UNIVERSITÀ DEGLI STUDI DI PADOVA
Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Unconditionally secure authentication for quantum key distribution

Relatore: Prof. Nicola Laurenti

Candidato: Simon Calimani
matr.: 607259

13 dicembre 2011
A. A. 2011/2012

Abstract

This work describes the using of Universal and Strongly Universal classes of hash functions for unconditionally secure message authentication in quantum cryptography. Different classes are compared and constructions of flexible ε-Almost Strongly Universal classes are described. A new upper bound on the lifetime of a single hash function in one-time padded tags is introduced and optimisation on the final QKD key rate is shown for the QuAKE experiment, a B92 based QKD system. The public channel communication protocol of QuAKE is described, with special stress on the security issues.

A Rifca, che attendeva

Ringraziamenti

Un ringraziamento particolare va al prof. Laurenti per la sua estrema disponibilità, e senza la cui infinita competenza questo lavoro non sarebbe stato completo.

Ringrazio i miei genitori e mia sorella per l'ENORME pazienza dimostrata durante tutti questi anni.

Ringrazio Alice, per aver finto interesse per tutto quello che stavo facendo.

Ringrazio inoltre i compagni di laboratorio Matteo Canale, Francesco Renna, Diego Altolini, Nicola Dalla Pozza e Paolo Baracca per l'aiuto che mi hanno dato durante lo svolgimento della tesi, per il sostegno nei momenti bui, e per le risate.

Ringrazio, inoltre, tutti quelli che mi hanno sfamato portandomi in Forcellini e in Murialdo in questi mesi.

Contents

Chapter

1

Quantum cryptography

Cryptography is the science that studies the methods of making messages meaningless to unauthorised third parties. The usual manner to attain this purpose is to use algorithms, called ciphers, that process the input message and whose output, called cyphertext, is determined by some additional secret information, called the *key*. Obtaining the original message without knowing the key must be computationally very hard; on the contrary, extracting the message from the cyphertext and the key must be reasonably fast.[1] By means of cryptography, it is possible to transmit a message on a public channel without anybody being able to understand its meaning, given that both the sender and the receiver share a secret key that must be delivered on a different, secret, channel.

Symmetric key cryptography is based on cryptosystems where the same key is used both by the sender to encrypt the message, and by the receiver to decrypt the cyphertext. Examples of symmetric key cryptosystems are DES, AES and RC4.

Asymmetric key cryptography, also called public key cryptography, uses two different keys to perform the two phases of encryption and decryption, so one of the two keys can be made public, while the other one is kept secret. This can be used in two ways: if the public key is used to encrypt messages, the cyphertexts will be decodable only by the owner of the private key. In the other way, the owner of the private key can encrypt information that will be decodable by everybody: this allows what is called "digital signature", that is anyone can decrypt the cyphertexts and be reasonably sure that the message may only have been sent by the owner of the private key. The most widespread asymmetric key cryptosystem is RSA.

[1] In terms of complexity theory, the deciphering problem should lie within the nondeterministic polynomial time (NP) class, and outside the bounded-error probabilistic polynomial time (BPP) class [KL07].

1.1 Key distribution

The length of the key being equal, symmetric key cryptosystems are generally faster and safer, so usually asymmetric key cyphers are used only to secretly transmit temporary keys for faster symmetric cyphers. Without using key agreement protocols, first published by Diffie and Hellman in 1976 ([DH76]), or asymmetric key cyphers, the only way to exchange secret keys would be by physically meeting the other party of the communication, or by using a trusted courier.

Almost every cryptosystem[2] grounds the secrecy of the cyphertext on the computational limitations of the possible adversaries. Most symmetric cryptosystems could be forced if the adversary could produce all the possible messages by trying every possible key on a given cyphertext, and then choose the message which sounds more sensible: this operation is generally hard, because the number of all the possible keys is chosen to be very high.

All asymmetric cryptosystems and Diffie-Hellman based key exchange protocols rely on functions which are extremely easy to calculate, but computationally hard to invert without a "trapdoor". These functions are called *one-way functions* and exploit mathematical problems like the factorisation of large numbers in two prime numbers or the computation of the logarithm in finite fields, or operations on elliptic curves.

The main problem with these systems is that none of the mathematical problems they are based on has ever been proven to be really intractable[3], so it is possible that one day an algorithm will be found which allows to "crack" all the cyphertexts produced until then.

1.2 Quantum properties

A way to answer the need for a secure key distribution system comes from quantum physics.

Quantum particles obey some laws which can be used as an advantage towards an attacker. For example:

- it is not possible to perform measures on an unknown quantum state without perturbing it (*uncertainty principle*);
- it is not possible to duplicate an unknown quantum state (*no cloning theorem*);
- it is not possible to measure a photon on two nonorthogonal bases at the same time.

If we transmit information by means of single photons, it is possible to turn these laws into a way to check if the transmission has been observed by an adversary. We can

[2] One-Time Pad is an example of symmetric key cryptosystem which is not based on computational complexity to achieve the secrecy of the encrypted message. See paragraph 4.2.

[3] That is, not to belong to the NPP class.

encode one bit of information in the quantum state of each transmitted photon: this unit of information is called *qubit*, from "quantum bit".[4]

1.3 BB84

The first protocol for quantum key distribution is BB84, and was introduced by Bennett and Brassard in [BB84]. It is based on the transmission of single photons where the information is encoded into the state of polarisation. The party that sends the photons is conventionally called "Alice", while the receiver is called "Bob".

Alice and Bob choose two polarisation bases that are not orthogonal to each other, and in each basis they assign value "0" to one state and value "1" to the orthogonal state. Let us assume that the two nonorthogonal bases are 0°/90° and 45°/-45° and the chosen values are:

Alice chooses two random binary sequences of the same length: one for the bases and one for the binary values to transmit. Then Alice sends the single photons polarised according to the values of the two random sequences. At the receiving side, for every received photon, Bob must randomly choose one of the two bases to perform his measure. If the chosen basis is the same as the one chosen by Alice, the photon is correctly measured, while if the basis is the wrong one, the photon will give unpredictable random results, evenly distributed between 0 and 1.

After the quantum transmission, Alice can reveal on the public channel the actual sequence of bases used for the polarisations. Since she is only telling the bases, and not the polarisation state, she is not disclosing any additional information on the transmitted bits. That is, if the attacker (called "Eve") did not perform any measure on the transmitted photons, the sequence of chosen bases does not carry any information on the actual values coded by each photon.

When Bob receives the bases used by Alice, he can reply on the public channel with a list of positions where Bob performed a successful measure on the photon *and* the two bases coincide. He can also append a random list of some of the outcomes of his measures, so that Alice can check that the transmission has been successful. The revealed values will then be discarded from the list of secret bits.

The only way for an attacker to extract some information from the whole process, is to perform some measures on the polarised photons and then let these photons continue their way to the receiver, or send some other polarised photons in place of the

[4]Formally, a qubit is a two-dimensional vector space over the complex numbers.

intercepted ones. In any case, for every possible strategy chosen by the opponent, these measures will introduce either errors, or losses, or both, on the quantum channel, and by comparing some random values of the quantum transmission it is possible to detect intrusions with high probability.

See table 1.1 for an example of realisation of the BB84 protocol.

Positions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alice's bases	✣	✣	⤫	✣	⤫	✣	⤫	⤫	✣	⤫	✣	⤫	✣	✣	⤫
Values	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1
Quantum channel															
Polarisation	↕	↔	⤢	↕	⤡	↕	⤢	⤡	↔	⤡	↔	⤢	↔	↕	⤢
Bob's bases	⤫	✣	⤫	✣	✣	⤫	✣	⤫	⤫	⤫	⤫	✣	✣	⤫	⤫
Bob's measures	⤢	↔		↕	↔	⤡		⤡	⤢	⤡		↕	↔		⤢
Bob's values	1	0		1	0	0		0	1	0		1	0		1
Public channel															
Bases comparison		✓	✓	✓				✓		✓			✓		✓
Sifted values		0		1				0		0			0		1
Compared values				1									0		
Comparison result				✓									✓		
Final values		0						0		0					1

Table 1.1: BB84 example.

QKD cannot be efficiently used as a system to transmit a given data form the sender to the receiver, because the received data is an unpredictable subset of the transmitted bits without knowing the exact bases used by Alice. Yet, this uncertainty can be turned into an advantage towards a possible attacker. Even if the final data shared by Alice and Bob cannot be predicted at the beginning of the whole process, the important fact is that Alice and Bob share the *very same* data, which, therefore, can be used as a shared secret key.

1.4 B92

The present work has been developed to provide authentication to the QuAKE project, developed by QuantumFuture, a strategic project of Padova University. QuAKE has been designed to implement B92, a simple protocol introduced by Bennett in [Ben92]. B92 works with only two nonorthogonal states and does not require Alice and Bob to

reveal the sequence of chosen bases. A possible setup is the following. Alice encodes her random bits into the following polarisation quits:

Alice's encoding	
↕	0
⤢	1

Bob, then, detects the received photons with two single-photon avalanche diodes (SPAD) screened by two polarising filters, each one oriented along one of the two orthogonal polarisations ⟷ and ⤡. When one of the two SPADs clicks, it is impossible that the detected photon had a polarisation orthogonal to that of the corresponding filter, so the only possibility is that the photon had the only other possible polarisation. So Bob's decision rule is:

Bob's decision rule	
⟷	1
⤡	0

An example of B92 protocol is described in table 1.2.

Positions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Values	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1
Quantum channel															
Polarisation	⤢	↕	⤢	⤢	↕	⤢	⤢	↕	↕	↕	↕	⤢	↕	⤢	⤢
Bob's measures		↕	⤢	⤢		⤢	⤢	↕		↕		⤢	↕		⤢
Bob's values		0	1	1		1	1	0		0		1	0		1
Public channel															
Compared values				1									0		
Comparison outcome				✓									✓		
Final values		0	1			1	1	0		0		1			1

Table 1.2: B92 example.

1.5 Other protocols

The set of QKD protocols is not limited to polarisation-based protocols. Other QKD systems use entangled photons and phase interference. An extensive survey on QKD can be found in [GRTZ02].

1.6 QKD distillation phases

Since the quantum channel is not perfect, some quits may be wrongly detected even if the chosen basis is the right one. Alice and Bob, therefore, must correct the transmission errors through a classical transmission on the public channel, and at the same time they have to keep the information that Eve can gather on the final data (both from observations on the quantum channel and from the messages that have been transmitted on the public channel) as low as possible.

In other words, there are three main aims when generating secret keys:

- *correctness*, i.e. Alice's key must be equal to Bob's key;
- *secrecy*, i.e. Eve's information on the key must be zero;
- *uniformity*, i.e. the entropy of the final key must be maximum.

Let k_a be the final key obtained by Alice, let k_b be Bob's final key and let L be the length of k_a. Let z be a random variable that describes all the observations performed by Eve on the quantum channel, and c all the messages exchanged by Alice and Bob on the public channel. In practice it is required that there exist three values $\varepsilon', \varepsilon'', \varepsilon'''$, small enough, such that:

$$P[k_a \neq k_b] < \varepsilon' \qquad \text{(correctness)}$$
$$I(k_a, k_b; z, c) < \varepsilon'' \qquad \text{(secrecy)}$$
$$L - H(k_a) < \varepsilon''' \qquad \text{(uniformity)}$$

The main way to achieve these purposes is to divide the key processing performed by Alice and Bob in distinct phases: quantum transmission, sifting, key reconciliation, and privacy amplification. Each of these phases aims at a different target.

Quantum transmission

During the quantum transmission Alice shares some randomness with Bob. The main aim is to maximise the mutual information between the key sent by Alice (the *raw key*) and the key received by Bob.

Sifting

The aim of sifting is to distill the advantage of Alice and Bob with respect to Eve, that is to have a mutual information between Alice's and Bob's data which is greater than the mutual information between Alice's key and the attacker's observations.

After the quantum transmission, Alice and Bob must agree on the positions of the correctly transmitted photons. In the case of BB84, Alice (or Bob) transmits the sequence of chosen bases to the other party, which replies with a list of the positions where the

bases chosen by Alice agree with the bases chosen by Bob. In B92 it is sufficient that Bob sends a list of the positions of all the received photons. Then a randomly chosen sample of bits is chosen by one of the two parties and is revealed, so that it is possible to estimate the amount of losses and errors.

The output of this phase is called *sifted key*.

Key reconciliation

This phase aims at reducing the probabilities that Alice's and Bob's final keys are different.

After the sifting phase, Alice and Bob must correct all the errors occurred during the quantum transmission, using reconciliation protocols like Cascade (see [BS93]), Winnow ([BLT+03]) or protocols based on LDPC ([ELAB09], [MDMD10]).

The key resulting from this phase is called *reconciled key*.

Privacy amplification

The aim of privacy amplification is to reduce the information that Eve still has on the reconciled key below a given threshold. This can be achieved by estimating an upper bound to the knowledge that Eve may have collected during all the previous phases and by compressing the reconciled key with an appropriate function, like multiplying the key by a random Toeplitz matrix (see paragraph 3.1).

1.7 The need for authentication

In principle, an opponent could pose as one of the legitimate parties and perform both the quantum and the classic transmission pretending to be the intended counterpart, without raising any suspect. The only way to prevent this kind of attack is by authenticating the messages exchanged on the public classical channel through an authentication code. Since every authentication code requires some preshared keys, Alice and Bob cannot initiate a QKD session without some common secret information. Therefore, QKD should actually be called quantum key *growing*, since it produces arbitrarily long secret keys starting from a small amount of secure key material.

Chapter

2

Unconditionally secure authentication

2.1 Authentication codes

When we deal with QKD, if one of the messages transmitted through the public channel is altered, the whole system could be prone to man-in-the-middle attacks, or, more generally, to denial-of-service attacks that would aim at disrupting one or all of the stages of key sifting, reconciliation, and privacy amplification.

Authentication is the mean by which we address the issue of being sure that the transmitted information has not been forged or modified by a third party. With the right authentication system we can assure ourselves both that a message has been sent by the person that claims to be the sender (i.e. the message *authenticity*), and that the message has not been modified on the way to the receiver (i.e. the *integrity* of the message).

An authentication framework We want to be able to send one or more messages belonging to the message set $\mathcal{M}$ from a legitimate *sender* to a legitimate *receiver*, and we want the additional property that an *opponent*, which is not a legitimate party of the transmission, had a very low probability of forging a fake message or substituting a legitimate message with a fake message without being detected. The situation where the opponent simply inserts a fake message into the channel is called an *impersonation attack*, while when the opponent intercepts a correct message and sends a forged message in place of the legitimate one is called a *substitution attack*.

For a chosen authentication system, we call P_I the *maximum* success probability over all impersonation attacks, and we define P_S as the *maximum* probability of success over all substitution attacks.

One may wonder if there exist an optimum authentication scheme for a given key length. Lower bounds were derived in [Mau00] for the success probability of impersonation and substitution attacks on a system with multiple messages authentication over a public error-free channel. By denoting with k the secret key shared by Alice and Bob,

with $m_1, \ldots, m_i$ the messages to be authenticated, and with $x_1, \ldots, x_i$ the authenticated messages transmitted over the channel, the following bound for the impersonation attack at the j-th message authentication holds [Mau00, Theorem 3]

$$P_I(j) \geq \frac{1}{2^{I(k;x_j|x_1,\ldots,x_{j-1})}} \qquad \forall\, j = 1, \ldots, i \tag{2.1}$$

whereas for the substitution attack [Mau00, Theorem 6]

$$P_S(j) \geq \frac{1}{2^{H(k|x_1,\ldots,x_{j-1},x_j)}} \geq \frac{1}{2^{H(k|x_1,\ldots,x_j)}} \qquad \forall\, j = 1, \ldots, i \tag{2.2}$$

The above bounds suggest partitioning the key k into $i+1$ independent and uniform subkeys $k_0, \ldots, k_i$, so that

$$x_j = f(m_j, k_0, k_j) \qquad \forall\, j = 1, \ldots, i \tag{2.3}$$

and $f(m_j, k_0, \cdot)$ is a one-to-one function of k_j for any value of m_j, k_0. A possible way to implement (2.3) is to use a simple OTP scheme, where some subvector t_j of x_j is given by

$$t_j = g(m_j, k_0) \oplus k_j \qquad \forall\, j = 1, \ldots, i \tag{2.4}$$

Then, from (2.3) we have

$$\begin{aligned} I(k; x_j|x_1, \ldots, x_{j-1}) = I(k_j; x_j) = H(x_j) - H(x_j|k_j) = \\ = H(k_j) = log_2(K_j) \qquad \forall\, i = 1, \ldots, n \end{aligned} \tag{2.5}$$

and

$$H(k|x_1, \ldots, x_i) = H(k_0) log_2(K_0) \tag{2.6}$$

where K_j is the number of possible values for k_j.

Observe that in this case, the lower bound in eq. (2.1) becomes

$$P_I(j) \geq \frac{1}{K_j} \qquad \forall\, j = 1, \ldots, i \tag{2.7}$$

and is attained by the OTP scheme (2.4), which is therefore optimal in this respect.

On the other hand, the bound in eq. (2.2), which becomes

$$P_S(j) \geq \frac{1}{K_0} \qquad \forall\, j = 1, \ldots, i \tag{2.8}$$

is derived under the assumption of a generic attack aimed at recovering the key, and is believed to be rather loose, when $H(k_0) < H(m_j)$.

Nevertheless, we consider systematic authentication schemes of the type (2.3)-(2.4) where $x_j = (m_j, t_j)$, g is a keyed hash function with m_j as input, k_0 as key, and tag t_j of the same length as k_j.

Therefore, the sender and the receiver share a secretly chosen encoding, and use it to generate and check the tags. The opponent does not know which is the selected encoding, but does know the complete description of all the possible encodings, their statistical distribution and the statistical distribution of the messages.

Therefore, if we define a couple (m, t) to be *valid* if t is the correct tag for the message m with the currently selected encoding, we can define:

$$P_I = \max_{\substack{\hat{m}\in\mathcal{M}\\ \hat{t}\in\mathcal{T}}} P[(\hat{m}, \hat{t}) \text{ is valid}]$$

$$P_S = \max_{\substack{\hat{m}\neq m\in\mathcal{M}\\ \hat{t},t\in\mathcal{T}}} P[(\hat{m}, \hat{t}) \text{ is valid}|(m, t) \text{ has been observed}]$$

We can view all the possible encodings as a multiset of *keyed hash functions*, that is, a pool $\mathcal{F}$ of hash functions, each of which is uniquely identified by an index. The sender and the receiver can agree on the selected hash function by sharing a secret key k chosen from the set $\mathcal{K}$ of all the possible indexes.

