circuit used to produce the $|\psi_{00}\rangle$ state requires two $|0\rangle$ qubits as input. However, in a practical implementation there will be noise present. If there is sufficient noise, the information encoded within the qubit (0) would be completely scrambled, meaning that were we to measure it (in the computational basis) it would collapse randomly onto 0 or 1 (meaning that i's state is $\rho_1 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$). We now use the density matrix formalism to quantify this nose. $\rho_0 = |0\rangle\langle 0|$ represents our ideal state. $\rho_1 = p_1|0\rangle\langle 0| + (1-p_1)|\psi_1\rangle\langle\psi_1|$ is then the mixture between the ideal and noisy state: The larger p_1 is, the more alike they are. The fidelity would then be calculated as $F(\rho_0, \rho_1)$.

It is not possible to determine the fidelity of an unknown state. The density formalism is merely a mathematical description of the underlying state. It is possible, however, to estimate this fidelity by performing continually reconstructing the same state and measuring its fidelity as part of a process called *quantum tomography* [39].

Lastly, this metric also lets us estimate the quality of entanglement. A fidelity of 1 would then indicate that we have, for instance, a perfect Bell pair.

2.1.6. Measuring Quantum Systems

One may ask how one extracts information encoded within quantum systems. The act of measuring may be described using a measurement operator. That is, before measurement, one must choose a basis to measure in. This may be the computational basis, meaning that measuring $|0\rangle$ will give 0 with probability 1, and likewise $|1\rangle$ gives 1 with probability 1. However, if we were to measure in the $\{|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}, |-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}\}$ basis, then measuring $|0\rangle$ would give outcome 0 with 0.5 probability and 1 with 0.5 probability.

Recall that that $H|0\rangle$ and $H|1\rangle$ (eq. 2.5), despite the sign, have the same probability of the outcome being 1 upon measurement: $\left|+1/\sqrt{2}\right|^2 = \left|-1/\sqrt{2}\right|^2 = 1/2$. However, this assumes that measurements are only performed in the computational basis, which we have implicitly done up until this point. If one were to measure in a different basis, then this would not be true. Here we cover the mathematical basis which describes such measurements, adapted from [79]. We do not cover the actual physical apparatus which performs such measurements.

A measurement operator is defined formally in [79] as follows:

Definition 2.1.8 (Quantum Measurements)**.** A quantum measurement is defined as a set $\{M_m\}$ of measurement operators. Index m refers to the outcome of the measurement, i.e. the information one extracts from the system. The probability that we get outcome m is

$$\Pr[m] = |\psi\rangle M_m^\dagger M_m \langle\psi| \tag{2.10}$$

The state after measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{|\psi\rangle M_m^\dagger M_m \langle\psi|}} \tag{2.11}$$

Lastly, the set must be complete, meaning that

$$\sum_m M_m^\dagger M_m = I \tag{2.12}$$

This ensures that the probabilities $\Pr[m]$ for all values m in $\{M_m\}$ sums up to one.

Example 2.1.3. Consider $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. If we measure the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability of obtaining outcome 0 is

$$\Pr[0] = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = |\alpha|^2 \tag{2.13}$$

Example 2.1.4. Tke $M_0 = |+\rangle\langle +|$ and $M_1 = |-\rangle\langle -|$. If we measure the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability of obtaining outcome 0 is

$$\Pr[0] = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = 1/2\alpha^2 + \alpha\beta + 1/2\beta^2 \tag{2.14}$$

Going back to equation 2.5, the probability of getting outcome 0 for $H|0\rangle = |+\rangle$ is:

$$\Pr[0] = \langle +| M_0^\dagger M_0 |+\rangle = \langle +| M_0 |+\rangle = 1 \tag{2.15}$$

The same could be obtained by filling in $\alpha = 1/\sqrt{2}$ and $\beta = 1/\sqrt{2}$ in equation 2.14. Doing a similar calculation for outcome 1 (using M_1) yields probability 0.

2.2. Networking Principles

Network stacks aim to facilitate information transfer from an arbitrary point in a network to another point (in the network). The classical network stack has been standardized as the Open Systems Interconnection (OSI) model. The internet, however, makes use of the TCP/IP stack, which is somewhat similar to the OSI stack but has a few differences. While OSI is a general model, the TCP/IP model was based on protocols that already existed [3]. We cover these here to provide context for the link layer protocol (QEGP) in [38] as well as the general stack they outline. For instance, the TCP/IP model does not distinguish between the link layer and physical layer, whereas the quantum stack in [38] and the OSI model do. However, as TCP/IP is also very popular, we cover it as well as to clearly state where it differs from the OSI model.

The quantum networks stack, however, is not as established. In chapter 3 we do delve into an effort by QuTech to develop such a quantum stack. To understand what a network stack is in the first place, we cover the classical stack.

2.2.1. The Classical Network Stack, The OSI And TCP/IP Models

The classical stack, as specified in the Open Systems Interconnection (OSI) is depicted in part in figure 2.4. The session and presentation layers are omitted in our depiction, much like in the TCP/IP stack. The general structure OSI structure has also served as inspiration for quantum network stacks [38, 64, 65, 84].

The following is a description of all the layers in the OSI model. We make note of where it diverges with the TCP/IP model. Some of the definitions taken below are from [3].

7. **Application** The services which the end-user makes use of are defined here. Communication protocols, such as HTTP(S), FTP(S), SSH, POP3 or IMAP3 (email protocols), and interface methods of the stack.

6. **Presentation** Determines how data is presented to the application layer. I.e., data may be compressed, encrypted, or translated from little-endian to big-endian encoding depending on the needs of the application layer. Within the TCP/IP model, this layer is incorporated within the application layer.

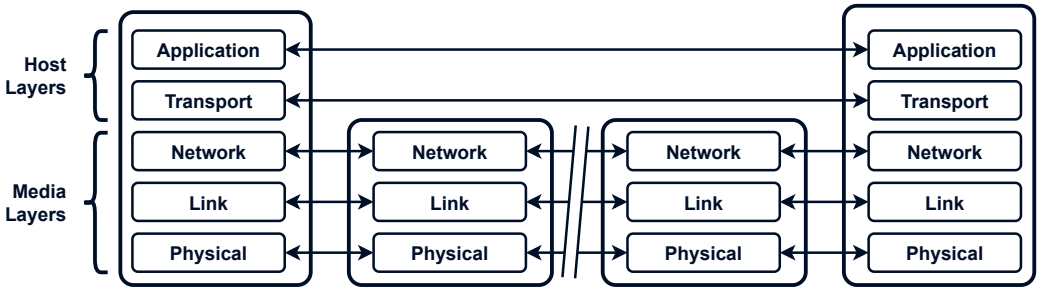


Figure 2.4: Classical network stack, as specified in the OSI model. The session and presentation layers are omitted. Each layer treats the underlying layer as a service. From the application layer perspective, there is a seamless (but potentially lossy) connection between it and the node on the other end of the line.

5. **Session** The main responsibility of the session layer is to give the presentation layer the ability to organize the communication for multiple sessions taking place concurrently. Within the TCP/IP model, this layer is incorporated within the application layer.

4. **Transport** End-to-end transport of data.

   Both TCP and UDP reside at this layer. TCP does ensure in-order delivery of packets to higher layers, and is more reliable, at the cost of performance. UDP makes no guarantees and is typically only used in scenario's where real-time transmission is more important than reliability (e.g. streaming protocols, where losing a few packets typically does not make that big of a difference).

   In the case of TCP, the protocol internally already performs session management, making the session layer from the original OSI model unnecessary within the TCP/IP stack.

3. **Network** Responsible for routing packets through intermediary nodes, from one end of the network to another using a best-effort approach.

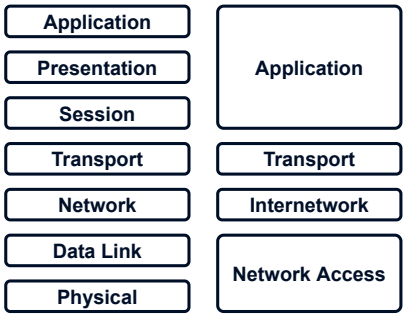
   Within the TCP/IP stack, this is referred to as the "internetwork layer". The IP protocol resides at this layer. Within the TCP/IP stack, there are several routing protocols which support IP, sch as the Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), and Reverse Address Resolution Protocol (RARP) [3].

2. **Data Link** Responsible for routing packets between connected nodes across the physical layer. Deals with point-to-point connections only. The lower half of this layer, called the Medium Access Control Sublayer, also deals with the mapping of bits (i.e. using a 4b/5b or 8b/10b encoding scheme), and re-transmission of lost frames across a single link. It also deals with collisions on the link, by for instance employing an exponential back-off scheme.

   Protocols include 802.3 Ethernet, 802.11 Wi-Fi, and 802.15.4 ZigBee. ZigBee is typically used by Low-Power Wide Area Network (LPWAN). This is in a sense also the lowest layer within the TCP/IP stack. The (data) link layer and physical layers are combined into the "network access layer". The TCP/IP model does not heavily specify this layer, it only requires that it can deliver IP packets [3]. In general, its responsibility is to map IP addresses to physical addresses (Media Access Control (MAC) addresses).

1. **Physical** Deals with the transmission of raw bits across different physical mediums. Such mediums may be fibre optic cables, copper cables, or satellite transmissions, to name a few. Other responsibilities, depending on the medium, include error detection and multiplexing and demultiplexing (if relevant).

   These include the Bluetooth, USB, and Controller Area Network (CAN) standards.



(a) OSI model. (b) TCP/IP model.

Figure 2.5: Comparison of the layers of the OSI model and the TCP/IP model.

The physical, link, and network layers form part of the *communication sub-net layers*. Within these layers, signals must be transported using both repeaters (across a single link) and switches (across multiple links and networks). This is because any signal degrades over time and distance. To higher layers, called the *host layers*, the connection appears to be a seamless end-to-end connection.

2.2.2. Quantum Repeaters

Just as with any classical network, a quantum network also require repeaters. This is because any signal will degrade over distance. The rate at which the signal (most likely consisting of photons) degrades is determined by the *attenuation coefficient* for said medium: A large value means the signal degrades quickly, whereas a small value means the signal is barely affected. However, due to the no cloning theorem, quantum repeaters cannot simply copy and re-transmit data. They must be more clever in how they handle the re-transmission of data.

Quantum repeater schemes are typically classified into three generations depending on how they deal with transmission loss and operational errors [77, 78]. Typically, photons are used to transmit quantum data across (large) distances.

For detecting (and handling) of loss of photons in transit, repeaters employ either *heralded entanglement generation* (also referred to as *entanglement distillation*) or *quantum error correction*. Repeaters themselves may, however, also introduce noise. For correcting this noise they perform either *entanglement purification* or again the aforementioned quantum error correction. How these tools are used in each generation of repeaters is depicted in table 2.2.

Table 2.2: Generations of quantum repeaters. Each subsequent generation promises higher throughput over long distances. Note that the third generation of quantum repeater makes exclusive use of error correcting codes. This means that these are used to both detect (and correct) both omission of transmitted qubits (typically photons) and corruption of data.

Errors	**Approach**	**Classical Comm.**	**1G**	**2G**	**3G**
Loss Correction	Heralded Entanglement Generation (HEG)	two-way	✓	✓	
	Quantum Error Correction (QEC)	one-way			✓
Operation Error Corr.	Heralded Entanglement Purification (HEP)	two-way	✓		
	Quantum Error Correction (QEC)	one-way		✓	✓

Each subsequent generation promises higher throughput over long distances. However, quantum error correction codes have more stringent hardware requirements, meaning that the second and third generation of repeaters are also more difficult to implement.

Heralded Entanglement Generation Heralded entanglement generation revolves around entanglement swapping. Additionally, it makes use of a heralding[IV] signal. This signal encodes the correction information required for entanglement swapping, as well as signifying to the two nodes which the repeater (or midpoint) is connected to that (1) the midpoint station did, indeed, receive two signals (typically photon) and (2) the swap itself was successful. Therefore, the classical success signal is conditioned on the detectors clicking. See figure 2.3 for a graphical overview of entanglement swapping, including the classical corrections performed. If no heralding (success) signal is received then it is assumed that the midpoint station has failed the swap (for any number of reasons). If it is received, then we know (with very high certainty) that the swap was successful.

Quantum Error Correction Quantum error correction, like its classical counterpart, makes use of redundancy to protect quantum information. One such code is the Calderbank-Shor-Steane (CCS) construction [101], which is used by second and third generation repeaters [3]. Quantum error correction has the advantage that it requires only one-way communication. Quantum data is sent using an array of qubits, and the accompanying classical information may be used to perform any corrections on said quantum data at the receiving end. The recipient may either be a repeater or a end-node.

When used for loss errors, in the case of third generation repeaters, repeaters must be placed close together [3]. This is because quantum error correction codes can only correct a relatively small amount of loss errors.

Heralded Entanglement Purification This is also referred to as *entanglement distillation*. It consumes multiple Bell pairs, combining them to produce a smaller set of Bell pairs with higher fidelity [41]. It uses only local quantum operations and classical communication to achieve this. This also referred to as entanglement distillation.

This also makes use of heralding signals, which means that classical communication is two-way. This, again, is to both announce success or failure, and for exchange of measurement results for classical corrections.

[IV] Heralds were messengers sent by noblemen to convey messages.

CHAPTER 3

PRELIMINARIES II: THE QUANTUM NETWORK STACK

Quantum communication networks are expected to have multiple uses. At long distances, quantum communication networks allow quantum computers to connect to a quantum network. Two applications of such networks include Quantum Key Distribution (QKD) and secure access of remote quantum computers [31]. These two fall under the umbrella of "quantum cryptography", which is an active area of research [30]. Further applications include accurate clock-synchronization [67], and extending the baseline of telescopes [48], to name a few. One application for short-distance communication is wiring lots of small quantum computers, which could for instance be used to build scalable quantum computers [107]. Other applications will likely be discovered as such networks mature in the coming years [108].

As it stands, all proposals of quantum network designs identified thus far, some being more concrete proposals than others, use some form of classical communication to coordinate quantum communication and entanglement [57, 75, 84, 92]. This also includes the quantum network stack developed at QuTech [38, 65]. They are used to either facilitate repeater stations (section 2.2.2) or to coordinate entanglement in a general sense between end-nodes.

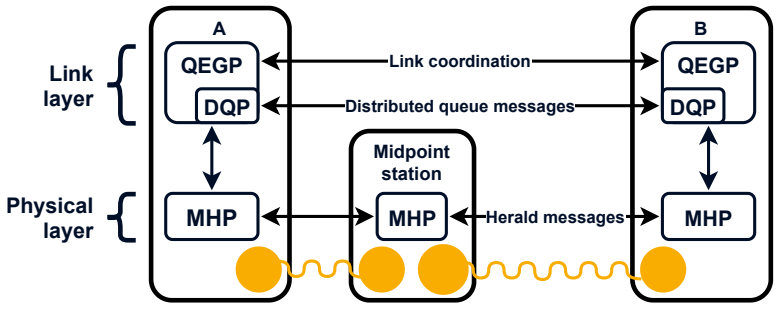


Figure 3.1: High-level view of QuTech stack up to link layer level, with a single midpoint station (quantum repeater). This includes the Quantum Entanglement Generation Protocol (QEGP) and Distributed Queue Protocol (DQP) at the link layer, and Midpoint Heralding Protocol (MHP) at the physical layer from [38].

As mentioned, QuTech is in the process of developing a quantum stack. This includes both

the general framework of the stack itself as well as the protocols which inhabit the several layers of the stack. At the link layer, Quantum Entanglement Generation Protocol (QEGP) aims to provide robust entanglement generation [38]. It makes use of a Distributed Queue Protocol (DQP) to align the requests of the nodes. QEGP also makes use of the Midpoint Heralding Protocol (MHP) at the physical layer to produce entangled pairs using heralded entanglement generation. Figure 3.1 demonstrates how these protocols and the midpoint station interact.

3.1. Overview

The Quantum Entanglement Generation Protocol (QEGP) in covered in section 3.2, Distributed Queue Protocol (DQP) in section 3.3, and Midpoint Heralding Protocol (MHP) in section 3.4. Figure 3.2 gives a birds-eye view of the inter-workings of these protocols. DQP functions as a sub-protocol of QEGP and may be considered an ad-hoc solution which would in a large-scale network be fulfilled by a more fully-fledged routing protocol.

For a more complete description of the protocols, including a more complete rundown of their inner workings, we refer the reader to [38, Appendix E].

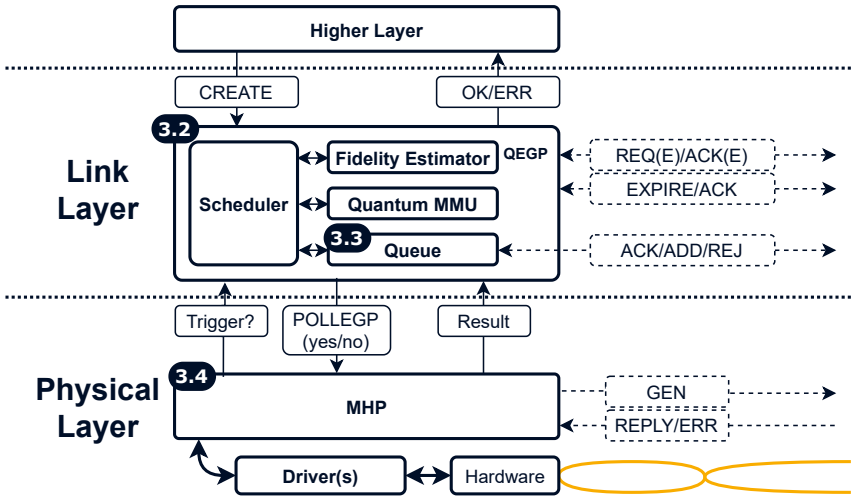


Figure 3.2: Flow diagram from [38] of QEGP, DQP (distributed queue), and MHP. Solid lines show inner node communication, dotted lines inter-node classical communication. The MMU (Memory Management Unit) and the Fidelity Estimator get the required parameters directly from hardware.

3.2. Quantum Entanglement Generation Protocol (QEGP)

The Quantum Entanglement Generation Protocol (QEGP) aims to provide robust entanglement generation between nodes along with a single link. QEGP's responsibilities are scheduling of requests from higher layers (section 3.3), estimating the fidelity of completed requests, and quantum memory management.

Say we have two nodes along a link: Alice and Bob, and a midpoint (repeater) station. When Alice wants to entangle with Bob she would issue a `CREATE` request to QEGP. In a fully-fledged quantum network stack, requests would be done at the application layer, which in turn would forward these to lower layers. However, for simplicity, we assume here that Alice polls QEGP directly. The parameters of the request are as follows:

- Minimum desired fidelity (①), denoted as F_{min}. If QEGP receives an entangled qubit and estimates that its fidelity is not high enough, QEGP will discard the qubit and instruct the physical layer to deliver another entangled qubit.

- Maximum time (②) we are willing to wait for an entangled qubit (followed by sending an `EXPIRE` message).
- Number of pairs to be created (③). This means that we can request the multiple pairs be created by the underling MHP without querying the QEGP multiple times (decrease inter-layer communication).
- Priority value (④), which is used by the scheduler.
- Type of request flag (⑤). Either create and keep (CK) or create and measure (MD) type request. CK request require both parties to store the pair (after successful entanglement) in quantum memory. Measure directly means that directly after establishing an entangled pair the qubits are measured.
- Atomic flag (⑥), used by a CK type request when multiple pairs must be made available at the same time.

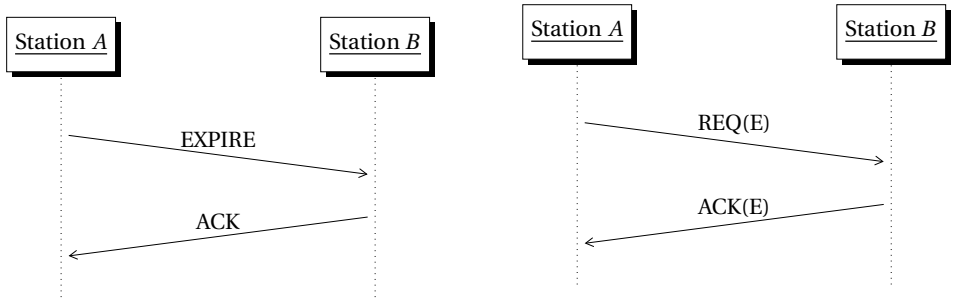
After receiving this request, QEGP will attempt to deliver an entangled pair using DQP and MHP. In the event of success, it will issue an `OK` response to the higher layer, with:

- The "Goodness", which is an estimate of fidelity in the case of create and keep requests as well as the quantum bit error rate for measure requests.

If it is not an `OK`, it will be one of the following error messages will be sent as a response to the higher layer. Note that these can either result from a failure within QEGP, DQP, or MHP:

- `TIMEOUT` a request could not be fulfilled in a specific maximum time (②).
- `UNSUPP` the requested minimum fidelity (①) is not achievable.
- `MEMEXCEEDED` quantum memory is too small to simultaneously store all pairs of an atomic request.
- `OUTOFMEM` quantum memory does not have enough space at this time to simultaneously store all pairs of an atomic request.
- `DENIED` refusal to participate from other party.

QEGP does have some responsibilities which involve communicating with Bob (which are not outsourced to DQP). This involves messages for (1) expired notices (fig. 3.3a) and (2) memory requests before starting entanglement (fig. 3.3b).



(a) Timeline of request expiration within QEGP.

(b) Timeline of memory advertisement requests within QEGP.

Figure 3.3: Classical messages transmitted by QEGP.

1. If Alice receives a successful MHP message (`REPLY`) with a sequence number that is too high, Alice will send an `EXPIRE` message to Bob to indicate that the pair will not be used. After, QEGP internally sends an `ERR_EXPIRE` message to higher layers.
2. QEGP internally performs some memory management (see MMU, fig 3.2). Alice and Bob also regularly keep each other updated on the number of available communication and storage qubits at each respective end. Figure A.5 in the appendix shows to format of the (small) memory request message.

3.3. Distributed Queue Protocol (DQP)

The distributed queue protocol is what allows Alice and Bob to communicate requests and align requests such that they are performed concurrently. DQP uses a parent-child configuration for scheduling. Figure 3.4 demonstrates the flow of messages within DQP. The parent queue may add items, and request the other to add an item as well. The child may then accept this request, and acknowledge or reject it.

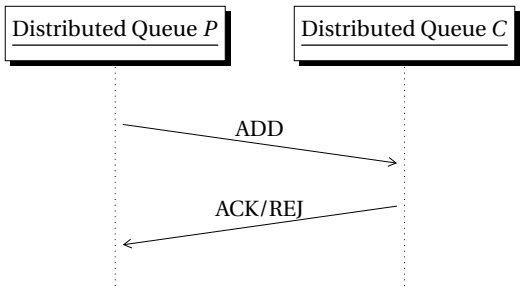


Figure 3.4: DQP operation timeline. Node *P* (parent) adds an item to the distributed queue by sending an `ADD` message to peer node *C* (child), which either acknowledges or rejects the request. This process is symmetric when *C* attempts to add an item to the queue.

The protocol ensures that (i) Alice and Bob have a synchronized queue and (ii) informs when entanglement generation should take place. If MHP asks if it should produce a pair, and QEGP detects an item at the front of the queue, it will pass the relevant parameters of the queue item to MHP.

If Alice is the parent, and she wants to add a request to her queue, she will first send an `ADD` request to Bob. This `ADD` request contains the following parameters relevant to the simulation later :

- Timeout, Derived from the `CREATE` Max Time field (②), which is the MHP cycle when we will timeout.
- Taken directly from the QEGP `CREATE` request: Minimum fidelity (①), number of Pairs (③), and request priority (④).
- Atomic flag (⑥) and measure directly flag (MD) which depends on request type (⑤).

Bob response with a `ACK` or `REJ`. An `ACK` will result in both parties adding the item to the queue, whereas a `REJ` will result in Alice dropping the request and QEGP to send a `DENIED` response to higher layers.

3.4. Midpoint Heralding Protocol (MHP)

Alice and Bob will use the Midpoint Heralding Protocol (MHP) to perform the actual entanglement using a midpoint station. It is a protocol that implements Heralded Entanglement Generation (HEG). For MHP to function, Alice and Bob must have tightly synchronized clocks, as MHP operates using time cycles (MHP cycles). DQP used a schedule cycle parameter to mark when entanglement may begin, and a timeout parameter to denote till which MHP cycle we may perform entanglement. The actual entanglement generation, using the midpoint station, is illustrated in figure 3.5.

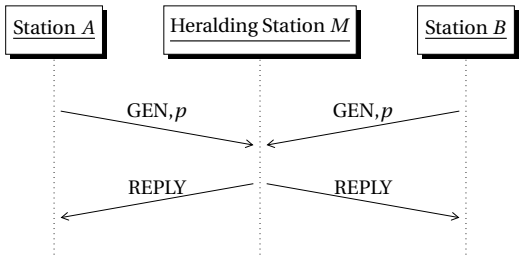
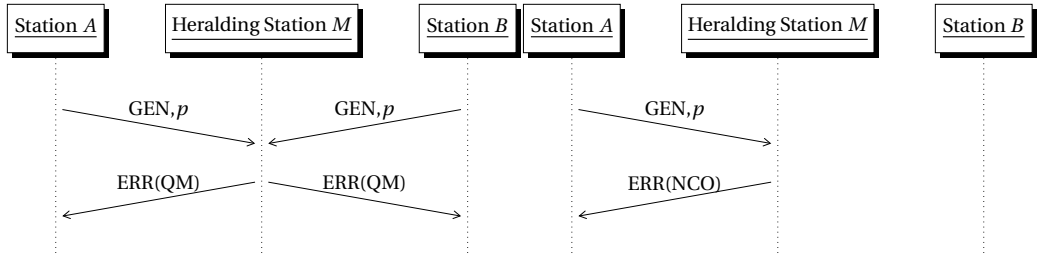


Figure 3.5: Timeline of the MHP message exchange with a successful reply by the heralding station; p is a photon associated with the `GEN` message.

Within the MHP, if Alice or Bob identify a request at the front of their queue, they will both send a `GEN` message to the midpoint station via the classical data plane. This `GEN` message is accompanied by a qubit sent via the quantum data plane, which is one half of a Bell pair. Likewise, Bob will also send one half of an entangled pair to the midpoint station accompanied by a `GEN` message. The midpoint will then attempt to perform an entanglement swap.

If the reply is an `OK` it means that Alice and Bob's two halves were successfully swapped at the midpoint station, meaning that Alice and Bob now have an entangled pair. This reply also contains the relevant classical correction information. Alternatively, the reply may be an `ERR(QM)` (`QUEUE_MISTMATCH`), or `ERR(NCO)` (`NO_MESSAGE_OTHER`) message, indicating failure (fig. 3.6). All packets contain qubit identifiers (QID) so that the QEGP upon success immediately knows where the qubit is stored in memory.



(a) Queue mismatch error (b) Single-sided transmission error

Figure 3.6: Timeline of two types of errors within MHP. QM and NCO refer to specific fields of the REPLY message (i.e. OT field), i.e. `QUEUE_MISMATCH` and `NO_MESSAGE_OTHER`, respectively.

Create and Keep (CK) type requests require that qubits are stored in memory. This means that requests are performed sequentially as in figure 3.5: A qubit may only be moved to memory after receiving the success heralding signal. However, in the case of Measure Directly (MD) type requests, MHP have to wait for an attempt to complete. One can continually attempt to generate entanglement in that case. Therefore, entanglement attempts may be performed in quick succession until success using a pipeline as depicted in figure 3.7.

Taking everything together, MHP, QEGP, and DQP, we demonstrate what a run to produce a single pair following a CK type request figure 3.8. Note that for MHP, due to the very tight

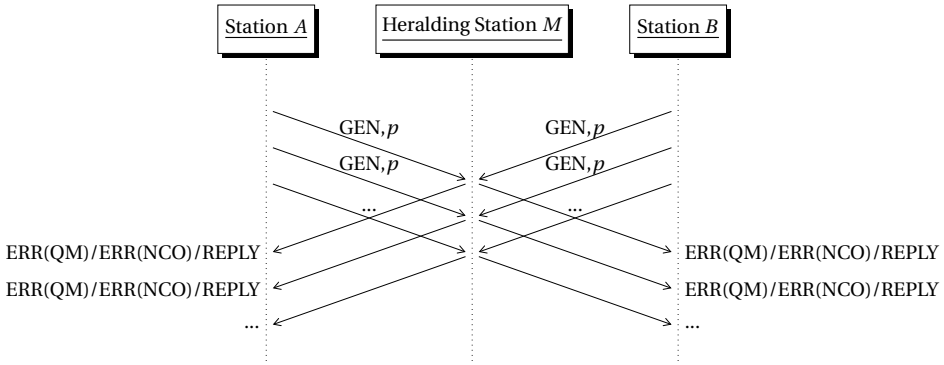


Figure 3.7: Timeline of multiplexing photon emission in the MHP. Multiple GEN, p messages are sent one by one for which any reply messages (ERR(QM), ERR(NCO) or REPLY) is possible.

timing constraints and (occasional) classical packet losses, it is to be expected that it takes a few attempts to succeed (many more than depicted here).

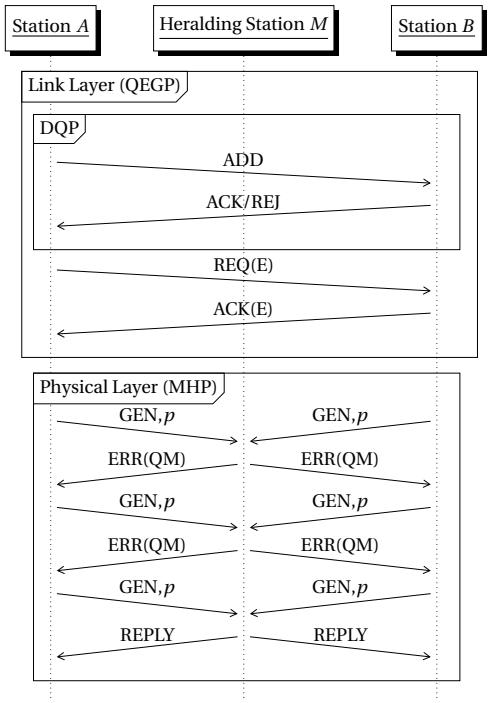


Figure 3.8: Example QEGP, MHP, and DQP protocol run, showing the interaction between the three protocols. Each photon p travels along the quantum channel.

CHAPTER 4

PRELIMINARIES III: DATA ORIGIN AUTHENTICATION AND QKD

We explain what it means to be information-theoretically secure in a data origin authentication setting. Data origin authentication refers to, for a given message, determining that (1) it was not altered during transit, and (2) sent by a legitimate user (specifically, the one we expect). That is, if Bob receives a message, they can determine both that it was from Alice and has that it was not modified while in transit. This is for instance useful when performing financial transactions, communicating with satellite control systems, and distributed control systems [86]. By extension, we also explain what we mean when we say something is "computationally secure" and how this differs from (). An information theoretically secure scheme either cannot be broken or, the chance of that happening (if non-zero) can be proven to be bound. In contrast, computationally secure schemes' security relies on the presumed, but unproven, difficulty of some problems. (This is also why information-theoretic security schemes are sometimes referred to as being unconditionally secure.) One such problem is the factorization of very large numbers, which is assumed to be difficult.