If the chosen hash function family exhibits some "good properties", the values P_I and P_S will be extremely low, and therefore it will be difficult[1] for an attacker to forge the correct tag for a chosen message.

It's important to stress the fact that in the current authentication model we are not making any assumption on the computational capabilities of the opponents, such as computer speed and storage memory, or time available for the attack. This is in contrast with all the those security methods based on the difficulty of solving given problems, for instance the factorisation of large prime numbers.

Our measure of security, that is, the limits on P_I and P_S that will be shown in this work, relies only on combinatorial features which are intrinsic traits of the classes of hash function that we will study, and for this reason this authentication is called *unconditionally secure*.

The other important aspect to stress is that our hash functions are not related to the well known *cryptographic hash function*, which are generally deterministic functions (not indexed), and which have features we are not interested in: we are not concerned by preimage resistance or collision resistance of the single hash functions of our classes. On the contrary, most of our hash functions exhibit a very simple structure.

General properties of authentication codes Let (m, t) be a valid couple formed by a message and its tag. When an attacker intercepts (m, t), it's easy to get the set of all the possible hash functions of the class $\mathcal{F}$ that could have generated t from m:

$$\mathcal{F}(m, t) \triangleq \{h \in \mathcal{F} : h(m) = t\}$$

[1] In all this work, the word *difficult*, when referred to an attack, means that the probability of success is extremely low, and is *not* related either to the difficulty of calculations or to the space and time taken by the computation.

Since all the hash functions are injective, it's easy to prove that, for a given message m taken from the message set $\mathcal{M}$, the sets $\mathcal{F}(m,t)$ form a partition of $\mathcal{F}$.

Now we can write explicitly the probability of impersonation, under the hypothesis that the hash functions are uniformly distributed:

$$P_I = \max_{\substack{m\in\mathcal{M}\\ t\in\mathcal{T}}} P[(m,t) \text{ is valid}] = \max_{\substack{m\in\mathcal{M}\\ t\in\mathcal{T}}} \frac{|\mathcal{F}(m,t)|}{|\mathcal{F}|} \tag{2.9}$$

There exist one or more couples (m,t) that maximise the right term which are the weakest couples of the code, that is, the couples that have exactly probability P_I of being valid.

The probability of substitution becomes:

$$\begin{aligned} P_S &= \max_{\substack{\hat{m}\neq m\in\mathcal{M}\\ \hat{t},t\in\mathcal{T}}} P[(\hat{m},\hat{t}) \text{ is valid}|(m,t) \text{ has been observed}] = \\ &= \max_{\substack{\hat{m}\neq m\in\mathcal{M}\\ \hat{t},t\in\mathcal{T}}} \frac{P[\mathcal{F}(\hat{m},\hat{t})\cap\mathcal{F}(m,t)]}{P[\mathcal{F}(m,t)]} = \\ &= \max_{\substack{\hat{m}\neq m\in\mathcal{M}\\ \hat{t},t\in\mathcal{T}}} \frac{|\{h\in\mathcal{F} : h(m)=t \wedge h(\hat{m})=\hat{t}\}|}{|\{h\in\mathcal{F} : h(m)=t\}|} \end{aligned} \tag{2.10}$$

In this case there is no determined couple that attains the maximum P_S, but one (or more) *pair of couples* $((\hat{m},\hat{t}),(m,t))$ that gives P_S. In general, for each legitimate couple (m,t) observed by the opponent, there are one or more couples that maximise the probability of being valid, but in any case this probability will be bounded by P_S.

2.2 Universal hashing

The concept of Universal hash functions was introduced in 1977 by Carter and Wegman in [CW77].

Definition 2.1. Let $\mathcal{F}$ be a class of hash functions from a set of messages $\mathcal{M}$ to a set of tags $\mathcal{T}$.

Let $F = |\mathcal{F}|$ and $N = |\mathcal{T}|$.

A family of hash functions is ε-Almost Universal (or ε-AU$_2$) if

$$|\{h\in\mathcal{F} : h(m_1)=h(m_2)\}| \leq \varepsilon F \qquad \forall\, m_1,m_2\in\mathcal{M}, \quad m_1\neq m_2$$

If $\varepsilon = \frac{1}{N}$, $\mathcal{F}$ is simply called Universal (or U$_2$). □

If we define a *collision* as the case where a certain hash function gives the same tag for two different messages, a U$_2$ class of hash function guarantees that the number of

collisions generated by a single function is bounded from above.[2] Moreover, if we assume to take the keys k uniformly over $\mathcal{K}$, ε is the collision probability between the hash values of two different messages.

It's important to notice that a U_2 class alone is *not* sufficient to give a valid authentication code.

Consider the following example:

Example 2.1. Let $\mathcal{M} = \mathcal{T} = \{0,1\}^a$, $\mathcal{K} = \{0,1,\ldots,a-1\}$ and let $\mathcal{F}$ be the class of functions:

$$h_k(m) = (t(1),\ldots,t(a)), \qquad t(i) = m((i+k) \mod a) \quad \forall\, i = 1,\ldots,a$$

It's easy to see that such a class is U_2, in fact two messages collide under the same hash function only if they are equal. □

The number of collisions for each couple of different messages is 0, but forging a tag for a fake message is very easy. If the original message m is not periodic, it suffices intercepting the couple (m,t) to recover the secret key k. Now we can forge a valid couple $(\hat{m},\hat{t})$ that will be accepted as a valid message by the receiver. This attack is possible because ε-AU_2 hash functions do not require any feature that allow to limit the probability of substitution P_S or the probability of impersonation P_I.

2.3 Strongly universal hashing

In [WC81], Wegman and Carter introduce a new kind of hash function class: [3]

Definition 2.2. Let $\mathcal{F}$ be a class of hash functions from a set of messages $\mathcal{M}$ to a set of tags $\mathcal{T}$. Let $\varepsilon \in (0,1)$.

Let $F = |\mathcal{F}|$ and $N = |\mathcal{T}|$.

A family of hash functions is ε-Almost Strongly Universal (or ε-ASU_2) if

1. $|\{h \in \mathcal{F} : h(m) = t\}| = \frac{F}{N} \qquad \forall\, m \in \mathcal{M}, \quad \forall\, t \in \mathcal{T}$
2. $|\{h \in \mathcal{F} : h(m_1) = t_1, h(m_2) = t_2\}| \leq \varepsilon \frac{F}{N} \quad \forall\, m_1 \neq m_2 \in \mathcal{M}, \forall\, t_1, t_2 \in \mathcal{T}$

If $\varepsilon = 1/N$ the class is simply called SU_2. □

These properties are crucial for our goal: let's suppose an opponent has observed a valid (m,t) couple. Now the attacker is able to identify the subset $\mathcal{F}'$ of hash functions that could have generated our t. Property 1 gives us the exact cardinality of this subset, so that (if $F \geq 2N$) it will be impossible to identify the right function with certainty. But

[2] The upper bound given by the definition is not the lowest possible: in [Sar80] an *optimally universal* (OU) class of hash functions is defined as a ε-AU_2-class with a collision number bounded by $\frac{M-N}{mn-N}$.

[3] In [WC81] Wegman and Carter define only SU_2 classes, but they describe the concept of ε-ASU_2 (calling it "almost strongly universal$_2$") without formalising it.

there could exist a "lucky" message $\hat{m}(\neq m)$ that is mapped to a single value $\hat{t}$ by each hash function $h \in \mathcal{F}'$. If this was the case, we would not need to find out the exact hash function, but we could be able to forge a valid couple $(\hat{m}, \hat{t})$ that would not be detected as a fake one.

Property 2 is what addresses the issue noted above. It states that, for every subset $\mathcal{F}'$ identified by a legitimate couple (m, t), the number of hash functions in $\mathcal{F}'$ which map a given message $\hat{m} \neq m$ into the same tag $\hat{t}$ is upper bounded to be a fraction ε of $|\mathcal{F}'|$. If ε is much smaller than 1, the success probability P_S of a substitution attack is very low, namely:

$$P_S \leq \varepsilon$$

where the maximum is attained by choosing a couple $(\hat{m}, \hat{t})$ that maximises $|\{h \in \mathcal{F} : h(m) = t, h(\hat{m}) = \hat{t}\}|$.

In other words, the two properties of ε-ASU$_2$ hash function classes allow to give an upper bound on the two probabilities P_I and P_S (see eqs. (2.9) and (2.10)).

The concept of SU$_2$ can be generalised to prevent substitution attacks based on the observation of up to $l-1$ valid couples (m, t).

Definition 2.3. [WC81] Let $\mathcal{F}$ be a class of hash functions from a set of messages $\mathcal{M}$ to a set of tags $\mathcal{T}$. Let $\varepsilon \in (0, 1)$.

Let $F = |\mathcal{F}|$ and $N = |\mathcal{T}|$.

A family of hash functions is Strongly Universal$_l$ (or SU$_l$) if, for every choice of l different values $m_1, \ldots, m_l \in \mathcal{M}$ and l (not necessarily different) values $t_1, \ldots, t_l \in \mathcal{T}$, there are exactly F/N^l functions which map m_i into t_i for each $i = 1, \ldots, l$.

Orthogonal arrays and SU$_2$ class

One method to get a strict SU$_2$ class is to build an *orthogonal array*.

Definition 2.4. [Bie96] An *orthogonal array* with parameters OA$_\lambda(l, M, N)$ is a multiset $\mathcal{F}$ of mappings from a M-set $\mathcal{M}$ into a N-set $\mathcal{T}$ such that for every choice of l distinct elements $m_1, m_2, \ldots, m_l \in \mathcal{M}$ and l (not necessarily distinct) elements $t_1, t_2, \ldots, t_l \in \mathcal{T}$ there are exactly λ elements $h \in \mathcal{F}$ affording the operation $h(m_i) = t_i \forall i = 1, \ldots, l$.

$\mathcal{F}$ can be visualised as a matrix where every row i corresponds to the message m_i, and every column j to a different mapping h_j. The elements of the matrix are the values $h_j(m_i)$.

It is easy to see that an OA$_\lambda(l, M, N)$ can be used as an authentication code for the set of messages $\mathcal{M}$ and the set of tags $\mathcal{T}$. More exactly, this is a SU$_l$ class of hash functions.

If we put $l = 2$, our OA$_\lambda(2, M, N)$ is described by a matrix with M rows and λN^2 columns [Sti88]. A lower bound for the value λ can be found in [PB45] and is given by

$$\lambda \geq \frac{M(N-1)+1}{N^2} \tag{2.11}$$

Moreover, in [Sti94a] Stinson shows that a SU_2 class $\mathcal{F}$ of hash functions from $\mathcal{M}$ to $\mathcal{T}$ reaches its minimum size[4] $|\mathcal{F}| = 1 + M(N-1)$ if and only if there exists a $OA_\lambda(2, M, N)$ with $\lambda = \frac{M(N-1)+1}{N^2}$.

With the help of this parallel, we can see that, keeping the same N to leave unchanged the probabilities of success of the opponent's attacks, the number of messages that can be authenticated by a SU_2 hash function class grows only linearly with the size of the class of hash functions. In fact we can derive by equation (2.11) that $\lambda N^2 \geq M(N-1)+1$. Recalling that in our optimal SU_2 construction $|\mathcal{F}| = \lambda N^2$, we obtain the bound

$$|\mathcal{F}| \geq \lambda N^2$$

Wegman and Carter in [WC81] note that if we don't require the probability of success P_S to be the theoretical minimum $\frac{1}{N}$, but we let it reach $\frac{1}{N} + \delta$, the size of the message set $\mathcal{M}$ grows exponentially.

2.4 Δ-universal hashing

Δ-universal families of hash functions (ε-ΔU) are a generalisation of Xor-universal families (which will be described subsequently) and their definition is due to Stinson in [Sti96].

Definition 2.5. Let $\mathcal{F}$ be a class of hash functions from a set of messages $\mathcal{M}$ to a set of tags $\mathcal{T}$ which is an additive abelian group. Let $\varepsilon \in (0, 1)$.

Let $F = |\mathcal{F}|$.

A family of hash functions is ε-Δ Universal (or ε-ΔU) if

$$|\{h \in \mathcal{F} : h(m_1) - h(m_2) = t\}| \leq \varepsilon F \qquad \forall\, m_1 \neq m_2 \in \mathcal{M}, \quad \forall\, t \in \mathcal{T}$$

□

Xor-universal hashing

This kind of hash functions have been first described by Krawczyk in [Kra94] who called them *ε-otp-secure* hash functions. Rogaway renamed them ε-Xor-universal in [Rog95].

A family of hash functions is defined Xor-universal if it is Δ-universal and $\mathcal{F} = (\mathbb{Z}_2)^l$ for some l.

In this case, the addition operation of the abelian group is the bitwise exclusive-or.

[4] see theorem 2.13 with $\varepsilon = 1/N$

2.5 Authentication and channel coding

The similarities between Authentication codes and forward error correcting codes have been noticed independently by Bierbrauer in [Bie97a] and by Johansson, Kabatianskii and Smeets in [JKS93]. The goal of a well designed authentication code is to have the minimum number of collisions among tags of different messages, generated by all the possible hash functions. The key observation is that such a requirement is the same as getting a high minimum distance between the words of a forward error correction code.

Let's formalise this observation. Suppose that we have a family of keyed hash functions (e.g. an ε-AU$_2$). We can index every hash function with the index $j \in [1, F]$, and every possible message with the index $i \in [1, M]$. The codomain of our hash function is $\{0,1\}^n$. We can think of the different tags as the symbols of our code, so we have an alphabet with cardinality 2^n. If, for each message m, we concatenate all the hash values generated by all the hash functions applied to m, ordered by index, we obtain the word that will code for the message m. Namely, if we call C the coding function:

$$C(m) = (h_1(m), h_2(m), \ldots, h_{F-1}(m), h_F(m))$$

Every codeword is formed by $2^n \cdot F$ bits, but we will not be required to ever generate the whole bit string.

Now we can choose two different messages m_1 and m_2 and see how many collisions there are between their hash values: it is sufficient to compare every symbol with the same index in their codewords. If they have the same symbol (that is, the same hash value) in the same position, we found a collision. So the number of symbols that do not get to a collision is the *distance* between the two codewords.

The *minimum distance* of a code is defined as

$$d = \min_{m_1 \neq m_2} d_H(C(m_1), C(m_2))$$

where d_H is the Hamming distance.[5] Usually the relative distance is defined as d/n, where n is the number of symbols in one codeword, so in our case it is in fact d/F.

Therefore, calling ε the ratio between the number of collisions and the total number of functions, we can state that:

$$\varepsilon = 1 - \frac{d}{F}$$

so an ε-AU$_2$ hash function family with cardinality F identifies a code with minimum distance $F(1-\varepsilon)$.

On the contrary, if we have a FEC code C with minimum distance d and block length F, we can use it, for the set of messages accepted by C, just as if it were an ε-AU$_2$ class. To generate the hash value for the message m, we choose a secret index k and take the k-th symbol as the hash value for m. For what we have just seen, this way of generating hash

[5] Note that the code symbols are not the single bits, but the hash values.

values forms exactly an ε-AU$_2$ hash function family, because it complies with definition 2.1.

We don't have, in principle, an actual construction for the generic hash function indexed by k: we could be required to calculate the whole codeword $C(m)$ before being able to select the only symbol k. To get a viable way of quickly hashing messages, we must find a code that allows us to limit the computational load by selectively generating single symbols of the codeword.

2.6 Relationships

It is possible to identify some useful relations among these hash function families.

Theorem 2.1. *[Sti96] If $\mathcal{F}$ is ε-ASU$_2$,*

1. *it is ε-AU$_2$.*
2. *Moreover, if $\mathcal{T}$ is an abelian group, $\mathcal{F}$ is ε-ΔU.*

Proof. 1. To show the first result we use property 1 of definition 2.2. Choosing a fixed value t, if we put $t_1 = t_2$ we get that

$$|\{h \in \mathcal{F} : h(m_1) = h(m_2) = t\}| \leq \varepsilon \frac{F}{N}$$

Since we have N different choices of the value t, we find that

$$|\{h \in \mathcal{F} : h(m_1) = h(m_2)\}| = \\ = \sum_{t \in \mathcal{T}} |\{h \in \mathcal{F} : h(m_1) = h(m_2) = t\}| \leq \varepsilon \frac{F}{N} = N \cdot \varepsilon \frac{F}{N} = \varepsilon F$$

2. For a given $m_1 \neq m_2 \in \mathcal{M}$ and $t \in \mathcal{T}$ we have:

$$|\{h \in \mathcal{F} : h(m_1) - h(m_2) = t\}| = \\ = \sum_{\bar{t} \in \mathcal{T}} |\{h \in \mathcal{F} : h(m_1) = t + \bar{t} \ \wedge \ h(m_2) = \bar{t}\}| \leq N \cdot \varepsilon \frac{F}{N} = \varepsilon F$$

So $\mathcal{F}$ is ε-ΔU.

□

Theorem 2.2. *[Sti96] If $\mathcal{F}$ is ε-ΔU, it is ε-AU$_2$.*

Proof. If we let $t = 0$ in the definition of ε-ΔU families, we get:

$$|\{h \in \mathcal{F} : h(m_1) - h(m_2) = 0\}| = |\{h \in \mathcal{F} : h(m_1) = h(m_2)\}| \leq \varepsilon F$$

□

Theorem 2.3. *[Sti96] If there exists a ε-ΔU hash function family $\mathcal{G}$ of cardinality G that maps $\mathcal{M}$ into $\mathcal{T}$, with $M = \mathcal{M}$ and $N = \mathcal{T}$ then there exists a ε-ASU$_2$ hash function family $\mathcal{F}$ of cardinality $F = Fn$ that maps $\mathcal{M}$ into $\mathcal{T}$.*

Proof. Let $h_{(f,\theta)}$ be the function defined by:

$$h_{(f,\theta)}(m) = f(m) + \theta$$

Let $\mathcal{F} = \{h_{(f,\theta)} \quad \forall f \in \mathcal{G}, \theta \in \mathcal{T}\}$. We must now prove the two properties of definition 2.2. Let $|\mathcal{G}| = G$ and $|\mathcal{F}| = F$. We have that $F = G \cdot N$.