Smart's "Cryptography Made Simple" gives a more elaborate introduction to the security concepts discussed here [99]. There, chapter 9 gives an introduction to information-theoretic security, and chapter 14 hash functions and message authentication codes. These are used to perform data origin authentication.

4.1. Data Origin Authentication

One of our goals is to guarantee the validity of messages. This is done by passing messages through some cryptographic object which either signs the messages or produces a tag, which is sent alongside the message. In [80], this is explained as follows, each building on the previous item:

1. Firstly, one wants to guarantee *data integrity*. Typically one uses a cryptographically secure hash function with the data as input to produce a digest which is then sent alongside the data. On the receiver end, one runs this function again to verify that the message contents are intact.
2. Second, one wishes to achieve *data origin authentication*, which in addition to integrity verifies that the sender is indeed whom they say they are. It is trivial, in the previous item,

for Eve to simply construct a message, compute a digest, and send said pair to either Alice or Bob.
To combat this, one makes use of a Message Authentication Code (MAC) which additionally takes some shared secret as input. Eve does not have access to this shared secret, so she is no longer able to (efficiently) compute a tag. However, Eve can still resend old (message, tag) pairs to either Alice or Bob, which will be accepted as valid even though Eve is the sender. To combat this, Alice and Bob may pass an additional nonce (number only once) as input to prevent replay attacks. As the name suggests, Alice and Bob may only use a nonce once, meaning that they must maintain state. This means that Eve when transmits and old (message, nonce, tag) triplets, the recipient will detect that the nonce was already used and discard the triplet.
However, as the secret is shared, a third party cannot resolve a dispute about whether the first or second party sent a message. This would be a problem if Alice and Bob ever have a falling out.

3. *Non-repudiation* ensures that once a party has signed a message, they cannot deny doing so. This is achieved through asymmetric cryptography, where signatures are performed using a private key, and verification using a public key. However, these are often more expensive, computationally speaking, than symmetric cryptographic primitives such as MACs.

We are interested in data origin authentication, which is performed using a MAC. The MAC is a function that takes two at least two inputs: (1) A secret (the session key) known only to the sender and receiver and (2) the message to be sent, and optionally (3) a nonce. It then produces a (small) tag which is then sent alongside the message. Let Z be the tag space, and $F = \{f_1, f_2, ..., f_n\}$ a set of n encoding which map the message space M to tag space T, meaning $\forall i, f_i : M \rightarrow T$. A tag is given by $t = f_k(m)$, with $t \in T, f \in F, m \in M$, and k being the secret key. This k therefore encodes the choice of f in F, and the tuple (m, t) is what is transmitted (along the public channel).

During verification, the message and tag are given as input, as well as the secret key. The verification passes if the tag produced given the same secret key and message as the sender produces the same tag. A system is functional if for all possible secret keys $k \in K$:

$$\mathrm{Ver}_k(m, f_k(m)) = 1 \tag{4.1}$$

Where an output of 1 means that it is a valid tag. When referring to the MAC we typically refer to the function itself, though the tag itself is occasionally also referred to as a MAC depending on the context. The overall structure of a MAC is illustrated in figure 4.1.

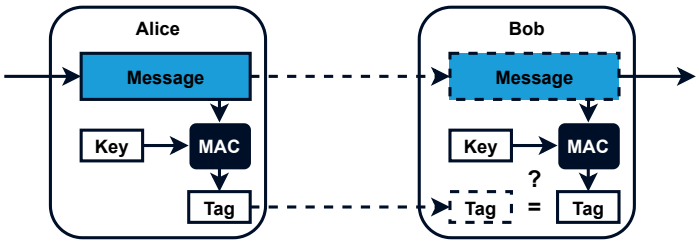


Figure 4.1: Overall structure of a MAC. The receiver does a check to verify that the message is unaltered. Typically, messages and tags are sent alongside another in one packet. The sender and receiver share a secret (key) for this scheme to function. Within a back-and-forth situation, Alice and Bob continually alternate between being sender and receiver

4.2. Notions of Security Of Data Origin Authentication

When talking about the security of data origin authentication, we care about forgeries: An attacker should not be able to forge tags for messages of their choosing. This gives the two following types of attack [58]:

Definition 4.2.1 (Impersonation attack)**.** When an attacker sends a message tag pair of their choosing, with the goal of it being accepted. This means that they succeed if, for a m' and t' of their own choosing:

$$\text{Ver}_k(m', t') = 1 \tag{4.2}$$

Definition 4.2.2 (Substitution attack)**.** When an attacker intercepts a message tag pair, replaces the message, and re-transmits it, with the goal of it being accepted. This means that they succeed if, for a m' of their own choosing, and for a $t = f_k(m)$:

$$\text{Ver}_k(m', t) = 1 \quad \text{and} \quad m' \neq m \tag{4.3}$$

We also care about the key:

Definition 4.2.3 (Key Recovery Attacks)**.** Tag message pairs do not reveal anything about the underlying key. In other words, they should not be vulnerable to *key recovery attacks* [86]. This should remain true even if the system itself is known to the adversary following *Kerkhoff's maxim* [61].

We distinguish between two notions of security: information-theoretic security and computational security. The latter assumes a computationally bound adversary, while the former does not. In other words, if a scheme is information-theoretic security, even an adversary with unbounded computational resources cannot break it. Computationally secure schemes rely on the presumed, but unproven, difficulty of some mathematical problem.

4.2.1. Information Theoretically Secure Data Origin Authentication

Such schemes are also sometimes referred to as *provably secure* schemes, as their security can be proven. Information-theoretically secure schemes are sometimes also referred to as *unconditionally secure* schemes, as their security is not conditioned on any assumption of difficulty. Therefore, an information-theoretically secure system is not susceptible to new algorithmic or hardware developments.

Kabatiansky and Smeets provide a clear definition of information-theoretic data origin authentication [58]:

1. Given a tag size of $\log|T|$ bits, if perfectly secure, then the chance of forging a tag is $1/|T|$ if the key is unknown. This is equivalent to a random guess.
2. Otherwise, a system may still be ϵ-secure. In such a system, the distribution of the tag space and the uniform space is distributed such that the statistical distance between them is ϵ:

   Definition 4.2.4 (Statistical distance)**.** Given two probability distributions p and q over the set Y. The statistical distance between p and q, denoted $\mathsf{d}(p,q)$, is as follows:

   $$\mathsf{d}(p,q) = \frac{1}{2}\sum_{y\in Y}|p(y) - q(y)| \tag{4.4}$$

   Where $p(y)$ and $q(y)$ are the probability values of event y within the distributions

p and q respectively.

Such a system does require a continuous stream of fresh key material.

4.2.2. Computationally Secure Data Origin Authentication

The security of a computationally secure MAC depends on the difficulty of solving some presumed difficult problem. It is difficult in the sense that the best currently *known* method cannot efficiently solve the problem. This also makes them more susceptible to advances in algorithmic theory: The Discovery of a new method to solve some previously difficult problems (far) more efficiently could render such a scheme insecure. A specific scheme is secure as long as no such efficient method (or methods) is found to solve (or, in this context, break) the underlying computationally difficult problem.

Generally speaking, we have the following notions of security when talking about MACs [99]:

Total Break The adversary can forge tags for any message of their choosing, as well as recover the underlying key used by Alice and Bob.

Universal Forgery An adversary is able to forge a tag for any message of their choosing, including both messages they choose and one provided to them.

Selective Forgery An adversary can forge a tag for a single message. This means that for a *given* m' (potentially of their own choosing) they are able to forge a tag.

Existential Forgery An adversary is able to produce a single (m', t') message pair for a message which was never signed. This differs from selective forgery in the sense that they must produce any valid (m', t'), whereas with selective forgery m' is given in advance and they must produce a tag valid for said message.

A computationally secure MAC should be *existentially unforgeable under chosen message attack* [99, Ch. 11]. This means that given many message tag pairs for messages of their choosing, it should not be possible for an adversary to efficiently forge a valid tag for any message. For instance, a message which is gibberish (with an accompanying tag) might still have parts that are interpreted as valid, thus interfering with the protocol. Therefore, existential forgeries should not be possible. If an adversary is able to perform universal (and to lesser extent selective) forgeries, then they are able to produce tags for any control message of their choosing, also interfering with the workings of the protocols in chapter 3. Examples of such interference are explored in section 5.2.1. Thus, a MAC is taken to be computationally secure according to the following paraphrased definition [99]:

Definition 4.2.5. A MAC is computationally secure if a polynomially bound adversary cannot produce an existential forgery. This is called a EUF-CMA MAC: It is Existentially UnForgeable against a Chosen Message Attack.

With:

Definition 4.2.6 (Polynominally bound adversary)**.** Say we have an algorithm A to break some scheme. (In the case of a computationally secure scheme, A would typically be the best known method to break the scheme.) There exists a polynomial $p()$ for which holds that, for any $x = \{0,1\}^*$, $A(x)$ terminates within at most $p(|x|)$ steps.

For a given MAC, the success probability of attacks succeeding is typically calculated as a function of the message input size and the number of queries q [86].

In general, when investigating MACS in the remainder of the document, we assume they are secure against existential forgery attacks. This is because we require that they have seen widespread use and that their security has been independently verified by many parties. If they were indeed not EUF-MAC secure, we assume they would not see wide-spread use.

4.3. Quantum Key Distribution

QKD allows parties to establish a shared secret in a provably secure fashion. Unlike classical handshake protocols, which derive their security from the assumed difficulty of solving some mathematical problem, QKD's security depends on the assumed correctness of the laws of physics.

There exist several QKD schemes, such as BB84 [11], B92 [12], E91 [45], and SARG04 [29], to name a few. BB84 is a relatively simple protocol, as well as being the first example of a quantum cryptographic protocol as far as we are able to identify. In the next chapter, we investigate in part the feasibility of using Quantum Key Distribution (QKD) for key establishment as part of a self-sustaining information-theoretically secure scheme.

4.3.1. BB84

We have the following four states:

$$\left|\psi_{00}\right\rangle = |0\rangle \qquad \left|\psi_{01}\right\rangle = |1\rangle$$
$$\left|\psi_{10}\right\rangle = \mathrm{H}|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \qquad \left|\psi_{11}\right\rangle = \mathrm{H}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$

Furthermore, BB84 makes use of a public authenticated classical channel to function. The protocol then functions as follows:

1. Alice generates two random strings of n bits long: a and b. She then encodes her state as follows:
$$\left|\psi\right\rangle = \bigotimes_{i=1}^{n} \left|\psi_{a_i b_i}\right\rangle \tag{4.5}$$
a determines whether bit i is a 0 or a 1, whereas b determines which basis bit i is encoded as. In other words, if $b = 0$, then the bit is encoded as $\{0 \to 0, 1 \to 1\}$, and if $b = 1$ then it is encoded as $\{0 \to -, 1 \to +\}$.
2. The qubits encoding the state are sent to Bob, who announces that he received them.
3. Bob measures each qubit in basis X and Z at random, determined by his own independent and random string b' also of length n.
4. Alice and Bob compare b and b' using the public channel and discard all measurement results where they used different bases. On average, this would mean that $1/2$ measurement results are discarded.
5. Alice picks a random subset of size t of the remaining bits $\approx n/2$ which are now denoted as test bits (Bennett and Brassard give as an example a third [11]) and announces their values to Bob.
6. Bob then compares these values to his own. If an acceptable amount are the same, Alice and Bob determine that the amount of interference (possibly by a third party) is within acceptable bounds and proceed. Else, they abort.
7. They perform *information reconciliation* and *privacy amplification* on the remaining $m = n/2 - t$ bits to produce k bits (where $k < m$) of shared key material.

Information Reconciliation This serves the purpose of error correction. After performing information reconciliation, Alice and Bob should with high probability have the same string. The goal of information reconciliation is therefore to ensure Alice and Bob have the same string while also ensuring that the process leaks as little information to the outside world as possible about said string. This is used as input to the privacy amplification stage to produce the final key. It may be performed wholly using classical communication.

Information reconciliation is defined as follows[I]:

[I] The definition has been taken from the TU Delft's *CS4090 Quantum Communication And Cryptography* lecture notes.

Definition 4.3.1 (Information Reconciliation)**.** Let X_A and X_B be distributed with joint probability distribution $P_{X_A X_B}$. An *information reconciliation* protocol for X_A, X_B is ϵ-correct and leaks $|C|$ bits if:

1. $\Pr[X_A \neq X_B] \leq \epsilon$.
2. The length of the message (or messages) exchanged on the public channel is $|C|$.

This process does, however, leak some information about the key to a third party. It is important that we at least have some randomness[II] left in the output to perform the next step of privacy amplification. The amount of randomness left decreases as $|C|$ increases. If the whole string is known by Eve, then there is nothing to amplify. Therefore, $|C|$ must not be too great, and so certainly not equal to the length of the entire string.

Privacy Amplification Privacy amplification is the final step and is a method for (significantly) reducing the information a third party has about the final key. It does this at the cost of some key bits, meaning that the final key length is smaller after performing amplification.

Definition 4.3.2 (Privacy Amplification)**.** Say that Alice and Bob have a shared string x [a], of which Eve has some information. Privacy amplification ensures Alice and Bob, each with string x from distribution X, produce the same string z from distribution Z which is close to uniformly distributed, including from the perspective of Eve.

[a] Assuming information reconciliation was executed successfully.

A simple example of a privacy amplification protocol is to make use of hash functions[III]. We refer to the key of the hash function, s, as the *seed* as it is public. Alice picks a random seed and passes the string x as input to the hash function: $f_s(x)$. Alice then transmits the seed over the public channel to Bob, who then performs the same function. Though Eve has access to the seed, she only has access to a part of the input, meaning that she won't know the final output of the extractor as its output is (close to) uniform.

4.3.2. Decoy State Protocols

Decoy states are meant to combat Photon Number Splitting (PNS) [110] attacks [72]. These attacks make use of the fact that while, say BB84, expects that a bit of information is encoded onto a single photons, in practice information is often (accidentally) encoded onto multiple photons before being sent to the other side. BB84 only guarantees unconditional security if the signal emitted by either party consists of a single photon each time.

Definition 4.3.3 (Photon Number Splitting Attack)**.** In a Photon Number Splitting (PNS) attack, Eve makes use of the fact that some pulses contain 2 or more photons. Eve can then keep one photon herself and transmit the second one onto Bob, keeping her own in memory. When Alice reveals string b (the basis announcements), Eve can measure the photon in the correct basis without introducing any detectable errors.

This attack in some sense circumvents the no cloning theorem: BB84's security relies on the fact that Eve cannot make a perfect copy of a quantum state herself, meaning that the signal cannot be split. However, if multiple photons are transmitted, then the signal can be split. Usage of decoy states to detect such attacks, as far as we can tell, was proposed first in [72] by Lo et al. [72]. The essence of decoy states is that Alice makes use of multiple signals, each signal having a different intensity associated with it. (A more intense signal transmits more photons.)

[II] In the context of information theory, randomness may be quantified using *information entropy*.
[III] Specifically, a *strongly universal*$_2$ *family* of hash functions.

The true signals, called the “signal” states, are used as normal, and all other states are referred to as “decoy” states. What the signal states were is announced publicly after completing the protocol. Any attempt by Eve to eavesdrop will, in theory, influence the probability that (1) Bob detects a given signal[IV] and (2) the Quantum Bit Error Rate (QBER) of the signals. This will then be detected by Alice and Bob.

Usage of such decoy states allows one to achieve high key rates over larger distances, all the while still providing unconditional security.

[IV] They state that it influences the yield Y_n of a signal, which is the probability that Bob detects a signal conditioned on an n photon signal being emitted by Alice.

CHAPTER 5

KEY CONSUMPTION: DATA ORIGIN AUTHENTICATION

> One should never try to prove anything that is not almost obvious.
>
> ALEXANDER GROTHENDIECK

We investigate in greater detail why data origin authentication is important using three concrete protocol proposals within the quantum stack: QEGP and MHP, and to a lesser extent DQP. Data origin authentication allows parties within the network to verify that messages are from legitimate parties. If no such authentication took place, it would be trivial for a third party to forge control messages which either alter the workings of the link or shut it down outright. Such authentication also prevents adversaries from submitting fraudulent entanglement generation requests which could rapidly destroy quantum memory. This also means that one must prevent replay-type attacks: An adversary must not be able to resend old legitimate message tag pairs which request entanglement. Control messages must additionally be verified in real-time: Nodes must only enact upon messages once they are verified to be from a legitimate party. As for encryption of classical messages, it does not protect the quantum data sent along the link, nor does it address the aforementioned issues. However, we conjecture that it could still prove useful in settings where one wishes to leak as little meta-data as possible.

Kozlowski and Wehner remarked that the key material produced by QKD could be used to perform information-theoretic security data origin authentication of classical control messages [64]. Such authentication is provably secure and virtually immune to advances in hardware and algorithmic theory. One should note that QKD does require some initial key material to function, which is why it often is referred to as a "key expansion primitive" [30].

information-theoretic security solutions require a constant stream of fresh key material which is (1) uniformly random and (2) unknown to (malicious) third parties. One such mechanism which in theory can generate such *shared* key material indefinitely is QKD. We calculate the rate at which QKD must generate key material to sustain real-time information-theoretically secure data origin authentication of control messages which accompany qubits. This is to show that it is indeed possible for those who require a high level of security. We also conclude that QKD must necessarily be run using some external mechanism. QKD cannot be run within the stack itself as the rate at which it produces key material in such a situation is too low.

We, however, also consider the question of information-theoretic security data origin authentication: Is the level of security it provides warranted? The only concern at the physical (and

link) layers is to uphold the availability of the link for the duration of the session and not make any other guarantees about say for instance confidentiality. This use case perhaps does not warrant the use of information-theoretic security solutions, opening the door to computationally secure solutions. The use of computationally secure data origin authentication solutions would most likely ease the constraint placed on the network and its designers. However, it was noted by Walenta et al. that QKD, which provides a stream of random key material, may be used to enhance the security of computationally secure schemes [105]. They use Advanced Encryption Standard (AES)[100] as an example, a block cipher, which is a symmetric-key algorithm. This would be an attractive solution if for instance one wishes to still use QKDto somewhat enhance the security of the network (specifically, the availability of the network), without requiring that QKD produce key material at (very) high rate.

5.1. Overview

We first establish why authentication is required in section 5.2. This consists of firstly performing a case study of the link-layer protocol from [38]. We also argue why encryption is of secondary importance here.

To understand the rate at which information-theoretically secure data origin authentication solutions consume key material, we explore an ITS solution in section 5.3: The Wegman-Carter MAC. Knowing how much key material we require per tag (and by extension per control message), we then investigate how much key material the quantum network stack consumes if we were to generate a tag for each control message separately in section 5.4. We also showcase some computationally secure data origin authentication alternatives in section 5.7. These, we argue, are viable alternatives in situations where (1) information-theoretic security is not necessary or (2) QKD cannot deliver key material at the rate required.

5.2. Motivation For Data Origin Authentication

Before investigating data origin authentication methods we must first establish why it is necessary within the quantum network stack. We consider it sufficient to show some examples of how adversaries can interfere with the system if there is no data origin authentication present. We are not the first to do so, however, as this was already noted in [64] and [38]. Satoh et al. also remark that forged control messages are a general concern for quantum networks [91]. This is with regard to maintaining the availability of the network.

5.2.1. Case Study Of QEGP, DQP, And MHP

The following is a non-exhaustive list of vulnerabilities we have identified. All following examples of vulnerabilities (or attacks) target the *availability* of the link itself. For a brief specification of the protocols, we refer to the reader to chapter 3. For a more complete specification, we refer the reader to [38, appendix E].

QEGP QEGP operates at the link-layer. With regards to classical communication, it is used to communicate the number of memory qubit a node has available. Second, it is also used to communicate the expiration of requests.

Example 5.2.1. Eve may modify the contents of `EXPIRE` messages and accompanying acknowledgements (`ACK`). If Eve continues to forge `EXPIRE` messages on Bob's behalf, Bob will always believe that their sequence numbers are misaligned and discard their half of the pair. If an accompanying `ACK` is delivered to Alice she will also discard her half of the pair.

Example 5.2.2. Eve may modify or forge memory requests and acknowledgements. These advertise the number of available communication and storage qubits. If either is set to 0, the receiver of the acknowledgement might assume that there are no communication and/or storage qubits available, preventing entanglement generation.

DQP DQP is used to communicate requests for entanglement. Therefore, if a third party may forge these, it is possible to continually request entanglement, which in turn destroy quantum memory as mentioned before. Alternatively, if a third party would continuously send create and keep type requests, this would also consume the memory of the respective node.

These examples are here to mainly showcase that any coordination protocol would almost certainly require some form of data origin authentication.

Example 5.2.3. If Eve modifies the parameters of a DQP `ADD` message sent by Alice, she could also cause a significant discrepancy in the queues of Alice and Bob, or halt communication outright.

Example 5.2.4. Eve could modify the requested minimum fidelity. After receiving a successful heralding signal from the midpoint, the fidelity estimator will estimate the fidelity of the pair. The discrepancy between the required minimum fidelity between two nodes might cause one node to proceed using a pair while the other disregards the pair. That is, MHP does not have a feedback loop for fidelity estimation [38, Protocol 2 - EGP, 3.c.v.B)], meaning that each node runs this independently and rejects/accepts the pair independently at this stage. It would therefore likely take some time to notice the discrepancy, causing even greater delays.

Example 5.2.5. Modify the sequence identifier within the request, disrupting the order in which pairs are generated.

Example 5.2.6. Change the qubit identifier (QID) such that Alice and Bob potentially entangle the wrong data qubits, causing cross-process interference. Any path running to this link could then be affected.

MHP Eve can also interfere with the MHP, which resides at the physical layer. This is not to be confused with any physical entanglement generation hardware, which itself may be attacked and is outside the scope of this work. MHP replies and requests may be modified or injected such that:

- The heralding station's resources are exhausted when continually receiving `GEN` requests.
- Modify the heralding signal (classical correction information) from the midpoint station so that Alice or Bob project onto the wrong basis in the event of a successful swap.
- Similarly, a successful heralding signal can be changed to an error code such Alice and Bob falsely conclude that the swap has failed.

We see that forged MHP control messages interfere with the availability of the link. Therefore, it must be possible to verify the validity of each control message.

5.2.2. Replay Attacks And Prevention

It remains vital that a third party cannot re-send old control messages as part of a replay attack. Such repeated requests may destroy quantum memory, particularly if Alice and Bob have no rate-limiting mechanism. More generally, re-sending old control messages interfere with the workings of these protocols. A simple example would be that MHP control messages are used to communicate the heralding signal. Continually re-sending old failures or incorrect correction information would significantly hamper the working of the link.

To prevent such replay attacks, we require more than a shared secret. Namely, if a MAC only takes as input a key and a message, it is trivial for an adversary to resend an old message-tag pair. The verifier cannot distinguish between messages sent by the third party and the legitimate party. Therefore, we require an additional mechanism such that the verifier can distinguish between fresh pairs and old (stale) pairs. The sender and verifier must maintain some sort of internal state to distinguish between the two. This state may be public, but must be continually updated to ensure old pairs are seen as stale. Alice and Bob then transmit (message_i, state_i, tag_i) tuples for verification each time. We illustrated this in figure 5.1. We identify two options: (1) A timestamp, and (2) a frame counter, or nonce, which stands for number used only once.

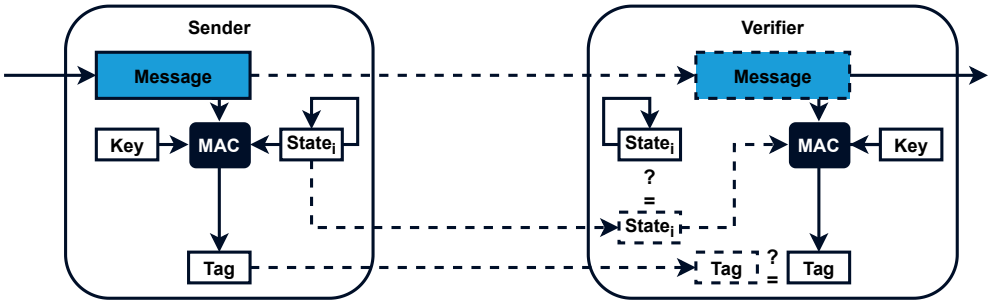


Figure 5.1: MAC combined with external state. The state is updated continually such that old (message$_i$, state$_i$, tag$_i$) triplets will fail the verification stage. Specifically, state$_i$ will then (ideally) not match the current internal state. Note that in the case of timestamps, one would have to introduce some tolerance into the equivalence function to account for e.g. transmission delays, as well as variances in said delays.

Timestamps Timestamps require that Alice and Bob's clocks are tightly synchronized, which is a necessity for a quantum link to function. Whenever a party sends a message, they pass both the message and the timestamp to the MAC. This timestamp is then sent alongside the message. Upon verification, the verifier checks that the timestamp sent alongside the message closely matches their own. If the adversary attempts to resend a message at some later time, the difference between the two will be too great and the verifier will reject this message.

One disadvantage of such a scheme is that it requires some tolerance to function. There is always some delay between transmitting and receiving messages. We would expect some variance in the delays as well, thus meaning that one must tolerate some variance in addition to a fixed delay. Even if the variance is minor, this would still leave the door open to occasional replay attacks.

Nonce Another option is to use a non-repeating number as input each time. That this number never repeat is the only requirement of the system: The number may be predictable and even known by the adversary. The easiest solution, then, is to simply increment the nonce with each use. A nonce also circumvents the need for tolerance as is the case with timestamps, making replay attacks nearly impossible. One disadvantage we is that a separate nonce must be maintained for each session, as opposed to maintaining maintaining a single clock for all sessions. However, we argue that a nonce does not consume space space for this to be of any real concern. This is especially true for a single link, which in all likelihood has only a single session associated with it. The largest nonce we identified which is used in practice is (only) 128 bits, used by Poly1305 [14].

5.2.3. Implications

The attacks we identify, on the classical plane, require Eve to be an active classical adversary: Eve must either forge or modify messages as part of her attack.

Data origin authentication enables nodes to verify both the integrity of messages and to identify legitimate senders. We assume that non-repudiation is not necessary. Within the scope of this work, we assume that Alice and Bob are well-behaved at the system level. This does not preclude that Alice and Bob don't trust one another at the application layer when performing say blind quantum computing.

One may ask whether encryption is necessary. It may be that Alice and Bob may want to leak as little information as possible. This may be desirable in a high-security setting. However, as encryption of system-level messages does not protect the quantum data in any direct way, we focus mainly on data origin authentication which is necessary to maintain the availability of the link. But we do not preclude the need for encryption and its potential usefulness at the system level in certain (high-security) settings.

5.3. Wegman And Carter

The Wegman-Carter style MAC, conceived by Wegman and Carter in 1981, is a well known authentication method which allows one to achieve information theoretic authentication [106]. We investigate the Wegman-Carter MAC to understand how much key material it requires.

If a message m and the tag $t = f_k(m)$ pair are known to the adversary the security is derived wholly from the adversary not knowing f_k. With this function, an adversary would be able to forge tags. Within their construct, f_k is taken from a set of strongly universal$_2$ hash functions. f must be chosen at random from this family for the scheme to be secure. Wegman and Carter note that a strongly universal$_2$ family is sufficient to build a secure MAC [106]. The definition for such a family, also given in [4], is as follows:

Definition 5.3.1 (Strongly Universal$_2$)**.** Let F be a universal hash family of functions $f : M \to T$. For any arbitrary and distinct $m_1, m_2 \in M$, and for any two necessarily distinct elements $t_1, t_2 \in T$, if F is a strongly universal$_2$ family then:

$$\Pr\left[f(m_1) = t_1, f(m_2) = t_2\right] \leq |T|^{-2} \tag{5.1}$$

The shared secret k is given as input to the family F, which outputs a function f_k, as illustrated in figure 5.2. The family is *strongly* universal$_2$ because the hash family is 2-wise independent. If it is not strongly universal, then it is only universal, meaning that it is not 2-wise independent[I]. n-wise independence is defined as follows:

Definition 5.3.2 (n-wise independence)**.** A class of hash functions $f \in F : M \to T$ is n-wise independent if for distinct $m_1, m_2, ..., m_n \in M$ and for any $\forall\, t \in T$

$$\Pr\left[f(m_1) = t_1, f(m_2) = t_2, ..., f(m_n) = t_n\right] = |T|^{-n} \tag{5.2}$$

Such a strongly universal construction is a one-time MACs: We may only use function f to authenticate a message once. They then give the following construct[II] for an information-theoretic security MAC for authenticating multiple messages, with f being from a almost-strongly universal$_2$ family:

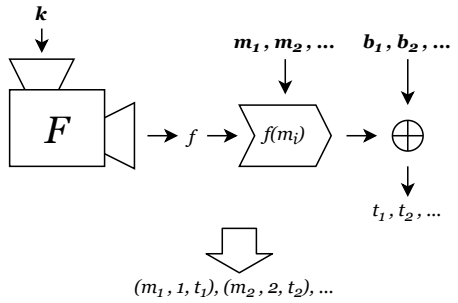
Theorem 5.3.1 (Wegman-Carter MAC for multiple messages)**.** Suppose some key $(f, (b_1, ..., b_n))$ has been chosen randomly from the set of keys. Let $m_1, ..., m_n$, be any n messages, with the restriction that the message numbers must all be different (we assume that m_i has number i.) Suppose a forger knows only the set F and the set of messages and their corresponding tags $t_i = f_k(m_i) \oplus b_i$. Then there is no new message (with any message number) for which the forger has a better than $|T|^{-1}$ chance of correctly guessing the tag.

The above construction allows one to tag multiple messages and is depicted in figure 5.2. This was proposed as an alternative to picking a new $f_k \in F$ for each message, which would consume key material at a higher rate. The output of the universal family is passed to a cryptographic object and encrypted, in this case by an OTP, which is provably secure [94]. Note that the index i may never be reused, as the OTP would otherwise catastrophically break down.

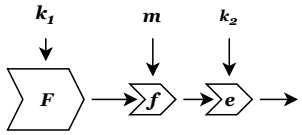
[I] A relatively recent (2014) work by Alomair shows that if a family is only universal, it is in some situations easy to find a collision if the family is a *dependent universal hash-function family* [4]. It is dependent if there exists a $m' \in M$, $m \in M$ for which $m \neq m'$, and at least one $f \in F$, for which $\Pr[f(m) = f(m')]$ is non-negligible.

[II] Carter and Wegman [32] give the following example of a strongly universal$_2$ family: Given message space $M = \{0, ..., |U| - 1\}$ and prime $p \geq |U|$, key $k = a, b$, where $a \neq 0$ and both are random integers modulo p, family $F = \{f_{a,b}\}$ is universal if:

$$f_{a,b}(m) = ((am + b) \mod p) \mod |T| \tag{5.3}$$



(a) Structure of ITS version. The output is encrypted using an OTP.