1. We use the fact that $\mathcal{T}$ is an abelian group.

$$|\{h_{(f,\theta)} \in \mathcal{F} : h_{(f,\theta)}(m) = t\}| = |\{(f,\theta) : f(m) + \theta = t\}| =$$
$$= \sum_{\theta \in \mathcal{T}} |\{f : f(m) = t - \theta\}| = \sum_{\bar{t} \in \mathcal{T}} |\{f : f(m) = \bar{t}\}| = G = \frac{F}{N}$$

2. We must prove that $|\{h_{(f,\theta)} \in \mathcal{F} : h_{(f,\theta)}(m_1) = t_1 \wedge h_{(f,\theta)}(m_2) = t_2\}| \leq \varepsilon \frac{F}{N}$

$$\begin{aligned}
&|\{h_{(f,\theta)} \in \mathcal{F} : h_{(f,\theta)}(m_1) = t_1 \wedge h_{(f,\theta)}(m_2) = t_2\}| = \\
&= |\{(f,\theta) : f(m_1) + \theta = t_1 \wedge f(m_2) + \theta = t_2\}| = \\
&= |\{(f,\theta) : f(m_1) - f(m_2) = t_1 - t_2 \wedge f(m_1) = t_1 - \theta\}| = \\
&= \sum_{\theta \in \mathcal{T}} |\{f \in \mathcal{G} : f(m_1) - f(m_2) = t_1 - t_2 \wedge f(m_1) = t_1 - \theta\}| = \\
&= \sum_{\bar{t} \in \mathcal{T}} |\{f \in \mathcal{G} : f(m_1) - f(m_2) = t_1 - t_2 \wedge f(m_1) = \bar{t}\}| = \\
&= |\{f \in \mathcal{G} : f(m_1) - f(m_2) = t_1 - t_2\}| \leq \varepsilon G = \varepsilon \frac{F}{N}
\end{aligned}$$

□

2.7 Compositions

To build a good authentication function, it is not necessary to study an ε-ASU$_2$ from scratch. We can instead compose different families of one or more classes of hash functions to obtain a new family with characteristics which are different from the formers. In this way we can focus our efforts on improving different characteristics of the components of a hash function family, and the best hash functions are in fact designed taking advantage of the following results.

Theorem 2.4. *[Sti94b] Let $\mathcal{F}$ be an ε-AU$_2$ class of F hash functions from $\mathcal{M}$ to $\mathcal{T}$. Then there exist an ε-AU$_2$ class $\mathcal{F}^i$ of F^i hash functions from $\mathcal{M}^i$ to $\mathcal{T}^i$.*

Proof. For each $h \in \mathcal{F}$, we define:

$$\begin{aligned} h^i : \quad & \mathcal{M}^i \to \mathcal{T}^i \\ & (m_1, \ldots, m_i) \mapsto (h(m_1), \ldots, h(m_i)) \end{aligned}$$

Let $\mathcal{F}^i = \{h^i : h \in \mathcal{F}\}$. □

Theorem 2.5. *[Sti94b] Let $\mathcal{F}_1$ be an ε_1-AU_2 class of F_1 hash functions from $\mathcal{M}_1$ to $\mathcal{T}_1$, and let $\mathcal{F}_2$ be an ε_2-AU_2 class of F_2 hash functions from $\mathcal{M}_2 = \mathcal{T}_1$ to $\mathcal{T}_2$. Then there exist an ε-AU_2 class $\mathcal{F}$ of F hash functions from $\mathcal{M}_1$ to $\mathcal{T}_2$ with*

$$\varepsilon = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$$

and

$$F = F_1F_2$$

Proof. For each couple $(h_i, h_j) \in \mathcal{F}_1 \times \mathcal{F}_2$, we define:

$$\begin{aligned} h_{i,j} : \quad & \mathcal{M}_1 \to \mathcal{T}_2 \\ & m \mapsto h_j(h_i(m)) \end{aligned}$$

Let $\mathcal{F} = \{h_{i,j} : (h_i, h_j) \in \mathcal{F}_1 \times \mathcal{F}_2\}$. □

Theorem 2.6. *[Sti94b] Let $\mathcal{F}_1$ be an ε_1-AU_2 class of F_1 hash functions from $\mathcal{M}$ to $\mathcal{T}_1$, and let $\mathcal{F}_2$ be an ε_2-ASU_2 class of F_2 hash functions from $\mathcal{T}_1$ to $\mathcal{T}_2$. Then there exist an ε-ASU_2 class $\mathcal{F}$ of F hash functions from $\mathcal{M}_1$ to $\mathcal{T}_2$ with*

$$\varepsilon = \varepsilon_1 + \varepsilon_2$$

and

$$F = F_1F_2$$

Proof. For each couple $(h_i, h_j) \in \mathcal{F}_1 \times \mathcal{F}_2$, we define:

$$\begin{aligned} h_{i,j} : \quad & \mathcal{M}_1 \to \mathcal{T}_2 \\ & m \mapsto h_j(h_i(m)) \end{aligned}$$

Let $\mathcal{F} = \{h_{i,j} : (h_i, h_j) \in \mathcal{F}_1 \times \mathcal{F}_2\}$. Let's prove the two properties of definition 2.2:

1. For a given couple (m, t):

$$\begin{aligned} & |\{h \in \mathcal{F} : h(m) = t\}| = |\{(h_1, h_2) \in \mathcal{F}_1 \times \mathcal{F}_2 : h_2(h_1((m)) = t\}| = \\ & = \sum_{\bar{t}} |\{(h_1, h_2) \in \mathcal{F}_1 \times \mathcal{F}_2 : h_1(m) = \bar{t} \wedge h_2(\bar{t}) = t\}| = \\ & = \sum_{\bar{t}} |\{h_1 \in \mathcal{F}_1 : h_1(m) = \bar{t}\}| \cdot \frac{F_2}{N} = \frac{F_1F_2}{N} = \frac{F}{N}. \end{aligned}$$

2. We have to prove that $|\{h \in \mathcal{F} : h(m_1) = t_1, h(m_2) = t_2\}| \leq \varepsilon \frac{F}{N} \quad \forall\, m_1 \neq m_2 \in \mathcal{M}, \forall\, t_1, t_2 \in \mathcal{T}$. First let's suppose that $t_1 = t_2 = t$:

$$
\begin{aligned}
&|\{h \in \mathcal{F} : h(m_1) = h(m_2) = t\}| = \\
&= |\{(h_1, h_2) \in \mathcal{F}_1 \times \mathcal{F}_2 : h_2(h_1(m_1)) = h_2(h_1(m_2)) = t\}| = \\
&= \sum_{\hat{t} \in \mathcal{T}_1} |\{(h_1, h_2) \in \mathcal{F}_1 \times \mathcal{F}_2 : h_1(m_1) = h_1(m_2) = \hat{t} \wedge h_2(\hat{t}) = t\}| + \\
&+ \sum_{\bar{t}_1 \neq \bar{t}_2} \Big(|\{h_1 \in \mathcal{F}_1 : h_1(m_1) = \bar{t}_1, h_1(m_2) = \bar{t}_2\}| \cdot |\{h_2 \in \mathcal{F}_2 : h_2(\bar{t}_1) = h_2(\bar{t}_2) = t\}| \Big) \leq \\
&= \sum_{\hat{t} \in \mathcal{T}_1} \Big(|\{h_1 \in \mathcal{F}_1 : h_1(m_1) = h_1(m_2) = \hat{t}\}| \cdot |\{h_2 \in \mathcal{F}_2 : h_2(\hat{t}) = t\}| \Big) + \\
&+ F_1 \cdot |\{h_2 \in \mathcal{F}_2 : h_2(\bar{t}_1) = h_2(\bar{t}_2)\}| \leq \\
&\leq \varepsilon_1 F_1 \cdot \frac{F_2}{N} + F_1 \cdot \varepsilon_2 \frac{F_2}{N} = (\varepsilon_1 + \varepsilon_2) \frac{F_1 F_2}{N} = \varepsilon \frac{F}{N}
\end{aligned}
$$

The second case is when $t_1 \neq t_2$:

$$
\begin{aligned}
&|\{h \in \mathcal{F} : h(m_1) = t_1, h(m_2) = t_2\}| = \\
&= |\{(h_1, h_2) \in \mathcal{F}_1 \times \mathcal{F}_2 : h_2(h_1(m_1)) = t_1, h_2(h_1(m_2)) = t_2\}| = \\
&= \sum_{\bar{t}_1 \neq \bar{t}_2} \Big(|\{h_1 \in \mathcal{F}_1 : h_1(m_1) = \bar{t}_1, h_1(m_2) = \bar{t}_2\}| \cdot \\
&\qquad \cdot |\{h_2 \in \mathcal{F}_2 : h_2(\bar{t}_1) = t_1, h_2(\bar{t}_2) = t_2\}| \Big) \leq \\
&\leq \sum_{\bar{t}_1 \neq \bar{t}_2} \Big(|\{h_1 \in \mathcal{F}_1 : h_1(m_1) = \bar{t}_1, h_1(m_2) = \bar{t}_2\}| \Big) \cdot \varepsilon_2 \frac{F_2}{N} \leq \\
&\leq F_1 \cdot \varepsilon_2 \frac{F_2}{N} \leq (\varepsilon_1 + \varepsilon_2) \frac{F_1 F_2}{N} = \varepsilon \frac{F}{N}
\end{aligned}
$$

□

2.8 Bounds

To analyse the features of our hash functions, we must introduce some bounds on the cardinality of specific families.

Bounds for AU_2 classes

Theorem 2.7. *[Sar80] Let $\mathcal{F}$ be an ε-AU_2 class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$\varepsilon \geq \frac{M - N}{N(M - 1)}$$

Theorem 2.8. *[Sti94b] Let $\mathcal{F}$ be an ε-AU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$F \geq \frac{M(N-1)}{M(\varepsilon N - 1) + N^2(1-\varepsilon)}$$

Theorem 2.9. *[NR10] Let $\mathcal{F}$ be an ε-AU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$F \geq \frac{1}{\varepsilon}\left\lceil \frac{\log_2 M}{\log_2 N} - 1 \right\rceil$$

Bounds for ΔU$_2$ classes

Theorem 2.10. *[Sti96] Let $\mathcal{F}$ be an ε-ΔU class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$\varepsilon \geq \frac{1}{N}$$

Theorem 2.11. *[Sti96] Let $\mathcal{F}$ be an ε-ΔU class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$F \geq \frac{M(N-1)}{N - M + N\varepsilon(M-1)}$$

Theorem 2.12. *[NR10] Let $\mathcal{F}$ be an ε-ΔU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. If*

$$\log_2 M < \log_2 N \cdot \left(\sqrt{2H\left(1 - \frac{1}{N}\right)} - \frac{1}{2} \right)$$

then

$$F \geq \frac{1}{\varepsilon}\left\lfloor \frac{\log_2 M}{\log_2 N} \right\rfloor$$

Bounds for ASU$_2$ classes

Theorem 2.13. *[Sti94b] Let $\mathcal{F}$ be an ε-ASU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$F \geq 1 + \frac{M(N-1)^2}{N\varepsilon(M-1) + N - M}$$

Theorem 2.14. *[GN93] Let $\mathcal{F}$ be an ε-ASU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. Then*

$$F \geq -\frac{M}{\varepsilon^2 \log \varepsilon} \tag{2.12}$$

	$\varepsilon < \left(1 + \frac{\mathbf{log_2}\, \boldsymbol{N}}{\mathbf{log_2}\, \boldsymbol{M} - \mathbf{log_2}\, \boldsymbol{N}}\right) \frac{\mathbf{1}}{\boldsymbol{N}}$	$\varepsilon > \left(1 + \frac{\mathbf{log_2}\, \boldsymbol{N}}{\mathbf{log_2}\, \boldsymbol{M} - \mathbf{log_2}\, \boldsymbol{N}}\right) \frac{\mathbf{1}}{\boldsymbol{N}}$
ε-AU	$\frac{M(N-1)}{M(\varepsilon N - 1) + N^2(1-\varepsilon)}$	$\frac{1}{\varepsilon}\left\lceil \frac{\log_2 M}{\log_2 N} - 1 \right\rceil$
ε-ΔU	$\frac{M(N-1)}{N - M + N\varepsilon(M-1)}$	$\frac{1}{\varepsilon}\left\lfloor \frac{\log_2 M}{\log_2 N} \right\rfloor$ if $\log_2 M < \log_2 N \cdot \left(\sqrt{2H\left(1 - \frac{1}{N}\right)} - \frac{1}{2}\right)$
ε-ASU	$1 + \frac{M(N-1)^2}{N\varepsilon(M-1) + N - M}$	$\frac{N}{\varepsilon}\left\lfloor \frac{\log_2 M}{\log_2 N} \right\rfloor$ if $\log_2 M < \log_2 N \cdot \left(\sqrt{2\frac{F}{N}\left(1 - \frac{1}{N}\right)} - \frac{1}{2}\right)$
		$-\frac{M}{\varepsilon^2 \log_2 \varepsilon}$

Table 2.1: (Adapted from [NR10]) Lower bounds for the cardinality F of different classes of hash functions. $M = |\mathcal{M}|$, $N = |\mathcal{N}|$.

Theorem 2.15. *[KSJ96] Let $\mathcal{F}$ be an ε-ASU$_2$ class of hash functions from $\mathcal{M}$ to $\mathcal{T}$. Let $F = |\mathcal{F}|$, $M = |\mathcal{M}|$, and $N = |\mathcal{T}|$. If*

$$\log_2 M < \log_2 N \cdot \left(\sqrt{2\frac{F}{N}\left(1 - \frac{1}{N}\right)} - \frac{1}{2}\right)$$

then

$$F \geq \frac{N}{\varepsilon}\left\lfloor \frac{\log_2 M}{\log_2 N} \right\rfloor \tag{2.13}$$

Nguyen and Roscoe point out in [NR10] that the bounds described in theorems 2.9, 2.12, 2.15, and 2.14 are better than the ones contained in 2.8, 2.10, 2.11, and 2.13 when the ε is higher than $\bar{\varepsilon} = \left(1 + \frac{\log_2 N}{\log_2 M - \log_2 N}\right)\frac{1}{N}$. The previous results are summarised in table 2.1.

ε-AU	$\log_2\left(\frac{1}{\varepsilon}\left\lceil\frac{m}{n}-1\right\rceil\right)$
ε-ΔU	$\log_2\left(\frac{1}{\varepsilon}\left\lfloor\frac{m}{n}\right\rfloor\right)$ if $m < n\cdot\left(\sqrt{2^{r+1}\left(1-\frac{1}{2^n}\right)}-\frac{1}{2}\right)$
ε-ASU	$n+\log_2\left(\varepsilon^{-1}\left\lfloor\frac{m}{n}\right\rfloor\right)$ if $m < n\cdot\left(\sqrt{2\frac{F}{N}\left(1-\frac{1}{N}\right)}-\frac{1}{2}\right)$
	$\log_2 m + 2\log_2\varepsilon^{-1} - \log_2\log_2\varepsilon^{-1}$

Table 2.2: (Adapted from [NR10]) Lower bounds for the logarithm k_h of the cardinality of different classes of hash functions $\mathcal{F}$. $k_h = \log_2|\mathcal{F}|$, $m = \log_2|\mathcal{M}|$, $n = \log_2|\mathcal{N}|$.

We can observe that, even if the bounds on the left side of table 2.1 can be more accurate for some values of ε, they become quite useless when we deal with very large message sets $\mathcal{M}$: in those cases it is impossible to exactly calculate the lower bound for F (e.g. M could be 2^{10^6} as well). Since in our environment we will authenticate messages as long as a million bits, we will always use the bounds on the right column (see table 2.2 for logarithmic bounds).

2.9 Attacks

While this authentication system looks like perfect, it doesn't deal with the case when the opponent has partial information on the one-time pad keys, as in the case of quantum key distribution. In fact, in QKD, after using all the keys that must be shared before the starting of the system, Alice and Bob will use keys that have been generated by the key generation process. As was seen in chapter 1, the opponent could have some residual information about the exchanged key, and our authentication system can not ignore this fact.

Partial knowledge on the key

The main threat to an ε-ASU$_2$ authentication has been proposed by Larsson and Cederlöf in [CL08].

They study the case where the opponent has partial knowledge on the key. This event is anything but unlikely, because during the quantum transmission a little amount of information could have been intercepted by Eve without being noticed, and also because all the data exchanged over the public channel is, by definition, public. The privacy amplification step in QKD is designed to make the information that Eve has on the final key as little as possible, but some small amount of information will still be retained by the opponent.

Cederlöf and Larsson highlight how Eve is, in principle, able to influence the messages that are exchanged on the public channel by altering the quantum transmission. Since she is not trying to extract information by the photons, but only to increase the impact of the information she *already has*, she is not required to make any kind of measures, and therefore altering the quantum states and finally being detected.[6]

When Eve intercepts a valid couple (m, t), where m is the message which could have been influenced by her, she can partition the family $\mathcal{F}$ in the classes

$$\mathcal{F}(m,t) = \{h \in \mathcal{F} : h(m) = t\}$$

Let's call this partition

$$\mathcal{F}_m$$

The blocks of $\mathcal{F}_m$ correspond to the sets $\mathcal{F}(m,t)$ determined by all the values of $t \in \mathcal{T}$, that is

$$\mathcal{F}_m = \sum_{t \in \mathcal{T}} \mathcal{F}(m,t)$$

Since Eve has some information on the actual hash function, she can use all her information with the purpose of excluding possible hash functions. Influencing the message m, she has some control on the partition. If she is able to exclude some hash functions by a single class $\mathcal{F}(m,t)$, and if the number of hash functions in it falls under $\varepsilon F/N$, it may be that all the remained functions map another message $\hat{m}$ into another tag $\hat{t}$. In this case, Eve would be sure that the substitute couple $(\hat{m}, \hat{t})$ would be valid, and therefore her substitution attack would be successful.

It is important to point out that, with this attack, the opponent knows exactly when the attack will be successful, and can choose to perform it only in this case.

The best situation the opponent could aim to, is the condition where every block of functions has either size $\varepsilon F/N$ or F/N. In fact, when the selected hash function falls into a block of size $\varepsilon F/N$, it could happen that there exist a message $\hat{m}$ that is mapped into a single tag $\hat{t}$.

[6] For instance, she could in some way just affect the position of the slots in which Bob receives some photons.

Chapter

3

Constructions of ε-ASU$_2$ classes

In this chapter we will show some fundamental and flexible constructions of ε-ASU$_2$ classes of hash functions. This is not meant to be a comprehensive list of the existing ε-ASU$_2$ classes.