(b) The general structure of the Wegman-Carter construct. The universal family F must be resilient to collisions, and the cryptographic object e must be non-invertible, i.e. an encryption function. F is sometimes also referred to as the compression function, as it typically reduces the size of the output. e may also be a PRF instead of a One Time Pad (OTP), making the scheme computationally secure instead (sec. 5.7).

Figure 5.2: Structure of a Wegman-Carter style MAC. F is an almost strongly universal hash family, with f_i being the functions produced for each message. m_i are the messages produced and b_i is from a non-repeating key sequence (B).

Why Encrypt The Output? On a more fundamental level, one might wonder why we have to encrypt the output of the universal family in the first place. Abidin demonstrate why this encryption is necessary using an example attack where one output is encrypted, and one is not [1].

Example 5.3.2 (Existential Forgery and Strongly Universal$_2$ Families)**.** Eve intercepts both $(m_1, 2, t_1 = f(m_1))$ and $(m_2, 2, t_2 = f(m_2) \oplus K_1$. The K_1 is the key used by the OTP, which is only performed from the second message onward. This allows eve to identify the set of f in F for which holds $\{f \in F : f(m_1) = t_1\}$, which has size $|F|/|T|$. Then, Eve finds a message m_E such that $f(m_2) \oplus f(m_E) = t$ for all f in the above set. The above set is smaller than F and thus easier to search through. She then produces the tuple $(m_E, 2, t \oplus t_2)$. Eventually, the tag will be accepted as:

$$h(m_E) \oplus K_1 = h(m_2) \oplus h(m_E) \oplus t_2 = t \oplus t_2 \tag{5.4}$$

A perhaps more straightforward example is given if the underlying family is XOR-linear.

Definition 5.3.3 (XOR Linear Family)**.** A family is XOR linear if for every function f from said family, it holds that:

$$f(m_1 \oplus m_2) = f(m_1) \oplus f(m_2) \tag{5.5}$$

The family used by Wegman and Carter in [106] is, for instance, an XOR-linear family.

Example 5.3.3 (Existential Forgery and XOR-Linear Families)**.** Say that Eve has observed the first tuple $(t_1 = f(m_1), 1, m_1)$. Eve then intercepts the second tuple $(t_2 = f(m_2) \oplus K_1, 2, m_2)$, and sends $(t_1 \oplus t_2, 2, m_1 \oplus m_2)$ instead. Note that the first output $f(m_1)$ was not encrypted, whereas $f(m_2)$ was using K_1. Because the family is XOR-linear, the hash will be accepted because:

$$f(m_1 \oplus m_2) \oplus k_1 = f(m_1) \oplus f(m_2) \oplus K_1 = t_1 \oplus t_2 \tag{5.6}$$

In general, Eve may for the i^{th} round intercept the tuple (t_i, i, t_i) to produce $(m_i \oplus m_1, i, t_1 \oplus t_i)$. This highlights why it is important to encrypt every output of f, including the very first, as Eve can otherwise trivially forge tags without ever knowing k_1.

It is not difficult to see, following the above two examples, why Eve can trivially forge tags if no encryption of the output of h is performed at all. The purpose of the universal family is to be resilient to message collisions, and the encryption (or any non-invertible cryptographic primitive) of the output hides the key in use by the universal family. In short, the output of a (strongly) universal family must be encrypted for the scheme to remain secure.

5.4. Rate Of Consumption Of Key Material By The Stack

We know that any deterministic system cannot produce more randomness than is inserted into the system. Shannon makes a similar remark when talking about the uncertainty about the key in a system with provable secrecy [94]:

> *the amount of uncertainty we can introduce into the solution cannot be greater than the key uncertainty.*

Any information-theoretic security system needs a constant supply of randomness (in the form of a key). The question then is how much. Atici and Stinson review in 1996 Wegman-Carter constructs and the key material they require [5]. They remark that while the amount of key material required to specify f_k in $\{f_i\}_K$ (k in figure 5.2) has been optimized over the years through the construction of more efficient families, the amount of key material used by the OTP remains the same:

Lemma 5.4.1 (Key Consumption Rate Of Real-Time information-theoretic security Data Origin Authentication)**.** Each message requires $\log|T|$ bits for each new message.

Consider the following:

Example 5.4.2. A tag with a maximum value of 1023 (meaning a tag may be one of $t \in \{0,1,...,1023\}$), meaning that $|T| = 1024$, requires $\log 1024 = 10$ bits of key material for each use of the OTP. In a typical setting, one might use 128-bit tags, which would require 128-bits of fresh key material for each message.

In general, as each qubit is accompanied by two control messages (MHPspecification, sec. 3.4), we require $2\log|T|$ bits of key material per qubit transmitted. To make clear that this is the set of possible tags accompanying *physical* layer control messages, we denote this set as T_1. To simplify we ignore the requirements of QEGP's messages, are these are sent intermittently. We then investigate the amount of key material required by a direct link between the nodes and the midpoint, or conversely, the maximum transmission rate of quantum data following the key rate produced. We operate at the physical layer, so calculate the transmission rate following the key rate of the Alice-midpoint and Bob-midpoint connections. The placement of QKD links that generate this key material, which we sometimes refer to as boxes, is illustrated in figure 5.3. The overall pair generation rate would then be capped by the minimum key generation rate, but not the sum of the rate of key generation.

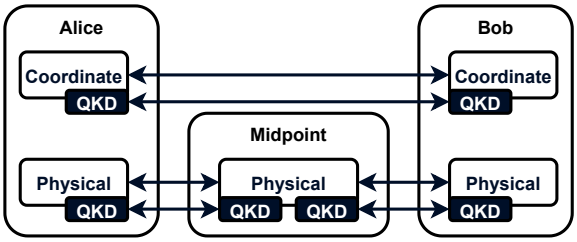


Figure 5.3: The placement of QKD links within the stack. We calculate here the required key rate of these QKD links. Or, conversely, we calculate the maximum transmission rate quantum data following the key generation rate of these links. The key material generated for coordination purposes could, in theory, be used for authentication of any direct communication between Alice and Bob.

Therefore, if the attempt-rate on the quantum channel is r_{attempt}, the rate at which keys must be produced must be at least

$$r_{k_1} = 2r_{\text{attempt}}\log|T_1| \qquad \text{[bit/s]} \quad (5.7)$$

if each control message is to be authenticated using an information-theoretic security scheme.

We may want to express the key rate in terms of the rate at which pairs are successfully produced. We denote this rate as r_{data_1}. We must then also factor in entanglement generation attempts that are unsuccessful, which also consume key material. This is achievable using a geometric distribution, which given the successful attempt of an entanglement generation attempt gives the expected number of attempts (independent trials) until successful entanglement generation. Given a probability of success p during each attempt, the expected number of trials until success is

$$\mathrm{E}[X] = \frac{1}{p_{\text{succ}}} \tag{5.8}$$

One can interpret this as the amount of swap attempts which must be made to produce a single pair. Following this, we then have:

$$r_{\text{data}_1} = r_{\text{attempt}} p_{\text{succ}} \quad [\text{bit/s}] \tag{5.9}$$

The key rate must therefore be, for a given probability p_{succ} of successful swapping, a tag size of $|T_1|$, and a transmission rate on the quantum channel of r_{attempt} be as follows:

$$r_{k_1} = \frac{2 r_{\text{data}_1} \log|T_1|}{p_{\text{succ}}} = 2 r_{\text{attempt}} \log|T_1| \quad [\text{bit/s}] \tag{5.10}$$

Example 5.4.3. Say that the probability of a successful generation attempt is $p_{\text{succ}} = 10^{-4}$. Say that the specification requires we produce $r_{\text{data}_1} = 64$ pairs per second. We assume each control bit has a tag of $\log|T_1| = 16$ bits. The consumption rate of the scheme is according to eq. 5.10:

$$\frac{2 \times 64 \times 16}{10^{-4}} = \frac{2048}{10^{-4}} = 20\,480\,000 \quad [\text{bit/s}] \tag{5.11}$$

Meaning that the rate at which key material should be generated at minimum along both links (fig. 5.3, physical layer) is 20.48 Mbit/s.

Alternatively, it might be useful to, given a certain key rate, determine how many entanglement generation attempts one may make at most. Given r_{k_1}, we rewrite eq. 5.10 to:

$$r^{\text{upper}}_{\text{attempt}} = \frac{r_{k_1}}{2\log|T_1|} \quad [\text{Hz}] \tag{5.12}$$

The true attempt rate r_{attempt} is influenced by many other factors, such as hardware. The above is only an upper bound on the attempt rate. Indirectly, however, this does also form an upper bound on the data transmission rate. We may insert $r^{\text{upper}}_{\text{attempt}}$ directly into eq. 5.9 to get the following:

$$r^{\text{upper}}_{\text{data}_1} = p_{\text{succ}} r^{\text{upper}}_{\text{attempt}} \quad [\text{bit/s}] \tag{5.13}$$

All this assumes that the key generation rate is the same everywhere. However, in the case of MHP (as it is now), Alice and Bob have a midpoint station between them. Therefore, Alice and Bob each have a separate session with the midpoint station, meaning that there are two separate mechanisms (QKD links) producing key material. In a more mature network, Alice and Bob could easily have multiple midpoint stations between them. We therefore generalize eq. 5.12 as follows: Say that there exist n links between Alice and Bob, depicted in figure 5.4. Each link has an accompanying QKD link which produces key material of a rate at $r_{k_{1,n}}$ for link n. As swaps are performed over increasingly large distances, nodes pass on the key material they generate to where it is required, meaning that they make use of *trusted node QKD*. Assuming there are no other bottlenecks, the maximum speed at which a station may swap is determined by the speed at which its immediate connections can generate key material.

$$r^{\text{upper}}_{\text{attempt}} = \min\left(\frac{r_{k_{1,1}}}{2\log|T_1|}, \frac{r_{k_{1,2}}}{2\log|T_1|}, ..., \frac{r_{k_{1,n}}}{2\log|T_1|}\right) \quad [\text{Hz}] \tag{5.14}$$

$$= \frac{\min\left(r_{k_{1,1}}, r_{k_{1,2}}, ..., r_{k_{1,n}}\right)}{2\log|T_1|} \quad [\text{Hz}] \tag{5.15}$$

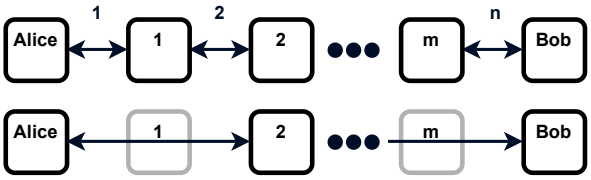


Figure 5.4: n direct links between Alice and Bob. Entanglements swapping is performed over increasingly large distances, requiring that key material also be available over increasingly large distances. One could make use of trusted node QKD to extent the distance over which QKD can supply key material. Another option is to have dedicated links between each station, though this (1) increases the complexity of the link (for m midpoint stations, for a total amount of nodes $n = m+2$, the link requires $n(n+1)/2$ dedicated QKD links), and (2) requires that QKD can operate at the required rate for the final swap by the midpoint station in the middle of the link.

The data transmission rate q additionally depends on the chance of success at each individual station. If we take the overall probability of a successful chain of swaps to be $p_{\text{succ}}^{\text{link}}$, eq. 5.13 for data rate remains similar:

$$r_{\text{data}}^{\text{upper}} = p_{\text{succ}}^{\text{link}} r_{\text{attempt}}^{\text{upper}} \quad \text{[bit/s]} \quad (5.16)$$

Total Amount Of Key Material Knowing the total amount of key material required could be useful if one wants to generate it in advance. To know the total amount of key material required, we must take into account the key material required to start the session itself: c_{k_1}. We require key material to jump-start QKD (as it is a key-expansion primitive), and a component to select $f_k \in F$ within the Wegman Carter construct:

$$c_{k_1} = c_{k_1}^{\text{QKD}} + c_{k_1}^{\text{MAC}} \quad \text{[bit]} \quad (5.17)$$

We motivate these as follows:

$c_{k_1}^{\text{QKD}}$ Assuming one uses QKD, this is the number of key bits QKD itself requires to start. For instance, the amount of key material required by its internal data origin authentication mechanism. In theory once QKD is running it could supply the key material required to initialize the Wegman-Carter MAC, meaning that in such a setup:

$$c_{k_1} = c_{k_1}^{\text{QKD}} \quad \text{[bit]} \quad (5.18)$$

$c_{k_1}^{\text{MAC}}$ The number of bits required by the Wegman Carter construct to specify $f_k \in F$. The actual number of bits then depends on the family in use and the maximum size of control messages ($|M|$), as well as of course the possible number of tags ($|T|$). E.g. Walenta et al. [105] make use of an ASU_2 family by Bierbrauer et al. [21] of approximate size $2\log\log|M| + 3\log|T|$[III].
If one does not wish to run QKD along a link, they key material is still required to specify $f_k \in F$ is:

$$c_{k_1} = c_{k_1}^{\text{MAC}} \quad \text{[bit]} \quad (5.19)$$

The total amount of key material required by a single session is then pertained by adapting the earlier derivations we performed for key rates. One then gets the following by adapting eq. 5.12 and eq. 5.10:

$$S_{\text{attempt}} = \frac{S_{k_1} - c_{k_1}}{2\log|T_1|} \quad \text{[bit]} \quad (5.20)$$

$$S_{k_1} = 2S_{\text{attempt}}\log|T_1| + c_{k_1} \quad \text{[bit]} \quad (5.21)$$

[III] Assuming all MHP messages are padded to 64 bits, and a tag length of 16 bits, then $c_{k_1}^{\text{MAC}} = 2 \times \log 64 + 3 \times 16 = 60$ bits.

Eq. 5.20 quantifies how many attempts are allowed given a certain amount of key material S_{k_1}. A certain amount of key material is consumed to start the session, which may not be used for entanglement attempts. Eq. 5.21 quantifies the total amount of key material required at the physical level. The amount of pairs which are produced calculated the same as before (eq. 5.13):

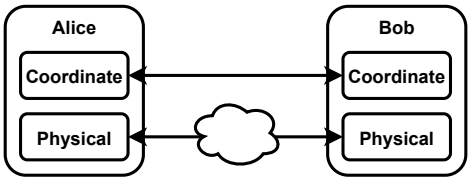
$$S_{\text{data}_1} = S_{\text{attempt}} p_{\text{succ}} \quad \text{[bit]} \quad (5.22)$$

5.4.1. Combined With Entanglement Coordination

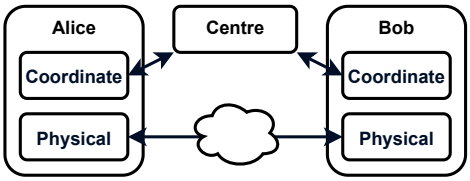
We would expect that any entanglement generation mechanism would:

1. Send request for entanglement messages to each node which each request $n \geq 1$ qubits.
2. Want an acknowledgement message for said message.

Whether the coordination mechanism uses a centralised node or distributed system is not of major significance when talking about key material. Either the required key material is generated between Alice and Bob, or between Alice and Bob, Alice and the central node, and Bob and the central node. Though the amount of messages sent is doubled in the latter case, the minimum amount of QKD poxes, due to a doubling in the number of links, has also doubled. The two alternatives here are illustrated in figure 5.5.



(a) Direct coordination between Alice and Bob.



(b) A centralised node coordinates entanglement. Other schemes are also possible, for instance one where there is a link between Alice and Bob *and* to the central node. We do not consider such a scheme, but propose using k_2 and $k_{3,n}$ to distinguish the key material used for direct coordination between Alice and Bob (bot not k_1, which is used at the physical layer) and the centralised node, as the key material would be generated by different QKD links in such a situation.

Figure 5.5: Two alternatives for facilitating entanglement coordination. Every link implicitly has one (or more) QKD link(s) supplying key material.

A centralised node does influence the amount of key material one needs to start a session. We refer to the key material required at this level as k_2. Given n links from a central node to n nodes, each node requires $c_{k_{2,n}}$ bits of key material to start the session. The amount of key material required along each link is determined by the QKD implementation and MAC in use (eq. 5.17). The central node must have the sum of this key material at the start of the session:

$$c_{k_2} = \sum_{i}^{n} c_{k_{2,i}} \quad \text{[bit]} \quad (5.23)$$

The rate at which we may submit requests is similar to the rate at which we may attempt entanglement generation (eq. 5.12), though it now depends on the rate of key material generation for the entanglement coordination mechanism.

$$r_{\text{request}}^{\text{upper}} = \frac{r_{k_2}}{2\log|T_2|} \quad \text{[Hz]} \quad (5.24)$$

If a centralised node is used, then this may be modified such that we take into account every link $1 < l \leq p$ between the central node and end-nodes (Alice and Bob in our example):

$$r_{\text{request}}^{\text{upper}} = \frac{\min\left(r_{k_{2,1}}, r_{k_{2,2}}, ..., r_{k_{2,p}}\right)}{2\log|T_2|} \quad \text{[Hz]} \quad (5.25)$$

And like with eq. 5.10, the rate at which we must produce key material along a direct link (with no centralised) node following a given request rate is:

$$r_{k_2} = 2r_{\text{request}} \log|T_2| \qquad \text{[bit/s]} \quad (5.26)$$

In the case of a centralised node, every link must supply a key rate equal to or greater than this value. r_{request} is the rate at which requests are submitted to the link as a whole.

Data rate Let n be the number of pairs requested. We now have three options when discussing the rate of key consumption with regards to the number of pairs a link produces:

n_{min} Assume the worst case where each request we receive requests only a single pair.
n_{avg} Average number of pairs requested. This is possible by simply analysing the link and determining the average number of requested pairs over time.
n_{max} Assume that each request requests the maximum amount of pairs allowed.

One may opt to use different size tags for control messages. As such, we denote the tag space for these messages as T_2 to differentiate it from T_1: The rate at which entanglement coordination messages are transmitted is only linked to the amount of pairs produced. Therefore, it is linked to r_{data_1}, and not r_{attempt}. The capped transmission rate, when taking into account physical layer communication and entanglement coordination, would then depend on the two key rates r_{k_1}, r_{k_2}. The rate $r^{\text{upper}}_{\text{data}_2}$ then is as follows:

$$r^{\text{upper}}_{\text{data}_2} = \min\left(r_{\text{data}_1}, r_{\text{request}} \times n\right) \qquad \text{[bit/s]} \quad (5.27)$$

This is the rate of data transmission when one takes into account both the physical layer and the entanglement coordination mechanism. A complete overview of the relationship between data rates is depicted in figure 5.6.

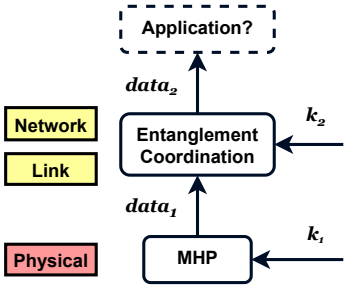


Figure 5.6: Overview data rate and key rates we defined concerning the bottlenecks formed by protocols.

Total Amount Of Key Material Like before, we derive the total amount of key material. The total amount of key material required may in this setup differs per node: In a centralised setup, the centralised node must store key material for each session it initiates with every node it is connected to along the link (fig. 5.5b and eq. 5.23):

$$S_{\text{request}} = \frac{S_{k_2} - c_{k_2}}{2\log|T_2|} \qquad \text{[bit]} \quad (5.28)$$

$$S_{k_2} = 2S_{\text{request}} \log|T_2| + c_{k_2} \qquad \text{[bit]} \quad (5.29)$$

We then modify eq. 5.27 (see eq. 5.22 for definition of S_{data_1}) to get the following:

$$S^{\text{approx}}_{\text{data}_2} = \min\left(S_{\text{data}_1}, S_{\text{request}} \times n\right) \qquad \text{[bit]} \quad (5.30)$$

Key Generation Over Larger Distances We expect that r_{k_2} is generally lower than r_{k_1}. A direct QKD link between Alice and Bob is almost certainly going to be longer than the direct links between them and the midpoint station. This places a certain amount of strain on QKD, whose key generation rate decreases typically with distance. For instance, a solution by Dixon et al. produces approx. $10^{6.05} \approx 1.12\,\text{Mbit/s}$ of key material at a distance of 10 km, and approx. $10^{5.5} \approx 316\,\text{kbit/s}$ at a distance of 20 km [40, fig. 3].

Fidelity Of Pairs The above rates form an upper bound on the number of qubits that may be transmitted, given a certain set of key rates. However, fidelity is also an important parameter when talking about entanglement. As remarked in [38], it generally takes longer to produce high-fidelity pairs. Within [38], the bright state population parameter (α) directly influences the average fidelity of pairs and the chance of a successful swap [38, fig. 8]. p_{succ} then directly encodes the minimum fidelity requirements. In the case of Heralded Entanglement Purification (HEP) (or more generally, entanglement distillation), multiple pairs are combined to produce a smaller set of higher fidelity pairs. If used, such repeated attempts would consume further key material and would need to be modelled as well. We do not provide derivations for the use of key material by Heralded Entanglement Purification (HEP) as [38] do not make use of distillation, though the stack they outline does not preclude its use.

5.4.2. Derivations Summarized

All derivations we have investigated are summarized in table 5.1.

Table 5.1: Entanglement attempt, pair generation, and key generation rates summarized. Coordination layer is a placeholder name for the coordination mechanism.

Layer	Value	Description	Formula
Physical	$r^{\text{upper}}_{\text{attempt}}$	Maximum rate of attempted heralded entanglement generation following a key rate of r_{k_1}.	$\frac{r_{k_1}}{2\log\lvert T_1\rvert}$
	$r^{\text{upper}}_{\text{data}_1}$	Maximum rate at which a link produces entangled pairs.	$r^{\text{upper}}_{\text{attempt}} p_{\text{succ}}$
	r_{k_1}	Required key generation rate at the physical layer for a given attempt rate.	$2r_{\text{attempt}}\log\lvert T_1\rvert$
	S_{attempt}	Total attempts. c_{k_1} is the key material required to initiate a session.	$\frac{S_{k_1}-c_{k_1}}{2\log\lvert T_1\rvert}$
	S_{k_1}	Total amount of key material.	$2S_{\text{attempt}}\log\lvert T_1\rvert + c_{k_1}$
	S_{data}	Total pairs delivered.	$S_{\text{attempt}} p_{\text{succ}}$
Coordination[a]	$r^{\text{upper}}_{\text{request}}$	Maximum rate at which entanglement requests are submitted to the link.	$\frac{r_{k_2}}{2\log\lvert T_2\rvert}$
	$r^{\text{upper}}_{\text{data}_2}$	Maximum rate at which pairs are produced by when taking into account request rate. n may be either the minimum, maximum, or average number of pairs specified per request.	$\min\left(r^{\text{upper}}_{\text{data}_1}, r^{\text{upper}}_{\text{request}} \times n\right)$
	r_{k_2}	Required key generation rate at the coordination layer for a given request rate.	$2r_{\text{request}}\log\lvert T_2\rvert$
	S_{request}	Total amount of requests made. c_{k_2} is the amount of key material required to initiate a session.	$\frac{S_{k_2}-c_{k_2}}{2\log\lvert T_2\rvert}$
	S_{k_2}	Total amount of key material.	$2S_{\text{request}}\log\lvert T_2\rvert + c_{k_2}$
	S_{data_2}	Total amount of pairs delivered.	$\min\left(S_{\text{data}_1}, S_{\text{request}} \times n\right)$

[a] See fig. 5.5 for architectures under consideration.

5.5. Tag (And Key) Size

As we shall see later (Ch. 8-9), the size of tags has a noticeable effect on the performance of the link. Logically, one would want to have as short tags as possible from a performance perspective. However from a security perspective, one wants as large a tags as possible.

Typically, when using computationally secure MAC solutions, there are already some well-established rules of thumb with regards to the minimum tag size. As of 2004, a 56-bit key is generally regarded as too short[IV] [86]. Preneel states that keys of 80-90 bits are sufficient for 10 to 15 years of security, and 128-bit keys are more suited for long-term protection (50 years) [86]. However, note that an adversary in our setting may only make use of a key if a session is still active: If they recover the key after a session, it is of no use to them. Therefore, a lifetime of 50 years is likely excessive. Therefore, 80 bits would already likely be more than sufficient. Seeing as information-theoretic security MACs provide a "higher" level of security, however, do these same rules of thumb hold?

If the tag is too short, it is trivial for Eve to perform forgery: If the tag size is $\log|T|$, the chance of Eve forging a valid tag for any message of their choosing purely through guessing is $2^{-|T|}$ each time. A short tag would be acceptable if we accept occasional forgery: Assume we use 16-bit tags. This results in a 2^{-16} chance of forgery. Furthermore, 12.5 million packets are transmitted per second on a 1 Gbit/s channel with 64-bit control messages and 16-bit tags[V], a message would be accepted on average once every $(64+16)/10^9 \times (2^{-16})^{-1} = 0.0052$ seconds with:

$$\text{exp. time between successful forgeries} = \frac{\text{packet size}}{\text{channel bit rate}} \times \text{forgery probability}^{-1}\text{[s]} \quad (5.31)$$

This would make the link nearly unusable, as forged packets would constantly be accepted. A tag of 56-bit reduces this value to roughly once every 3.15×10^7 seconds (which equates to roughly once every year). See figure 5.7 for the relationship between packet size and chance of forgery.

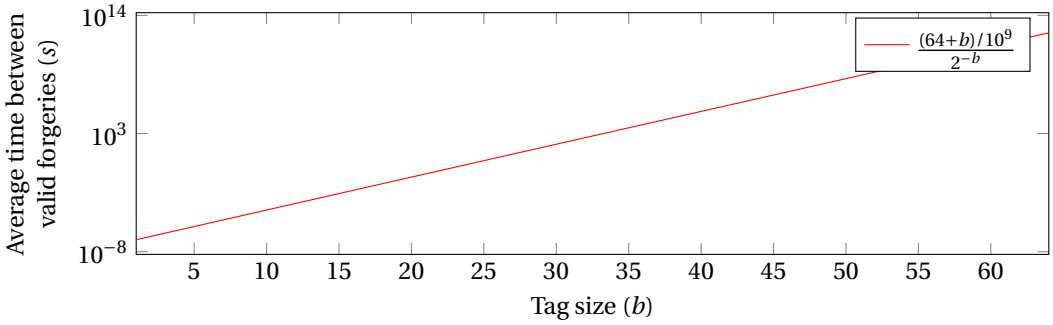


Figure 5.7: Tag size versus average occurrence of forgery if Eve continuously send messages with random b-bit tags along a 1 Gbit/s classical channel. This assumes 64-bit control messages for simplicity.

Use of an information-theoretic security MAC, which uses a OTP, would imply that a successful forgery does not reveal anything about the underlying f_k. However, Handschuh and Preneel reveal that this is not necessarily the case depending on which universal family is used as part of a Person-In-The-Middle (PITM) attack [50]: Alice sends a valid message, nonce, tag triplet, Eve then modifies *only* the message and sends this as a verification query[VI] *without*

[IV] For instance, the authors of Chaskey recommend using a tag size of at least 64 bits if one wants to keep the chance of accepting an inauthentic message to a minimum [76].

[V] The IEEE 802.3 specification does specify that there should be 12 bytes between each packet. The rate at which Eve may transmit packets might therefore be slightly lower than presented here.

[VI] A query where the adversary receives a yes/no answer for whether a message-tag pair (or message-nonce-tag triplet) is valid.

knowing the key used by the OTP. If the modified triplet is accepted as valid, Eve knows that both the original message and the modified messages (and nonce pairs) produced the same hash value. Assuming the OTP used key k_i (nonce inputs should be identical, so omitted):

$$\text{if} \quad f_k(m_i) \oplus k_i = f_k(m_i') \oplus k_i \quad \text{, then} \quad f_k(m_i) = f_k(m_i') \tag{5.32}$$

Recall that we have shown that knowing just a few input output pairs of a (almost-)strongly universal family may reveal the index k (sec. 5.3). If an adversary knows the key of the strongly universal family, they will be able to perform existential forgery attacks in future without knowing the key used by the OTP for messages. Handschuh and Preneel recommend that [50]:

1. Care must be taken to never re-use a nonce.
2. Care must be taken not to leak parts of the key used to index the universal family through side channels. Partially revealing the key makes certain key recovery attacks possible.

It is best to pick a new $f_k \in F$ every so often before Eve is able to perform a successful forgery. Shorter tags imply that a new f_k must be chosen more often. For instance, using a 32-bit tag means Eve can on average produce forgeries once every 412 seconds, requiring that $f_k \in F$ be changed relatively often, but far less so than if we use 16-bit tags. In practice, however, this might actually be much worse, as Eve must perform a PITM attack. This attack is limited not by the bit-rate of the channel, but the rate at which control messages are transmitted, which in all likelihood will be much lower. Therefore, the time between successful forgeries then becomes:

$$\text{exp. time between successful forgeries} = \text{packet rate} \times \text{forgery probability}^{-1} \quad \text{[s]} \tag{5.33}$$

Therefore, 32-bit tags we consider to be more suited to our use case. This also because the size of F is limited, in turn meaning that we do not need much key material[VII] to specify $f_k \in F$. Therefore, we assume that refreshing the choice of $f_k \in F$ every few hundred seconds (in the worst case) costs less key material than consuming (a few) extra bits of key material for each control message transmitted.

5.6. Supplying Information Theoretic Key Material

For the authentication mechanism to be information-theoretically secure, it requires that the key be uniformly random. While we could assume that this key material is initially available, such a system would not scale: The longer the system is in use, the more key material is required. Kozlowski and Wehner conjectured that it may be possible to leverage the quantum network to run QKD [64]. This in theory should be able to supply this key material indefinitely.

5.6.1. Using The Stack Itself

Figure 5.8 illustrates how QKD would be used if it were to run at the application layer. We also show how QKD consumes some of the key material it produces to perform authentication of its basis announcement.

Every qubit must be accompanied by a control message and tag pair which must be verified in real-time. If a node were to execute an instruction and only afterwards discover that it is fraudulent, it is already too late. It is therefore not possible to batch multiple control messages together and produce a tag for this batch. A similar approach is used in [105], a QKD algorithm which batches many basis announcements together before generating a tag for many announcements.

First, consider how many control messages are required to produce a single pair. There are session keys between Alice and the midpoint, the midpoint and Bob, and Alice and Bob. We for each qubit send a control message to and from the midpoint. This is done once using the

[VII] We previously showed that $\log|F| = 60$ bits when control messages are all 64-bits in length (sec. 5.4, Total Amount Of Key Material).