3.1 Toeplitz matrices

Toeplitz matrices have constant diagonals. A binary Toeplitz matrix A has elements $a_{i,j} \in \{0,1\}$ such that

$$a_{i,j} = a_{k,l} \quad \forall\, (i,j,k,l) : k - i = l - j$$

Such matrices can be completely characterised by expressing only the first row and first column, as the remaining elements are equal to the first item of the diagonal they belong to. Since the element in position $(1,1)$ is shared by the first row and the first column, we can describe a Toeplitz matrix using only an array of $n + m - 1$ bits, which is called the *seed* of the matrix.

Let's define a hash function which uses Toeplitz matrices. Let $\mathcal{M} = \{0,1\}^m$ and $\mathcal{T} = \{0,1\}^n$. For each pair (A, b) where A is a Toeplitz matrix with n rows and m columns and $b \in \mathcal{T}$ is an array, we can define a hash function:

$$\begin{aligned} h_{(A,b)} : \quad & \mathcal{M} \to \mathcal{T} \\ & x \mapsto Ax + b \end{aligned}$$

Let $\mathcal{F}$ be the set of all these functions.

Theorem 3.1. *[MNT90] $\mathcal{F}$ is $\frac{1}{n}$-ASU$_2$ (that is, SU$_2$).*

Proof. Let's prove property 2 of the definition 2.2. For each choice of $x_1 \neq x_2 \in \mathcal{M}$ and $y_1, y_2 \in \mathcal{T}$:

$$\begin{aligned}&|\{(A,b) : Ax_1 + b = y_1, Ax_2 + b = y_2\}| = \\ &= |\{(A,b) : A(x_1 + x_2) = y_1 + y_2, Ax_1 + b = y_1\}|\end{aligned}$$

For each matrix A, there is only one array $b = y_1 - Ax_1$ such as $Ax_1 + b = y_1$, so we can count only the number of matrices A which satisfies the equality $A(x_1 + x_2) = y_1 + y_2$:

$$\begin{aligned}&|\{(A,b) : Ax_1 + b = y_1, Ax_2 + b = y_2\}| = \\ &= |\{A : A(x_1 + x_2) = y_1 + y_2\}| = |\{(A,b) : Ax = y\}|\end{aligned}$$

where we let $x = x_1 + x_2$ and $y = y_1 + y_2$. Note that $x \neq 0$, because $x_1 \neq x_2$ and we are using the binary sum. If we write the matrix element $a_{i,j}$ by its *seed* value, that is as α_{n-i+j}, [1] we can write the expression $Ax = y$ as the following system of linear equations:

$$\begin{cases} \sum_{i=1}^{m} \alpha_i x_i = y_n \\ \sum_{i=1}^{m} \alpha_{i+1} x_i = y_{n-1} \\ \dots \\ \sum_{i=1}^{m} \alpha_{i+n-1} x_i = y_1 \end{cases}$$

Since $x \neq 0$, the n equations are linearly independent[2], and so the dimension of the solution space is exactly $m - 1$. Each set $\{(A,b) : Ax_1 + b = y_1, Ax_2 + b = y_2\}$ has therefore size 2^{m-1}. The number of all the possible hash functions in $\mathcal{F}$ is

$$|\mathcal{F}| = 2^{n+m-1} \cdot 2^n = 2^{2n+m-1}$$

[1] That is, we write the matrix

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m-1} & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m-1} & a_{2,m} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,m-1} & a_{n-1,m} \\ a_{n,1} & a_{n,2} & \dots & a_{n,m-1} & a_{n,m} \end{pmatrix}$$

as:

$$\begin{pmatrix} \alpha_n & \alpha_{n+1} & \dots & \alpha_{n+m-2} & \alpha_{n+m-1} \\ \alpha_{n-1} & \alpha_n & \dots & \alpha_{n+m-3} & \alpha_{n+m-2} \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \alpha_2 & \alpha_3 & \dots & \alpha_m & \alpha_{m+1} \\ \alpha_1 & \alpha_2 & \dots & \alpha_{m-1} & \alpha_m \end{pmatrix}$$

[2] The same system of linear equations obtained by the multiplication of a Toeplitz matrix A by an array x:

$$\begin{pmatrix} a_n & a_{n+1} & \dots & a_{n+m-2} & a_{n+m-1} \\ a_{n-1} & a_n & \dots & a_{n+m-3} & a_{n+m-2} \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ a_2 & a_3 & \dots & a_m & a_{m+1} \\ a_1 & a_2 & \dots & a_{m-1} & a_m \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \\ x_m \end{pmatrix}$$

and so

$$|\{(A,b) : Ax_1 + b = y_1, Ax_2 + b = y_2\}| = 2^{m-1} = \frac{|\mathcal{F}|}{2^{2n}}$$

We have thus proved property 2. Property 1 can be easily proved using the last part of the proof of property 2. □

Features

Class size The class size is exactly

$$|\mathcal{F}| = 2^{2n+m+1}$$

Indexing We can index every hash function of this class by concatenating the seed[3] $s \in \{0,1\}^{n+m-1}$ of the Toeplitz matrix and the array $b \in \mathcal{T}$. Every hash function is indexed by the $2n + m - 1$ binary array $[s, b]$.

Implementation The implementation is straightforward: one could develop the actual Toeplitz matrix from the seed and then perform a matrix-array multiplication, otherwise one could implement the same multiplication in an efficient way taking advantage of the special structure of the Toeplitz matrices. Tang, Duraiswami, and Gumerov [TDG04], and Bodrato ([Bod07]) give an example of an efficient algorithm to perform this task. In this way it is not necessary to explicitly express (and therefore store in memory) the whole matrix, which in many cases may have as many as hundreds of rows and millions of columns.

To authenticate messages which are shorter than m bits, it is enough to truncate the number of columns of the Toeplitz matrix to the actual length of the message, if we are using the classical authentication, or to shorten the seed accordingly, if we are using an efficient algorithm which only uses the seed.

The binary addition between arrays adds a negligible complexity to the previous step.

---

can be described by:

$$\begin{pmatrix} 0 & 0 & \dots & 0 & x_1 & x_2 & \dots & \dots & x_{m-1} & x_m \\ 0 & 0 & \dots & x_1 & x_2 & x_3 & \dots & \dots & x_m & 0 \\ \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots \\ 0 & x_1 & \dots & \dots & \dots & x_{m-1} & x_m & \dots & 0 & 0 \\ x_1 & x_2 & \dots & \dots & \dots & x_m & 0 & \dots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n+m-2} \\ a_{n+m-1} \end{pmatrix}$$

Now it becomes evident that, when at least one of the coefficient of x is not null, the equations are independent.

[3] That is, every element $a_{i,j}$ of the Toeplitz matrix is equal to the seed element s_{n-i+j}.

Remarks This class of hash functions is not suitable to be used only once, in the context of QKD, because it needs an amount of random data (the key length) that is greater than the message to be authenticated.

3.2 Binary matrices

A class of SU$_2$ hash functions for messages of length m and hash length n is obtained by taking a random binary matrix A of size $n \times m$ and an array b of length n. The hash value for a message x is calculated as $Ax + b$. This class has the same properties as the Toeplitz based class, but its size is much larger and its hash computation is slower, so it is rarely used in practice.

Features

Class size The class size is exactly

$$|\mathcal{F}| = 2^{(m+1)n}$$

Indexing We can index each hash function by concatenating of the matrix A (either in row-major or in column-major) and of the array b. Every hash function is indexed by a binary array of size $(m + 1)n$.

3.3 Stinson

In [Sti94b], Stinson refines a construction due to Wegman and Carter in [WC81].

The final ε-ASU$_2$ class is built on a U$_2$ class $\mathcal{G}_1$ and a SU$_2$ class $\mathcal{G}_2$.

Let p be a prime power, and let $\mathcal{M}_1 = \mathbb{F}_p \times \mathbb{F}_p$ and $\mathcal{T}_1 = \mathbb{F}_p$. So $|\mathcal{M}_1| = M_1 = p^2$ and $|\mathcal{T}_1| = N_1 = p$. For each $x \in \mathbb{F}_p$ the function g_x is defined as:

$$\begin{aligned} g_x : \mathcal{M}_1 = \mathbb{F}_p \times \mathbb{F}_p &\to \mathcal{T}_1 \\ (y, z) &\mapsto xy + z \end{aligned}$$

Let $\mathcal{G}_1$ be the set of all the functions g_x:

$$\mathcal{G}_1 = \{g_x : x \in \mathbb{F}_p\}$$

It is straightforward to prove that $\mathcal{G}_1$ is a U$_2$ class of hash functions.

Applying theorem 2.4, for each $j = 1, \ldots, i$ it is possible to build a $1/p$-AU$_2$ class $\mathcal{G}_1^{2^j}$ of p hash functions from $\mathcal{M}_1^{2^j}$ to $\mathcal{T}_1^{2^{j-1}}$. Then it is possible to use theorem 2.5 to combine all the classes $\mathcal{G}_1^{2^j}$ in a single i/p-AU$_2$ class $\mathcal{G}_1^{2^i}$ of p^i hash functions from $\mathcal{M}_1^{2^i}$ to $\mathcal{T}_1$, with $|\mathcal{M}_1^{2^i}| = p^{2^i}$.

To define $\mathcal{G}_2$, let q be a prime power and let s and t be two integers so that $s > t$. Let $\mathcal{M}_2 = \mathbb{F}_{q^s}$ and $\mathcal{T}_2 = \mathbb{F}_{q^t}$. $|\mathcal{M}_2| = M_2 = q^s$ and $|\mathcal{T}_2| = N_2 = q^t$. For each $x \in \mathbb{F}_q$. Let ϕ be any surjective linear transformation from $\mathcal{M}_2$ to $\mathcal{T}_2$. For each $(x, y) \in \mathcal{M}_2 \times \mathcal{T}_2$ let the function g_{xy} be defined as:

$$\begin{aligned} g_{xy} : \mathcal{M}_2 &\to \mathcal{T}_2 \\ z &\mapsto \phi(xz) + y \end{aligned}$$

Let $\mathcal{G}_2$ be the set of all the functions g_{xy}:

$$\mathcal{G}_2 = \{g_{xy} : (x, y) \in \mathbb{F}_{q^s} \times \mathbb{F}_{q^t}\}$$

It is possible to prove that $\mathcal{G}_2$ is a SU_2 class of hash functions (see [Sti94b]).

If the elements in $\mathbb{F}_{q^s}$ and $\mathbb{F}_{q^t}$ are represented by vectors on $\mathbb{F}_q$, it is possible to define $\phi(x)$ as any fixed subset of the coefficients of size t, that is:

$$\phi(x) = \phi\left((x_1, \ldots, x_s)\right) = (x_{i_1}, \ldots, x_{i_t})$$

Now we can build the final ε-ASU_2 class $\mathcal{F}$ of hash functions.

Choosing the class $\mathcal{G}_1^{2^i}$ with $p = q^s$, we obtain an i/q^s-AU_2 class of hash functions from $\mathcal{M}_1^{2^i}$ to $\mathcal{T}_1$, with $|\mathcal{M}_1^{2^i}| = q^{2^i s}$ and $|\mathcal{T}_1| = q^s$. Then it is possible to compose the i/q^s-AU_2 class $\mathcal{G}_1^{2^i}$ with the SU_2 class $\mathcal{G}_2$ using theorem 2.6. The result is the $(i/q^s + 1/q^t)$-ASU_2 class $\mathcal{F}$.

Now we can choose the right parameters: let $q = 2$, $t = n$, $\mathcal{M} = \mathcal{M}_1^{2^i}$, and $\mathcal{T} = \mathcal{T}_2$, with $|\mathcal{M}| = 2^m$ and $|\mathcal{T}| = 2^n$. Let

$$s = n + \lceil \log_2 \log_2 m \rceil \tag{3.1}$$

$$i = \left\lceil \log_2 \frac{m}{s} \right\rceil \tag{3.2}$$

It is possible to check that

$$\frac{i}{2^s} \leq \frac{1}{2^n}$$

so $\mathcal{F}$ is a $1/2^{n-1}$-ASU_2 class of F hash functions, with

$$F = 2^{(i+1)s+n}$$

Features

Class size The class size is exactly

$$|\mathcal{F}| = 2^{(i+1)s+n}$$

with $s = n + \lceil \log_2 \log_2 m \rceil$ and $i = \left\lceil \log_2 \frac{m}{s} \right\rceil$.

Indexing We can index every hash function of this class by concatenating i binary arrays of size s, which are the keys for the i steps of universal functions, plus one more binary array of size s and another one of size n, which are the keys of the strongly universal function. So every hash function is indexed by a binary array of size $(i+1)s+n$.

Implementation The message is processed in $i+1$ rounds. At the beginning of the first i rounds, the input is padded with zeros up to a length multiple of $2s$. Let k_j be the key for the round j: in the first i rounds the message is repeatedly split in blocks of size $2s$ and the hash function g_{k_j} is applied to each block. The concatenation of the results is the input for the next round. The final round consists in the application of the strongly universal hash function $g_{k_a k_b}$, where k_a and k_b are the two keys for the $\mathcal{G}_2$ function. The linear transformation can be implemented by simply taking the first (or the last) n bits.

Optimisations A possible improvement could be that of applying the universal hash functions only for the number of steps that are strictly necessary to reduce the size of the message to s. Then it is possible to skip to the application of the final strongly universal hash function.

Remarks The key size is much shorter than m, that is the size of the messages in $\mathcal{M}$.

3.4 Krawczyk

Krawczyk constructions are three different constructions of ε-ASU$_2$ classes introduced in [Kra94] and [Kra95]. These are all based on ε-AXU classes (see theorem 2.3), formed by Toeplitz matrices which seed has been obtained from so-called ε-biased distributions (described by Naor and Naor in [NN90]). The main problem in two out of three of the constructions is that the key is formed by the description of an irreducible polynomial, which is not efficiently encoded, so the length of the key is larger than $\log_2 |\mathcal{F}|$. Moreover, in the third construction (the *scalar product construction*) it is possible to notice that different keys produce the same hash function, and this problem worsens as the class size increases, so the entropy of the key is not completely used. In any case, the Reed-Solomon class addresses these problems with a smaller size.

3.5 Reed-Solomon

Using the similarities between ε-ASU$_2$ classes shown in paragraph 2.5, we can use Reed-Solomon codes as a component of an ε-ASU$_2$ hash function class. This construction is due to Bierbrauer, Johansson, Kabatianskii and Smeets (see [BJKS93]).

Using coding theory notation, a $[n, k, d]_q$-code is a code with minimum distance d that processes messages of length k on a q-sized alphabet and returns codewords of length

n on the same alphabet. In general, a (n, k)-code is a code which processes messages of length k and returns codewords of length k. The Reed-Solomon code is optimal, in the sense that it is an (n, k)-code that reaches the minimum distance $d = n - k + 1$.

In Reed-Solomon encoding (see [RS60]), the message is seen as a concatenation of k coefficients of a polynomial on a finite field $\mathbb{F}_q$. The codeword is obtained by oversampling the polynomial in $n > k$ different points of $\mathbb{F}_q$ and concatenating the results. If up to $n - k$ symbols get lost, the original message is still recoverable.

We choose the Reed-Solomon $[n, k, d]_q$-code with the following parameters:

$$\begin{aligned} n &= 2^{r+s} \\ k &= 1 + 2^s \\ d &= n - k + 1 = 2^{r+s} - 2^s \\ q &= 2^{r+s} \end{aligned}$$

Going back to authentication codes notation, this Reed-Solomon code is capable of encoding binary messages of length up to $m = 2^{(n+s)(1+2^s)}$ (from now on, n is the length of the final tag) and produces a ε_1-AU$_2$ class of hash functions with

$$\varepsilon_1 = 1 - \frac{d}{n} = 1 - \frac{2^{n+s}}{2^{n+s}} + \frac{2^s}{2^{n+s}} = \frac{1}{2^n}$$

Now it is possible to compose the ε_1-AU$_2$ class with the ε_2-ASU$_2$ class $\mathcal{G}_2$ (see paragraph 3.3) of the hash functions from $\mathbb{F}_{2^s}$ to $\mathbb{F}_{2^n}$. Therefore, by theorem 2.6, the final class $\mathcal{F}$ of hash functions is ε-ASU$_2$, with

$$\varepsilon = \varepsilon_1 + \varepsilon_2 = \frac{1}{2^n} + \frac{1}{2^n} = \frac{1}{2^{n-1}}$$

Features

Class size The class size is exactly

$$|\mathcal{F}| = 2^{3n+2s}$$

Let m be the maximum message length: s must be chosen to be the smallest integer such that

$$m < (n + s)(1 + 2^s) \tag{3.3}$$

Usually $m \gg n$, so it is possible to approximate $s \approx \lceil \log_2 m - \log_2 n \rceil$.[4]

Indexing We can index every hash function of this class by the concatenation of a binary array of size $n+s$, which is the key for the universal function, with a binary array of size $n+s$ and another one of size n, which are the keys of the strongly universal function. So every hash function is indexed by a binary array of size $3n + 2s$.

[4] If the requirement of equation (3.3) does not hold, it is sufficient to choose s as the smaller integer that makes (3.3) feasible.

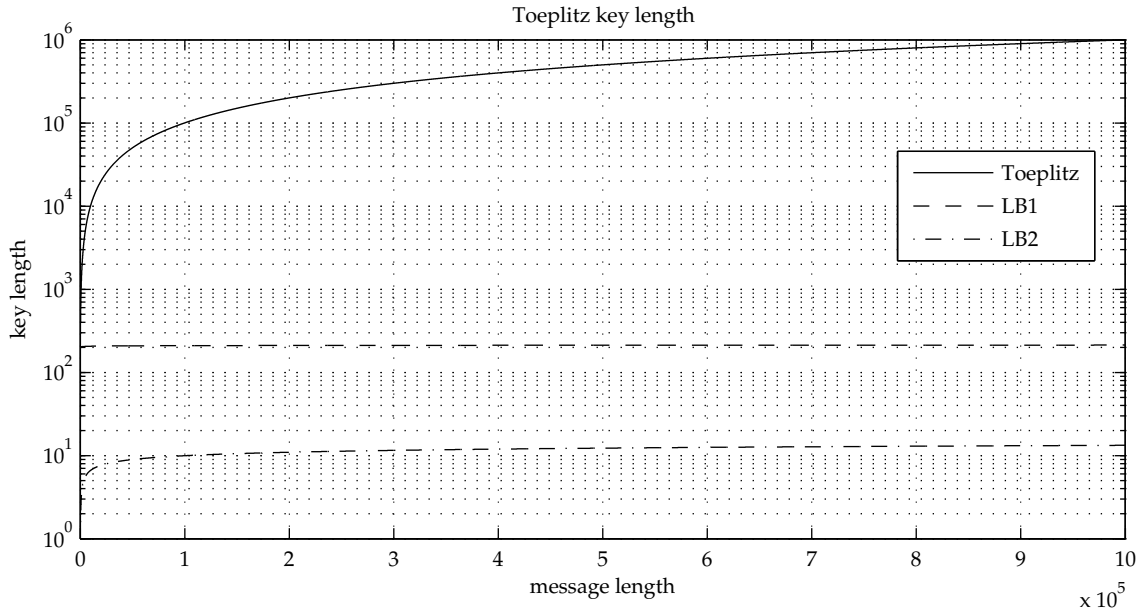
Implementation Let $k_1 \in \{0,1\}^{n+s}$ be the key for the universal hash function and $k_a \in \{0,1\}^{n+s}$ and $k_b \in \{0,1\}^n$ The message is padded up to a length multiple of $n+s$ and then is split in i blocks of length $n+s$. Every block j is multiplied by k_1^{j-1} for each $j = 1, \ldots, i$, then all the results are summed. The multiplications and exponentiations are calculated in $\mathbb{F}_{2^{r+s}}$.

The second phase consists in the application of the strongly universal hash function $g_{k_a k_b}$, as in Stinson's construction.

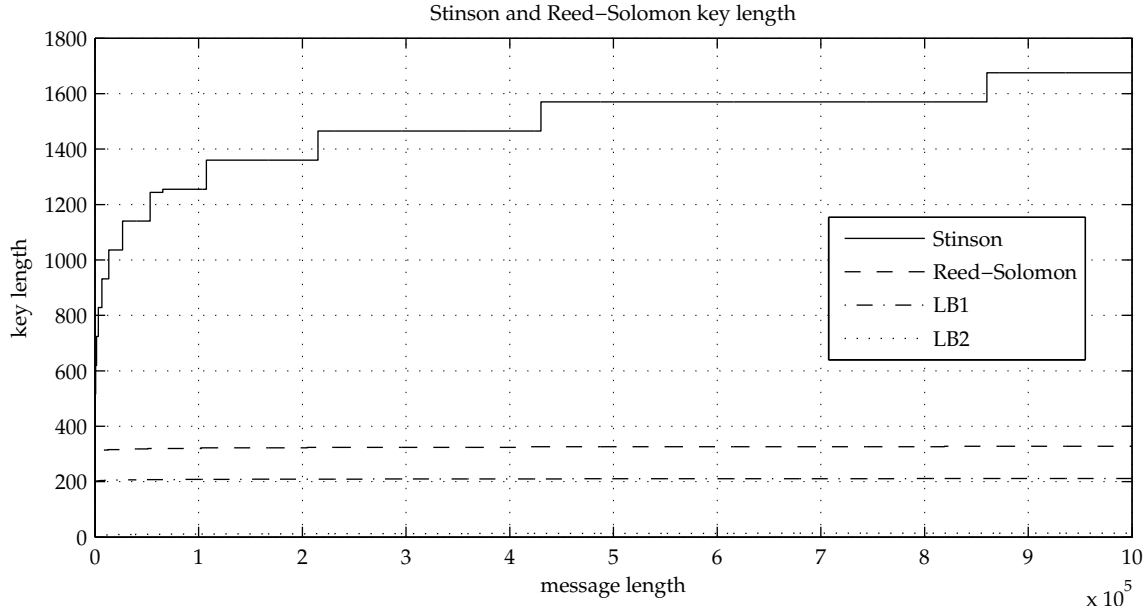
Remarks The key size is the shortest in all the constructions considered in this work.

3.6 Comparison between class sizes and lower bounds

We can compare the different sizes of these constructions with the lower bounds (2.12) and (2.13). It is possible to see that Reed-Solomon construction has good performances versus both m and n.



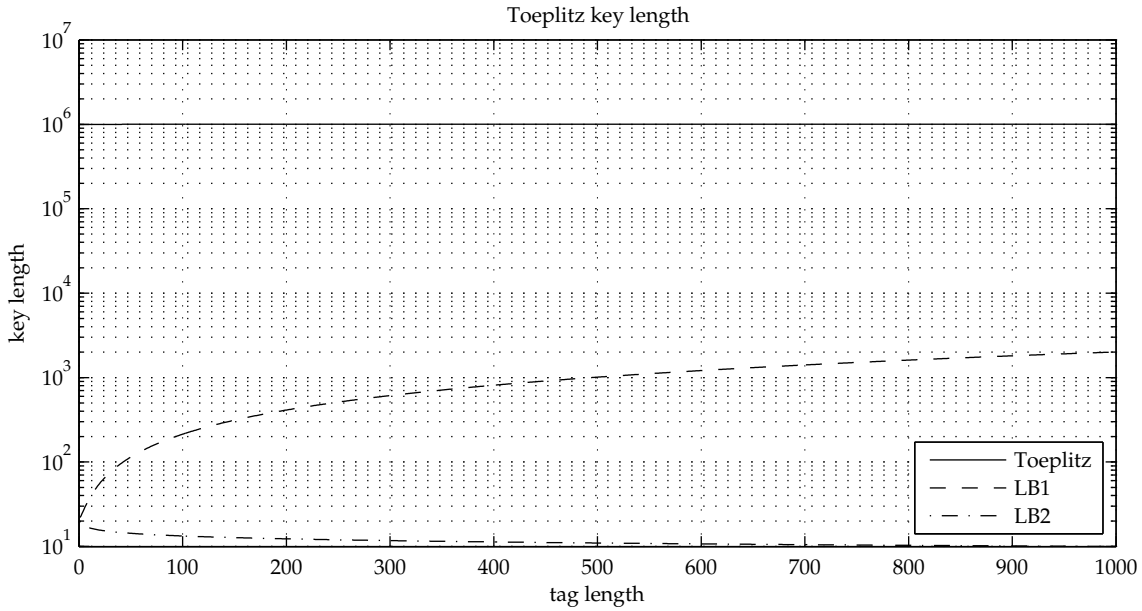
(a) Key length for SU_2 class based on Toeplitz matrices.



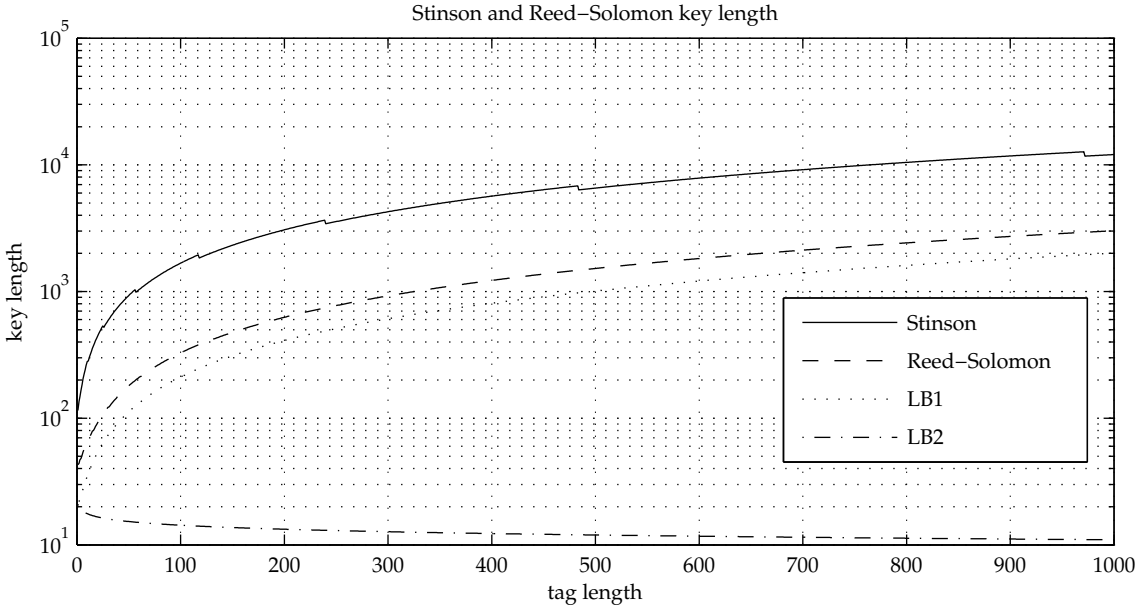
(b) Key length for ε-ASU_2 Stinson and Reed-Solomon classes.

Parameters
n 100

Figure 3.1: Key length as a function of message length m, compared with lower bounds (2.12) and (2.13).



(a) Key length for SU$_2$ class based on Toeplitz matrices.



(b) Key length for ε-ASU$_2$ Stinson and Reed-Solomon classes.

Parameters	
m	10^6

Figure 3.2: Key length as a function of tag length n, compared with lower bounds (2.12) and (2.13).

Chapter

4

Reusing the same key

In the previous chapters it has been seen that changing the selected hash function is very expensive, because many secret bits have to be used to form a key which will define the new hash function.

In the context of QKD, the rate at which we use secret bits is fundamental to determine the soundness of the whole key generating system. In fact, the random data used by Alice and Bob as the index for the new hash function are extracted by the same stream of secret bits generated by the quantum system. Therefore, the overall net key rate produced by the QKD is given by the rate of secret data stream at the end of privacy amplification to which the rate of key consumed by the authentication process has to be subtracted.

If we are able to reduce the rate of the key used by the authentication, all the saved bits will increase the net key rate, so it is crucial to use as little key as possible for the authentication purpose.

In QKD, it is necessary to authenticate one or more messages during a single round, so it is necessary to find a way to use the same hash function as long as possible, without letting the security of the whole process to diminish under a fixed threshold.

4.1 One-time pad

One-time pad (or OTP), that is a Vernam cypher with a keystream which is random, uniformly distributed and is used only once, is an encryption method that attains Shannon's perfect secrecy (see [Sha49]). This means that, if m is the message and c is the cyphertext that codes for m,

$$H(m) = H(m|c) \tag{4.1}$$

that is to say that the observation of c gives absolutely no information about the original message.

One-time pad encryption is very easy to perform: if the message m is formed by the concatenation of many symbols $(a_1, \ldots, a_n)$, with $a_i \in \mathcal{A}$, and $(\mathcal{A}, +)$ is an algebraic group, a one-time pad key k must be composed by as many random symbols (of the same alphabet $\mathcal{A}$) as the message m:

$$k = (k_1, \ldots, k_n)$$

The cyphertext c is given by:

$$c = (c_1, \ldots, c_n)$$
$$c_i = m_i + k_i$$

4.2 Strongly Universal Hashing and One-Time Pad

To answer the need for multiple authentication, Wegman and Carter in [WC81] mention the possible use of SU_l hash function families to reuse a single hash function up to $l - 1$ times: nevertheless, the number of functions in such a class increases exponentially with l (see Atici and Stinson in [AS96]). Another method suggested by Wegman and Carter is to encrypt the tag obtained by a (ε-A)SU_2 hash function with one-time pad. Since the one-time pad changes at every authentication round, it is impossible for the attacker to ever identify the selected hash function, basically because he has no information about the hash value generated by the hash function, as shown in equation (4.1).

In [WC81] Wegman and Carter also show that the choice of encoding the tag with a one-time pad key leads to an authentication code that is *asymptotically optimal*. Their proof is reported here for completeness.

Theorem 4.1. *[WC81] Let $\mathcal{F}$ be a ε-ASU$_2$ class of hash functions from $\mathcal{M}$ to the group $\mathcal{T}$. Let's choose a hash function h with uniform distribution in $\mathcal{F}$. If we authenticate a sequence of messages m_i by first calculating their hash value with h and then encoding the hash values with a one-time pad key k_i chosen with a uniform distribution in $\mathcal{T}$, so $t_i = h(m_i) + k_i$, then an attacker who only sees the couples (m_i, t_i) has a probability P_D of successfully forging a valid tag $\hat{t}$ for a message $\hat{m}$ of his choice that is upper bounded by*

$$P_D \leq \varepsilon$$

Proof. Let $\hat{m}$ be a generic message chosen by the opponent. Let $\mathcal{F}(m, t, \hat{t}) = \{(h, k) \in \mathcal{F} \times \mathcal{T} : h(m) + k = t \wedge h(\hat{m}) + k = \hat{t}\}$. Since $\mathcal{F}$ is ε-ASU$_2$, for each value of k there are at most $\varepsilon|\mathcal{F}|$ functions that map m into $t - k$ and $\hat{m}$ into $\hat{t} - k$, so

$$|\mathcal{F}(m_1, t_1, \hat{t}_1)| \leq \varepsilon \frac{|\mathcal{F}|}{|\mathcal{T}|} \cdot |\mathcal{T}| = \varepsilon|\mathcal{F}|$$

Moreover, for each couple $(h, k_1) \in \mathcal{F}(m_1, t_1, \hat{t}_1)$, there is exactly one sequence $(k_1, \ldots, k_n)$ that satisfies the set of equalities

$$\begin{cases} h(m_1) + k_1 = t_1 \\ \ldots \\ h(m_n) + k_n = t_n \end{cases}$$

so the opponent will not be able to gain any additional information on the set of possible hash functions h. The chances of successfully substituting a valid couple m_i, t_i with a valid forged couple $(\hat{m}, \hat{t})$, therefore, are upper bounded by

$$P_S = \max_{\substack{m_i \neq \hat{m} \in \mathcal{M} \\ t_i, \hat{t} \in \mathcal{T}}} \frac{|\{(h,k) \in \mathcal{F} \times \mathcal{T} : h(m_i) + k = t_i, h(\hat{m}) + k = \hat{t}\}|}{|\mathcal{F} \times \mathcal{T}|} \leq \varepsilon \frac{|\mathcal{F}||\mathcal{T}|}{|\mathcal{F}||\mathcal{T}|} = \varepsilon$$

□

Now let's consider the case where, for n times, an opponent can choose a message m_i, tries to produce a valid tag $\hat{t}_i$, and then is given the right tag t_i.

Theorem 4.2. *[WC81] If the opponent's probability of forging the right tag at round i ($\hat{t}_i = t_i$) is*

$$P_D \leq p_i \quad \forall i = 1, \ldots, n$$

then the size of the set of hash functions must be

$$|\mathcal{F}| \geq \frac{1}{p_1 \cdot \ldots \cdot p_n}$$

Proof. Let $\mathcal{F}_0 = \mathcal{F}$ and $\mathcal{F}_i = \{h \in \mathcal{F} | h(m_l) = t_l \; \forall l = 1, \ldots, i\}$. The strategy of the opponent could be that of choosing, at round i, a random element of the set F_i to calculate $\hat{t}_i$. If $P_D \leq p_i$, then $|\{h \in \mathcal{F}_{i-1} : h(m_i = t_i)\}| \leq p_i |\mathcal{F}_{i-1}|$, so

$$|\mathcal{F}_i| \leq p_i |\mathcal{F}_{i-1}| \quad \forall i = 1, \ldots, n$$

Considering the last round, we obtain

$$|\mathcal{F}_n| \leq p_1 \cdot \ldots \cdot p_n |\mathcal{F}_0|$$

and, therefore,

$$|\mathcal{F}| \geq \frac{1}{p_1 \cdot \ldots \cdot p_n}$$

because $\mathcal{F}_0 = \mathcal{F}$ and $|\mathcal{F}_n| \geq 1$. □

Corollary 4.1. *If we require $p_1 = \ldots = p_n = p$, then we need $|\mathcal{F}| \geq \frac{1}{p^n}$ and at least $-n \log_2 p$ to identify a hash function in $\mathcal{F}$.*

If we used a SU_2 hash function with OTP encoding for the tag, it would take us $k - n \log_2 p$ bits for n rounds, where k is the key size for the SU_2 function, we can state that the key consumption would be asintotically optimal.

Remarks The secrecy of the OTP key alone is completely unsecure, because, knowing just the selected hash function, the opponent is able to calculate the hash value of an intercepted message and then to recover the OTP key, so the substitution attack is trivial. So it is not enough to have a perfect secrecy on the OTP encoding.

OTP advantages

In [AS96], Atici and Stinson show that it is possible to save some key by not encoding the first tag with OTP, relying only on the properties of ε-ASU$_2$ classes. In this way, they expose the authentication code, at least for the first couple (m, t), to the Cederlöf-Larsson attack (see paragraph 2.9). Encrypting all the hash values, as in Wegman and Carter original idea, on the other hand, protects the authentication code by this threat, while, at the same time, the optimal asymptotical behaviour is not affected.

4.3 Partial knowledge on the key, revised

With the OTP encryption of the tag, the Cederlöf-Larsson attack (paragraph) is no longer possible: since the tag t is encrypted, most of the times the opponent is not able to determine which block $\mathcal{F}(m, t)$ of the partition $\mathcal{F}_m$ is identified by the intercepted couple (m, t).

While OTP encryption on the tag allows a single hash function to be used more than once, though, it also has a new weak point. Since the hash function is used many times, the opponent can store all the information extracted in a series of observed valid authentication. As a consequence, the more a single hash function is used, the higher are the opponent's chances of a successful attack.

Abidin and Larsson study the same problem in [AL11]. They give simplified models to calculate the probability of success of the opponent's attacks and describe the results of simulations of attacks. We will give, instead, an upper bound to the probabilities of success of the attacks.

Hypothesis

In the QKD, the privacy amplification phase is designed to leave the opponent with not more than a given amount of information. Let z be a random variable which describes all the observations made by the opponent on both the quantum and the public channel during the quantum key generation phase. Let k_h the random variable which represents the key of the ε-ASU$_2$ hash function, and k_{OTP} the random variable which represents a generic key of the OTP encryption.

We may postulate that Eve has at the most i_h bits of information on the selected hash function and i_{OTP} bits of information on each OTP key used to encrypt the tag, that in

symbol	meaning
i_{OTP}	maximum leaked information on each OTP key
i_h	maximum leaked information on the selected hash function
ℓ	information leaked per binary symbol
n	tag length
k	hash function key length

Table 4.1: List of parameters for the analysis of a new attack to authentication codes based on ε-ASU$_2$ hash function classes with OTP-encrypted tags.

terms of mutual information means:

$$I(k_h; z) \leq i_h$$
$$I(k_{\text{OTP}}; z) \leq i_{\text{OTP}}$$

Since the key length required to select a hash function is larger than the length of the OTP key, tipically $i_h > i_{\text{OTP}}$. We can hypothesise that in the long run the hash function keys will be taken by the same repository where the OTP keys come from, so that the opponent has the same average information on them. Namely, if k is the length of k_h and n the tag length, we can assume that:

$$i_h = \ell \cdot n \tag{4.2}$$
$$i_{\text{OTP}} = \ell \cdot k \tag{4.3}$$

We can assume, like in Larsson-Cederlöf [CL08], that Eve can influence the messages m exchanged between Alice and Bob on the public channel, but now it is impossible for the attacker to try to break the authentication code in the same way described by Cederlöf and Larsson, simply because the hash value v is encrypted into the tag t. As a consequence, Eve cannot identify the corresponding block $\mathcal{F}(m, v)$.