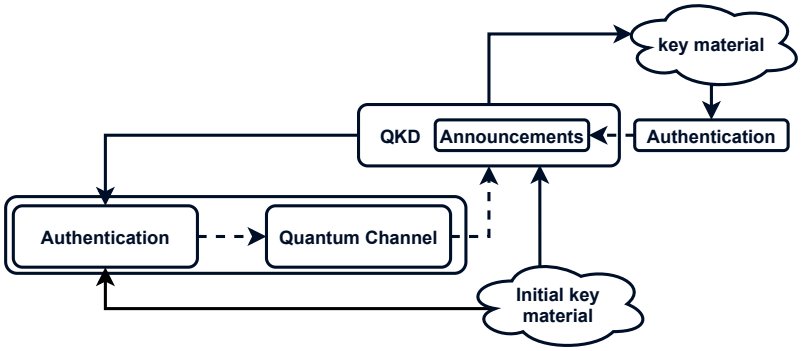


Figure 5.8: Sketch of system with information-theoretic authentication and QKD for key distribution. QKD is run at the application layer within the stack. The solid lines represent the flow of (fresh) key material. The dotted lines represent a supporting role, i.e. "Authentication supports the workings of the quantum channel".
After an initial jump start (using initial key material), the authentication mechanism should have supported a run of QKD which is used to supply the authentication mechanism. Note that QKD itself also needs some initial key material to function. For this reason, it is also often referred to as a key expansion primitive [30].

session key between Alice and the midpoint. The same holds for the connection between the midpoint and Bob. Therefore, we must send at least two control MHP messages per qubit at the physical layer, which leads us to conclude the following:

Lemma 5.6.1. It is not possible to produce a new tag for each new control messages if each qubit is accompanied by two, or more, information-theoretic security authenticated control messages supplied by key material from QKD which runs on the same stack.

Proof. Take $|t| \geq 1$ to be the size of tags. QKD can at most produce 1 bit of key material per qubit pair[VIII], whereas a single pair requires at least two MHP messages accompanying it. Always $1 < 2 \times |t|$ if $|t| \geq 1$, so the scheme cannot be self-sustaining. □

5.6.2. Using A Separate Module

Another solution is to use an off-the-shelf QKD module as depicted in figure 5.9.

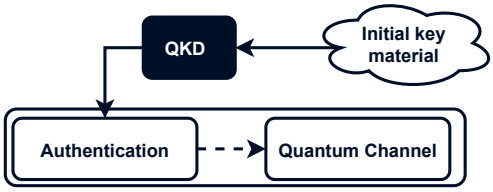


Figure 5.9: Using a dedicated QKD module with its link to supply key material. We treat it as a black box here; once it has been supplied with some initial key material it should continue working indefinitely.

We consider the rate at which we may perform entanglement attempts (r_{attempt}) at the physical layer using the solution by Walenta et al. [105]. We consider 32-bit tags, which we concluded previously is still usable while also being more favourable towards QKD:

[VIII] This is already an unrealistic assumption. In the case of BB84, on average half of the produced key material is thrown away, and privacy amplification further reduces the key size.

Example 5.6.2. Say we use the solution by Walenta et al., which produces up to 61.2 kbit/s[IX] of key material at a distance of 12.5 km [105, tab. 1]. If Alice and Bob each have a QKD link with the midpoint station, each of 12.5km, use tags of $\log|T_1| = 32$, the rate at which they may perform entanglement attempt, following from eq. 5.12, is as follows:

$$r_{\text{attempt}}^{\text{upper}} = \frac{r_{k_1}}{2\log|T_1|} = \frac{61\,200}{2\times 32} = 956.250 \quad \text{[Hz]} \quad (5.34)$$

Therefore, they may attempt to entangle at approximately 0.956 kbit/s. (The actual rate at which pairs are produced then depends on p_{succ}.) For the QL2020 setup[X] described in [38], where Alice is 10 km away from the midpoint and Bob 15 km, the attempt rate is $r_{\text{attempt}} = 6897$ Hz ($1/r_{\text{attempt}} = 10.120\,\mu$s) for MD type requests. Clearly, this QKD links would form a significant bottleneck. If we wish to operate the QL2020 setup as described in [38] without any problem, we would require the following amount of key material

$$r_{k_1} = 2\times 6897\times 56 = 772\,464 \quad \text{[bit/s]} \quad (5.35)$$

Meaning that we would require a solution which can generate approx. 772 kbit/s over at least 15 km if the QL2020 setup from [38] to remain unimpeded (for MD type requests).

We demonstrate some real-world implementations of QKD implementations in table 5.2 as well as the resulting upper bound on entanglement generation attempts in a similar manner. We see that while [105] does form a bottleneck, the solution by [54] does not. This is over a distance of 12.5km, assuming that the internal data origin authentication mechanism would consume key material at roughly the same rate as in [105]. In other words, we assume that the rate at which *authenticated key material* is produced would be similarly high in such a case. This showcases the key rates which have been achieved in real-life settings, representing the start-of-the-art as far as we can tell. These values then directly translate to the r_{attempt} rate from above provided one using ITS data origin authentication of all control messages.

5.6.3. Implications

We find that it is possible to use QKD to supply key material for information-theoretic security data origin authentication of control messages, provided one uses relatively short (16-bit) tags. In the case of the QL2020 setup, for MD requests (which have the higher request rate), the key rate needs to be a minimum of approximately 774 kbit/s in such a case, assuming 32-bit tags for control messages. They required key rate scales directly with the tag size. I.e. Using 32-bit tags requires 440 kbit/s, 64-bit 880 kbit/s, and so forth. In section 5.5, we discuss in greater detail the implications of tag size in an information theoretic setting.

Alternatively, one may use a hybrid approach, where QKD is used to intermittently refresh a fixed amount of key material. This is motivated in part by the remark made in [105], which state that QKD may be used to enhance the security of the computationally secure solution. Specifically, they propose using QKD to refresh the key material used by AES [100] (a computationally secure scheme) every so often.

5.6.4. Is ITS Data Origin Authentication Necessary?

As we investigate the feasibility of using information-theoretic security data origin authentication methods, it is worthwhile to ask why in the first place. Generally speaking, the main

[IX] Some key material is used to perform data origin authentication of QKD's internal classical communication. They require 127 bits of key material per 2^{20} ($\approx 10^6$) bits of classical communication. On a fibre of 25 km, accounting for all corrections, roughly 412 bits of classical communication are required per bit of key material produced. This means that the implementation consumes $\approx 5.0\%$ of the produced key material (or $\approx 2.7\%$ when the fibre is 1 km), leaving the remaining 95% as output by the protocol. The remaining key rate, which they refer to as *authenticated key rate*, is 61.2 kbit/s over 12.5km. This is the key rate we use.

[X] A quantum link between two hypothetical cities within Europe separated by 25 km.

Table 5.2: Examples of QKD rates of actual implementations. All of these are decoy-QKD protocols. We also make note of the distance over which these rates were achieved. This is by no means an exhaustive list. It is simply illustrative of what rates are achievable over distances that would be required by the QL2020 setup. We also show the resulting $r^{\text{upper}}_{\text{attempt}}$ and $r^{\text{upper}}_{\text{data}_1}$ following these key rates. Recall that in [38], $r_{\text{attempt}} = 6897\,\text{Hz}$ following the QL2020 setup. Any solution whose $r^{\text{upper}}_{\text{attempt}}$ is higher than this value should not affect the link.

					$r^{\text{upper}}_{\text{data}_1}$ [bit/s]	
Authors	**Date**	**Rate**	**Distance**	$r^{\text{upper}}_{\text{attempt}}$ [a][Hz]	p_{succ} [b] $= 10^{-4}$	p_{succ} [c] $= 1.88 \times 10^{-3}$
Dixon et al. [40]	2008	1.02 Mbit/s	20 km	15 937.500	1.594	29.963
Dixon et al. [40]	2008	10.1 kbit/s	100 km	157.813	0.016	0.297
Walenta et al. [105][e]	2014	61.2 kbit/s	12.5 km	957.031	0.096	1.799
Walenta et al. [105][e]	2014	21.4 kbit/s	25 km	334.375	0.033	0.629
Huang et al. [54]	2015	≈3.1 Mbit/s[f]	25 km	48 437.500	4.844	91.063
Huang et al. [54]	2015	1 Mbit/s	12.5 km	15 625.000	1.563	29.375
Korzh et al. [63]	2015	12.3 kbit/s[d]	100 km	192.188	0.019	0.361
Zhang et al. [112]	2018	1 Gbit/s	-	15 625 000.000	1562.500	29 375

[a] $\log|T_1| = 32$.

[b] $p_{\text{succ}} = \alpha \times 10^{-3} = 10^{-4}$ for $\alpha = 0.1$. This is the bright state population (α) when $F_{\text{min}} \approx 0.8$ [38, fig. 8].

[c] We also pick $p_{\text{succ}} = 1.88 \times 10^{-3}$ as this does more closely reflect the results of MD type requests along the link when $F_{min} = 0.8$. That is, the rate of pair production is approx. $r_{\text{data}_1} = 13$ pairs/s [38, fig. 6]. Assuming $r_{\text{attempt}} = 6897\,\text{Hz}$ for MD type request; $p_{\text{succ}} = r_{\text{data}_1} / r_{\text{attempt}} = 13/6897 = 1.88 \times 10^{-3}$

[d] Approximated [63, fig. 2b].

[e] Walenta et al. mention explicitly that this is the key rate accounting for key consumption by the internal (ITS) data origin authentication mechanism [105]. They refer to this as *authenticated key rate*.

[f] Approximated [54, fig. 4].

advantage of information-theoretic security authentication methods is that they are not susceptible to advances in hardware and algorithmic theory. This would mean that such schemes are, in theory, future-proof. A scheme that may be intractable[XI] for classical adversaries may not be so for adversaries which may execute quantum algorithms or react quantumly with the scheme. A general framework has been proposed to analyze whether a MAC is existentially unforgeable against a quantum chosen message attack [27]. They distinguish two situations: An adversary may either only interact with the environment classically and run quantum algorithms locally on the obtained data, or they may interact with the environment quantumly. The data origin authentication solution, in our use case, is expected to only run on the classical systems. The general consensus appears to be that data origin authentication schemes appear unaffected by adversaries with quantum computing resources[XII] [15]. Certain solutions are still affected if an adversary may submit queries that are in superposition, meaning that they may interact quantumly, certain authentication solutions are no longer secure [111]. However, we assume that no such queries may be made in the system under investigation.

Symmetric encryption schemes do see a reduction in security, but it is not so severe: Their security level is (only) halved. This is easily rectified by simply doubling the key size [15], as symmetric schemes can only be brute-forced using a generic search (generally speaking), for which Grover's algorithm [49] provides (only) a quadratic speedup. This is in contrast to Shor's algorithm [96], which provides a exponential speedup for number factorisation, breaking RSA, a popular public asymmetric cryptographic scheme.

[XI] Meaning that it is computationally infeasible for an adversary to break the scheme. See definition 4.2.6.

[XII] Bernstein and Lange give GMAC [73] and Poly1305 [14] as examples of MACs which are not affected at all by adversaries with (only) quantum computing resources [15].

On a more fundamental level, is the level of security provided by information-theoretic security data origin authentication warranted at this level? Recall that the link itself, included the physical layer, makes no actual security guarantees about the data itself. We only, at this level, care about the availability of the link. If sessions along a link are sufficiently short, a computationally secure scheme may prove sufficiently secure. This is not to say that there is no reason to use an information theoretically secure scheme. However, these schemes require a lot of key material to function as we shall see later. Therefore, these may not always be a viable option if it is say not possible to produce enough key material, or the production of key material would itself become a significant bottleneck.

5.7. Computationally Secure Authentication

In some situations, it may not be possible to supply the key material required by information-theoretic security authentication (of all control messages). For instance, the QKD module may be defective. This, combined with the fact that information-theoretic security data origin authentication may not be necessary (sec. 5.6.4), means that a computationally secure solution might be a more viable alternative. These in turn would also allow larger tag sizes without consuming any extra key material.

In such situations, we require a data origin authentication mechanism that requires only a small or fixed, amount of key material. This is where computationally secure schemes come into play. These have also seen widespread use for this very reason. Potential methods for session key establishment would include using keys shared via an external channel, key handshakes (potentially using post-quantum cryptographic methods), or QKD, to name a few. An analysis of the different methods of session key establishment and what would be most suitable is left out of scope, however. Computationally secure schemes offer a more practical alternative at the cost of reduced security level, being more susceptible to:

1. Advances in hardware, such as advances in quantum hardware used by adversaries.
2. Advances in algorithmic theory.

When using a computationally secure scheme, an adversary has enough information to derive the key (and/or forge tags). Such a scheme is considered secure if *using the best-known method* it would take an adversary a considerable amount of time to break said scheme, however. This is also why it is in some regards difficult to compare the security of different computationally secure schemes, as mentioned in [42]. For this reason, we take the security of these schemes as-is. We shall go briefly over some more concrete schemes in the next chapter, only showcasing the general structures we identified.

The Wegman-Carter-Shoup Construct In the case of Wegman-Carter style MACs, more modern proposals use a PRF instead of a non-repeating key sequence B (in fig. 5.2). The Wegman Carter Shoup construct instead combines the output of the universal family with a block cipher which takes a publicly known, but non-repeating, nonce N as input, depicted in figure 5.10.

$$\mathrm{WCS}[\mathrm{PRF},\mathrm{F}]_{k_1,k_2}(m,n) = \mathrm{F}_{k_1}(m) \oplus \mathrm{PRF}_{k_2}(n) \tag{5.36}$$

The main advantage of such a scheme is that the key size does not scale with the amount of (control) messages sent. Alice and Bob need to only establish a fixed amount of key material before commencing with communication. The security of the scheme does gradually degrade with the use of this session key (or keys). But it does not catastrophically break down when reusing the same key, unlike its ITS counterpart. The security of such a scheme is almost wholly on the strength of PRF in use. It is the only object that hides the output of $f_k(M)$. The author of Poly1305-AES (a Wegman-Carter-Shoup style MAC), Bernstein, mention this explicitly [14]:

> *If anything does go wrong with AES, users can switch from Poly1305-AES to Poly1305-AnotherFunction, with an identical security guarantee. All the effort invested in the*

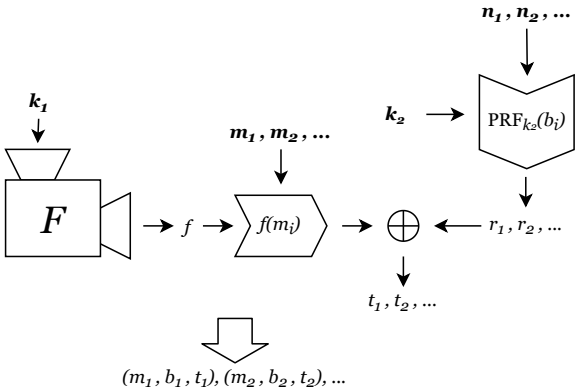


Figure 5.10: The Wegman-Carter-Shoup [97] construct. The second key (sequence), instead of $b_1, b_2, ...$, is now fixed as k_2, meaning that the amount of key material required is now fixed. In turn, the sequence $n_1, n_2, ...$ is no longer required to be random (i.e., we may use $n_i = n_{i-1} + 1$): The PRF supplies the randomness. Care must be taken to never reuse a n_i. All this does, however, mean that the scheme is no longer ITS, but computationally secure instead.

> *non-AES part of Poly1305- AES can be reused; the non-AES part of Poly1305-AES cannot be broken.*

WMAC construct A somewhat similar construct is the WMAC construct, which also makes use of a PRF. However, the PRF takes as input the output of the universal family appended with the nonce directly:

$$\text{WMAC}[\text{PRF}, \text{F}]_{k_1,k_2}(m, n) = \text{PRF}_{k_2}(\text{F}_{k_1}(m)||n) \tag{5.37}$$

MACs such as UMAC (1999) and MAC611 ([42]) have such a structure. UMAC is the first MAC with such a structure, as far as we were able to identify. It is referred to as "WMAC" in the security analysis in [26]. It is somewhat similar to the Wegman-Carter-Shoup construct. However, instead of XORing the output of the PRF with the output of the family, the output of the family is passed directly to the PRF with a nonce appended. One disadvantage of such a system is that the universal family and PRF cannot be evaluated in parallel, however.

Keyed PRFs A keyed Pseudorandom Function (PRF) may also function as a MAC. In such constructions, we may pass the nonce n as such:

$$\text{PRF}_k(n||m) \quad \text{or} \quad \text{PRF}_k(m||n) \tag{5.38}$$

As an adversary should not be able to distinguish a PRF from a truly random function "efficiently" (in polynomial time) [99, Security game, figure 11.3], this is a valid way to use these PRFs. Provided, of course, that the PRF is not susceptible to extension attacks. This includes Merkel-Damgård based hash functions, such as SHA-2.

We remark that a universal family is not a PRF in and of itself. By definition, for a PRF f_k, if an adversary has knowledge of $f_k(m_1), f_k(m_2), ..., f_k(m_p)$, they should not be able to guess the tag for $f_k(m_{p+1})$ in polynomial time. From this it follows trivially then that provided the PRF f_k is secure, revealing multiple message tag pairs at most leaks a negligible amount of information about k. In the case of universal families, however, even a single repeated use opens up the door to forgery by an adversary if the result is left unencrypted as was shown in section 5.3.

5.8. Summary RQ1

We have shown that data origin authentication is required to ensure a reliable quantum link. It ensures that the link remains operational by preventing fraudulent (error) messages from being

processed. Additionally, this has the potential to prevent cross-process interference and the destruction of quantum memory due to fraudulent requests for entanglement.

As for encryption, we have shown this to be of secondary importance. While this does not protect the quantum data in any significant manner (as far as we can identify), it may be of use in high-security settings where meta-data may be also classified as sensitive data.

RQ 1 | Why, And How, Should Control Messages Be Authenticated?

RQ 1.1 | Why is data origin authentication necessary?

Authentication is necessary to uphold the availability of the network. Without authentication, an adversary can destroy quantum memory through forgery of control messages. Additionally, it is also possible to forge error messages or other types of messages which may interfere with the inner workings of system-level protocols. Therefore, to maintain the availability of the network, an adversary must not be able to forge control messages. They must also not be able to resend old control messages as part of a replay attack, as this would still allow them to destroy quantum memory due to repeated requests for entanglement.

RQ 1.2 | When would encryption be necessary?

Encryption is not necessary from the perspective of confidentiality of quantum information, nor does it on its own prevent the destruction of quantum memory. However, we conjecture that encryption does have a use case if we wish to hide meta-information regarding entanglement. Such meta-information would for instance be which parties are communicating or information about the internal state of the protocols. We do not investigate the security implications of leaking this information to an adversary, however.

RQ 1.3 | Can a scheme which uses information-theoretic real-time data origin authentication be self-sustaining?

The most straightforward way to achieve this is through the use of QKD. While it's not possible to run this on the stack itself in a self-sustaining manner, a QKD box does make it possible. The rate maximum rate at which we are allowed to transmit qubits, particularly the accompanying control messages, is then determined by the rate this box produces key material and the tag size we use. We have presented multiple derivations which allow one to calculate the minimum key rates, or total required amount of key material, given link requirements such as attempt rate or the total amount of pairs required.
We deemed that a tag of 32-bits provides sufficient security for a quantum link. This does require that once in a while (412 seconds) a small fixed amount of key material is used to pick a new hash function with a strongly universal family. Within the QL2020 setup, use of tags of 32 bits for each individual control message would require that QKD can generate approximately 772 kbit/s of key material over a distance of 15 km.

CHAPTER 6

AUTHENTICATION PERFORMANCE: STATE-OF-THE-ART

We motivated previously that computationally secure data origin authentication solutions are sufficiently secure for direct links. We now aim to investigate how they influence the link's performance: Any data origin authentication introduces some latency along the link.

We investigate the fastest "mainstream" data origin authentication solutions. Their estimated performance, in terms of evaluation times with regards to input size (control message size), will then be used as input for our model. We also investigate some further optimizations in the field which aim to minimize the amount of transmission overhead of an authentication solution. This is particularly useful if the bit rate of the classical channel is low.

It should be noted, however, that the performance indicators of MAC evaluation times should be taken with a grain of salt. While they give a general indication of what we should expect in terms of performance, it is difficult to compare MAC solutions across different platforms due to the many factors which influence performance [46]. Seeing as new authentication solutions are constantly being developed (the most recent one under investigation being published in 2019 [42]), it is not possible to produce an exhaustive list that will remain relevant indefinitely. We can, however, produce a range of performance indicators for use in our simulation. Using this range should produce a model which gives a general overview of the impact of data-origin authentication solutions on the link. Furthermore, these solutions appear much slower than information-theoretic solutions by a considerable margin[I] [50]. Therefore, if we show that data origin authentication using fast computationally secure solutions has a marginal effect on link performance, we argue that using information-theoretically secure solution would have a similar or smaller effect on throughput. This does assume, of course, that the mechanism used to supply a continuous stream of fresh key material does itself not become a bottleneck (including but not limited to moving large amounts of key material to and from memory).

We do not investigate the security of multiple MAC solutions or their implementations (which themselves may again have security flaws, e.g. in the form of side-channels). This is due to the difficult of comparing the security of different MAC solutions as the security of computationally secure solutions is usually not easily reducible to a single value. Therefore, we take the security of all solutions at face value: We assume they are secure due to widespread use and the fact that no serious flaws have been found in literature as far as we were able to identify.

[I] As of 2008, Wegman-Carter style information-theoretically secure MACs have shown to have a throughput 15 times greater than CBC-MAC with AES and HMAC with SHA-1 [50].

6.1. Overview

We first perform a survey of commonly used MAC solutions. We explain briefly our method of selection in section 6.2. They are then grouped as follows:

1. We first cover MACs which are inspired by the Wegman-Carter construct in section 6.3.
2. Next, we cover MACs which are built using (unkeyed) hash functions in section 6.4.
3. Next, we cover MACs which are built using block ciphers in section 6.5.
4. Last, we cover custom (meaning, not derived or based heavily on any pre-defined structure) keyed Pseudorandom Functions (PRFs) in section 6.6.

We also analyse some optimizations to limit the number of transmitted bits in section 6.7. This is crucial, as transmission time quickly overtakes computation time if sufficiently many bits are transmitted (which we shall further investigate in chap. 8). We give an overview of the expected performance of the MAC solutions in section 6.8. The performance numbers in section 6.8 will also form the basis of our experimental setup in later chapters.

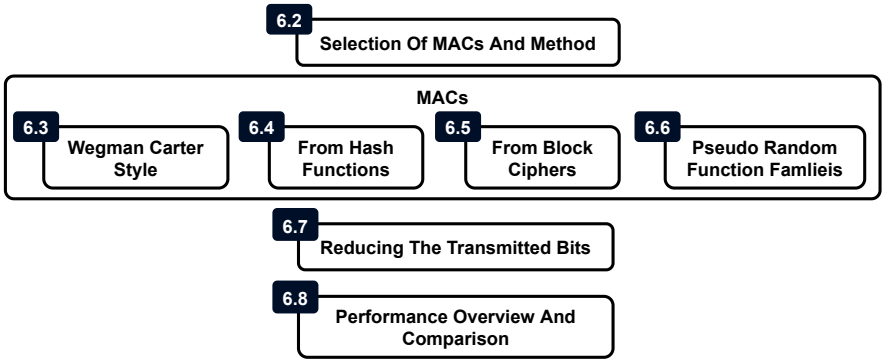


Figure 6.1: Overview of this chapter.

6.2. Selection Of MACs Under Evaluation And Method

Table 6.1 gives an overview of all MAC solutions under evaluation. It is difficult to make a definitive selection of data origin authentication solutions. There exist many alternatives, as well as implementations, from which one may pick. Generally, we focus on:

1. Solutions which are have seen somewhat widespread use. We operate under the assumption that speed is a common concern: If a proven faster solution is available, we would expect that the community over time would migrate to such a solution where possible.
2. Solutions which fall under the umbrella term of "lightweight cryptography". There are very few MAC solutions which fall specifically under the umbrella of "lightweight cryptography" as pointed out in a 2017 survey [22]. In fact, they only mention SipHash [8] and Chaskey [76]. This might also follow from the fact that while there are many block ciphers and hash functions, there are relatively few dedicated MAC algorithm proposals [86]. However, as MACs are often derived from other primitives, we do investigate a lightweight block cipher called LEA [53], as we did encounter it in a speed benchmark ([95]) during our review.

The above general criteria led to our final selection, shown in table 6.1. We also showcase the minimum message length requirements, key lengths, and nonce lengths. Certain MAC solutions do not specify a nonce. For instance, SipHash does this by design so that a user may opt to not maintain any state at all [8]. In such situations, it is valid to simply append the nonce to the message before passing this to the MAC itself. For instance, UMAC [23] performs this operation

internally does this by appending a nonce to the output of a universal family before passing this to HMAC, a MAC which does not specify explicitly a nonce.

Table 6.1: MAC's under investigation. All solutions are computationally secure. A minimum length of 0 means no minimum was explicitly specified. However, it may still be that the MAC internally still pads the message so that it is some minimum length before proceeding.

		Length (bits)			
Type	**Name**	**Min. Message**[a]	**Key**	**Nonce**	**Tag**
Hash function based	HMAC [10]	(PRF)[b]	(PRF)[b]	-	(PRF)[b]
Hash function	KMACK-128 [19][g]	0	0	-	128
Hash function	KMACK-256 [19][g]	0	0	-	256
ARX	SipHash-64 [8][h]	0	128	-	64
ARX	Chaskey [76]	0	64	-	64[e]
Block cipher based	CBC-MAC	(PRF)[b]	(PRF)[d,i]	(PRF)[b]	(PRF)[b]
Block cipher based	CMAC [56]	(PRF)[b]	(PRF)[b]	(PRF)[b]	(PRF)[b]
Block cipher based	GMAC [43]	(PRF)[b]	(PRF)[b]	(PRF)[b]	(PRF)[b]
WMAC	UMAC [23]	32	512	0[d]	(PRF)[b]
WMAC	MAC611 [42]	56	128 (Noekeon [36])	64	64
Wegman-Carter-Shoup	VMAC-AES [66]	0	128 + 2×(PRF)[b]	(PRF)[b]	(PRF)[b]
Wegman-Carter-Shoup	Poly1305-AES [14]	0	256	128	128
Universal Hash Family	CLHASH [69]	64	8192[c]	318	64

[a] If a message happens to be shorter it will simply be padded until it is the minimum required length.
[b] Amount of bits depends on input or output of underlying PRF (cryptographic primitive) in use.
[c] Similar to UMAC, it should be possible to pass a smaller 512-bit key to a Psuedo Random Number Generator (PRNG) to generate the 128×64-bit $(k_1, ..., k_{128})$ of key material required by the CLNH family.
[d] No minimum specified, but should be large enough to be non-repeating during a session.
[e] Minimum of 64 bits is recommended, though other tag size may be specified. Additionally, Chaskey authors mention that it is not vulnerable to tag truncation.
[f] The nonce must be encrypted (or any one-way cryptographic function) before given as input.
[g] KECCAK family [18].
[h] There exists a 128-bit tag version called SipHash-128 [7].

This non-exhaustive list under instigation is meant to be indicative of the performance of the state-of-the-art. Our goal here is to not determine the fastest available solution. That would require a more extensive investigation, and would in all likelihood be an ongoing investigation.

6.3. Wegman-Carter Style solutions

Like before, we first delve into Wegman-Carter style MACs. These are solutions that make use of a strongly universal hash family. However, to authenticate many messages, these solutions do not make use of a One Time Pad (OTP), but instead make use of a PRF. Either the output, combined with a nonce, is passed to a PRF (WMAC), or the output of the family is passed to a PRF with a nonce as input (Wegman-Carter-Shoup).

6.3.1. WMAC

The first variant is the WMAC, originally the UMAC construct [23]. This term was first used in [25] in their analysis of UMAC according to [42]. This construct passes the output of the universal family F (with as input m), appended with nonce n, to a PRF:

$$\text{WMAC}[\text{PRF}, \text{F}]_{k_1,k_2}(m,n) = \text{PRF}_{k_2}(\text{F}_{k_1}(m)||n)$$

The security now depends in large part on the presumed difficulty of breaking the PRF.

UMAC (1999) UMAC was developed by Black et al. in 1999. They claim that it is roughly an order of magnitude faster than HMAC-SHA1 [23], a popular MAC at the time. The "U" stands for the fact that it uses a strongly universal hash family, called NH. This strongly universal family forms part of UHASH, which is a hash function whose output is then passed to an encryption function.

This output is then given as input to a PRF, in addition to a 64-bit nonce n. It is the responsibility of the sender and the receiver to ensure that a nonce is never used twice. The authors use HMAC-SHA1 as the PRF of choice:

$$t = \text{HMAC-SHA1}_A(n||\text{UHASH}_k(m))$$

One might remark why we do not use HMAC directly as such: $t = \text{HMAC}_k(n||m)$. The main reason for not doing so, the authors state, is that HMAC is inefficient for long messages, whereas the universal family NH is.

UMAC (version 2, 2000) Black et al. do make an amendment one year later in 2000 to the above by introducing an extra stage between the compression by NH and running the function through a PRF [24]. This extra stage reduces the reduced variable-length message to a fixed-length hash, which is then passed to the PRF. This (1) makes the security independent of the length of the original message and (2) minimizes further the use of cryptography.

MAC611 (2019) Duval and Leurent build a MAC specifically for micro-controllers [42], making it a lightweight MAC solution. As for HMAC, they remark that lightweight hash functions may be used in conjunction with HMAC, but that a dedicated MAC is typically more efficient due to the larger internal state of hash functions. This is a sentiment that the authors of SipHash share [8], which is one such dedicated MAC.

6.3.2. Wegman-Carter-Shoup MACs

Like the WMAC, the Wegman-Carter-Shoup construct uses PRF instead of a OTP [98]. However, instead of passing the output of the universal family appended with a nonce to the cryptographic object (PRF), the Wegman-Carter-Shoup construct XORs the output of the universal family with the output of the PRF with the nonce as input:

$$\text{WCS}[\text{PRF}, \text{F}]_{k_1,k_2}(m, n) = \text{F}_{k_1}(m) \oplus \text{PRF}_{k_2}(n)$$

The Wegman-Carter-Shoup construct was previously illustrated in figure 5.10. An advantage of this construct is that the PRF may be evaluated in parallel with the hash function. Alternatively, one may use the + (not to be confused with $\oplus$) operator combined with modular arithmetic. As far as we can tell, this also falls under the definition of Wegman-Carter-Shoup.

VHASH And VMAC-AES (2007) Krovetz developed VMAC in 2007 [66]. While similar to UMAC, which was the fastest reported MAC at the time, VMAC is optimized for 64-bit architectures as opposed to 32-bit architectures: It makes use of newer instruction sets found on 64-bit architectures. The internal hash is referred to as VHASH, and the output is passed to AES instead of HMAC-SHA1:

$$t = \text{AES}_{k_2}(n||\text{VHASH}_{k_1}(m)) \tag{6.1}$$

The authors make note that VMAC is patent- and copy-right free, and like UMAC, parallelizable. Though parallelizability would likely not offer any significant gains for very short messages we should note.

Poly1305-AES (2005) Bernstein create a Wegman-Carter-Shoup style MAC which unlike UMAC and VMAC, has no key expansion. It requires a 128-bit of key material k for AES, and a further 128-bit of key material, referred to as r.

Poly1305 requires 128-bit nonces. The authors also state that AES may be exchanged with any keyed function which maps nonces to 16-byte strings [14]. Lastly, Poly1305-AES is also used in combination with the ChaCha20 cipher [17] in the TLS versions 1.2 [68] and 1.3.

CLHASH (2016) Lemire and Kaser build a universal 64-bit hash family called CLASH which is similar to the VHASH. Internally, it makes use of a family called CLNH, which is similar in structure to the aforementioned NH family. CLHASH appears to be 60% faster than VHASH, achieved in part by exploiting the 64-bit carry-less multiplication (CMUL) instruction set in x64 processors of both Intel and AMD. This is an illustrative example of how solutions may be optimized concerning their host architecture. Carry-less multiplication is also referred to as XOR multiplication. The authors go on to note that VHASH is designed mostly for speed on long messages, whereas CLHASH also shows favourable performance on short messages.

6.4. From Hash Functions

A MAC solution may be built using cryptographic hash functions. This might prove useful if for instance they have already been implemented in hardware (in a cryptographic module), promising great speed.

HMAC and NMAC (1996) Mechanisms to guarantee the authenticity and integrity of messages have been around for some time. However, as Bellare et al. pointed out in 1996, a lot of these solutions were ad-hoc [10]. Therefore, they provided the HMAC and NMAC constructs, which prevent an adversary from learning the state of a cryptographic hash function to produce tags for messages through clever padding techniques.

Definition 6.4.1 (Length Extension Attack)**.** A PRF is susceptible to a length extension attack if knowing $H(k||m_1)$ and the length $|m_1|$ an attacker is able to for a chosen message m_2 construct $H(k||m_1||m_2)$ without knowing k.

For such constructs, the performance depends heavily on the underlying hash function used. The HMAC construct is as follows, which turns an unkeyed underlying hash function h (such as SHA-1) into a keyed MAC (or keyed PRF):

$$\text{HMAC}_k(m) = h((k \oplus \text{opad})||h((k \oplus \text{ipad})||m))$$

opad is formed by repeating the `0x36` byte, and ipad by repeating the `0x5c` byte. $k\oplus$opad and $k\oplus$ipad must the length of a block size of the hash function H. The tag size is determined by the output of the hash function H. The underlying Hash functions could for instance be SHA-2256. To use a nonce, we could do something akin to the WMAC constructs by simply appending it before giving it as input the PRF:

$$\text{HMAC}_k(m, n) = h((k \oplus \text{opad})||H((k \oplus \text{ipad})||m||n))$$

SHA-3 And The Keccak Family (2016) One might wonder if the HMAC construct is of any use if the underlying hash function is not susceptible to length extension attacks. SHA-3, a member of the Keccak family of cryptographic primitives and winner of the 2012 SHA-3 competition [19], is not susceptible to such attacks. This is because it is a sponge-like construction, as opposed to a Merkel-Damgård like construction as is the case with SHA-1 and SHA-2.

A 2016 NIST publication specifies how one may produce such a a MAC derived from the Keccak family: Keccak MAC (KMAC) [60]. It makes use of SHAKE128 or SHAKE256 from the

Keccak family, which are supposed to be faster than SHA-3. There are multiple functions in the family, such as the KangarooTwelve hash [20], another function within the Keccak family that promises better performance for short messages. However, we are unable to identify benchmarks that verify the performance on (many) short messages. It is therefore not possible to say for certain whether SHA-3 is suited.

6.5. From Block Ciphers

An alternative approach is to build a MAC from block ciphers instead of hash functions. Using one or the other may be preferable depending on the platform on which we operate or which primitives have been implemented, for instance in hardware. We find no proof, however, that one is faster than the other in a most general sense. It should be noted that multiple block ciphers under investigation, such as Poly1305-AES and VMAC make use of AES, a block cipher, as a PRF. However, we don't consider this to be

CBC-MAC A CBC-MAC allows authentication by running a message through a block cipher. In such an implementation, the message is first split into blocks. The first block is XORed with the initialization vector (IV) before running through a keyed cipher E_k. This output is then XORed with the second message block before running through E_k, and so forth until the last message block. The structure is illustrated in figure 6.2.

It should be noted that the initialization vector must be fixed (typically set to **0**), else it can be abused by attackers. One could still use the initialization vector as a nonce, however, if it is encrypted each time before use.

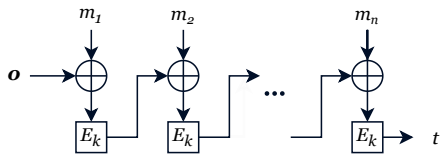


Figure 6.2: Illustration of the cipher block chaining MAC (CBC-MAC) technique for constructing a MAC. The input is a single message $m = m_1||m_2||...||m_n$, which is then split up into equal sized blocks $m_1, m_2, ..., m_n$. The block cipher E_k may be any keyed block cipher. It is a block cipher, and not a hash, because it (1) operates on a fixed-length amount of bits and (2) the size of the input and output are equal (excluding the key itself).

One problem with this construct is that it is only secure for fixed-length messages. One option is to encrypt the last block a second time using a second key, in a mode of operation called ECBC-MAC (Encrypt-last-block CBC-MAC). Another is to prepend the length of the message, which in our case is easy to do like the length of each message is known in advance. Another option is to use CMAC.

CMAC (OMAC1) CMAC was published in 2005 by NIST, though it has been superseded by the SP 800-38B 2016 publication [44]. It is equivalent to the OMAC1 (One-key CBC-MAC) construct from [56]. It is called a One-key CBC-MAC (OMAC) construct as it requires only one key. It addresses the aforementioned security concerns of CBC-MAC such as tagging multiple-length messages, without requiring multiple keys as is the case with the ECBC-MAC construct.

GMAC (2007) GMAC also uses block ciphers to produce a tag but instead uses the Galois mode of operation. This allows block ciphers to be evaluated in parallel, promising higher throughput [43]. However, we wouldn't expect that the parallel nature of GMAC would provide any significant performance gains as MHP messages are at most 64 bits, which is less than the 128-bit block size of AES.

The message is split up into blocks before being fed to a universal family named GHASH, after which the output is encrypted by XORing the result with the original message. The IV must receive a nonce as input, which also means that GMAC is resistant to replay attacks.

LEA LEA is a block cipher (and not a MAC in and of itself) which is supposed to be even faster than AES. We elaborate on LEA as it was shown to be one of the fastest lightweight block ciphers in [95]. There are, of course, many other block ciphers built for constrained environments (e.g. SPECK [9]). LEA was developed by Hong et al. [53], and is a block cipher with 128-bit blocks and key lengths of 128, 192, and 256-bits (similar to AES). The authors of LEA claim that it is faster than Advanced Encryption Standard (AES) on ARM platforms, even when hardware-accelerated AES.

We have found LEA used in practice as LEA-CMAC. This uses the aforementioned CMAC construct but replaces the default AES block cipher with LEA. Such a solution was benchmarked in [95].

6.6. Pseudo Random Function Families

Certain solutions have been built from the ground up. This means that they are built specifically as MAC solutions, and do not make use of other cryptographic primitives, such as a universal family (e.g. VMAC), block ciphers (e.g. CMAC), or hash functions (e.g. HMAC).

SipHash (2012) SipHash was developed by Aumasson and Bernstein in 2012 [8]. It is a MAC heavily optimized for sending many short messages on a network. It may be regarded as a lightweight message authentication solution, it being present in the lightweight cryptography survey by Biryukov and Perrin [22].

SipHash is mentioned as being a viable alternative to UMAC and VMAC [8]. These, while fast they state, have a lot of overhead and are more suited for longer messages, HMAC they state also has the overhead of calling two cryptographic hash functions, which perform many additional computations such that they are collision resistant. Collision resistance would only be relevant if the function were unkeyed.

Chaskey (2014) Mouha et al. construct a permutation-based MAC called Chaskey for 32-bit microcontrollers [76]. It was inspired in part by SipHash but designed for 32-bit microcontrollers while also achieving 128-bit security.

6.7. Optimisation: Reducing The Transmitted Bits

Transmission time can easily exceed computation time. Therefore, we explore solutions that aim to minimize the number of transmitted bits. Figure 6.3 demonstrates for instance what the new MHP `GEN` packet would look like with a 128-bit tag and 128-bit nonce. A reduction in nonce and tag size could offer a significant speedup, proportionally speaking, for these relatively small payloads.

6.7.1. Reduction Of Transmitted Nonce

Lin and Sangiovanni-Vincentelli propose sending only the least significant bits of the nonce, which is then reassembled using the most significant of the nonce at the receiving end. The MAC itself still takes the full nonce as input, meaning that the amount of communication overhead is reduced without any loss of security[II].

The nonce is divided into a most significant bits section n^M, and least significant bits section n^L. It is stated that if the probability of a network fault is p_{fault}, then the probability that the

[II] This was originally meant as an optimization for the TESLA (Timed Efficient Stream Loss-tolerant Authentication) protocol [83].

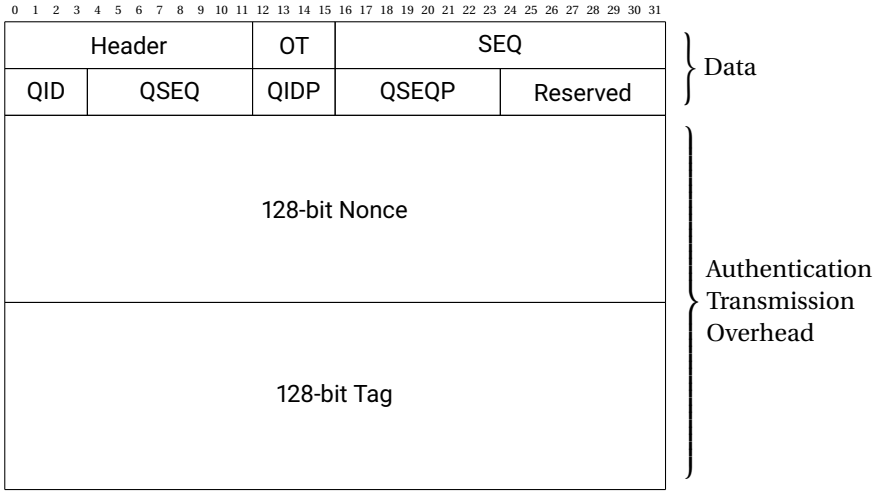


Figure 6.3: MHP `REPLY` packet format with tag and nonce to demonstrate increase in message size. This output would for instance be given by Poly1305, which produces 128-bit tags and uses 128-bit nonces. The nonce prevents replay attacks.

nonces will become out of synchronization is $p_{\text{fault}}^{2^{|n^L|}}$. Once out of synchronization, messages will start getting rejected, causing a link failure. The authors state that there is no loss of security.

Example 6.7.1 (Link Failure)**.** Consider the chance of a message being lost $p_{\text{fault}} = 0.01$ and a nonce of 128 bits. We transmit, however, only the $\left|n^L\right| = 16$ least significant bits. That would mean that the probability of the counters going out of synchronization being $0.1^{2^{16}} = 10^{-65536}$, an exceptionally small probability.

6.7.2. Cumulative MAC With Speculation

Li et al. note that Internet of Things (IoT) networks cannot use MACs in a straightforward manner due to the large outputs of MACs [70]. They also make a similar remark for Controller Area Networks (CANs): There is a need for security, but the format of CAN packets places significant size constraints on tags. A non-ideal solution is to simply truncate the output of the MAC at the cost of security. This assumes that the tag may be safely truncated, which for instance is not the case with GMAC as the creators of Chaskey note [76]. Solutions have been proposed to calculate MACS for multiple messages before being transmitted over several successive packets [81]:

CuMAC Nilsson et al. propose a solution called Cumulative Messages Authentication Code (CuMAC) [70]. The tag sent is a product of several segments of multiple generated MACs, thus reducing network traffic. The receiver then performs the same process during verification, thus reducing the number of tags that have to be sent while keeping the security level the same.

CuMAC/S If there is an accurate model for predicting future messages using the current and past messages, it is possible to calculate MACs of future messages. Note that speculation is deterministic, so both sender and receiver run this system without needing additional communication. They refer to this setup as CuMAC/S (CuMAC with Speculation)[III].

We expect that control messages, particularly at the physical layer in the quantum stack, are relatively easy to predict. This is also where such an optimisation would have the highest impact due to the relatively large amount of control messages exchanged at this level.

[III] The original authors predict future messages in their setup using an Autoregressive Integrated Moving Average (ARIMA) model and validate their solution in a prototype implementation.

6.8. Performance Overview and Comparison

It is difficult to predict how MACs perform in practice. Hardware, underlying architecture, compiler optimizations, and interaction with other systems all influence performance, to name a few. We investigate multiple independent benchmarks to uncover their performance in a somewhat "realistic" setting.

Benchmark by Shin et al. (2017) Shin et al. provide experimental performance analysis of several lightweight block ciphers and message authentication codes [95], benchmarking on three real-world devices: 8-bit AVR (Arduino Uno), 16-bit MSP (Tmote), and the 32-bit ARM (Raspberry Pi 2).

They consider CMAC [44] and HMAC [10]. For CMAC, they analyse the performance of three variants which make use of one the following three lightweight block ciphers: SIMON [9], SPECK [9], and LEA [53]. Like LEA, SPECK is also an ARX-based cipher. They also analyse the use of AES, which is not a lightweight block cipher. For HMACs, they opt to use SHA-256 [87] as the underlying hash function. The results are then obtained by measuring how many clock cycles were required for generating a tag for a 23-byte payload.

Benchmark by Rashwan et al. (2012) MACs have also been analyzed in the context of mobile computing by Rashwan et al. [88]. They additionally make the following observations:

1. The amount of cores involved influences performance. Most previous studies they state do not take into account the number of cores utilized by a function.
2. Extra-processor communication, for instance with an external cryptographic processor, additionally influences performance.
3. Implementation heterogeneity, meaning the same MAC might perform differently under the same architecture, depending on how the OS schedules execution, I/O, and bus delays.
4. Workload concurrency. Other processes, such as scheduling, memory, and I/O demands can affect the processing time of generating and verifying MACs as well.

Items 3 emphasizes that one should not expect that performance indicators are an exact representation of how a solution will function in practice. Items 1 and 4 would become more relevant as networks grow. Nodes may have to in those situations maintain multiple sessions, and whether the performance will remain (roughly) equivalent is not clear.

Benchmark by Czybik et al. (2013) Czybik et al. evaluate HMAC (Using SHA-256), CMAC and GMACs (both using AES-128) for a real-time 32-bit Ethernet of a field bus system [35]. Lastly, their benchmark runs in software. The authors do suggest that a hardware implementation would have provided a speedup.

The performance of several MACs for messages of several lengths in bytes is shown in table 6.2. We show performance in terms of cycles per byte required, not the total amount of cycles required to compute the tag for a given message size. Also note that the numbers are, as far as we can tell, based only on the message size and are irrespective of the nonce size.

6.9. Summary RQ2

When generating tags for short messages, all MACs under evaluation seem to have comparable performance (table 8.2). Even supposedly inefficient solutions for many short messages, such as VMAC, which performs a lot of internal key expansion, performs well when heavily optimized for its host architecture as in [69]. Therefore, all MACs under evaluation appear suited for at the very least a quantum link. The only exception perhaps is KMAC from the KECCAK family of hash functions, which is still relatively new and for which we do not find clear performance indicators for (many) short messages.

Table 6.2: Cycles per byte required to compute a MAC for different message sizes.

MAC	Architecture And Benchmark	Message Size (Bytes) 8	16	23	32	56	64
CLHASH [69]	Intel Haswell [69]	4.6[a]	1.4[a]	-	0.7[a]	0.45	
SipHash [8]	Intel Haswell, [69]	9.2[a]	6.5[a]	4[a]	3.1	-	-
SipHash-64 [8]	AMD FX-8150 “Zambezi”, [8]	15.50	8.38	5.34	3.66	-	-
Chaskey [76]	ARM Cortex-M3/M4, [76]	-	10.6	-	-	-	-
CMAC-AES-128 [44]	ARM8[b], [35]	-	3[a]	-	5[a]	-	8[a]
HMAC-SHA-256** [10]	ARM8[b], [35]	-	6[a]	-	6[a]	-	9.5[a]
GMAC [43]	ARM8[b], [35]	-	12[a]	-	145[a]	-	205[a]
VMAC-AES-128	AMD Opteron 2354, [88]	-	11.5[a]	-	9.5[a]	-	6.5[a]
VMAC-AES-128	TI DM3730 ARM Cortex A8, [88]	-	17.8[a]	-	15.7[a]	-	13.9[a]
LEA-CMAC [44, 53]	ARM Cortex-A7[b], [95]	-	-	1	-	-	-
SPECK-CMAC [9, 44]	ARM Cortex-A7[b], [95]	-	-	1.52	-	-	-
HMAC-SHA-256** [10]	ARM Cortext-A7[b], [95]	-	-	4.87	-	-	-
Chaskey [76]	Cortex-M4[b] (OpenSSL impl.), [42]	-	-	-	-	26.8	-
MAC611 [42]	Cortex-M4[b], [42]	-	-	-	-	78.6	-
CMAC-AES-128 [44]	ARM Cortex-M3/M4, [76]	-	-	-	-	-	89.4
VMAC-AES-128 [66]	AMD Opteron 2354, [88]	-	2700[a]	-	1448[a]	-	730[a]
UMAC-64 [23]	AMD Athlon 64 “Manchester”, [66]	-	-	-	5.6[a]	-	6[a]
Poly1305-AES [14]	AMD Athlon 64 “Manchestor”, [66]	-	-	-	7.5[a]	-	5.3[a]
UMAC-128 [23]	AMD Athlon 64 “Manchester”, [66]	-	-	-	4.75[a]	-	2.75[a]
VMAC [66]	AMD Athlon 64 “Manchester”, [66]	-	-	-	6.82[a]	-	4[a]

[a] Approximated from figure.
[b] 32-bit system.

RQ 2 | How fast are the fastest data origin authentication solutions?

We find that every solution under investigation, except perhaps KMAC (from the KECCAK family, which also contains the SHA-3 secure hash function), shows favourable performance for many short messages. For short messages (of 8 bytes), SipHash and CLHASH in particular shows favourable performance, requiring 4.6 and 9.2 cycles per byte respectively in the most favourable setups we find. This is most relevant for use as MHP messages are between 4 and 8 bytes in length and are sent most often within the current stack. Other notable solutions are LEA-CMAC and SPECK-CMAC, which require only 1 and 1.52 cycles per byte respectively for a 23-byte message, thus outperforming the aforementioned solutions.
We do identify some outliers regarding performance, such as a benchmark of VMAC-AES-128 which require 2700 cycles to process an input of 16 bytes. VMAC-AES-128, however, requires 17.8 cycles per byte for similarly sized inputs in a different benchmark. This range (4.6-17.8) we, therefore, consider being an approximate range of cycles per byte required to produce tags for MHP messages. Actual performance will however also depend on other factors such as host architecture and architecture-specific optimizations, for instance.

CHAPTER 7

SIMULATION SETUP I: EXISTING SETUP

To gain a deeper understanding of how the overhead of data origin authentication affects a quantum network, we simulate a single link with such overhead present. Such a link has already been simulated previously in [38], though without any data origin authentication or transmission delays. Transmission time[I] is not to be confused with propagation delay, which is the time it takes for the first bit (of a packet) to reach a destination and was simulated. Propagation delay was already modeled within [38]. In this chapter, we give a brief overview of their work. In the next chapter, we outline how we extended their simulation to include delays incurred by data origin authentication.

Quantum networks are complex in nature. There are many points of failure (swaps, emissions, detection, etc.) and we have to contend with a low probability of success. The combination of the above means that a simulation is a suited tool for gaining a deeper understanding of the behaviour of such networks. It forms a counterpart to the more analytical approach of observing the performance of such networks.

The quantum stack under investigation (QEGP, DQP, and MHP, Ch. 3), has been simulated previously using NetSquid [38]. NetSquid, a tool by Coopmans et al., is a discrete event simulator for quantum networks [33]. Such a tool allows network and protocol designers to simulate and test quantum networks. This tool, by extension, forms the basis of our simulation. We first provide a high-level description of the setup in [38] before describing our contributions to the simulation in the next chapter.

7.1. Overview

The hardware, and physics, underpinning the simulation are summarized in section 7.2. How heralded entanglement generation is simulated is covered in section 7.3. For a complete overview of the original simulation than presented in this chapter, we direct the reader to [38, Appendix C]. The setup under simulation consists of a single link, with two end-nodes and a single repeater station between them. The simulation setup is as follows:

1. **Quantum Processing Devices** Two end-nodes which consist of:
   (a) Memory, communication, and transmission qubits.

[I] Transmission time factors in the packet size and bit-rate of the channel to calculate how long it takes to transmit and read an *entire* packet placed on the link.

(b) One or two-qubit quantum gates for corrections.
(c) Trigger which allows entanglement between communication qubit and transmission qubit. To clarify, the communication bit physically remains at the end node, which is entangled with the communication qubit which is sent to the midpoint station. After entanglement, the communication qubits of the two end-nodes are entangled, without them having ever physically left the end-node stations.
(d) Measurement apparatus for qubits.

2. **Heralding Station** The midpoint station measures arrived transmission qubits. This midpoint station performs entanglement swapping so that the communication qubits of two end-nodes are entangled.
3. **Quantum Connections** A physical medium through which qubits are transported, in this case photons. In this setup, this is assumed to be a fibre connection. The propagation speed along the fibre is 206 753 km s^{-1}.
4. **Classical Connections** A physical medium through which classical messages are transported. In this setup, this is assumed to be a fibre connection (with propagation transmission speeds).

7.2. Qubits, Noise, and the NV Platform

The following contextualizes briefly the system which was simulated previously in [38].

NV Centres diamonds The setup uses Nitrogen-Vacancy (NV) centres in diamonds. This is formed by replacing a carbon atom in a diamond lattice with a nitrogen atom and removing a neighbouring carbon atom. The neighbouring carbon atoms are then used as storage qubits and the electrons as the communication qubits. Photons are used as transmission qubits.

Manipulating qubits take time and introduce some noise. The simulation parameters, mainly the operations, the time they take and their fidelity, are given in table 7.3. These are based on experimentally realized values. T_1 is the thermal relaxation time (decoherence due to the presence of heat), and T_2 the dephasing time (from [79, 7.2.1]). T_2 is also referred to as coherence time, which is the time during which the phase of the system (qubit) is predictable.

Definition 7.2.1 (Relaxation Time)**.** The time it takes for a system to go back to equilibrium. This is the time it takes for all information contained within a system to be irretrievably lost.

These two numbers encapsulate how long qubits may store state. As table 7.3 shows, the particles used have a lifetime in the range of a few milliseconds, illustrating the tight time window we are dealing with.

Table 7.1: Thermal relaxation and dephasing times of electrons and carbon atoms in the NV center in diamond setup. L_{AB} are experimentally realised values between two nodes in a lab spaced two metres apart. These are the parameters which are also used in simulation runs in [38]. Experimentally realised values have been achieved in isolation, but not within a complete system.

	L_{AB} **Value**	**Experimentally Realized**
Electron T_1	2.860 ms	1 h [2]
Electron T_2	1.000 ms	1.460 s [2]
Carbon T_1	∞	6 min [28]
Carbon T_2	3.500 ms	$\approx$ 10 ms [28]

Qubit initialisation Electrons are initialized through a process of optical pumping, where one shines a laser onto the electron. Each time, it may fall back onto a lower energy level with decreasing probability. This means that after many repetitions, with a high probability the electron will be in a high-energy state ($|0\rangle$). The times are shown in the first half of table 7.2

Table 7.2: L_{AB} initialisation and readout fidelities and times.

	L_{AB} Value		Experimentally Realized	
	Fidelity	**Duration**	**Fidelity**	**Duration**
Electron initialization in $\|0\rangle$	0.950	2 µs	0.990	2 µs [89]
Carbon initialization in $\|0\rangle$	0.950	310 µs	0.950	300 µs [34]
Electron readout of $\|0\rangle$	0.950	3.700 µs	0.950	3 µs - 10 µs [55]
Electron readout of $\|1\rangle$	0.995	3.700 µs	0.995	3 µs - 10 µs [55]

Quantum Gates Quantum gates, in addition to taking time to execute, also introduce noise. Noise in gates is modelled by first applying a perfect gate U, and then applying dephasing noise:

$$U_{\text{noisy}}(\rho) = \mathcal{N}^{f}_{\text{dephase}} \circ U_{\text{perfect}}(\rho) \tag{7.1}$$

$$\mathcal{N}^{p}_{\text{dephase}} : \rho \mapsto f\rho + (1-f)Z\rho Z \tag{7.2}$$

f is the gate fidelity as given in table 7.3. ρ is the state (qubit) to which we apply the gate, which is represented using the density matrix notation. State initialization also suffers from noise. All qubits are initialized by applying depolarization noise to the $|0\rangle$ state in the X, Y, and Z axis:

$$f|0\rangle\langle 0| + \frac{1-f}{3}\left[X\rho X + Y\rho Y + Z\rho Z\right] \tag{7.3}$$

Table 7.3: L_{AB} Gate fidelities and times.

	L_{AB} Value		Experimentally Realized	
	Fidelity	**Duration**	**Fidelity**	**Duration**
Electron single-qubit gate	1.000	5 ns	> 0.955	100 ns [59]
E-C Controlled-$\sqrt{X}$-gate	0.992	500 µs	0.992	500 µs - 1000 µs [59]
Carbon Rot-Z-gate	0.999	20 µs	1.000	20 µs [102]

Interfacing With Communication Qubits Swapping the state of the electron and the carbon atom is done using two E-C (Electron-Carbon) Controlled-$\sqrt{X}$-gates and single-qubit gates with a total time of 1054 µs. Within the E-C gate, the electron functions as the control qubit, and the carbon atom as the qubit upon which the gate acts (in the X-axis, see section 2.1.3).

Reading Data In the case of a measure directly request (MD), we measure the electron state directly instead of transferring its state to a carbon atom. If we wish to measure a memory qubit instead (following a CK request), an electron is initialized. Then, an E-C controlled gate and single-qubit gates are used to transfer the spin from the carbon to the electron, and the electron spin is measured.

Readout is modeled using a Positive Operator-Valued Measure (POVM) [79, 2.2.6]. f_0 is the readout fidelity of the $|0\rangle$ state, and f_1 that of the $|1\rangle$ state (see tab. 7.2 for f_1 and f_2 values).

$$M_0 = \begin{pmatrix} \sqrt{f_0} & 0 \\ 0 & \sqrt{1-f_1} \end{pmatrix}, \qquad M_1 = \begin{pmatrix} \sqrt{1-f_0} & 0 \\ 0 & \sqrt{f_1} \end{pmatrix}$$

7.3. Physical Entanglement Generation

The midpoint station measures the incoming qubits in the Bell basis. On either end, an electron is initialized in the following state:

$$\sqrt{\alpha}\,|0\rangle + \sqrt{1-\alpha}\,|1\rangle$$

This is the communication qubit. Note that we wish to initialize in the $|0\rangle$, which is referred to as the bright state. The bright state population can be chosen per entanglement attempt. A higher bright state population results in higher fidelity but does also mean that initialisation takes longer.

A laser pulse then causes an emission of a photon. This creates the following state at both Alice's and Bob's end, with a communication qubit C and emitted photon P:

$$\sqrt{\alpha}\,|0\rangle_C\,|1\rangle_P + \sqrt{1-\alpha}\,|1\rangle_C\,|0\rangle_P$$

The photons are transmitted to the heralding station. After measurement, the station emits a success signal if the communication qubits at A and B have been projected onto either the state $\left|\Psi^+\right\rangle = 1/\sqrt{2}(|0\rangle_A\,|1\rangle_B + |1\rangle_A\,|0\rangle_B)$ (only left detector clicks) or state $|\Psi^-\rangle = 1/\sqrt{2}(|0\rangle_A\,|1\rangle_B - |1\rangle_A\,|0\rangle_B)$ (only right detector clicks). If neither or both detectors click, failure is declared. The probability that a swap is successful is $p_{\text{succ}} \approx 2\alpha\, p_{\text{det}}$. Here, p_{det} the the probability of detecting an emitted photon.

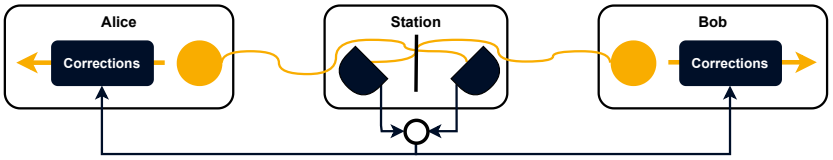


Figure 7.1: High-level depiction of heralding station. If left detector clicks, we are in state $^1/_{\sqrt{2}}(|01\rangle + |10\rangle)$. If right clicks, we are in state $^1/_{\sqrt{2}}(|01\rangle - |10\rangle)$. Else failure: it could be in more than one possible state.

See [38, App. E] for the list of noise processes modeled along the link. This includes both processes at the end-nodes, and at the mid-point station when it performs entanglement swaps.

CHAPTER 8

SIMULATION SETUP II: EXTENSIONS

We extend the simulations from [38] to understand how the overhead of an authenticated classical channel affects throughput. The extended simulation is illustrated in figure 8.1. As MHP control messages are sent most often, with every qubit being accompanied by two messages (and every pair by at least four), we expect that delays at the MHP level should affect link performance the most. We account for the transmission time of packets, which is affected both by their size and the bit rate of the channel. This also takes into account additional classical overhead induced by ethernet, the Internet Protocol (IP) (version 6), and the User Datagram Protocol (UDP).

As nodes should only perform operations after verification, MAC evaluation is a blocking operation. A range of evaluation times, taken from chapter 6, is used to evaluate their influence on link throughput. The actual evaluation time is influenced both by the speed of the MAC, and the size of the packet. The additional time to transmit the tags and nonces is also accounted for.

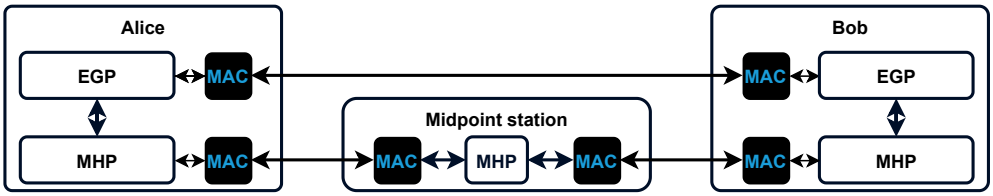


Figure 8.1: Data plane elements and our own extensions to the simulation. We additionally also model transmission time, influenced by the bit-rate of the channel and the amount of bits transmitted.

8.1. Overview

How we model transmission delays is elaborated upon in section 8.3.1. We explain how we calculate MAC delays in section 8.3.2, as depicted in figure 8.1. Note that we do not simulate any key distribution or evaluation of these functions. We consider only the delays in our simulation of evaluating these MACs. Key distribution, we assume, is done in advance and should therefore not influence throughput of a single link.

How we extend the model and insert these delays is covered in section 8.2. How we extend the actual codebase of the simulation in [38] is discussed in section 8.4. The final simulation

parameters are covered in section 8.5.