All the attacker could do is trying either an impersonation attack, a collision attack, or wait until it is possible to recover the exact OTP key used and then trying the original Cederlöf and Larsson attack.

Let (m_j, t_j) be a generic intercepted message with its tag. Since the hash value v_j of the message m_j is encrypted with a one-time pad key, we can state that

$$H(k_h|m_j, t_j) \geq H(k_h) - i_{\text{OTP}} \tag{4.4}$$

and therefore:

$$H(k_h|m_1, t_1, \ldots, m_j, t_j, z) \geq H(k_h) - (j \cdot i_{\text{OTP}} + i_h) \tag{4.5}$$

In other words, if the maximum information possessed by Eve on a single OTP key is i_{OTP}, this is the exact amount of information that she has on the unencrypted hash value

of the message m, and therefore she can not obtain, with each observation, more than i_{OTP} bits of information on the original tag.

It is possible to relate all the possible attacks against an authentication code made of an OTP encrypted ε-ASU$_2$ hash function class $\mathcal{F}$ to the task of identificating a certain subset $\bar{\mathcal{F}} \subset \mathcal{F}$. That is to say that, for each type of attack and for each selected function h_k, the probabilities of success are maximised if the opponent is able to identify a certain set $\bar{\mathcal{F}}$ which the selected hash function k_h belongs to.

Hypothesis for attack description

For a simple description and a rapid estimate of the probability of success of each possible attack, we can suppose that the opponent uses all the information obtained only to rule out as many hash functions as possible from the set of all the possible candidates, so that, if $\mathcal{F}_e$ is the set of possible hash functions, each $h \in \mathcal{F}_e$ has en even probability $1/|\mathcal{F}_e|$ of being the selected function.

When Alice and Bob start using a new hash function, and no couples (m, t) have yet been transmitted, all the information that Eve has on the hash function is limited to i_h bits, and therefore the entropy of the hash function key k_h, that is equal to the entropy of the hash function, is

$$H(k_h) = k - i_h$$

The entropy of a uniformly distributed random variable $x \in \mathcal{X}$ is $H(x) = \log_2 |\mathcal{X}|$, so the size of the set $\mathcal{F}_e$ is upper bounded by $2^{\lceil k - i_h \rceil}$.

After having intercepted j authenticated messages, the entropy of k_h is at most $k - (j \cdot i_{\text{OTP}} + i_h)$ and the size of the set $\mathcal{F}_e$ is upper bounded by

$$|\mathcal{F}_e| \leq 2^{\lceil k - i_h - j \cdot i_{\text{OTP}} \rceil} \tag{4.6}$$

Hypothesis for upper bounds

In order to give an upper bound to the probabilities of success of each attack, we have to study Eve's best way of exploiting her gathered information on h_k.

Let $\bar{\mathcal{F}}$ be a subset of $\mathcal{F}$ that give the maximum probability of success to the opponent. Let the size of $\bar{\mathcal{F}}$ be upper bounded by some value X:

$$|\bar{\mathcal{F}}| \leq X$$

We want to know how much information is needed by Eve to have a certain probability of success $\hat{P}$.

Let k_h be the random variable that describes the current selected hash function and let $p(\cdot)$ be its probabilty mass distribution:

$$\begin{aligned} p: \quad & \mathcal{F} \to [0, 1] \\ & p(h_i) = p_i = P[k_h = h_i] \qquad \forall h_i \in \mathcal{F} \end{aligned}$$

Without loss of generality, let the hash functions $h_1, \ldots, h_F$ be ordered in descending order by their probability p_i, so that

$$p_1 \geq p_2 \geq \ldots \geq p_F$$

It is straightforward to see that:

$$P[k_h \in \bar{\mathcal{F}}] = \sum_{h \in \bar{\mathcal{F}}} P[k_h = h] \leq p_1 + \ldots + p_{|\bar{\mathcal{F}}|} \leq p_1 + \ldots + p_X \tag{4.7}$$

Let $\bar{P}$ be the sum of the X highest probabilities p_i:

$$\bar{P} \triangleq p_1 + \ldots + p_X$$

The minimum information needed by Eve to have probability $P[k_h \in \bar{\mathcal{F}}] = \bar{P}$ is lower bounded by the difference between the initial entropy $H(k_h)$ and the maximum entropy $\bar{H}_{\max}$ of a probability distribution that satisfies $\bar{P} = p_1 + \ldots + p_X$. The distribution that maximises the final entropy is obtained when exactly X hash functions have probability $\bar{P}/X$ and the remaining ones have probability $(1 - \bar{P})/(F - X)$.

This can be proved as follows. Let $\mathcal{F}_X$ be a subset of $\mathcal{F}$ of size X, and let's define two conditioned p.m.d.:

$$\begin{aligned} p' : \quad & \mathcal{F} \to [0,1] \\ & p'(h_i) = p'_i = P[k_h = h_i | k_h \in \mathcal{F}_X] \qquad \forall h_i \in \mathcal{F} \end{aligned}$$

$$\begin{aligned} p'' : \quad & \mathcal{F} \to [0,1] \\ & p''(h_i) = p''_i = P[k_h = h_i | k_h \notin \mathcal{F}_X] \qquad \forall h_i \in \mathcal{F} \end{aligned}$$

and let $\bar{P} = \sum_{h_i \in \mathcal{F}_X} p_i = P[k_h \in \mathcal{F}_X]$. Note that $p_i = p'_i \cdot \bar{P}$ if $h_i \in \mathcal{F}_X$, and $p_i = p''_i \cdot (1 - \bar{P})$ if $h_i \notin \mathcal{F}_X$.

Thus we can write:

$$\begin{aligned} H(p) &= \sum_{h_i \in \mathcal{F}_X} p_i \log_{\frac{1}{2}} p_i + \sum_{h_i \notin \mathcal{F}_X} p_i \log_{\frac{1}{2}} p_i = \\ &= \sum_{h_i \in \mathcal{F}_X} \bar{P} p'_i \log_{\frac{1}{2}} \bar{P} p'_i + \sum_{h_i \notin \mathcal{F}_X} (1 - \bar{P}) p''_i \log_{\frac{1}{2}} (1 - \bar{P}) p''_i = \\ &= \bar{P} \left(\sum_{h_i \in \mathcal{F}_X} p'_i \log_{\frac{1}{2}} \bar{P} + \sum_{h_i \in \mathcal{F}_X} p'_i \log_{\frac{1}{2}} p'_i \right) + \\ &\qquad + (1 - \bar{P}) \left(\sum_{h_i \notin \mathcal{F}_X} p''_i \log_{\frac{1}{2}} (1 - \bar{P}) + \sum_{h_i \notin \mathcal{F}_X} p''_i \log_{\frac{1}{2}} p''_i \right) = \\ &= \bar{P} H(p') + (1 - \bar{P}) H(p'') + \bar{P} \log_{\frac{1}{2}} \bar{P} + (1 - \bar{P}) \log_{\frac{1}{2}} (1 - \bar{P}) = \\ &= \bar{P} H(p') + (1 - \bar{P}) H(p'') + h_2(\bar{P}) \end{aligned} \tag{4.8}$$

where $h_2(\bar{P})$ is the binary entropy of the random variable $\bar{P}$.

Since $\bar{P}$ is given, the maximum of $H(p)$ is reached only if p' and p'' are uniformly distributions, namely:

$$\max H(p) = \max_{p',p''} \left(\bar{P}H(p') + (1-\bar{P})H(p'') + h_2(\bar{P})\right) = \\ = \bar{P}\log_2 X + (1-\bar{P})\log_2(F-X) + h_2(\bar{P}) \quad (4.9)$$

The corresponding p.m.d. is:

$$p: \quad \mathcal{F} \to [0,1] \\ \begin{cases} p(h_i) = \frac{\bar{P}}{X} & \forall h_i \in \mathcal{F}_X \\ p(h_i) = \frac{1-\bar{P}}{F-X} & \forall h_i \notin \mathcal{F}_X \end{cases}$$

Impersonation attack

The probability of success P_I of an impersonation attack is limited by the knowledge on the selected hash function that the opponent has gathered during the previous rounds of authentication and, much more severely, by the knowlege on the OTP key.

Even if Eve knows the selected hash function, she has only i_{OTP} bits of information on the OTP key.

In this context, Eve has to guess the right OTP key among all the possible ones. This means that the "good" subset X is made of only one element. Let $\widehat{k_{\text{OTP}}}$ be the only element in X. The p.m.d. that maximises $H(k_{\text{OTP}})$, given that the probability of X is $\bar{P}$, is:

$$\begin{cases} P[k_{\text{OTP}} = k_i] = p_S = \bar{P} & k_i = \widehat{k_{\text{OTP}}} \\ P[k_{\text{OTP}} = k_i] = p_F = \frac{1-\bar{P}}{N-1} & \forall k_i \neq \widehat{k_{\text{OTP}}} \end{cases}$$

and consequently the entropy of the random variable k_{OTP} is:

$$H(k_{\text{OTP}}) = p_S \log_{\frac{1}{2}} p_S + \sum_{h \neq \hat{h}} p_F \log_{\frac{1}{2}} p_F = \\ = \bar{P}\log_{\frac{1}{2}} \bar{P} + (1-\bar{P})\log_{\frac{1}{2}} \frac{1-\bar{P}}{N-1}$$

The information available on k_{OTP} is upper bounded by i_{OTP}, so the maximum $\bar{P}$ obtainable is given by the following equation:

$$i_{\text{OTP}} = \Delta H(k_{\text{OTP}}) = n - \bar{P}\log_{\frac{1}{2}} \bar{P} - (1-\bar{P})\log_{\frac{1}{2}} \frac{1-\bar{P}}{N-1} \quad (4.10)$$

Threrefore, the probability of success of the impersonation attack is upper bounded by

$$P_I \leq \bar{P} \quad (4.11)$$

aside from any gathered information on the selected ε-ASU$_2$ hash function.

Collision attack

Eve could try to find a collision between the intercepted message m and any another message $\hat{m}$. In this case it is not necessary to find the exact OTP key, because, since the hash value of the fake message $\hat{m}$ is unvaried, so is the tag.

The selected hash function can be viewed as a random variable on the alphabet $\mathcal{F}$. When the hash functions are uniformly distributed, the entropy of the random variable is $\log_2 F = k$.

Since an ε-ASU$_2$ hash function class is also ε-AU$_2$ (see theorem 2.1), the following bound is valid:

$$|\{h \in \mathcal{F} : h(m) = h(\hat{m})\}| \leq \varepsilon F$$

If Eve succeeds in eliminating all the hash functions but εF, she could be able to substitute a valid message m with another message $\hat{m}$ being confident that the couple $(\hat{m}, t)$ is valid.

It is possible to consider the correct hash function as a realisation of a random variable on the alphabet of the εF remaining hash functions. The maximum entropy of this random variable is $\log_2 \varepsilon F$, and in typical constructions of ε-ASU$_2$ functions ε is equal to $2^{-(n-1)}$.

So, when the opponent collects $n-1$ bits, the size of the $\mathcal{F}_e$ could reach 2^{k-n+1} and the attack could be successful.

Probability of success in case of larger sets $\mathcal{F}_e$

Even if the opponent is not able to shrink the set of candidate hash functions $\mathcal{F}_e$ to a size of εF, it is possible to try an attack by substituting the message m with the fake message $\hat{m}$ which maximises the probability of success of collision.

When m is intercepted, for each $\hat{m} \neq m \in \mathcal{M}$ the opponent can build the set

$$\mathcal{F}(m, \hat{m}) \triangleq \{h \in \mathcal{F}_e : h(m) = h(\hat{m})\}$$

The maximum probability of success of the attack, if all the functions in $\mathcal{F}$ are equiprobable, is given by:

$$P_S = \max_{\hat{m} \in \mathcal{M}} \frac{|\mathcal{F}(m, \hat{m})|}{|\mathcal{F}_e|}$$

The strategy that gets to the highest probability of success, therefore, is choosing a set $\mathcal{F}(m, \hat{m})$ and trying to rule out all the hash functions which do not belong to it. This gives:

$$P_S = \frac{\varepsilon F}{|\mathcal{F}_e|}$$

If i_e is the information collected by Eve, the size of $\mathcal{F}_e$ is:

$$|\mathcal{F}_e| \in [2^{\lfloor k - i_e \rfloor}, 2^{\lceil k - i_e \rceil}]$$

so, if $\varepsilon = 2^{-(n-1)}$, we can write

$$P_S \leq \frac{\varepsilon F}{2^{\lfloor k - i_e \rfloor}} \sim 2^{i_e - n + 1}$$

Upper bound on the probability of success

With a different strategy, the opponent could be interested in maximizing the probabilities of success of a substitution attack, without the need of being sure that the attack will succed, that is, the opponent could not be interested in ruling out completely some hash functions, but he could just aim at increasing as much as possible the difference of probability between "good" and "bad" hash functions.

Consider the case of a collision attack: Eve's target is to divide a given set $\mathcal{F}(m_1, m_2)$, that will be the "good" set, from the remaining set $\mathcal{F}(m_1, m_2)^C$. The maximum size for $\mathcal{F}(m_1, m_2)$ is εF.

Using the terminology of paragraph 4.3, $\mathcal{F}(m_1, m_2)$ corresponds to the set X.

When Eve has no information about h, the entropy of the random variable k_h is k.

After gathering some information, the hash functions in $\bar{\mathcal{F}}$ and those in $\bar{\mathcal{F}}^C$ will have different probabilities. Let $\bar{P} = P[k_h \in \bar{\mathcal{F}}]$ be the probability that k_h is in the set $\bar{\mathcal{F}}$. The p.m.d that gives the maximum entropy for a given $\bar{P}$ is:

$$\begin{cases} P[k_h = h] = p_S = \frac{\bar{P}}{\varepsilon F} & \forall h \in \bar{\mathcal{F}} \\ P[k_h = h] = p_F = \frac{1-\bar{P}}{F - \varepsilon F} & \forall h \in \bar{\mathcal{F}}^C \end{cases}$$

and consequently the entropy of the random variable k_h is:

$$\begin{aligned} H(k_h) &= \sum_{h \in \bar{\mathcal{F}}} p_S \log_{\frac{1}{2}} p_S + \sum_{h \in \bar{\mathcal{F}}^C} p_F \log_{\frac{1}{2}} p_F = \\ &= \bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon F} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{F - \varepsilon F} = \\ &= k + \bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \varepsilon} \end{aligned}$$

The loss of entropy after i authentications, therefore, is:

$$\Delta H = H(k_h) - H(k_h | m_1, t_1, \ldots, m_i, t_i)_{max} = -\bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon} - (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \varepsilon} \quad (4.12)$$

and the corresponding probability of success of a collision attack is upper bounded by

$$P_{\text{coll}} \leq \bar{P} \quad (4.13)$$

This probability of success of a collision attack is valid only when the εF hash functions contained in $\bar{\mathcal{F}}$ correspond to some set $\mathcal{F}(m, \hat{m})$. Nevertheless, it is a valid upper bound to the probability against all collision-based attacks.

Substitution attack

Another attack that an opponent could try to accomplish is trying to recover the OTP key used in an intercepted couple (m, t), then trace the unencrypted hash value v and then perform the original Cederlöf-Larssen attack.

OTP key recovery

To recover the OTP key of an intercepted couple (m, t), the opponent must know the exact hash value v of the intercepted message m, since $k_{\text{OTP}} = t \oplus v$.

The hash function class is ε-ASU$_2$, so by definition 2.2:

$$|\mathcal{F}(m, v)| = |\{h \in \mathcal{F} : h(m) = v\}| = \frac{F}{N} \quad \forall m \in \mathcal{M}, \forall v \in \mathcal{T}$$

To reduce the size of $\mathcal{F}_e$ to $\frac{F}{N}$, which corresponds to a maximum entropy of $k-n$ bits, Eve has to gain at least n bits of information on the hash function. When this happens, it may be that all the candidate functions belong to a single block $\mathcal{F}(m, v)$ for some $v \in \mathcal{T}$. Then v is determined, and, as a consequence, so is the OTP key k_{OTP}.

Probability of success in case of larger sets $\mathcal{F}_e$ We can repeat the procedure of the collision attack. To study the probability of decryption of the OTP key, we note that it is upper bounded by the sum of the probabilities of the $\frac{F}{N}$ most probable hash functions. In fact, if this set $\bar{\mathcal{F}}$ is equal to one of the sets $\mathcal{F}(m, v)$ for some m and some v, the opponent would have exactly this probability of recovering the OTP key.

If the opponent uses the strategy of using all the information to rule out as many hash functions as possible and to have a uniform distribution of probability on all the candidates in $\mathcal{F}_e$, the probability of success of the attack is given by:

$$P_S = \max_{\substack{m \in \mathcal{M} \\ v \in \mathcal{T}}} \frac{|\mathcal{F}(m, v)|}{|\mathcal{F}_e|} = \frac{|\bar{\mathcal{F}}|}{|\mathcal{F}_e|} = \frac{F}{N \cdot |\mathcal{F}_e|}$$

If i_e is the information collected by Eve, the size of $\mathcal{F}_e$ is:

$$|\mathcal{F}_e| \in [2^{\lfloor k - i_e \rfloor}, 2^{\lceil k - i_e \rceil}]$$

so we can write

$$P_S \leq \frac{F}{N \cdot 2^{\lfloor k - i_e \rfloor}} \sim 2^{i_e - n}$$

Upper bound on the probability of successful recovery of k_{OTP}

As for the collision attack, we can study the case where the opponent just aims at increasing as much as possible the difference of probability between "good" and "bad" hash functions. In this case the set X of paragraph 4.3 is a given $\mathcal{F}(m, v)$.

Given a certain amount of extracted information, Eve wants to maximise the probabilities of the $\frac{F}{N}$ most probable functions. If this set $\bar{\mathcal{F}}$ is one of the sets $\mathcal{F}(m, v)$ for some $m \in \mathcal{M}$ and $v \in \mathcal{T}$, than Eve is able to accomplish the substitution attack with a probability of success that is the sum of all the hash functions contained in $\bar{\mathcal{F}}$.