8.2. Extending The Model And Simulation Goals

Our goals when extending the model are as follows:

1. Add transmission delays, which are affected by the packet sizes and classical channel bit rate.
2. Add data origin authentication delays to QEGP and MHP (figure 8.1).

To accomplish the above, we first add delays to both QEGP and MHP. We request enough pairs of QEGP such that only a single DQP exchange is performed at the start. As such, only the memory advertisements and expire messages within QEGP (Ch. 3) should affect throughput. Upon sending, we add delays which are the result of MAC evaluation times. Upon receiving a message (which is simulated), we add delay both MAC evaluation and transmission time delays.

Similar to in [38], we aim to analyze the trade-off of throughput and fidelity by sweeping incoming request frequency and requested minimum fidelity for Network Layer (NL) and Measure Directly (MD) type requests. Pairs produced for MD type requests may be measured as soon as they are available. NL and CK type requests however require that the pair be transferred to memory first, meaning that we would, like in [38], expect a lower throughput. Overall, we expect to see the following:

1. A decrease in throughput as classical delays are increased.
2. A decrease in throughput as minimum fidelity requirements are increased, which was already observed previously in [38].

8.3. Modeling Delays

We outline how we model both transmission delays and computation delays. After, we outline how we integrate this into the codebase, as well as the inputs we give to the extended model.

8.3.1. Modeling Transmission Delay

End-to-end classical delay is as follows:

$$\text{end-to-end delay} = \text{propagation delay} + \text{transmission delay} + \text{processing delay} \tag{8.1}$$

Transmission delay is determined by (i) the packet size and (ii) bit rate of the channel. Therefore, to model this, we need to take into account the size of the control messages, and the size of the accompanying tags. We assume a classical bit rate of 1 Gbit/s, following the simulated use of a BASE1000-ZX connection in [38].

Within the simulation, immediately after receiving a message, we insert a delay which we calculate using the message size and bit rate of the channel before any other operations are performed. This has the same effect as delays which are a property of the channel itself, such as propagation delay.

For transmission delay, however, we also need to consider the size of the packets that are transmitted. The packets as presented in [38] would likely not be transmitted directly via a dedicated physical fibre. Instead, they would likely be routed along an existing classical network. At the very least, they must almost certainly contain some meta-information for routing purposes, especially in a larger network (with multiple links). Therefore, we in likelihood will have to also contend with Ethernet, Internet Protocol (IP), and User Datagram Protocol (UDP) overhead in a more mature network implementation. We assume use of UDP instead of the Transmission Control Protocol (TCP) as the latter (1) introduces extra latency due to its retry mechanism and (2) Dahlberg et al. already performed their evaluation without assuming any underlying retry-mechanism on the classical link [38]. The final resulting Ethernet frame can be seen in figure 8.2.

Algorithm 1 Transmission delay for given message. See snippet 1 (p. 98) for the actual Python implementation

Require: bit_rate (bits/second), $message\_length$ (bits), tag_length (bits), $nonce\_length$ (bits)
$bit\_rate > 0$, $message\_length \geq 0$

$bits\_per\_nanosecond \leftarrow bit\_rate \times 10^{-9}$
$ethernet \leftarrow 5.5 \times (4 \times 8) + 4 \times 8$
$ipv6 \leftarrow 8 \times (4 \times 8)$
$udp \leftarrow 2 \times (4 \times 8)$
$min\_ethernet\_payload \leftarrow 46 \times 8$
$payload\_size \leftarrow ipv6 + udp + message\_length + tag\_length + nonce\_length$
$actual\_payload\_size \leftarrow \max(min\_ethernet\_payload, payload\_size)$
return $\dfrac{actual\_payload\_size + ethernet}{bits\_per\_nanosecond}$

In this example, for MHP `GEN` messages Ethernet adds some extra padding. This padding results from the IEEE 802.3 (Ethernet) specification stating that the payload (IPv6, UDP, and MHP `GEN` in figure 8.2) must be between 46 and 1500 bytes in length[I]. The model of transmission delays is then as presented in algorithm 1.

The sizes of packets we take into consideration, based on the protocol specification, is shown in table 8.1. A full overview of the packets is shown in appendix A.

Table 8.1: Simplified overview of packet sizes in bits of EGP/MHP. See chapter 3 for complete overview of packets.

Packet	Description	Bytes
MHP `CREATE`	Request to entangle with midpoint.	4
MHP `OK/ERR`	Response to request (from midpoint).	8
QEGP `EXPIRE`	Notify that qubit reserved for request has expired.	12
QEGP `ACK`	Acknowledge expire message.	8
QEGP `REQ(E)/ACK(E)`	Memory request and response.	4

[I] The IEEE 802.3 specification also states that there should be a 12-byte packet gap between successive packets. On a 1 Gbit/s channel, that would mean that we must have a gap of at least 12 ns between packet transmissions. However, we do not consider this within our simulation.

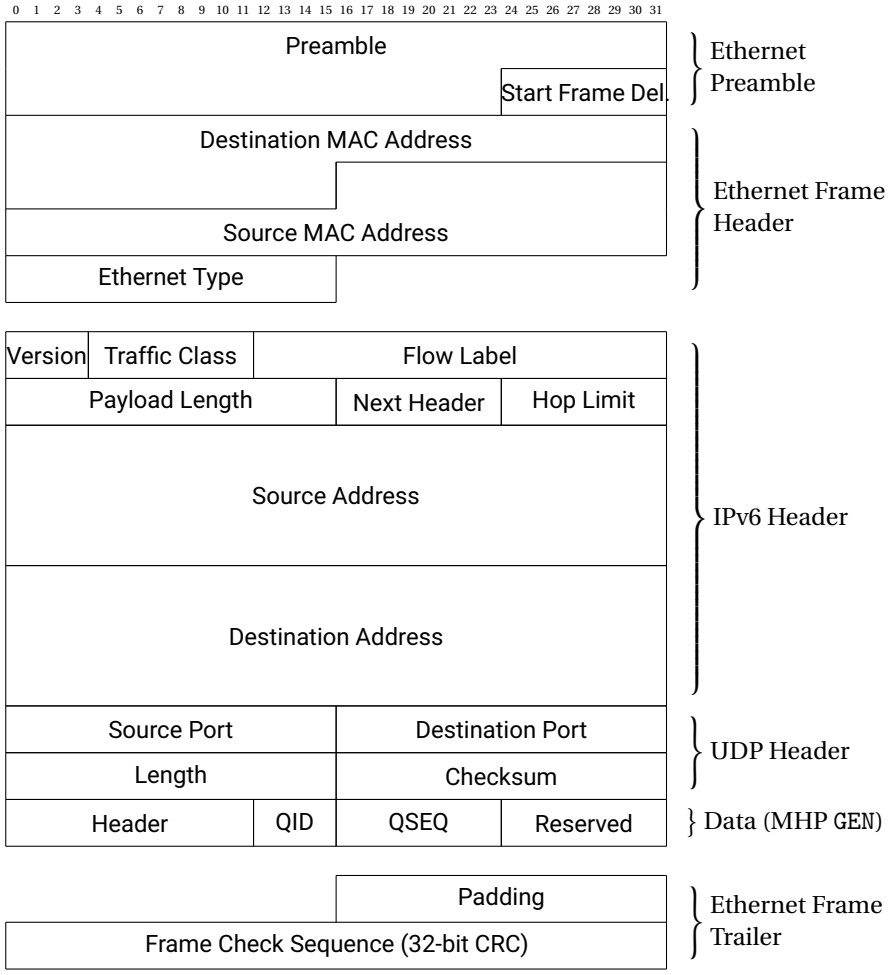


Figure 8.2: MHP `GEN` sent using the Ethernet, IPv6, and UDP protocols. The MAC address (Medium Access Control address) is not to be confused with the Message Authentication Code. Note the amount of overhead of the three protocols necessary for classical communication. We also note that here the Ethernet payload is only 36 bytes, so we add 2 bytes (16 bits) of padding to reach the minimum of 46 bytes of payload (if no 801.1Q tag is present) as per the Ethernet specification.

8.3.2. Modeling Classical Computation Delay

We model the delay of calculating tags for control messages. With regards to end-to-end delay (eq. 8.1), the falls under the processing delay. Computation delay (in nanoseconds) as a result of evaluating a MAC is calculated using the clock frequency, known computation times for different input sizes, and the message size. We illustrate this using algorithm 2. We do not model the overhead of calling these functions, maintaining a nonce or any append operators. Therefore, we do not simulate this overhead As for evaluation time, we distinguish three situations:

1. The evaluation time for a certain message type is known. In such a case, getting the number of cycles required is a simple lookup in table 8.2.
2. If the input is shorter than the shortest known time, we use the time for the smallest known length. If say we only know the computation time for 8 bytes, and a message is 4 bytes, we use the evaluation time of 8 bytes.
3. If the input is larger than the largest known, we use a greedy algorithm[II] to estimate the computation time. We start with the time for the largest known size. We keep adding

[II] An algorithm which makes a locally optimal choice at each stage.

this until the total size would be larger than the true input size. We then pick the second-largest known size and repeat this process until we have arrived at the smallest known size. The process terminates when the estimated size is greater than or equal to the true input size.

Algorithm 2 Calculating computation delay for given message. See snippet 2 (p. 99) for full Python implementation. If no known length, uses greedy algorithm to get computation delay for smallest possible input larger than or equal to true input.

Require: $clock\_speed$ (cycles/second), $input\_length$ (bits), $computation\_delays$ (bytes → cycles), $clock\_speed > 0$, $message\_length \geq 0$

$cycles\_per\_nanosecond \leftarrow clock\_speed \times 10^{-9}$
$lengths\_ascending \leftarrow$ sorted_ascending($computation\_delays.keys$)
$input\_bytes \leftarrow \left\lceil \frac{input\_length}{8} \right\rceil$
$delay, length, length\_idx \leftarrow 0, 0, 0$
while $length < input\_bytes$ **do**
  $current\_length \leftarrow lengths\_ascending$ **get** $length\_idx$
  if $length + current\_length \leq input\_bytes$ **or** $length\_idx =$ size($lengths\_ascending$) $- 1$ **then**
    $delay \leftarrow delay + (computation\_delays$ **get** $current\_length) \times current\_length$
    $length \leftarrow length + current\_length$
  else
    $length\_idx \leftarrow length\_idx + 1$
  end if
end while
return $\frac{delay}{cycles\_per\_nanosecond}$

Certain MACs do not explicitly specify a nonce, and instead, a nonce must be prepended or appended to a message. In practice, this would increase message size and thus evaluation time. For simplicity, however, we only consider evaluation times of the message proper, ignoring increases in message size due to appending a nonce.

We make one final remark about MHP. According to the specification, messages are transmitted at fixed time slots within the specification (indicated by the MHP sequence number). This allows us to distinguish two situations:

1. Send messages as soon as available (after the appropriate delays). This is how the current simulation operates.
2. Check, before sending if after inserting the delay we are still within the same time slot. If not, send it first thing during the next MHP cycle. However, in the current simulation time windows for MHP are not explicitly taken into account.

As one of our goals is to compare our results to [38], we opt not to implement the latter. This also limits the complexity of our additions, lessening the chance of errors.

8.4. Extending The Python Implementation

We add the following parameters to the implementation:

1. The expected number of classical computation cycles to compute MACs for messages of 4, 8, and 16 bytes in length. We require at that least one be known. The others may be calculated using a greedy algorithm (algorithm 2) when required.
2. The clock speed of the classical system.

3. The bit rate of the classical channel.
4. The size of nonces and tags. This is the size of these two values in terms of bits as they are transmitted. In other words, their size post-optimization (section 6.7).

We create the following helper functions to simulate delays at the sending and receiving end:

$$\text{send\_delay}(msg\_size) = \text{computation\_delay}(msg\_size) \tag{8.2}$$

$$\text{recv\_delay}(msg\_size) = \text{transmission\_delay}(msg\_size) + \text{computation\_delay}(msg\_size) \tag{8.3}$$

If the channel is not authenticated, then the computation_delay_model always returns a delay of 0 ns. Computation time depends on the clock speed, known MAC evaluation times, and message size. Transmission time depends on the message, nonce, and tag size, as well as the bit rate of the channel.

QEGP Within QEGP, we make the following modifications:

1. Within send_msg$(self, cmd, msg)$, add item to queue and use the delay from send_delay(msg_size) to schedule a NetSquid event before adding the message to the queue. We use a map from message type to message size (see table 8.1). As before, a separate internal method pops this item from the queue before sending it to the other node.
2. Within process_data$(self)$, add items to queue and instruct NetSquid to process data after a timeout which is calculated using receive_delay(msg_size). Message size depends on the message type. Receive delay additionally takes into account transmission delays.
3. Additional account of send and receive delays as a result of transmission time and authentication overhead (if relevant) in the schedule timeout method.

MHP Within MHP, we add the following function for convenience:

$$\text{transmit\_delay}(msg\_size) = \text{send\_delay}(msg\_size) + \text{receive\_delay}(msg\_size) \tag{8.4}$$

We make the following modifications to the magic distributor:

1. Transmit delay of 64-bit `OK/ERR` message from midpoint to nodes.
2. Transmit delay of 32-bit `CREATE` message from nodes to midpoint.
3. Transmit delay of 32-bit `CREATE` and transmit delay of 64-bit `OK/ERR` message to round-trip delay to the midpoint, and attempted entanglement generations for a specific node, and full-cycle time for attempted entanglement generations for a specific node.

Simulacron The simulation uses Simulacron [37] to pass messages to QEGP internally. This is a general framework for software development within the quantum internet. Within the simulation, this is used to interface with the QEGP implementation. Within the simulation, it is limited such that we may request at most $2^8 - 1 = 255$ pairs of QEGP. [III]

To increase this limit, we modify the CQCEPRRequestHeader(Header) within cqcHeader.py, within the cqc library, such that the number of pairs requested is no longer encoded as an 8-bit unsigned integer. We chance the PACKAGING_FORMAT value such that it is encoded by a 32-bit unsigned integer, meaning that we can request up to approximately 4.3×10^9 pairs. Additionally, we chance the header length to 19, to account for the 3 additional bytes per header:

1. CQC_EPR_REQ_LENGTH is set to 19 at the top of cqcHeader.py.

[III] This is despite a `CREATE` message of QEGP (fig. A.7) specifying that we may also request up to $2^{16} - 1 = 65\,535$ pairs.

2. HDR_LENGTH is set to 19 within the CQCEPRREQUEST class in cqcHeader.py.

We request 131 072 = 2^{17} pairs each time to circumvent DQP. This means that $k = k_{\max} =$ 131 072. This is because this is more than than the expected number of pairs to be delivered during the 2000 simulation seconds. We demonstrate this in the following example:

Example 8.4.1. We assume a best case of throughput of 30 pairs per second [38, fig. 3, with small extra margin] using the QL2020 setup. During 2000 simulations seconds, we would expect that only 60 000 = 30 × 2000 pairs will be delivered. 131 072 ≫ 60 000, so we expect only a single DQP exchange at the start of the simulation.

Though we have circumvented CQC, the maximum number of pairs one may request as part of a single request, according to the specification of DQP, is 65 535 = $2^{16} - 1$. This is due to DQP internally using 16 bits in the package to specify the number of pairs requested (fig. A.6). However, within the simulation DQP packets are not used at all. Instead, only the propagation delays as a result of a hypothetical exchange are simulated. No check is done to see whether the requested amount of pairs can be encoded using 16 bits, so we do not need to change the simulation here.

All this should result in only a single DQP exchange at the start of the simulation, followed by many MHP exchanges with the midpoint station, as well as intermittent QEGP exchanges. The single DQP exchange at the start should then not affect the throughput, as this only presents a fixed offset at the start of the simulation.

8.5. Model Parameters

Having outlined and described the model, we now outline and motivate the parameters for the said model.

8.5.1. The Network Setup: L_{AB} And Quantum Link 2020 Scenario's

We may either consider L_{AB}, or the Quantum Link 2020 (QL2020) scenario. The L_{AB} setup has been realized in a lab, where A and B are 2 metres apart. QL2020 parameters are based on an intended setup that connects Alice and Bob over 25 km between two hypothetical European cities. In that setup, the distance between Alice and the midpoint is 10 km, and Bob to the midpoint is 15 km. All other hardware parameters (table 7.3) are identical.

We choose to focus on the QL2020 scenario as this more closely resembles a quantum network setup as envisioned by the quantum network alliance.

8.5.2. Transmission Time

We first simulate the addition of transmission time. This is to enhance the simulation in [38], such that it more accurately reflects reality. This means that both propagation and transmission time are accounted for. We also run simulations with the parameters described in the QL2020 setup in [38] to verify their results. This gives the following:

1. An unmodified protocol and model. This allows us to verify that we can reproduce the results in [38].
2. Unmodified protocol with transmission delay present. We assume a channel of 1 Gbit/s, as the previous simulation assumed the use of a 1000BASE-ZX connection [38, D.6].

8.5.3. Computation And Additional Transmission Time

Next, we investigate the effects of data origin authentication. We therefore simulate the following:

1. Protocol with MAC evaluation times present, as well as 128-bit tags and 16-bit nonces. We do not test larger nonces as we consider the LSB transmission modification of the nonce

to be straightforward (sec. 6.7.1). We pick 16-bit tags as this is the shortest example we investigated previously, though we would recommend slightly larger tags (e.g. 32-bit tags). See section for the discussion on tag sizes 5.5.

2. Protocol with MAC evaluation times present, as well as 16-bit tags and 16-bit nonces. Shorter tags could be achieved by simply producing shorter tags, truncating tags, or by performing tag accumulation with speculation[IV] (sec. 6.7.2). Additionally, we do not distinguish between QEGP and MHP-level messages as to not complicate the simulation.

The MAC times we use are illustrated in table 8.2.

Table 8.2: Computation times under evaluation, in cycles per byte, for different packet sizes. Based on numbers from table 6.2. The CLHASH multiples (2, 4, and 16) are presented alongside the actual values in brackets to illustrate how we attempted to roughly mimic an exponential distribution of evaluation times using benchmark values. As we do not find numbers for 4 bytes (length of MHP `GEN` message), we pad this till 8 bytes, similar to how algorithm 2 calculates computation delays.

	Message Length (bytes)		
	8	**16**	**MAC (Benchmark)**
1	4.6	1.4	CLHASH [69] ([69])
2	7.0 (9.1)	3.50 (3.50)	VHASH[a] [66] ([69]) (≈ 2×CLHASH)
3	9.2 (18.2)	6.5 (7.00)	SipHash [8] ([69]) (≈ 4×CLHASH))
4	- (73.6)	17.8 (22.4)	VMAC-AES-128 [66] ([88]) (≈ 16×CLHASH)
5	588.8	179.2	128×CLHASH

[a] A component within VMAC [66]. Included as it is approximately half the speed of CLHASH, which is a complete MAC solution, and because it was run in the same setup as CLHASH [69]. An alternative would be CMAC-AES-128 [44] from the benchmark in [35], which has somewhat similar performance characteristics.

CLHASH×128 is the worst case under evaluation and presents a poorly optimized MAC (e.g. poor optimization for host architecture or latency when communicating with the cryptographic module, etc.).

Example 8.5.1 (Transmitting a MHP `GEN` packet)**.** Take 10 km between Alice and the midpoint, and 15 between the midpoint and Bob. We assume a classical channel with a bit rate of 1 Gbit/s. The propagation delay along classical fibre is taken to be 2×10^8 m/s, roughly two-thirds the speed of light.

a The length from A to the midpoint is 10 kilometers, which gives a propagation delay of 50 000 ns($= \frac{10\,000\,\mathrm{m}}{200\,000\,000\,\mathrm{m\,s^{-1}}} \times 1 \times 10^9$ ns).
b If we take into account transmission delay, we must then transmit 576 bits (32×18, see figure 8.2) over a 1 Gbit/s (1 000 000 000 bit/s) channel, which results in a transmission time of 576 ns. The total transmission time, which is now 50 576 ns (50000 + 576). As we see, the overall latency of the classical transmission does not increase dramatically over sufficiently large distances.
c Transmitting a 16-bit nonce and 16-bit tag requires 32 ns. Transmitting a 128-bit tag with 16-bit nonce instead requires 144 ns. Transmitting a 128-bit tag and 128-bit nonce requires 256 ns.
d For the MAC evaluation time, we refer to table 8.2. We require 73.6 cycles to compute the tag of a 8 byte message. On a 4GHz processor, this results in a computation delay of 73.600 ns ($= \frac{73.6}{1 \times 10^9\,\mathrm{instr./s}} \times 1 \times 10^9$ ns) on both ends, giving us a total delay of 179.200 ns ($= 32\,\mathrm{ns} + 73.600\,\mathrm{ns} \times 2$) if we assume a 16-bit tag and nonce.

[IV] We do not model the additional cited computational overhead of ≈10% [70].

Figure 8.3 illustrates how the delays are distributed for these situations. We would expect that the choice of MAC would have a small effect on link performance due to propagation delays and to a lesser extent transmission time being more prevalent. That is unless MAC evaluation times are sufficiently large (i.e. 128×CLHASH evaluation times), for which we would expect to see a noticeable drop in throughput.

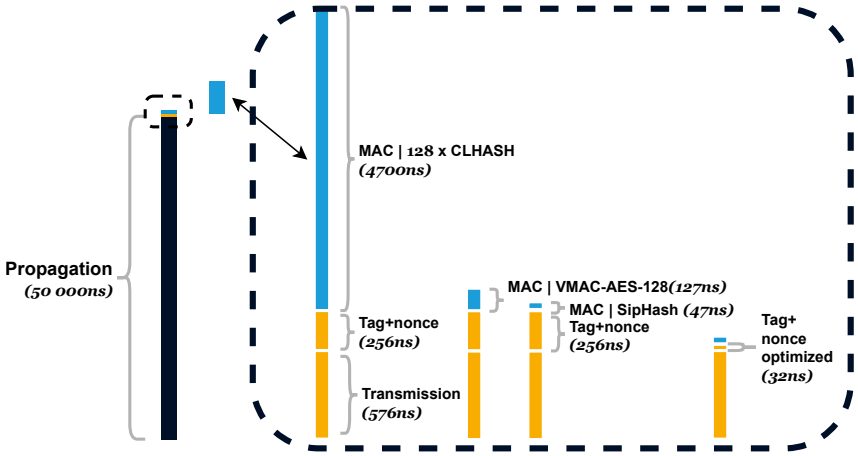


Figure 8.3: The distribution of classical delays for communication between Alice and the midpoint station, at a distance of 10 km. Transmission time is influenced by packet size and the bit rate of the channel and is depicted in yellow. MAC evaluation times, which must be accounted for twice (at the sending *and* receiving end) are depicted in light blue.

We have the following set of simulations, shown in table 8.3. For the minimum fidelity sweep, we choose a greater number of bright state populations to allow a finer granularity in the achievable set of minimum fidelities.

Table 8.3: Simulation parameters. Note that alpha is a single value, which is a list, whereas F_{min} consists of 8 different values.

Simulation	α	F_{min}
F_{min} sweep	[0, 0.005, ..., 0.5][a]	0.5, 0.55, ..., 0.85

[a] 100 evenly spaced numbers between 0 and 0.5

The amount of simulations follows from the combination of these parameters. This is expressed in table 8.4.

Table 8.4: Counts of each parameters, giving an overview of the total amount of simulations performed. Includes implicit total simulation count as we perform 5 runs for each parameter combination.

		Quantum			**Classical**		
Simulation	**Mix**	α	F_{min}	**Gbit/s**	**Computation Time**	**Tag size**	**Total**
F_{min} sweep	1	1	8	2	5	2	960 (×5)

See appendix B for an overview of the platform on which we run the simulations.

CHAPTER 9

EVALUATION

Our goal is to understand how overhead from authentication affects the performance of QEGP and MHP. We measure this in terms of throughput, which is the rate at which successfully entangled pairs are produced.

We first verify that we can reproduce the results from [38]. We next introduce transmission delay to observe the effects on these protocols. To investigate the effects of data origin authentication we evaluate the results of using the timings of the MACs outlined in section 8.5. This is to evaluate how the simulated link functions when taking into account a range of what we consider to be realistic performance indicators of data origin solutions. This also takes into account the resulting extra transmission delays of the 16-bit nonces and 16- and 128-bit tags.

We use the model with transmission time present as a baseline comparison. When doing so, we conclude that MAC evaluation time has a noticeable effect on throughput if they become excessive. However, if they remain approximately within the range of the fastest solutions we find (see table 8.2), the performance remains mostly unchanged. Therefore, we conclude that, if we use the results with transmission delays present as the baseline, data origin authentication has little effect on the performance of the link.

9.1. Overview

We first briefly elaborate on how the graphs were created, and how to interpret them in section 9.2. We delve into the effects of adding transmission delay in section 9.3. This represents a baseline, where the protocol itself remains unmodified. We then investigate the effects of augmenting the protocol with data origin authentication by observing how the delays affect throughput in 9.4. We draw our conclusions in section 9.5.

9.2. Result Preliminaries

We calculate the mean of the mean scaled latency of 5 runs and the 95% confidence interval. We perform only 5 runs due to (1) limited computational resources (2) a relatively high rate of agreement between simulations and (3) validation of the protocol was already performed in [38]. To calculate the confidence interval, we first calculate the standard deviation:

$$\sigma = \sqrt{\frac{\sum_i (x_i - \bar{x})}{n}} \qquad \text{and} \qquad \bar{x} = \frac{1}{n}\sum_i x_i$$

Where $\bar{x}$ is the mean throughput of each run, and n the number of runs (5). This means that $\forall i, x_i$ is itself already a mean of the throughput of all requests of run $i \in \{1,2,...,n\}$. Seeing

as we have a relatively small number of runs, we use the t-distribution to calculate the 95% confidence intervals. This distribution has larger tails due to the larger amount of uncertainty about the data when using a smaller sample size. The confidence interval, for a given set of runs, is calculated as follows:

$$\bar{x} \pm 2.571 \times \frac{\sigma}{\sqrt{n}}$$

Where $\sqrt{n} = \sqrt{5}$ and σ is the standard deviation of the runs as calculated above. The value 2.571, the t-value, follows from a using 95% confidence interval with 4 degrees of freedom (following from 5 runs). Note that this value is 2.04 when we have 30 degrees of freedom (31 runs), which is very similar to the z^*-value of 1.96 for the 95% confidence interval. Doing more runs would allow us to approach this value to further decrease the size of our confidence interval. However, we determine that for our purposes and considering limited computational resources for simulation purposes, this value is to approximate the effects of authentication. This is in part because the protocols have already extensively been analysed in a similar setup in [38].

A Note On The Strict Frequentist Interpretation Of Confidence Intervals This range only tells us that if were to perform repeated experiments that the expected value of 95% of *many* runs will be within this range. We cannot for a *single* run make any statement about where the expected value lies. For simplicity, however, we consider it sufficient to interpret the results as "there is a 95% chance that the average latency of the protocol given these parameters will be within this range for this type of request".

9.3. Introduction Of Transmission Time

As mentioned, an increase in F_{min} results in the probability of an entanglement attempt being successful being lower. We, therefore, expect the throughput to be lower. We measure the throughput as the number of successful pairs delivered by the protocol within a 1 second time window. The addition of transmission delay is illustrated in figure 9.1.

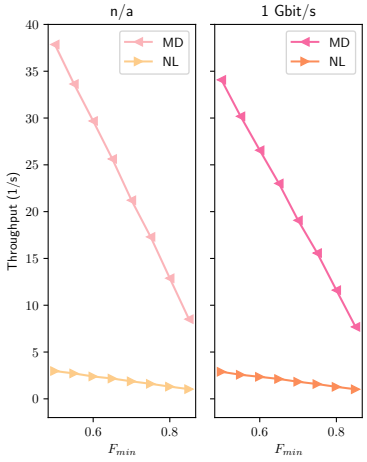


Figure 9.1: F_{min} versus throughput of the system before and after the introduction of transmission time. The introduction of transmission time already cases a noticeable decrease in throughput along the link. MD type requests, for the $0.50 \leq F_{\text{min}} \leq 0.85$ range, see an average throughput drop of 10.06%. NL type requests, for the same range, see an average drop of 2.34%. This might be explained by the larger amount of overhead already present in NL type requests, as qubits are moved to memory first.

As mentioned in [38], MD requests likely have a much higher throughput as qubits do not need to be moved to memory. The addition of transmission delay does cause a decrease in

throughput. For MD type requests, the throughput decrease from 37.86 pairs per second for $F_{\text{min}} = 0.50$ to 34.08 pairs per second when transmission time is introduced. This represents a $\approx 9.98\%$ decrease. At $F_{\text{min}} = 0.85$, the throughput drops from 8.51 to 7.69, representing a $\approx 9.64\%$ decrease. The decrease in throughput is therefore proportionally nearly identical for both F_{min} values (and by extension, the absolute difference is not). The average decrease in 10.06%, caused mainly by a slightly larger decrease for the $0.55 \leq F_{\text{min}} \leq 0.65$ range (tab. C.2).

For NL type requests, at $F_{\text{min}} = 0.50$ it drops from 2.97 to 2.89 pairs per second, a 2.69% drop in throughput. For $F_{\text{min}} = 0.85$ it drops from 1.04 to 1.02 pairs per second, a 1.92% drop. The average drop in throughput for all requested minimum fidelities after the introduction of transmission time is 2.34%. This smaller drop, compared to MD type requests, might be explained by NL type requests having more overhead as they also perform memory operations. The addition of transmission time would therefore represent a smaller increase, proportionally speaking.

We conclude from this that the rate at which MHP may deliver pairs is slightly worse than suggested in [38] as they did not account for transmission delay. The introduction of transmission delays does appear to affect the throughput for all requested F_{min} in approximately the same manner, as the above percentages demonstrate.

See table C.2 for a complete overview of all throughput values.

9.4. Introduction Of Data Origin Authentication Delays

Next, we investigate the effects of data origin authentication on throughput. We use 5 different evaluation times (tab.. 8.2). We investigate both the effect of using 16-, and 128-bit tags. The effects of an authenticated channel are illustrated in figure 9.2.