Again, we study how much information Eve has to gather in order to have a certain increase in her probability of success.

The probability that k_h is in the set $\bar{\mathcal{F}}$ is given by $\bar{P} = P[k_h \in \bar{\mathcal{F}}] = \sum_{h \in \bar{\mathcal{F}}} P[k_h = h]$. The p.m.d that gives the maximum entropy and, at the same time, the maximum probability $\bar{P}$, is:

$$\begin{cases} P[k_h = h] = p_S = \frac{\bar{P}}{F/N} & \forall h \in \bar{\mathcal{F}} \\ P[k_h = h] = p_F = \frac{1-\bar{P}}{F - \frac{F}{N}} & \forall h \in \bar{\mathcal{F}}^C \end{cases}$$

and consequently the entropy of the random variable h is:

$$\begin{aligned} H(k_h) &= \sum_{h \in \bar{\mathcal{F}}} p_S \log_{\frac{1}{2}} p_S + \sum_{h \in \bar{\mathcal{F}}^C} p_F \log_{\frac{1}{2}} p_F = \\ &= \bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\frac{F}{N}} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{F - \frac{F}{N}} = \\ &= k + \bar{P} \log_{\frac{1}{2}} N\bar{P} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \frac{1}{N}} \end{aligned}$$

The loss of entropy after i authentications, therefore, is:

$$\Delta H = H(k_h) - H(k_h | m_1, t_1, \ldots, m_i, t_i)_{max} = -\bar{P} \log_{\frac{1}{2}} N\bar{P} - (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \frac{1}{N}} \quad (4.14)$$

and the corresponding probability of succeeding in recovering the OTP key is upper bounded by

$$P_{\text{OTP}} \leq \bar{P} \quad (4.15)$$

This upper bound gives a limit to the probability of finding out a single key of the OTP encryption, and therefore the value of a single hash value v. This information alone does not lead to an effective attack, except for the substitution attack described in the following paragraph.

Upper bound on the probability of success of Cederlöf-Larsson attack

When the opponent has identified the right OTP key k_{OTP}, it is possible to identify the right couple (m, v), and therefore deploy the original Cederlöf-Larsson attack. While their original paper just describe the attack, we use the same procedure of the previous paragraphs to give an upper bound on the probability of success of the substitution attack of an ε-ASU$_2$.

With the hash value v, Eve is able to identify the right partition $\mathcal{F}(m, v)$ that contains the selected hash function, and can find the set of candidate functions calculating the intersection between the previous set of candidate functions and $\mathcal{F}(m, v)$. Let $\mathcal{F}_e$ be this intersection, that is the updated set of all the possible hash functions. If $\mathcal{F}_e$ corresponds to some $\mathcal{F}(m, \hat{m})$ (as defined in paragraph 2.9), Eve would be able to deploy a successful substitution attack.

Since $\mathcal{F}(m, \hat{m}) \leq \varepsilon \frac{F}{N}$ for definition 2.2, Eve's best strategy is to maximise the probabilities of the $\varepsilon \frac{F}{N}$ most probable functions in $\mathcal{F}_e$.

Let $\bar{\mathcal{F}}$ be the set of the $\varepsilon \frac{F}{N}$ most probable functions. The probability that the selected hash function k_h is in the set $\bar{\mathcal{F}}$ is given by $\bar{P} = P[k_h \in \bar{\mathcal{F}}] = \sum_{h \in \bar{\mathcal{F}}} P[k_h = h]$. Let $\bar{\mathcal{F}}^C$ be defined as $\mathcal{F}_e \setminus \bar{\mathcal{F}}$. The p.m.d that gives the maximum entropy and, at the same time, the maximum probability $\bar{P}$ is:

$$\begin{cases} P[k_h = h] = p_S = \frac{\bar{P}}{\varepsilon F/N} & \forall h \in \bar{\mathcal{F}} \\ P[k_h = h] = p_F = \frac{1-\bar{P}}{\frac{F}{N} - \varepsilon \frac{F}{N}} & \forall h \in \bar{\mathcal{F}}^C \end{cases}$$

and consequently the entropy of the random variable h is:

$$\begin{aligned} H(h) &= \sum_{h \in \bar{\mathcal{F}}} p_S \log_{\frac{1}{2}} p_S + \sum_{h \in \bar{\mathcal{F}}^C} p_F \log_{\frac{1}{2}} p_F = \\ &= \bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon \frac{F}{N}} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{\frac{F}{N} - \varepsilon \frac{F}{N}} = \\ &= k - n + \bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon} + (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \varepsilon} \end{aligned}$$

Since the initial entropy of the hash function, given the observed couple (m, v) is $k-n$, the loss of entropy is:

$$\Delta H = H(h) - H(h|m, v) = -\bar{P} \log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon} - (1 - \bar{P}) \log_{\frac{1}{2}} \frac{1 - \bar{P}}{1 - \varepsilon} \tag{4.16}$$

and the corresponding probability of succeeding in substituting (m, t) with a different pair $(\hat{m}, \hat{t})$ is upper bounded by

$$P_{\text{ced}} \leq \bar{P} \tag{4.17}$$

The final probability of success of a combined attack aimed at finding the right OTP and then perform a Cederlöf -Larsson attack is therefore upper bounded by:

$$P_{\text{comb}} = P_{\text{OTP}} \cdot P_{\text{ced}} \leq \bar{P}_{\text{OTP}} \cdot \bar{P}_{\text{ced}} \tag{4.18}$$

where $\bar{P}_{\text{OTP}}$ is $\bar{P}$ as defined in paragraph 4.3 and $\bar{P}_{\text{ced}}$ is $\bar{P}$ as defined in the current paragraph.

Upper bound on the probability of success of a substitution attack

The actual probability of success of a substitution attack can be upper bounded by considering the "good" subset $\bar{\mathcal{F}}$ as a set with $\varepsilon\frac{F}{N}$ elements, which could be all the functions which belong to a same set $\mathcal{F}(m, \hat{m})$.

In this case, $|\bar{\mathcal{F}}| = \varepsilon\frac{F}{N}$ and $|\mathcal{F}_e| = F - \varepsilon\frac{F}{N}$.

The probability that the selected hash function k_h is in the set $\bar{\mathcal{F}}$ is given by $\bar{P} = P[k_h \in \bar{\mathcal{F}}] = \sum_{h\in\bar{\mathcal{F}}} P[k_h = h]$. Let $\bar{\mathcal{F}}^C$ be defined as $\mathcal{F}_e \setminus \bar{\mathcal{F}}$. The p.m.d that gives the maximum entropy and, at the same time, the maximum probability $\bar{P}$, is:

$$\begin{cases} P[k_h = h] = p_S = \frac{\bar{P}}{\varepsilon F/N} & \forall h \in \bar{\mathcal{F}} \\ P[k_h = h] = p_F = \frac{1-\bar{P}}{F-\varepsilon\frac{F}{N}} & \forall h \in \bar{\mathcal{F}}^C \end{cases}$$

and consequently the entropy of the random variable h is:

$$\begin{aligned} H(h) &= \sum_{h\in\bar{\mathcal{F}}} p_S \log_{\frac{1}{2}} p_S + \sum_{h\in\bar{\mathcal{F}}^C} p_F \log_{\frac{1}{2}} p_F = \\ &= \bar{P}\log_{\frac{1}{2}} \frac{\bar{P}}{\varepsilon\frac{F}{N}} + (1-\bar{P})\log_{\frac{1}{2}} \frac{1-\bar{P}}{F-\varepsilon\frac{F}{N}} = \\ &= k + \bar{P}\log_{\frac{1}{2}} \frac{\bar{P}}{\frac{\varepsilon}{N}} + (1-\bar{P})\log_{\frac{1}{2}} \frac{1-\bar{P}}{1-\frac{\varepsilon}{N}} \end{aligned}$$

Since the initial entropy of the hash function is k, the loss of entropy after i authentications is:

$$\Delta H = H(k_h) - H(k_h|m_1, t_1, \ldots, m_i, t_i)_{max} = -\bar{P}\log_{\frac{1}{2}} \frac{\bar{P}}{\frac{\varepsilon}{N}} - (1-\bar{P})\log_{\frac{1}{2}} \frac{1-\bar{P}}{1-\frac{\varepsilon}{N}} \quad (4.19)$$

and the corresponding probability of succeeding in substituting (m, t) with a different pair $(\hat{m}, \hat{t})$ is upper bounded by

$$P_{\text{sub}} \leq \bar{P} \quad (4.20)$$

Hash function lifetime

Let P_D be Eve's global probability of a successful deception attack. P_D is upper bounded by the sum of the probabilities of success of all the possible attacks, that we can summarise in impersonation, collision, and substitution attacks:

$$P_D \leq P_I + P_{\text{coll}} + P_{\text{sub}}$$

To get an upper bound to the number of times a single hash function can be used for, we fix a maximum deception probability $P_D^{\max}$ that Eve can reach, after which we change hash function.

We choose to evenly distribute the maximum probability of deception among the three attacks. Let $P_{\max} = P_D^{\max}/3$. Then:

$$P_I \leq P_{\max} \tag{4.21}$$
$$P_{\text{coll}} \leq P_{\max} \tag{4.22}$$
$$P_{\text{sub}} \leq P_{\max} \tag{4.23}$$

To satisfy equation (4.21), we must have $2^{-n} \leq P_{\max}$, so a necessary condition to have feasible solutions is:

$$P_D^{\max} \geq 3 \cdot 2^{-n} \tag{4.24}$$

The upper bound on P_I derived by equation (4.10) on P_I does not depend on the number of observation, so it does not weaken when a new valid authentication is seen.

With the equations (4.12) and (4.19) we can calculate the minimum information that the opponent must have in order to be able to accomplish an attack with given maximum probability of success.

Now it is possible to calculate the lifetime of each ε-ASU$_2$ hash function under each attack: it is sufficient to calculate first the minimum information ΔH needed by the opponent to accomplish an attack with success probability $P_{\max}$, and then the number of authentication rounds needed to obtain the given amount of information:

$$\Lambda = \frac{\Delta H - i_h}{i_{\text{OTP}}} = \frac{\Delta H - \ell \cdot k}{\ell \cdot n} = \frac{\Delta H}{\ell \cdot n} - \frac{k}{n} \tag{4.25}$$

Lifetime for collision attack

$$\Delta H_{\text{coll}} = -P_{\max} \log_{\frac{1}{2}} \frac{P_{\max}}{\varepsilon} - (1 - P_{\max}) \log_{\frac{1}{2}} \frac{1 - P_{\max}}{1 - \varepsilon} \tag{4.26}$$

$$\Lambda_{\text{coll}} = \frac{\Delta H_{\text{coll}}}{\ell \cdot n} - \frac{k}{n} \tag{4.27}$$

Lifetime for substitution attack

$$\Delta H_{\text{sub}} = -P_{\max} \log_{\frac{1}{2}} \frac{P_{\max}}{\frac{\varepsilon}{N}} - (1 - P_{\max}) \log_{\frac{1}{2}} \frac{1 - P_{\max}}{1 - \frac{\varepsilon}{N}} \tag{4.28}$$

$$\Lambda_{\text{sub}} = \frac{\Delta H_{\text{sub}}}{\ell \cdot n} - \frac{k}{n} \tag{4.29}$$

The global lifetime matches the highest between (4.27) and (4.29):

$$\Lambda = \max\{\Lambda_{\text{coll}}, \Lambda_{\text{sub}}\} \tag{4.30}$$

Parameters optimisation

Starting from a required $P_D^{\max}$, a set M of messages to be authenticated, and a given class of hash functions, it is possible to optimise the sizes of both the hash function class and the tag length n, in order to minimise the rate of the secret key needed for the authentication.

The key rate R is given by

$$R = \frac{k}{\Lambda} + n \tag{4.31}$$

We note that k can be expressed as a function of n and the size M of the message set, while Λ is a function of $P_D^{\max}$, k, n and ℓ. M and ℓ are parameters given by the specific QKD system, and $P_D^{\max}$ is the required maximum probability of deception, so we can plot the rate R in function of M, ℓ, n and $P_D^{\max}$.

The minimum authentication key rate R is obtained with the lowest possible value of ℓ, which is achieved only with a considerable loss of the privacy amplification output rate R_{pa}. The net key rate R_f obtained by the QKD is given by

$$R_f = R_{\text{pa}} - R \tag{4.32}$$

and therefore the maximum R_f is attained by maximising the difference between R_{pa} and R, both calculated with the same parameter ℓ.

We can express the length of the final key after the privacy amplification as a function of the target average information leaked to Eve during the QKD process. Using the results of [BBCM95] it is possible to derive the following result (see [CBC+11]).

Let N_{rec} be the length of the reconciled key, N_{sec} the length of secure key obtained after one round of privacy amplification, N_{rev} the number of bits revealed during the key reconciliation phase, and t Eve's Rényi information on the reconciled key. Let P_{miss} represent the probability that Eve is observing on average more photons than the number predicted by the estimation made during the previous phases of QKD.

With probability $1-P_{\text{miss}}$, for every possible value of b the following bound is verified [CBC+11]:

$$I_{\text{leak}} \leq I_{\text{tar}}(N_{\text{sec}}, b) = N_{\text{sec}}\mathrm{P}[t > b] + \frac{1}{2^{N_{\text{rec}}-N_{\text{rev}}-N_{\text{sec}}-b}\ln 2} \tag{4.33}$$

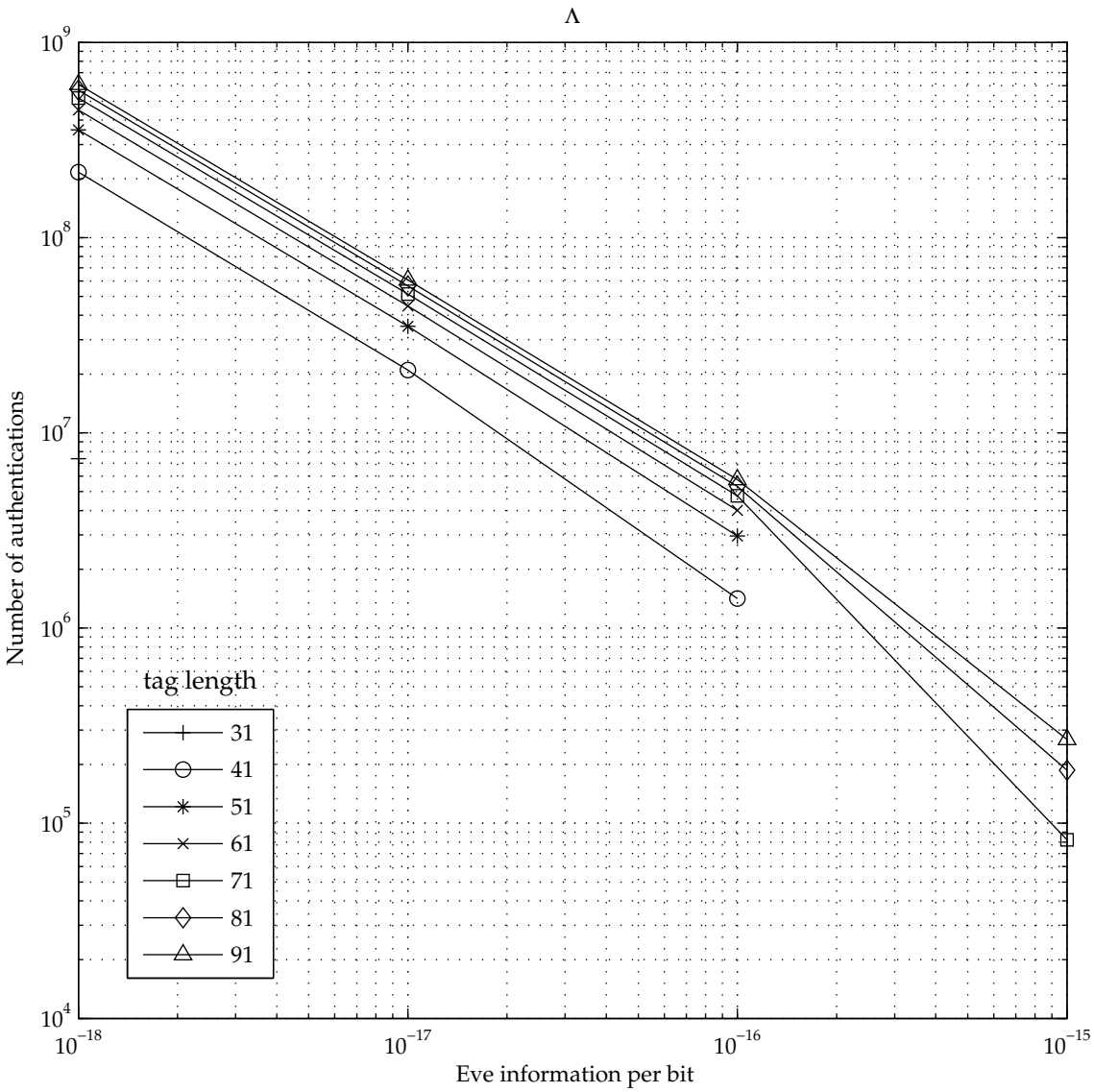
If we bind I_{leak} to be smaller than a fixed value δ, the maximum N_{sec} is obtained by the following result [CBC+11]:

$$N_{\text{sec}} = \max\left\{a : \min_b I_{\text{tar}}(a, b) \leq \delta\right\}$$