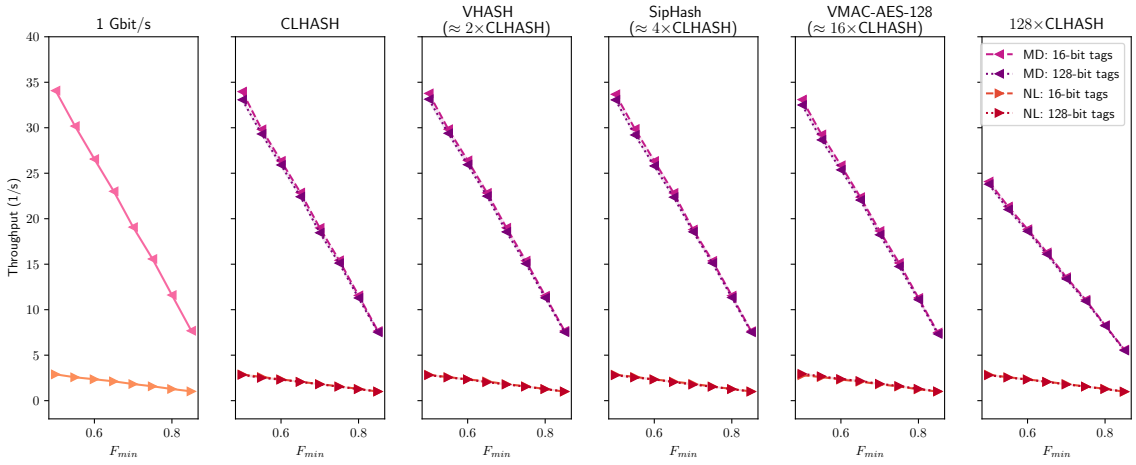


Figure 9.2: F_{min} versus throughput of the system with different MAC evaluation times. We also show throughput with no data origin authentication present on the very left. When using both 16-bit tags and 128-bit tags, throughput generally decreases between 0-3% (barring a few outliers) for both MD and NL type requests. The only exception is when using 128×CLHASH evaluation times, representing a poorly optimized MAC. MD type requests see a reduction of throughput between 27-30% for all requested fidelities. NL type requests are not noticeably influenced by this extra overhead it seems, with throughput decreases remaining with the 0-3% range. Lastly, using 128-bit tags instead of 16-bit tags, for both types of requests generally decreases throughput by 1-2%. In some situations, throughput increase for NL type requests, mostly when using 128-bit tags instead of 16-bit tags. We conclude that this is a simulation artefact, likely due to throughput for NL requests being lower: Small fluctuations in the absolute throughput represent a large proportional change (and thus percentage value).

First, we see that for the first four data origin solutions, which represent a range of the fastest

solutions identified, the drop in throughput is less pronounced than when transmission delays were introduced. We consider MD type requests. When $F_{\min} = 0.50$ throughput is 34.08 pairs per second when only transmission time is present. When introducing SipHash with 128-bit tags, the throughput is 33.06 pairs per second for MD type requests, a 2.99% drop in throughput. For 16-bit tags, the throughput is 33.68 pairs per second, giving a smaller 1.17% drop. For $F_{\min} = 0.85$, throughput is 7.69 pairs per second without data origin authentication, dropping to 7.52 pairs per second when SipHash was introduced with 128-bit tags, a 2.21% drop. When using 16-bit tags, the amount of pairs drops to 7.63 instead, giving a 0.78% drop in throughput.

For NL Type requests, the absolute chance in throughput is less pronounced. However, the relative change is comparable. For $F_{\min} = 0.50$, throughput is 2.89 pairs per second when only transmission time is present. When introducing SipHash with 128-bit tags, the throughput is 2.80 pairs per second, a 3.11% drop in throughput. When using 16-bit tags, throughput drops to 2.85 pairs instead, a 1.38% drop. For $F_{\min} = 0.85$, throughput is 1.02 pairs per second without data origin authentication, dropping to 1.00 pairs per second when SipHash was introduced with 128-bit tags, a 1.96% drop in throughput. When using 16-bit tags the throughput drops to 1.02 pairs per second when SipHash was introduced with 128-bit tags, meaning that SipHash has no influence. Therefore, like with MD type requests, the introduction of data origin authentication delays has a less pronounced effect on throughput than the introduction of transmission delays. The decrease in throughput is also slightly smaller than with MD type requests, similar to when transmission times were introduced.

See table C.3 for a complete overview of all throughput values.

9.5. Summary RQ3

We draw the following conclusions:

- Throughput decreases as classical delays are increased. This is because MHP cycle times are affected directly by these delays.
- The QEGP and MHP would in practice likely perform worse than suggested in [38]. We find that when we account for transmission delays, within any data origin authentication delays, the throughput of NL type requests is not affected greatly ($\approx 2.96 - 1.92\%$ decrease). The throughput of MD type requests is affected more noticeably ($\approx 8.8 - 9.2\%$ decrease).
- Data origin authentication has very little influence on link performance in the QL2020 setup. When using SipHash (a relatively popular MAC), throughput drops by $0.78 - 2.96\%$ depending on the requested fidelity and the type of request (MD or NL). This is compared to the throughput when transmission overhead of the original protocol was already introduced.
- When using data origin authentication, transmitting fewer classical bits, by for instance using 16-bit instead of 128-bit tags, does have an observable effect on throughput.
- Overall, the QL2020 setup does remain functional when we introduce delays from data origin authentication. The addition of transmission delay more noticeably influences throughput than the fastest data origin authentication solutions. By extension, this also means that the QL2020 link will likely perform slightly worse than the simulation from [38] suggests, as they did not account for transmission time.

RQ 3 | How does the latency of data origin authentication on the classical data plane affect link throughput?

RQ 3.1 | How does the introduction of transmission delays affect link throughput?

The introduction of transmission time overhead, within the QL2020 setup, causes a noticeable decrease in throughput of the link. For MD type requests, throughput decreases by 9.98% for $F_{\text{min}} = 0.50$, and 9.64% for $F_{\text{min}} = 0.85$. For NL, these values are 2.96% and 1.92% respectively. This smaller drop, compared to MD type requests, might be explained by NL type requests having more overhead as they also perform memory operations.

RQ 3.2 | How does the introduction of data origin authentication delays affect link throughput?

When taking the evaluation times of SipHash, we find that throughput does not decrease further as significantly within the QL2020 setup. For the following, we use the throughput values after transmission time has been introduced to the simulation as reference point. When using 128-bit tags and SipHash, a popular MAC, MD type requests see a throughput decreases of 2.99% for $F_{\text{min}} = 0.50$, and 2.21% when $F_{\text{min}} = 0.85$. For NL, these values are 3.11% and 1.92% respectively. The drop in throughput is less pronounced when using 16-bit tags instead of 128-bit tags. For MD type requests, when $F_{\text{min}} = 0.50$, throughput drops by a smaller 1.17% when using 16-bit tags instead, and when $F_{\text{min}} = 0.85$ it drops by 0.78%. For NL type requests, again with 16-bit tags, when $F_{\text{min}} = 0.50$, throughput drops by 1.38% instead, and by not at all when $F_{\text{min}} = 0.85$.

CHAPTER 10

DISCUSSION AND FUTURE WORK

Quantum networks are in the early stages of deployment. As with classical networks, both their technical feasibility as well as their security considerations must be taken into account during the design phase. We in this work investigated the need for data origin authentication of classical control messages at the physical and link layer of the stack We also investigated the effects of the delays of said authentication on direct quantum links in terms of performance, specifically throughput.

10.1. Conclusions

Based on the previous conclusions drawn in chapters 5, 6, and 9, we present the following.

RQ1 | Why, And How, Should Control Messages Be Authenticated? We have presented a non-exhaustive list of attacks on the QEGP and MHP from [38]. The existence of these attacks we argue is sufficient reason to perform data origin authentication of control messages at both the link and physical layers. Generally speaking, such authentication is necessary to help ensure the availability of individual links in a quantum network.

We provide derivations for the amount of key material that an information-theoretic security data origin authentication solution consumes in terms of the number of qubits that are transmitted along the link. Specifically, we perform these derivations for heralded entanglement generation. Using these derivations, we can give the upper rate of entanglement generation based on either the total amount of key material present or the rate at which it is produced. Conversely, we also derive the amount of key material needed to sustain a given rate of entanglement generation. Any key generation mechanism exceeding this rate will not form a bottleneck in [38] when using information-theoretic data origin authentication of MHP control messages. The rate of key production over a distance of 15 km (and over a distance of 10 km) in the QL2020 setup[I], must be roughly greater than 772 kbit/s when using 32 bit tags for the control messages for the key generation mechanism itself to not be a bottleneck. The best performing QKD solutions we identify appear to deliver key material at a sufficiently high rate over the specified distances (of the QL2020 setup) as to not become a bottleneck themselves (tab. 5.2).

However, we also argue that information-theoretically secure data origin authentication might not be warranted at this level within the stack. Computationally secure data origin authentication solutions appear sufficiently secure against adversaries with quantum computing resources at their disposal, provided they may not also interact with the authenticated channel

[I] A setup based on two hypothetical European cities separated by 25km.

quantumly. That is, perform queries in superposition. These MACs are likely sufficiently secure for the duration of a session, as these sessions have limited lifetimes. However, their use does not preclude the use of QKD to establish short session keys, which are then in turn used by computationally secure solutions as part of a hybrid solution.

RQ2 | How fast are the fastest data origin authentication solutions? We investigate a range of data origin authentications solutions that have seen some use in the industry. We find that nearly every solution presented shows similar performance characteristics when producing tags for short messages. That is, we identify in our survey no MAC which is noticeably slower than other implementations. This conclusion is based upon three benchmarks identified. One MAC for which we find now performance indicators for short messages is KMAC, which is from the same family as SHA-3. Performance appears to be more so affected by the actual implementation and to what extent it is optimized for the host architecture than the actual underlying algorithm.

RQ3 | How does the latency of data origin authentication on the classical data plane affect link throughput? As expected, an increase in latency on the classical channel reduces the throughput of the quantum link. Throughput already noticeably decreases when transmission delays are introduced. This means that the QL2020 setup, using the solution proposed in [38] will likely perform slightly worse (in terms of throughput) than originally indicated. When modeling transmission time, for MD type requests, throughput decreases by 9.98% when the minimum requested fidelity is $F_{\text{min}} = 0.50$, and 9.64% when $F_{\text{min}} = 0.85$. For NL, these values are 2.96% and 1.92% respectively.

The introduction of data origin authentication delays has a less pronounced impact on throughput of the link provided that the MAC evaluation times are within the range of evaluation times based on our survey. For the following, we use the throughput values after transmission time has been introduced to the simulation as reference point. When using 128-bit tags and SipHash, a popular MAC, MD type requests see a throughput decreases of 2.99% for $F_{\text{min}} = 0.50$, and 2.21% when $F_{\text{min}} = 0.85$. For NL, these values are 3.11% and 1.92% respectively. For MD type requests, when $F_{\text{min}} = 0.50$, throughput drops by a smaller 1.17% when using 16-bit tags instead, and when $F_{\text{min}} = 0.85$ it drops by 0.78%. For NL type requests, again with 16-bit tags, when $F_{\text{min}} = 0.50$, throughput drops by 1.38%, and not at all when $F_{\text{min}} = 0.85$.

The MAC evaluation time has a less pronounced effect on throughput, with there being much overlap between the drop in throughput among the MAC solutions evaluated. This is also illustrated in table C.3. This falls in line with our expectations, as transmission time is greater, proportionally speaking, as illustrated in figure 8.3. This implies that any MAC under investigation in this document is likely suited for the use case of performing data origin authentication of control messages along with quantum network links. The amount of transmitted bits has a far greater effect on throughput, and should therefore be kept to a minimum. We first recommend using truncated nonces. Tag length should also be kept to a minimum. Either one accepts a lower level of security, or one could use a system such as Cumulative MAC with Speculation (CuMAC/s). CuMAC/s does require future messages be accurately predicated but allows one to achieve higher levels of security while still having short(er) tags.

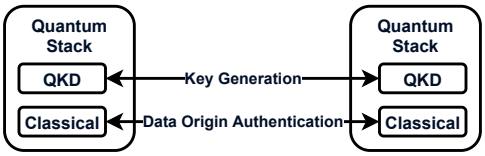
10.2. Future Work

Based on our findings, we propose the following three directions for further areas of research.

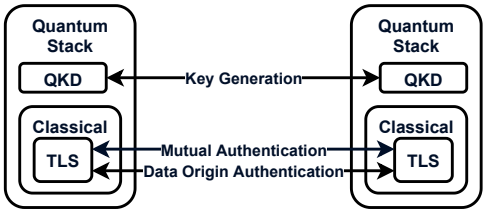
1. Analyse the cost of data origin authentication within larger networks. This could prove especially interesting if we were to consider midpoint stations that dynamically append information to incoming packets to keep classical latencies to a minimum ([65, Lazy entanglement tracking]).

2. While we do not outright preclude the use of encryption within this work, we do not yet identify any strong reasons for doing so. It may, however, prove useful to do a more extensive analysis of information that is leaked during entanglement generation via the control messages. Showcasing an attack or the type of information that may indirectly be leaked (e.g. the current state of the hardware or similar meta-information) would make a stronger case for why encryption of control messages could still prove useful.
3. In future, it would become more relevant to consider how we implement data origin authentication. In this work, we consider mainly performance and how this affects a single link.
   A more mature solution network stack might outsource some functionality to more established systems, such as Transport Layer Security (TLS). Figure 10.1 illustrates some of the possibilities as we see them. TLS 1.3, the most recent version introduced in 2018, operates exclusively using Authenticated Encryption (AE) [90][II]. One would therefore have to contend with the extra overhead of authenticated encryption when using TLS. We would argue that outsourcing data origin authentication to TLS has the advantage that it simplifies the protocol specification, as well as leveraging the industry trust of TLS. However, if the overhead is deemed too great (or encryption completely unnecessary), a call to (a fast) MAC function could be part of the protocol specification itself.

[II] TLS 1.3 supports the symmetric AES in GCM (Galois Counter Mode), Chacha20 combined with Poly1305, and AES in CCM (Cipher block Chaining Mode) ciphers.

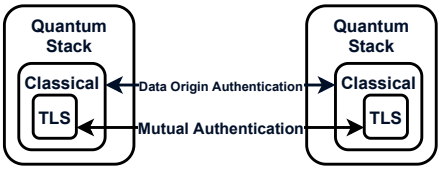


(a) The stack as we have explored it. Key material is assumed to be available in advance to both parties. Further generation of key material, in the case we use a information-theoretic security MAC, is assumed to be generated using a QKD link.



(b) A more mature solution might opt to first perform mutual authentication, probably via TLS. This process also creates a shared secret which they can use to generate further key material using i.e. QKD. This means that the key generated between Alice and Bob would have to remain secure until QKD is run. Data origin authentication can likely also be outsourced to TLS in this setup, which might either be computationally or information-theoretically secure. The latter would place a larger strain on QKD. This additionally assumes that the data origin authentication mechanism accepts new key material.

(c) One could opt to forego QKD. One would then rely wholly on both computationally secure keys and schemes to perform data origin authentication.



(d) Another option is to outsource mutual authentication to TLS. However, data origin authentication is then performed internally using the session key provided by TLS. This could be useful if a classical channel authenticated using TLS introduced too much overhead.

Figure 10.1: Exploration of using TLS to outsource data origin authentication, mutual authentication, and generation of further key material within a quantum link. After successfully performing mutual authentication, we assume Alice and Bob have a certain amount of shared key material at their disposal. Lastly, we consider here only data origin authentication of classical control messages along a link within the quantum stack.

SPECIAL TERMS

bell state specific two-qubit quantum states which are maximally entangled. 11

bit Smallest unit of information in computation. May store 0 or 1. 7

bit rate Measure of how many bits travel per second on a channel. 5, 67, 68, 74, 75

computational security A scheme is computationally secure if it is secure against a computationally bound adversary. Such schemes are susceptible to advances in algorithmic theory and hardware. 27

entanglement distillation combining multiple entangled pairs to produce a smaller set of Bell pairs, using only local operations and classical communication. 18, 44

entanglement swap Operation where two nodes use a third node two produce an entangled pair. Useful when the two nodes are not directly connected. 12, 18, 23

ethernet A networking standard, operating at the link layer of the OSI model. 5, 16, 67

heralded entanglement Entanglement conditioned on the successful measurement of a midpoint station. Onto which basis the communication qubits have been projected is known only by midpoint station and must be communicated to end-stations. 20

information reconciliation An act of error correction which aims to ensure that two pieces of data are, indeed, equal. 30

information-theoretic security A notion of security which states that even with unlimited computational power, an adversary cannot break a cryptosystem. 3–6, 25, 27, 33, 34, 37, 39, 40, 45, 47–50, 83, 86

no cloning theorem A theorem in quantum mechanics which states that it is impossible to create a perfect copy of an unknown quantum state. 3, 11, 17, 30

nonce A number which may only be used once. 4, 5, 26

privacy amplification The act of converting a larger set of bits which may have partially been leaked to the outside world to a smaller set of bits while effectively eliminating all partial knowledge from the outside world. 30

protocol System of rules which specify procedures to be followed. 2, 3, 5, 6, 15, 16, 20, 22, 24, 29, 33, 48, 49, 52, 63, 69, 73, 74, 77, 78, 80

Python Popular high-level programming language, including with data scientists. 69, 71, 93, 97

quantum tomography The process through which a quantum state is (continually) reconstructed. 14

qubit Quantum equivalent of a bit. Smallest unit of information in a quantum computing system. May be any linear combination of 0 or 1, and may be entangled with other qubits. A Qubit is a two-state quantum-mechanical system whose amplitudes may be described using a unit vector in two dimensional complex Hilbert space. 2, 7, 8

APPENDIX A

MHP, QEGP, AND DQP: PACKET FORMATS

All packets formats retrieved directly from [38]. To clarify, these are packets used for inter-node communication. We do not specify here any inter-layer packets.

A.1. Midpoint Heralding Protocol (MHP)

Packets of Midpoint Heralding Protocol. These packets are used to communicate with the midpoint (heralding) station.

The fields are as follows:

- Header: A CQC (classical-quantum combiner) header. See [37].
- QID: ID of queue from which request originated.
- QSEQ: Sequence number within specified queue.

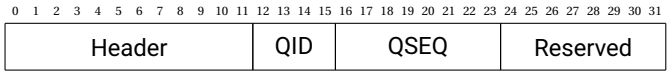


Figure A.1: `GEN` packet format (used in MHP) sent by end stations to heralding station (midpoint).

With the reply (from the midpoint station) containing the following:

- OT: Outcome or error reported by the midpoint station.
- SEQ: Sequence number.
- QIDP: Qubit identifier within sequence of peer.
- QSEQP: Sequence identifier.

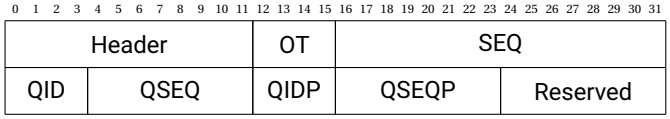


Figure A.2: `REPLY/ERR` packet format (used in MHP) for replies by midpoint with no error.

A.2. Quantum Entanglement Generation Protocol (QEGP)

Packets of the Quantum Entanglement Generation Protocol (QEGP). The QEGP, to facilitate reliable entanglement generation, QEGP communicates with other nodes that either a request has expired. It also performs

Expire

The expire messages are used to communicate mismatches in MHP sequence number. The fields are as follows, with the reply only containing a subset of the `EXPIRE` packet:

- Origin ID: Identifier of node where request came from.
- Create ID: Creation identifier of request.
- SEQ: Up-to-date expected MHP sequence number at the node EXPIRE originated from.

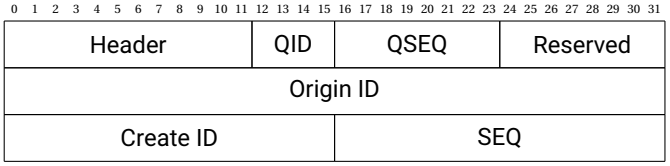


Figure A.3: Packet format for `EXPIRE` message.

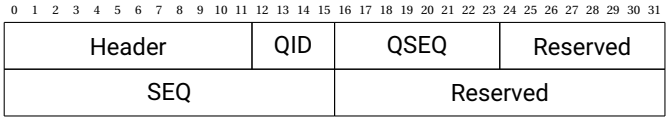


Figure A.4: Packet format for `ACK` message.

Memory Request

A memory request packet, used to request and advertise the amount of qubits available in memory. The fields are as follows:

- Type - Type of message, either request (`REQ(E)`) or acknowledgement/response (`ACK(E)`).
- CMS - The amount of communication qubits available.
- STRG - The amount of available qubits in memory.

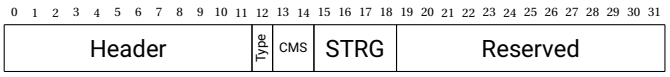


Figure A.5: `REQ(E)`/`ACK(E)` Packet format for EGP memory requests.

A.3. Distributed Queue Protocol (DQP)

These packets facilitate coordination between Alice and Bob as part of the distributed queue protocol. Both the request and reply packet have the same structure. The fields are as follows:

- FT: Frame type. Either `ADD`, `ACK`, or `REJ`.
- CSEQ: Communication sequence number of message.
- QID: ID of queue to add request to.
- QSEQ: Sequence number within specified queue.
- Schedule Cycle: First MHP cycle when request may begin.
- Timeout: MHP cycle when request wil time out.

- Minimum Fidelity: Minimum requested fidelity.
- Purpose ID: Purpose ID. For use by higher layers.
- Create ID
- Number of Pairs
- Priority
- Initial Virtual Finish: Scheduling information for weighted fair queuing (WFQ).
- Estimated Cycles/Pair
- STR: Store flag.
- ATM: Atomic flag.
- MD: Measure directly flag.
- MR: Parent request flag.

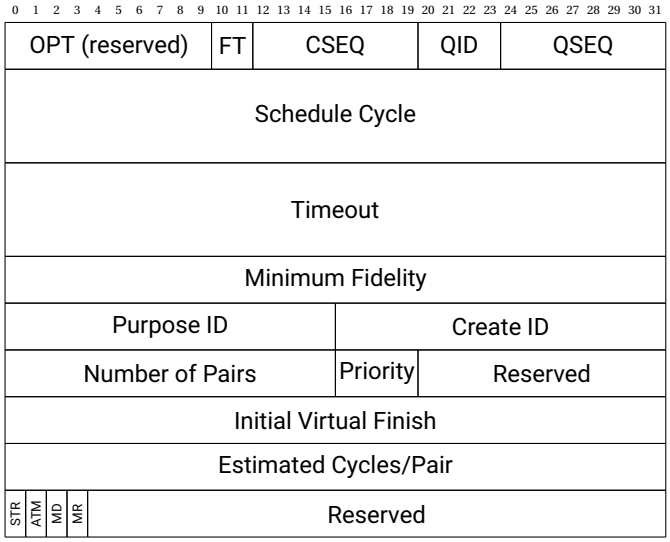


Figure A.6: Packet format for ADD, ACK, and REJ.

A.4. QEGP interface

These packets are used by higher layers to communicate with QEGP. These are internal, meaning that these are not transmitted.

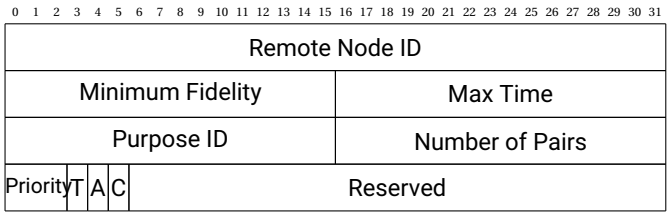


Figure A.7: Packet format for `CREATE` message to QEGP.

A.5. MHP interface

These packets are used by QEGP to communicate with MHP. These are internal, meaning that these should not be transmitted.

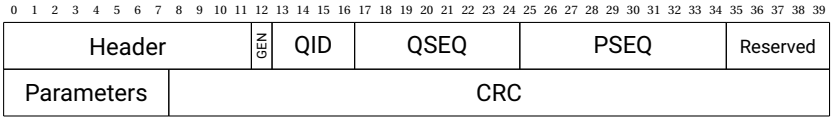


Figure A.8: Packet format of `POLLEGP` messages sent from QEGP to MHP.

APPENDIX B

SIMULATION PLATFORM

The versions of software used are shown in table B.1. NetSquid snippets[I] also shown.

Table B.1: Versions of software used for the simulation.

	Item	Version	Description
System	Ubuntu	20.04.1 LTS	Popular Linux based operating system. Installed as sub-system within Windows 10 OS.
		GNU/Linux 4.4.0-18362-Microsoft x86_64	
	Python	3.8.5 (GCC 9.3.0 on Linux)	General purpose programming language used to build the simulation.
	pip3	20.0.2	Python package management tool.
Python Packages	netsquid	0.7.5	Simulation engine.
	numpy	1.20.1	Math library.
	cqc	3.2.3	Classical-quantum combiner (CQC) interface. Interface between higher layers and hardware in a Quantum Internet.
NetSquid Snippets	physlayer	0.1.8	Nodes and fibres for connections.
	nv	4.2.0	Process quantum information using Nitrogen-Vacancy (NV) centres in diamonds.
	magic	0.1.1	Generate states between nodes using analytical data/simulation data.
	netconf	0.1.3	Easy configuration of networks.
	cartesiussim	0.1.2	Running simulation on cartesius system of SURFsara. However, we do run these experiments locally, with the help of the NetSquid-Magic snippet.

[I] `https://netsquid.org/snippets/`

APPENDIX C

SIMULATION RESULTS

Throughput is the number of pairs which are successfully generated per second of 5 runs. The mean value for both is calculated during each run. The values here are the mean and variation values of these mean values. The mean ($\bar{x}$) and standard deviation (σ) of a run are calculated as follows:

$$\bar{x} = \frac{1}{n}\sum_i x_i \qquad \text{and} \qquad \sigma = \sqrt{\frac{\sum_i (x_i - \bar{x})}{n}}$$

The MAC computation times are illustrated in table C.1.

Table C.1: Reference table of computation times we take into account. See chapter 8 for a more complete explanation. Identical to table 8.2.

	Message Length (bytes)		
	8	**16**	**MAC (Benchmark)**
1	4.6	1.4	CLHASH [69] ([69])
2	7.0 (9.1)	3.50 (3.50)	VHASH [66] ([69]) (≈ 2×CLHASH)
3	9.2 (18.2)	6.5 (7.00)	SipHash [8] ([69]) (≈ 4×CLHASH))
4	- (73.6)	17.8 (22.4)	VMAC-AES-128 [66] ([88]) (≈ 16×CLHASH)
5	588.8	179.2	128×CLHASH

The observed throughputs along the link are illustrated in table C.2 and table C.3

Table C.2: MD and NL mean and standard deviation of throughput. Transmission time and without transmission time comparison. Decrease in throughput also shown as percentage.

	No transmission time				**1 Gbit/s classical channel**					
	MD		**NL**		**MD**			**NL**		
$F_{\min}$	**mean**	**std.**	**mean**	**std.**	**mean**	**diff.**	**std.**	**mean**	**diff.**	**std.**
0.50	37.86	0.15	2.97	0.05	34.08	-9.98%	0.06	2.89	-2.69%	0.02
0.55	33.64	0.11	2.71	0.06	30.18	-10.29%	0.08	2.57	-5.17%	0.03
0.60	29.69	0.09	2.39	0.03	26.56	-10.39%	0.13	2.35	-1.67%	0.05
0.65	25.63	0.05	2.17	0.03	23.01	-10.22%	0.07	2.12	2.30%	0.04
0.70	21.22	0.07	1.87	0.04	19.07	-9.13%	0.10	1.83	2.14%	0.02
0.75	17.30	0.02	1.60	0.02	15.58	-9.94%	0.09	1.58	-1.25%	0.02
0.80	12.88	0.12	1.30	0.02	11.61	-9.86%	0.11	1.28	1.54%	0.04
0.85	8.51	0.04	1.04	0.02	7.69	-9.64%	0.04	1.02	-1.92%	0.03
					Average	-10.06%			-2.34%	

Table C.3: MD and NL mean and standard deviation of throughput. Authentication with multiple computation times comparison. Difference for 16-bit tags is difference between 1 Gbit/s classical channel and 16-bit tags. Difference for 128-bit tags is difference between 1 Gbit/s classical channel and 128-bit tags, and between 16-bit tags and 128-bit tags.

		Tag (bits): 16						**Tag (bits): 128**						
		MD			**NL**				**MD diff.**			**NL diff.**		
$F_{\min}$	**MAC**	**mean**	**diff.**	**std.**	**mean**	**diff.**	**std.**	**mean**	**1 Gbit/s**	**16-bit**	**std.**	**mean**	**1 Gbit/s**	**16-bit**	**std.**
0.50	CLHASH	33.97	-0.32%	0.11	2.88	-0.35%	0.03	33.07	-2.96%	-2.65%	0.11	2.84	-1.73%	-1.39%	0.08
	VHASH (≈ 2×CLHASH)	33.78	-0.88%	0.06	2.85	-1.38%	0.04	33.13	-2.79%	-1.92%	0.18	2.80	-3.11%	-1.75%	0.03
	SipHash (≈ 4×CLHASH)	33.68	-1.17%	0.14	2.85	-1.38%	0.03	33.06	-2.99%	-1.84%	0.10	2.80	-3.11%	-1.75%	0.06
	VMAC-AES-128 (≈ 16×CLHASH)	33.09	-2.90%	0.07	2.80	-3.11%	0.03	32.50	-4.64%	-1.78%	0.10	2.95	2.08%	5.36%	0.02
	128×CLHASH	24.08	-29.34%	0.11	2.86	-1.04%	0.03	23.79	-30.19%	-1.20%	0.08	2.79	-3.46%	-3.85%	0.03
0.55	CLHASH	29.82	-1.19%	0.13	2.60	1.17%	0.06	29.31	-2.88%	-1.71%	0.08	2.50	-2.72%	-3.85%	0.04
	VHASH (≈ 2×CLHASH)	29.84	-1.13%	0.10	2.59	0.78%	0.03	29.39	-2.62%	-1.61%	0.15	2.55	-0.78%	-1.54%	0.04
	SipHash (≈ 4×CLHASH)	29.87	-24.22%	0.11	2.61	1.56%	0.04	29.20	-3.25%	27.68%	0.08	2.54	-1.17%	-2.68%	0.06
	VMAC-AES-128 (≈ 16×CLHASH)	29.26	-3.05%	0.12	2.59	0.78%	0.02	28.66	-5.04%	-2.05%	0.13	2.68	4.28%	3.47%	0.04
	128×CLHASH	21.33	-29.32%	0.06	2.58	0.39%	0.03	21.01	-30.38%	-1.50%	0.12	2.54	-1.17%	-1.55%	0.04
0.60	CLHASH	26.36	-0.75%	0.10	2.35	0.00%	0.02	25.91	-2.45%	-1.71%	0.19	2.31	-1.70%	-1.70%	0.03
	VHASH (≈ 2×CLHASH)	26.41	-0.56%	0.11	2.35	0.00%	0.05	25.94	-2.33%	-1.78%	0.09	2.34	-0.43%	-0.43%	0.04
	SipHash (≈ 4×CLHASH)	26.34	-0.83%	0.15	2.34	-0.43%	0.02	25.80	-2.86%	-2.05%	0.10	2.33	-0.85%	-0.43%	0.04
	VMAC-AES-128 (≈ 16×CLHASH)	25.87	-2.60%	0.08	2.32	-1.28%	0.06	25.37	-4.48%	-1.93%	0.07	2.38	1.28%	2.59%	0.04
	128×CLHASH	18.84	-29.07%	0.08	2.35	0.00%	0.03	18.62	-29.89%	-1.17%	0.06	2.32	-1.28%	-1.28%	0.03
0.65	CLHASH	22.87	-0.61%	0.06	2.08	-1.89%	0.02	22.42	-2.56%	-1.97%	0.05	2.06	-2.83%	-0.96%	0.01
	VHASH (≈ 2×CLHASH)	22.84	-0.74%	0.15	2.11	0.47%	0.04	22.48	-2.30%	-1.58%	0.03	2.03	-4.25%	-3.79%	0.04
	SipHash (≈ 4×CLHASH)	22.83	-0.78%	0.10	2.10	-0.94%	0.04	22.37	-2.78%	-2.01%	0.10	2.03	-4.25%	-3.33%	0.03
	VMAC-AES-128 (≈ 16×CLHASH)	22.37	-2.78%	0.12	2.09	-1.42%	0.05	22.05	-4.17%	-1.43%	0.11	2.16	1.89%	3.35%	0.04
	128×CLHASH	16.29	-29.20%	0.09	2.07	-2.36%	0.03	16.09	-30.07%	-1.23%	0.14	2.07	-2.36%	0.00%	0.04
0.70	CLHASH	18.99	-0.42%	0.05	1.83	0.00%	0.02	18.46	-3.20%	-2.79%	0.13	1.80	-1.64%	-1.64%	0.03
	VHASH (≈ 2×CLHASH)	19.00	-0.37%	0.13	1.82	-0.55%	0.04	18.56	-2.67%	-2.32%	0.09	1.78	-2.73%	-2.20%	0.04
	SipHash (≈ 4×CLHASH)	18.80	-1.42%	0.08	1.85	-1.09%	0.05	18.56	-2.67%	-2.32%	0.07	1.76	-3.83%	-4.86%	0.04
	VMAC-AES-128 (≈ 16×CLHASH)	18.65	-2.20%	0.04	1.80	-1.64%	0.02	18.23	-4.40%	-2.23%	0.10	1.86	1.64%	3.33%	0.03
	128×CLHASH	13.56	-28.89%	0.09	1.84	0.55%	0.04	13.38	-29.84%	-1.33%	0.07	1.78	-2.73%	-3.26%	0.02
0.75	CLHASH	15.42	-1.03%	0.02	1.57	-0.63%	0.02	15.12	-2.95%	-1.95%	0.09	1.54	-2.53%	-1.91%	0.02
	VHASH (≈ 2×CLHASH)	15.38	-1.28%	0.09	1.58	0.00%	0.03	15.06	-3.34%	-2.08%	0.04	1.54	-2.53%	-1.91%	0.03
	SipHash (≈ 4×CLHASH)	15.36	-1.41%	0.07	1.56	-1.27%	0.03	15.13	-2.89%	-1.50%	0.11	1.53	-3.16%	-1.92%	0.02
	VMAC-AES-128 (≈ 16×CLHASH)	15.14	-2.82%	0.14	1.56	-1.27%	0.04	14.75	-5.33%	-2.58%	0.05	1.61	1.90%	3.21%	0.04
	128×CLHASH	11.08	-28.88%	0.07	1.55	-1.90%	0.03	10.94	-29.78%	-1.26%	0.11	1.55	-1.90%	0.00%	0.04
0.80	CLHASH	11.57	-0.34%	0.10	1.29	0.78%	0.02	11.29	-2.76%	-1.26%	0.06	1.26	-1.56%	-2.33%	0.04
	VHASH (≈ 2×CLHASH)	11.52	-0.78%	0.05	1.28	0.00%	0.04	11.30	-2.7%	-2.42%	0.02	1.27	-0.78%	-0.78%	0.04
	SipHash (≈ 4×CLHASH)	11.54	-0.60%	0.04	1.28	0.00%	0.02	11.33	-2.41%	-1.82%	0.05	1.27	-0.78%	-0.78&	0.02
	VMAC-AES-128 (≈ 16×CLHASH)	11.29	-2.76%	0.04	1.25	-2.34%	0.03	11.10	-4.39%	-1.68%	0.05	1.30	1.56%	4.00%	0.05
	128×CLHASH	8.29	-28.60%	0.06	1.31	2.34%	0.03	8.24	-29.03%	-0.60%	0.02	1.27	-0.78%	-3.05%	0.04
0.85	CLHASH	7.65	-0.52%	0.08	1.00	-1.96%	0.01	7.52	-2.21%	-1.70%	0.03	1.01	-0.98%	1.00%	0.03
	VHASH (≈ 2×CLHASH)	7.66	-0.39%	0.06	0.99	-2.94%	0.02	7.53	-2.08%	-1.70%	0.06	1.02	-0.98%	3.03%	0.02
	SipHash (≈ 4×CLHASH)	7.63	-0.78%	0.05	1.02	-2.94%	0.02	7.52	-2.21%	-1.44%	0.04	1.00	-1.96%	-1.96%	0.03
	VMAC-AES-128 (≈ 16×CLHASH)	7.49	-2.60%	0.05	1.00	-1.96%	0.04	7.33	-4.68%	-2.14%	0.06	1.03	0.98%	3.00%	0.04
	128×CLHASH	5.60	-27.18%	0.04	0.97	-4.90%	0.02	5.52	-28.22%	-1.43%	0.03	1.00	-1.96%	3.09%	0.02

APPENDIX D

CODE SNIPPETS

Transmission Delay

To calculate transmission delay, as shown in snippet 1, we take as input:

- Bit rate of the channel. Default is 1Gbit/s (1 000 000 000 bits per second).
- Amount of bits that must be transmitted.

Returns amount of nanoseconds it takes to transmit said bits. See section 8.3.1 for pseudocode implementation. Python implementation shown in snippet 1.

Computation Delay

To calculate computation delay, as shown in snippet 2, we take as input:

- Clock speed of the platform running the function, in cycles per second. Default is 1GHz (1 000 000 000 cycles per second).
- Mappings from input sizes in bits to total amount of cycles required to compute the function for said input size in cycles.

Returns amount of nanoseconds it takes to evaluate the function for said input size. Note that if no direct mapping from input size to cycles, a greedy algorithm is used to give an estimate. See section 8.3.2 for pseudocode implementation. Python implementation shown in snippet 2.

Code Snippet 1 Python snippet for calculating transmission delay. PyDoc is omitted.

```
ETHERNET_HEADER_TRAILER = int(6.5 * (4 * 8))
IPV6_HEADER = 8 * (4 * 8)
UDP_HEADER = 2 * (4 * 8)

MIN_ETHERNET_PAYLOAD = 46 * 8


class TransmissionDelay:
    def __init__(self, bit_rate: int = None, tag_size=None, nonce_size=None):
        if bit_rate is None:
            self._bits_per_nanosecond = None
        elif not isinstance(bit_rate, int) or bit_rate <= 0:
            raise ValueError(f"Bit rate must be greater than 0, was {bit_rate} ({type(bit_rate).__name__}).")
        else:
            self._bits_per_nanosecond = bit_rate * 10 ** (-9)

        if tag_size is None:
            self._tag_size = 0
        elif not isinstance(tag_size, int) or tag_size < 0:
            raise ValueError(f"Tag size must be 0 or greater, was {tag_size} ({type(tag_size).__name__}).")
        else:
            self._tag_size = tag_size

        if nonce_size is None:
            self._nonce_size = 0
        elif not isinstance(nonce_size, int) or nonce_size < 0:
            raise ValueError(f"Nonce size must be 0 or greater, was {nonce_size} ({type(nonce_size).__name__}).")
        else:
            self._nonce_size = nonce_size

    def calculate_bits(self, bits) -> int:
        payload_size = IPV6_HEADER + UDP_HEADER + bits + self._tag_size + self._nonce_size

        # Payload wrapped in Ethernet header and trailer.
        return max(payload_size, MIN_ETHERNET_PAYLOAD) + ETHERNET_HEADER_TRAILER

    def calculate(self, bits: int) -> float:
        return (self.calculate_bits(bits) / self._bits_per_nanosecond) if self._bits_per_nanosecond else 0.0
```

Code Snippet 2 Python snippet for calculating computation delay. PyDoc and imports omitted.

```
class ComputationDelay:
    def __init__(self, clock_speed: int = None, delays_p_byte: Dict[int, float] = None):
        if clock_speed is None:
            self._cycles_per_nanosecond = None
        elif not isinstance(clock_speed, int) or clock_speed <= 0:
            raise ValueError(f"Clock speed must be greater than 0, was {clock_speed} ({type(clock_speed).__name__}).")
        else:
            self._cycles_per_nanosecond = clock_speed * 10 ** (-9)

        if delays_p_byte is None:
            self._delays = None
        else:
            for (key, value) in delays_p_byte.items():
                if not isinstance(key, int) or key < 0:
                    raise ValueError(f"Delays contained key less than 0, was {key} ({type(key).__name__}).")
                elif (not isinstance(value, int) and not isinstance(value, float)) or value < 0:
                    raise ValueError(f"Delays contained value less than 0, was {value} ({type(value).__name__}).")
            self._delays = delays_p_byte
            self._lengths_asc = sorted(delays_p_byte.keys(), reverse=True)

    def calculate(self, bits: int) -> float:
        if self._delays is None or self._cycles_per_nanosecond is None:
            return 0.0  # Delays is empty
        else:
            no_of_bytes = ceil(bits / 8)
            delay, length, length_idx = 0, 0, 0
            while length < no_of_bytes:
                if length_idx == len(self._lengths_asc) - 1 or length + self._lengths_asc[length_idx] <= no_of_bytes:
                    known_length = self._lengths_asc[length_idx]

                    length = length + known_length
                    delay = delay + self._delays[known_length] * known_length
                else:
                    # Move onto next known length
                    length_idx = length_idx + 1

        return delay / self._cycles_per_nanosecond
```

BIBLIOGRAPHY

[**1**] Aysajan Abidin. On Security of Universal Hash Function Based Multiple Authentication. In Tat Wing Chim and Tsz Hon Yuen, editors, *Information and Communications Security*, Lecture Notes in Computer Science, pages 303–310, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-34129-8. doi: 10.1007/978-3-642-34129-8_27.

[**2**] Mohamed H Abobeih, Julia Cramer, Michiel A Bakker, Norbert Kalb, Matthew Markham, Daniel J Twitchen, and Tim H Taminiau. One-second coherence for a single electron spin coupled to a multi-qubit nuclear-spin environment. *Nature communications*, 9(1):1–8, 2018.

[**3**] Mohammed M. Alani. *OSI Model*, pages 5–17. Springer International Publishing, Cham, 2014. ISBN 978-3-319-05152-9. doi: 10.1007/978-3-319-05152-9_2.

[**4**] Basel Alomair. Universal hash-function families: From hashing to authentication. In *International Conference on Cryptology in Africa*, pages 459–474. Springer, 2014.

[**5**] M. Atici and D. R. Stinson. Universal Hashing and Multiple Authentication. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, Lecture Notes in Computer Science, pages 16–30, Berlin, Heidelberg, 1996. Springer. ISBN 978-3-540-68697-2. doi: 10.1007/3-540-68697-5_2.

[**6**] Jean-Philippe Aumasson. The impact of quantum computing on cryptography. *Computer Fraud & Security*, 2017(6):8–11, June 2017. ISSN 1361-3723. doi: 10.1016/S1361-3723(17)30051-9.

[**7**] Jean-Philippe Aumasson. SipHash, November 2021. URL `https://github.com/veorq/SipHash`. original-date: 2014-03-23T09:51:00Z.

[**8**] Jean-Philippe Aumasson and Daniel J. Bernstein. Siphash: A fast short-input prf. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012*, Lecture Notes in Computer Science, pages 489–508, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-34931-7. doi: 10.1007/978-3-642-34931-7_28.

[**9**] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[**10**] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, Lecture

Notes in Computer Science, pages 1–15, Berlin, Heidelberg, 1996. Springer. ISBN 978-3-540-68697-2. doi: 10.1007/3-540-68697-5_1.

[**11**] CH Bennett and G Brassard. Bb84. In *IEEE International Conference on Computers, Systems, and Signal Processing*, 1984.

[**12**] Charles H. Bennett. Quantum cryptography using any two nonorthogonal states. *Phys. Rev. Lett.*, 68:3121–3124, May 1992. doi: 10.1103/PhysRevLett.68.3121.

[**13**] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, December 2014. ISSN 0304-3975. doi: 10.1016/j.tcs.2014.05.025.

[**14**] Daniel J Bernstein. The poly1305-aes message-authentication code. In *International Workshop on Fast Software Encryption*, pages 32–49. Springer, 2005.

[**15**] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671): 188–194, September 2017. ISSN 1476-4687. doi: 10.1038/nature23461.

[**16**] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-88702-7. doi: 10.1007/978-3-540-88702-7_1.

[**17**] Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5, 2008.

[**18**] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30):320–337, 2009.

[**19**] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, Lecture Notes in Computer Science, pages 313–314, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-38348-9. doi: 10.1007/978-3-642-38348-9_19.

[**20**] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Benoît Viguier. KangarooTwelve: Fast Hashing Based on Keccak-p. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 400–418. Springer, 2018. doi: 10.1007/978-3-319-93387-0_21.

[**21**] Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, and Ben Smeets. On families of hash functions via geometric codes and concatenation. In *Annual International Cryptology Conference*, pages 331–342. Springer, 1993.

[**22**] Alex Biryukov and Leo Perrin. State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive, Report 2017/511, 2017.

[**23**] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, Lecture Notes in Computer Science, pages 216–233, Berlin, Heidelberg, 1999. Springer. ISBN 978-3-540-48405-9. doi: 10.1007/3-540-48405-1_14.

[**24**] J Black, S Halevi, H Krawczyk, T Krovetz, and P Rogaway. Update on umac fast message authentication, 2000.

[**25**] John Black and Martin Cochran. MAC Reforgeability. In Orr Dunkelman, editor, *Fast Software Encryption*, Lecture Notes in Computer Science, pages 345–362, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-03317-9. doi: 10.1007/978-3-642-03317-9_21.

[**26**] John Black and Martin Cochran. Mac reforgeability. In Orr Dunkelman, editor, *Fast Software Encryption*, pages 345–362, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03317-9.

[**27**] Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, Lecture Notes in Computer Science, pages 592–608. Springer, 2013. ISBN 978-3-642-38348-9. doi: 10.1007/978-3-642-38348-9_35.

[**28**] CE Bradley, Joe Randall, Mohamed H Abobeih, RC Berrevoets, MJ Degen, MA Bakker, Matthew Markham, DJ Twitchen, and Tim H Taminiau. A ten-qubit solid-state spin register with quantum memory up to one minute. *Physical Review X*, 9(3):031045, 2019.

[**29**] Cyril Branciard, Nicolas Gisin, Barbara Kraus, and Valerio Scarani. Security of two quantum cryptography protocols using the same four qubit states. *Physical Review A*, 72(3): 032301, 2005.

[**30**] Anne Broadbent and Christian Schaffner. Quantum cryptography beyond quantum key distribution. *Designs, Codes and Cryptography*, 78(1):351–382, 2016. ISSN 1573-7586. doi: 10.1007/s10623-015-0157-4.

[**31**] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526, 2009. doi: 10.1109/FOCS.2009.36.

[**32**] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. ISSN 0022-0000. doi: 10.1016/0022-0000(79)90044-8.

[**33**] Tim Coopmans, Robert Knegjens, Axel Dahlberg, David Maier, Loek Nijsten, Julio Oliveira, Martijn Papendrecht, Julian Rabbie, Filip Rozpędek, Matthew Skrzypczyk, Leon Wubben, Walter de Jong, Damian Podareanu, Ariana Torres Knoop, David Elkouss, and Stephanie Wehner. NetSquid, a discrete-event simulation platform for quantum networks. *arXiv:2010.12535 [quant-ph]*, October 2020.

[**34**] Julia Cramer, Norbert Kalb, M Adriaan Rol, Bas Hensen, Machiel S Blok, Matthew Markham, Daniel J Twitchen, Ronald Hanson, and Tim H Taminiau. Repeated quantum error correction on a continuously encoded qubit by real-time feedback. *Nature communications*, 7(1):1–7, 2016.

[**35**] B. Czybik, S. Hausmann, S. Heiss, and J. Jasperneite. Performance evaluation of mac algorithms for real-time ethernet communication systems. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pages 676–681, 2013. doi: 10.1109/INDIN.2013.6622965.

[**36**] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie proposal: Noekeon. In *First Open NESSIE Workshop*, pages 213–230, 2000.

[**37**] Axel Dahlberg and Stephanie Wehner. SimulaQron—a simulator for developing quantum internet software. *Quantum Science and Technology*, 4(1):015001, September 2018. ISSN 2058-9565. doi: 10.1088/2058-9565/aad56e.

[**38**] Axel Dahlberg, Matthew Skrzypczyk, Tim Coopmans, Leon Wubben, Filip Rozpędek, Matteo Pompili, Arian Stolk, Przemysław Pawełczak, Robert Knegjens, Julio de Oliveira Filho, Ronald Hanson, and Stephanie Wehner. A link layer protocol for quantum networks. *Proceedings of the ACM Special Interest Group on Data Communication*, pages 159–173, 2019. doi: 10.1145/3341302.3342070.

[**39**] G Mauro D'Ariano, Martina De Laurentis, Matteo GA Paris, Alberto Porzio, and Salvatore Solimeno. Quantum tomography as a tool for the characterization of optical devices. *Journal of Optics B: Quantum and Semiclassical Optics*, 4(3):S127, 2002.

[**40**] A. R. Dixon, Z. L. Yuan, J. F. Dynes, A. W. Sharpe, and A. J. Shields. Gigahertz decoy quantum key distribution with 1 mbit/s secure key rate. *Opt. Express*, 16(23):18790–18797, Nov 2008. doi: 10.1364/OE.16.018790.

[**41**] Wolfgang Dür, H-J Briegel, Juan Ignacio Cirac, and Peter Zoller. Quantum repeaters based on entanglement purification. *Physical Review A*, 59(1):169, 1999.

[**42**] Sébastien Duval and Gaëtan Leurent. Lightweight macs from universal hash functions. *Smart Card Research and Advanced Applications*, 11833:195–215, 2019.

[**43**] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical Report NIST Special Publication (SP) 800-38D, National Institute of Standards and Technology, November 2007.

[**44**] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication. Technical Report NIST Special Publication (SP) 800-38B, National Institute of Standards and Technology, October 2016.

[**45**] Artur K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67: 661–663, Aug 1991. doi: 10.1103/PhysRevLett.67.661.

[**46**] Mark Etzel, Sarvar Patel, and Zulfikar Ramzan. Square Hash: Fast Message Authentication via Optimized Universal Hash Functions. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, Lecture Notes in Computer Science, pages 234–251, Berlin, Heidelberg, 1999. Springer. ISBN 978-3-540-48405-9. doi: 10.1007/3-540-48405-1_15.

[**47**] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, June 1982. ISSN 1572-9575. doi: 10.1007/BF02650179.

[**48**] Daniel Gottesman, Thomas Jennewein, and Sarah Croke. Longer-Baseline Telescopes Using Quantum Repeaters. *Physical Review Letters*, 109(7):070503, August 2012. doi: 10.1103/PhysRevLett.109.070503. Publisher: American Physical Society.

[**49**] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv:quant-ph/9605043*, November 1996.

[**50**] Helena Handschuh and Bart Preneel. Key-recovery attacks on universal hash function based mac algorithms. In *Annual International Cryptology Conference*, pages 144–161. Springer, 2008.

[**51**] A. M. Helmenstine. How many atoms exist in the universe? August 2020. URL `thoughtco.com/number-of-atoms-in-the-universe-603795`.

[**52**] David Hilbert and Wilhelm Ackerman. *Grundzüge der theoretischen Logik*. Julius Springer, Berlin, 1928.

[**53**] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Dong-Geon Lee. LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In Yongdae Kim, Heejo Lee, and Adrian Perrig, editors, *Information Security Applications*, Lecture Notes in Computer Science, pages 3–27, Cham, 2014. Springer International Publishing. ISBN 978-3-319-05149-9. doi: 10.1007/978-3-319-05149-9_1.

[**54**] Duan Huang, Dakai Lin, Chao Wang, Weiqi Liu, Shuanghong Fang, Jinye Peng, Peng Huang, and Guihua Zeng. Continuous-variable quantum key distribution with 1 mbps secure key rate. *Opt. Express*, 23(13):17511–17519, Jun 2015. doi: 10.1364/OE.23.017511.

[**55**] Peter C Humphreys, Norbert Kalb, Jaco PJ Morits, Raymond N Schouten, Raymond FL Vermeulen, Daniel J Twitchen, Matthew Markham, and Ronald Hanson. Deterministic delivery of remote entanglement on a quantum network. *Nature*, 558(7709):268–273, 2018.

[**56**] Tetsu Iwata and Kaoru Kurosawa. Omac: One-key cbc mac. In Thomas Johansson, editor, *Fast Software Encryption*, pages 129–153, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39887-5.

[**57**] Siddarth Koduru Joshi, Djeylan Aktas, Sören Wengerowsky, Martin Lončarić, Sebastian Philipp Neumann, Bo Liu, Thomas Scheidl, Guillermo Currás Lorenzo, Željko Samec, Laurent Kling, Alex Qiu, Mohsen Razavi, Mario Stipčević, John G. Rarity, and Rupert Ursin. A trusted node–free eight-user metropolitan quantum communication network. *Science Advances*, 6(36):eaba0959, 2020. ISSN 2375-2548. doi: 10.1126/sciadv.aba0959.

[**58**] Gregory Kabatiansky and Ben Smeets. *Authentication, From an Information Theoretic Perspective*, pages 63–65. Springer US, Boston, MA, 2011. ISBN 978-1-4419-5906-5. doi: 10.1007/978-1-4419-5906-5_377.

[**59**] Norbert Kalb, Andreas A Reiserer, Peter C Humphreys, Jacob JW Bakermans, Sten J Kamerling, Naomi H Nickerson, Simon C Benjamin, Daniel J Twitchen, Matthew Markham, and Ronald Hanson. Entanglement distillation between solid-state quantum network nodes. *Science*, 356(6341):928–932, 2017.

[**60**] John Kelsey, Shu-jen Change, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Technical Report NIST SP 800-185, National Institute of Standards and Technology, Gaithersburg, MD, December 2016.

[**61**] Auguste Kerckhoffs. La cryptographic militaire. *Journal des sciences militaires*, pages 5–38, 1883.

[**62**] Martin J. Klein. Max Planck and the beginnings of the quantum theory. *Archive for History of Exact Sciences*, 1(5):459–479, October 1961. ISSN 1432-0657. doi: 10.1007/BF00327765. Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 5 Publisher: Springer-Verlag.

[**63**] Boris Korzh, Charles Ci Wen Lim, Raphael Houlmann, Nicolas Gisin, Ming Jun Li, Daniel Nolan, Bruno Sanguinetti, Rob Thew, and Hugo Zbinden. Provably secure and practical quantum key distribution over 307 km of optical fibre. *Nature Photonics*, 9(3):163–168, 2015.

[**64**] Wojciech Kozlowski and Stephanie Wehner. Towards large-scale quantum networks. In *Proceedings of the Sixth Annual ACM International Conference on Nanoscale Computing and Communication*, pages 1–7, 2019.

[**65**] Wojciech Kozlowski, Axel Dahlberg, and Stephanie Wehner. Designing a quantum network protocol. *arXiv:2010.02575 [quant-ph]*, 2020.

[**66**] Ted Krovetz. Message Authentication on 64-Bit Architectures. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, pages 327–341, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-74462-7. doi: 10.1007/978-3-540-74462-7_23.

[**67**] P. Kómár, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin. A quantum network of clocks. *Nature Physics*, 10(8):582–587, August 2014. ISSN 1745-2481. doi: 10.1038/nphys3000.

[**68**] Adam Langley, Wan-Teh Chang, Nikos Mavrogiannopoulos, Joachim Strombergson, and Simon Josefsson. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). RFC 7905, June 2016.

[**69**] Daniel Lemire and Owen Kaser. Faster 64-bit universal hashing using carry-less multiplications. *Journal of Cryptographic Engineering*, 6(3):171–185, September 2016. ISSN 2190-8516. doi: 10.1007/s13389-015-0110-5.

[**70**] He Li, Vireshwar Kumar, Jung-Min Jerry Park, and Yaling Yang. Cumulative message authentication codes for resource-constrained networks. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2020.

[**71**] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli. *Security Mechanisms for CAN Protocol*, pages 17–34. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-51328-7_4.

[**72**] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. Decoy state quantum key distribution. *Physical Review Letters*, 94(23):230504, 2005. doi: 10.1103/PhysRevLett.94.230504.

[**73**] David A McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In *International Conference on Cryptology in India*, pages 343–355. Springer, 2004.

[**74**] Miralem Mehic, Oliver Maurhart, Stefan Rass, Dan Komosny, Filip Rezac, and Miroslav Voznak. Analysis of the public channel of quantum key distribution link. *IEEE Journal of Quantum Electronics*, 53(5):1–8, 2017. ISSN 1558-1713. doi: 10.1109/JQE.2017.2740426. Conference Name: IEEE Journal of Quantum Electronics.

[**75**] Rodney Van Meter and Joe Touch. Designing quantum repeater networks. *IEEE Communications Magazine*, 51(8):64–71, 2013. doi: 10.1109/MCOM.2013.6576340.

[**76**] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In Antoine Joux and Amr Youssef, editors, *Selected Areas in Cryptography – SAC 2014*, Lecture Notes in Computer Science, pages 306–323, Cham, 2014. Springer International Publishing. ISBN 978-3-319-13051-4. doi: 10.1007/978-3-319-13051-4_19.

[**77**] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto. Inside quantum repeaters. *IEEE Journal of Selected Topics in Quantum Electronics*, 21(3):78–90, 2015. doi: 10.1109/JSTQE.2015.2392076.

[**78**] Sreraman Muralidharan, Linshu Li, Jungsang Kim, Norbert Lütkenhaus, Mikhail D Lukin, and Liang Jiang. Optimal architectures for long distance quantum communication. *Scientific reports*, 6(1):1–10, 2016.

[**79**] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 10th anniversary ed edition, 2010. ISBN 978-1-107-00217-3.

[**80**] Georgios M. Nikolopoulos and Marc Fischlin. Information-theoretically secure data origin authentication with quantum and classical resources. *Cryptography*, 4(4), 2020. ISSN 2410-387X. doi: 10.3390/cryptography4040031.

[**81**] D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *2008 IEEE 68th Vehicular Technology Conference*, pages 1–5, 2008. doi: 10.1109/VETECF.2008.259. ISSN: 1090-3038.

[**82**] Nobel Lectures. Max Planck – biographical. *Nobel Lectures, Physics, 1901-1921*, 1967.

[**83**] A. Perrig, R. Canetti, J. D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, pages 56–73, 2000. doi: 10.1109/SECPRI.2000.848446.

[**84**] A Pirker and W Dür. A quantum network stack and protocols for reliable entanglement-based networks. *New Journal of Physics*, 21(3):033003, Mar 2019. ISSN 1367-2630. doi: 10.1088/1367-2630/ab05f7.

[**85**] A Pirker, J Wallnöfer, and W Dür. Modular architectures for quantum networks. *New Journal of Physics*, 20(5):053054, may 2018. doi: 10.1088/1367-2630/aac2aa.

[**86**] B. Preneel. *MAC Algorithms*, pages 361–368. Springer US, Boston, MA, 2005. ISBN 978-0-387-23483-0. doi: 10.1007/0-387-23483-7_240.

[**87**] NIST FIPS Pub. 180-2. *Secure Hash Standard", National Institute of Standards and Technology, US Department of Commerce*, 2002.

[**88**] A. M. Rashwan, A. M. Taha, and H. S. Hassanein. Benchmarking message authentication code functions for mobile computing. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 2585–2590, 2012. doi: 10.1109/GLOCOM.2012.6503506. ISSN: 1930-529X.

[**89**] Andreas Reiserer, Norbert Kalb, Machiel S Blok, Koen JM van Bemmelen, Tim H Taminiau, Ronald Hanson, Daniel J Twitchen, and Matthew Markham. Robust quantum-network memory using decoherence-protected subspaces of nuclear spins. *Physical Review X*, 6 (2):021040, 2016.

[**90**] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Request for Comments RFC 8446, Internet Engineering Task Force, August 2018. Num Pages: 160.

[**91**] Takahiko Satoh, Shota Nagayama, Shigeya Suzuki, Takaaki Matsuo, and Rodney Van Meter. Attacking the quantum internet. *arXiv:2005.04617 [quant-ph]*, 2020.

[**92**] Eddie Schoute, Laura Mancinska, Tanvirul Islam, Iordanis Kerenidis, and Stephanie Wehner. Shortcuts to quantum network routing, 2016.

[**93**] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[**94**] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x.

[**95**] Sooyeon Shin, Minwoo Kim, and Taekyoung Kwon. Experimental performance analysis of lightweight block ciphers and message authentication codes for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 13(11):1550147717744169, 2017. doi: 10.1177/1550147717744169.

[**96**] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. ISSN 0097-5397, 1095-7111. doi: 10.1137/S0097539795293172. arXiv: quant-ph/9508027.

[**97**] Victor Shoup. On fast and provably secure message authentication based on universal hashing. In *Annual International Cryptology Conference*, pages 313–328. Springer, 1996.

[**98**] Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, Lecture Notes in Computer Science, pages 313–328, Berlin, Heidelberg, 1996. Springer. ISBN 978-3-540-68697-2. doi: 10.1007/3-540-68697-5_24.

[**99**] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer International Publishing, Cham, 2016. ISBN 978-3-319-21936-3. doi: 10.1007/978-3-319-21936-3_13.

[**100**] FIP Standard. Advanced encryption standard (aes). *National Institute of Standards and Technology (NIST)*, 2001.

[**101**] Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77 (5):793, 1996.

[**102**] Tim Hugo Taminiau, Julia Cramer, Toeno van der Sar, Viatcheslav V Dobrovitski, and Ronald Hanson. Universal control and error correction in multi-qubit spin registers in diamond. *Nature nanotechnology*, 9(3):171–176, 2014.

[**103**] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. ISSN 1460-244X. doi: https://doi.org/10.1112/plms/s2-42.1.230.

[**104**] Dermot Turing. *X, Y & Z: The Real Story of How Enigma Was Broken*. The History Press, 2018.

[**105**] Nino Walenta, Andreas Burg, Dario Caselunghe, Jeremy Constantin, Nicolas Gisin, Olivier Guinnard, Raphaël Houlmann, Pascal Junod, Boris Korzh, Natalia Kulesza, et al. A fast and versatile quantum key distribution system with hardware key distillation and wavelength multiplexing. *New Journal of Physics*, 16(1):013047, 2014.

[**106**] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

[**107**] Stephanie Wehner. Quantum networks: From a physics experiment to a quantum network system. Association for Computing Machinery (ACM), Sep 2020.

[**108**] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412), 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aam9288.

[**109**] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[**110**] Horace P Yuen. Some physics and system issues in the security analysis of quantum key distribution protocols. *Quantum information processing*, 13(10):2241–2254, 2014.

[**111**] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. Cryptology ePrint Archive, Report 2012/076, 2012.

[**112**] Zheshen Zhang, Changchen Chen, Quntao Zhuang, Franco NC Wong, and Jeffrey H Shapiro. Experimental quantum key distribution at 1.3 gigabit-per-second secret-key rate over a 10 db loss channel. *Quantum Science and Technology*, 3(2):025007, 2018.