Lifetime and rates for different classes of hash functions

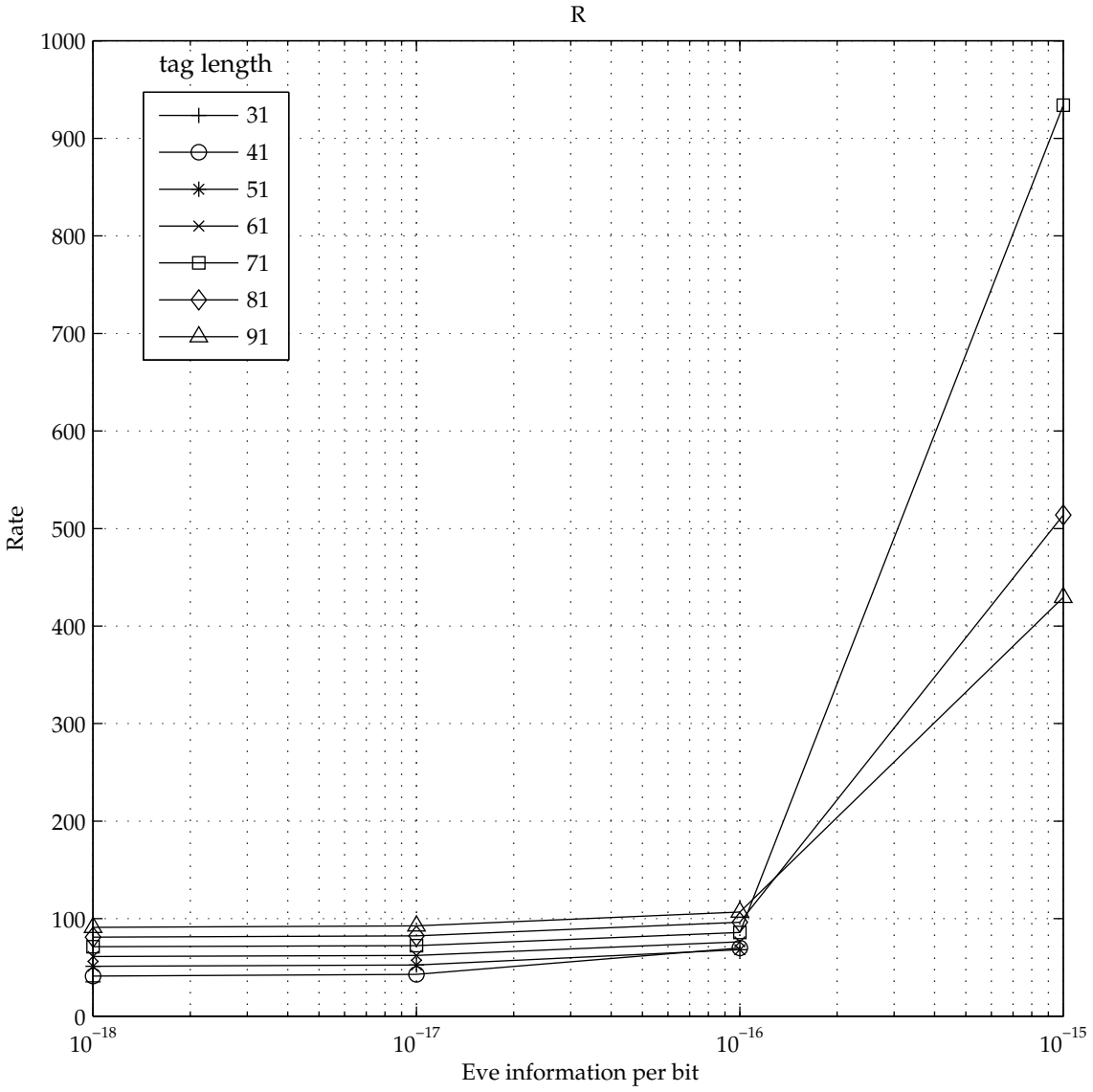
The following figures show the performances of different classes of hash functions designed to authenticate messages long up to 10^6 bits. The graphs plot lifetime Λ and rate R against the tag length n for a given P_{max} of 2^{-30}.

It is possible to note that Reed-Solomon based class performs quite as well as the other classes in terms of rate, and is able to authenticate with the required security target even with high values of ℓ (figures 4.7 and 4.8).



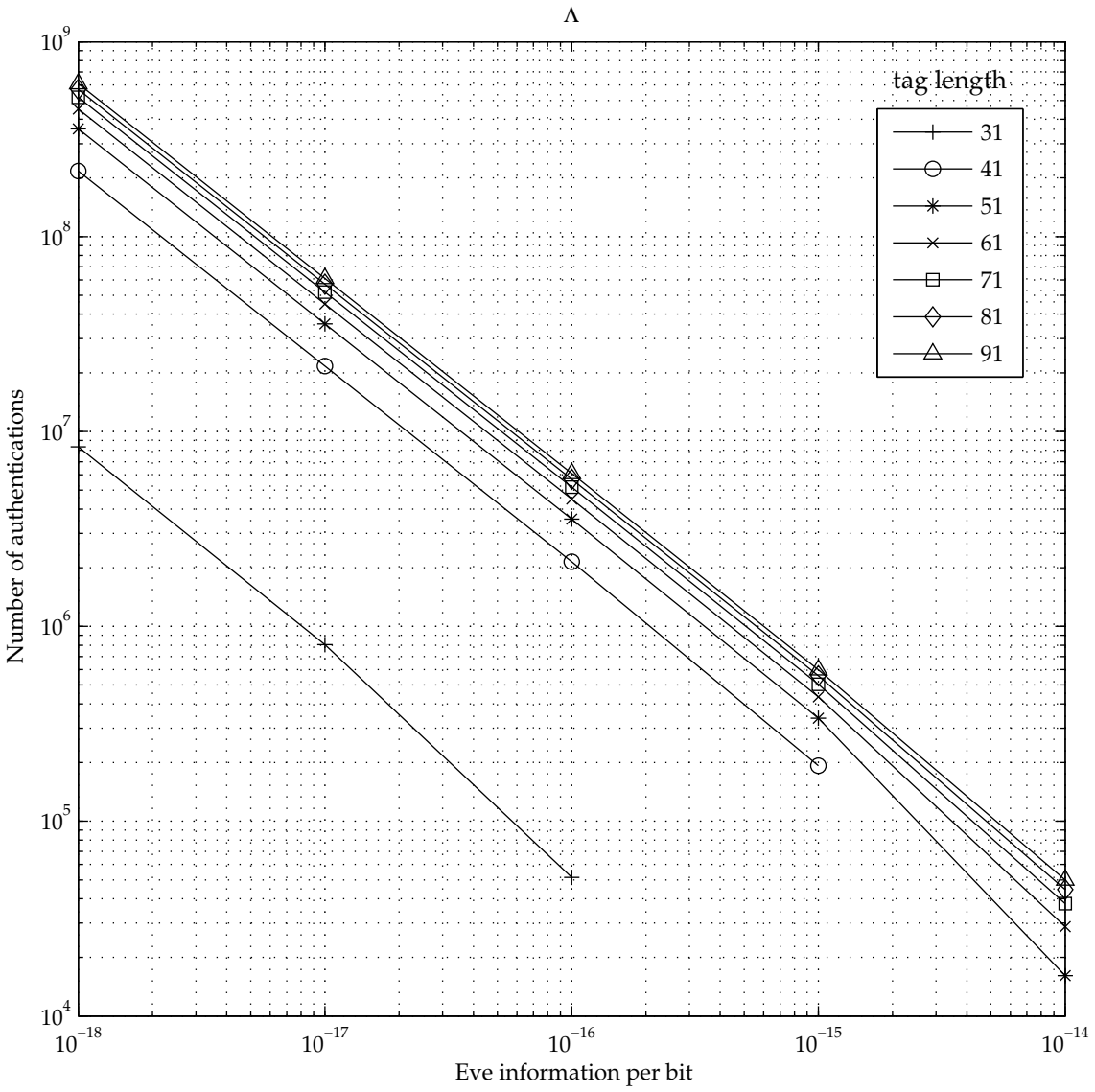
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.1: Lifetime for SU_2 hash function class obtained by random binary matrices.



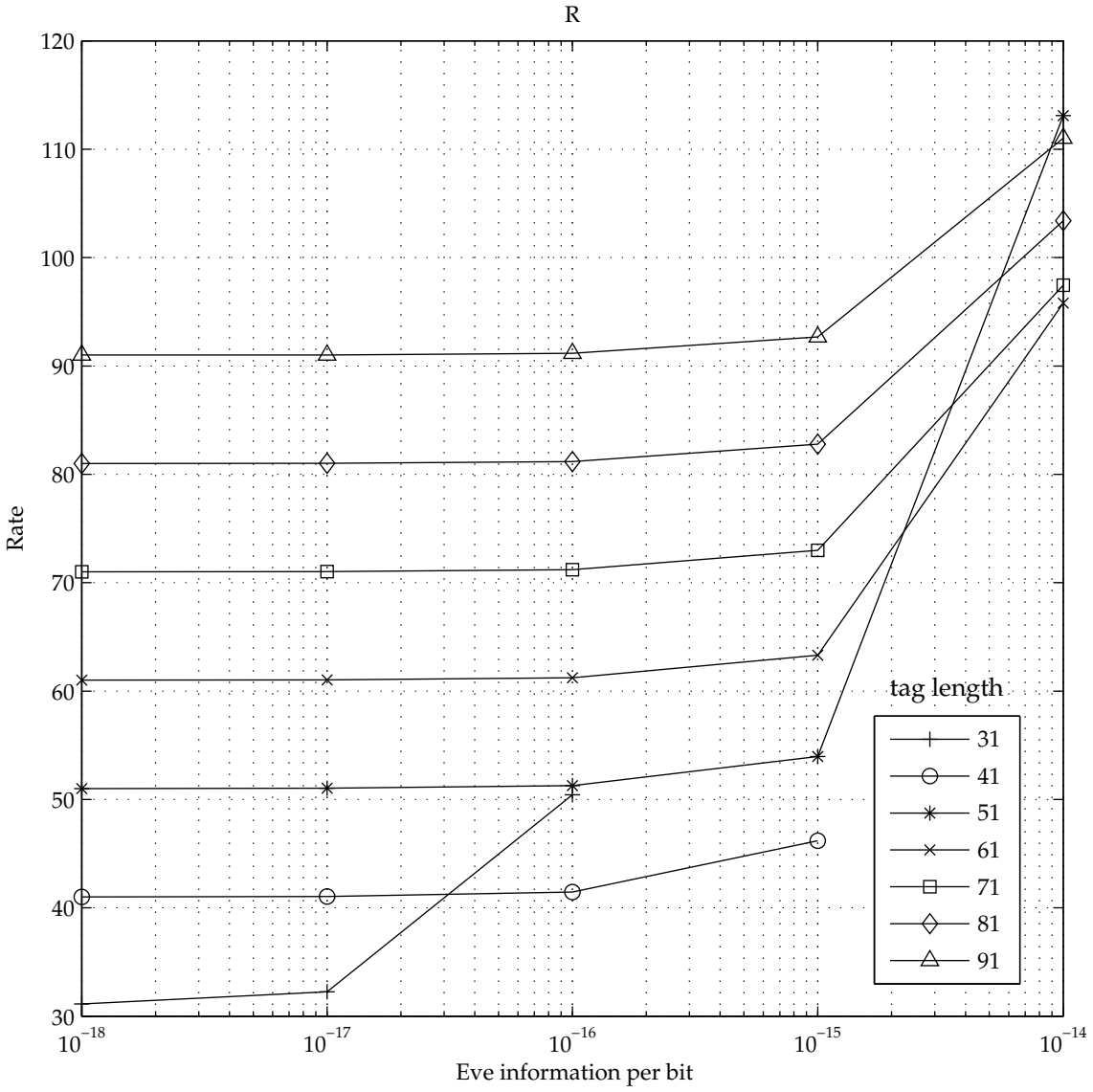
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.2: Rate for SU_2 hash function class obtained by random binary matrices.



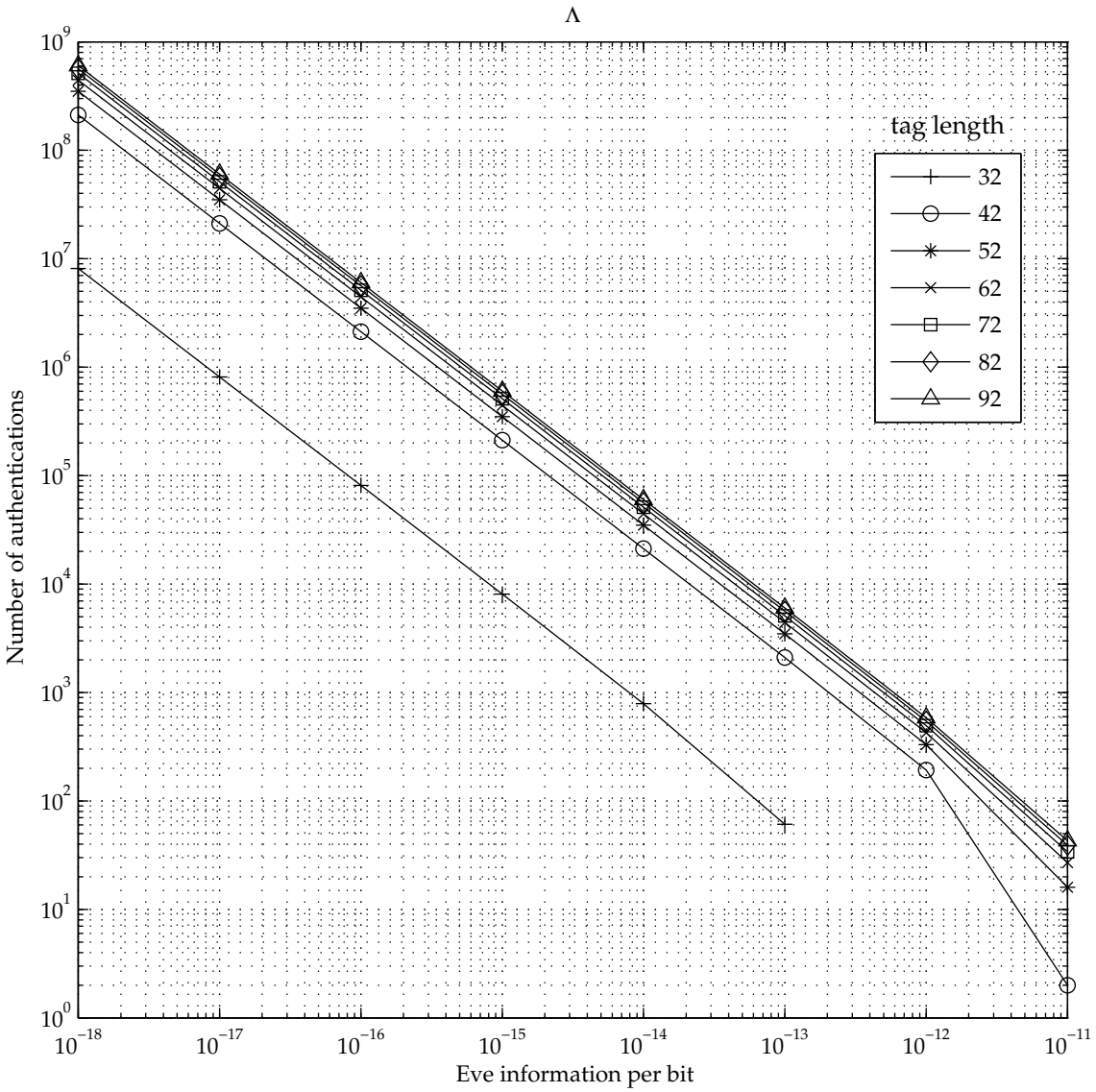
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.3: Lifetime for SU_2 hash function class obtained by Toeplitz binary matrices.



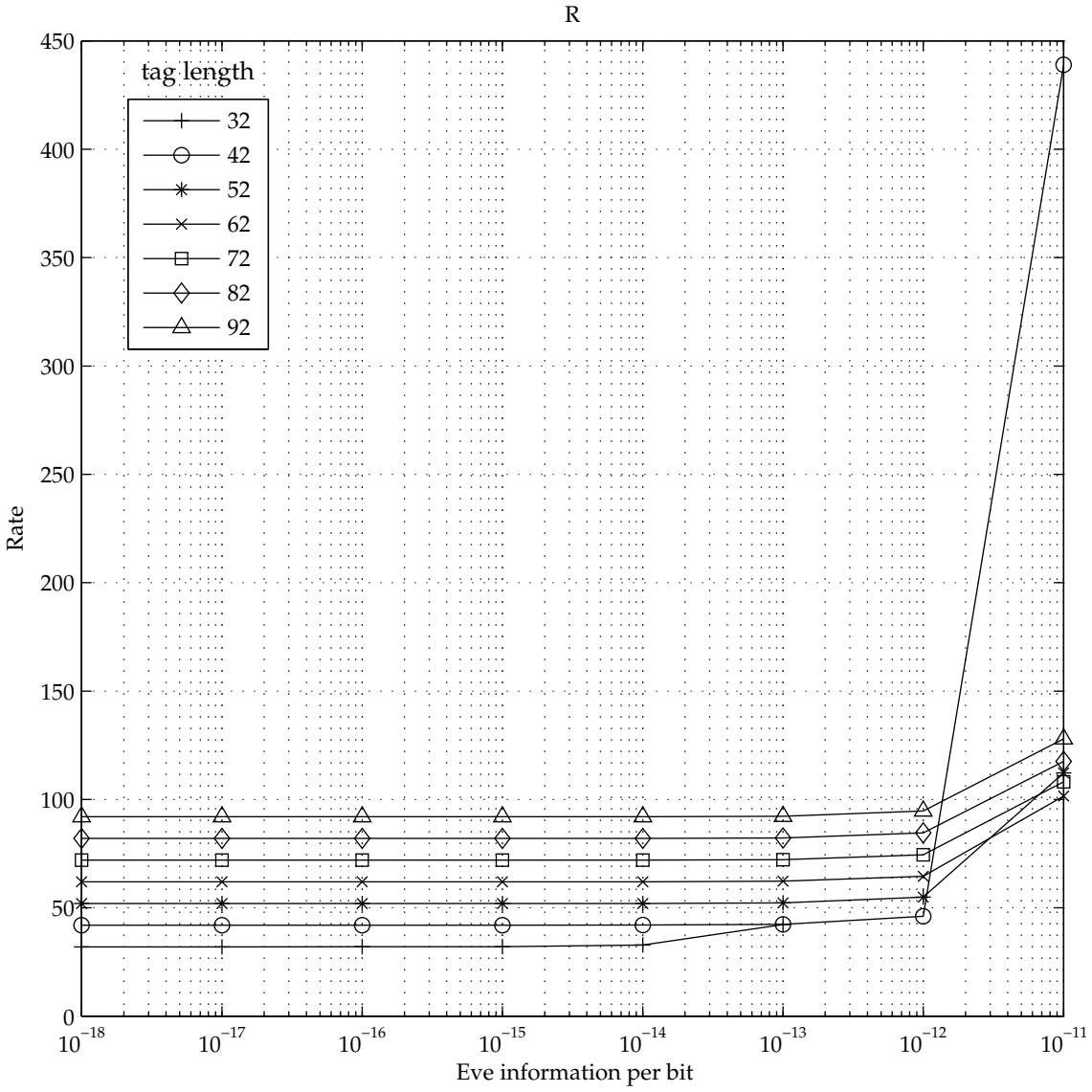
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.4: Rate for SU_2 hash function class obtained by Toeplitz binary matrices.



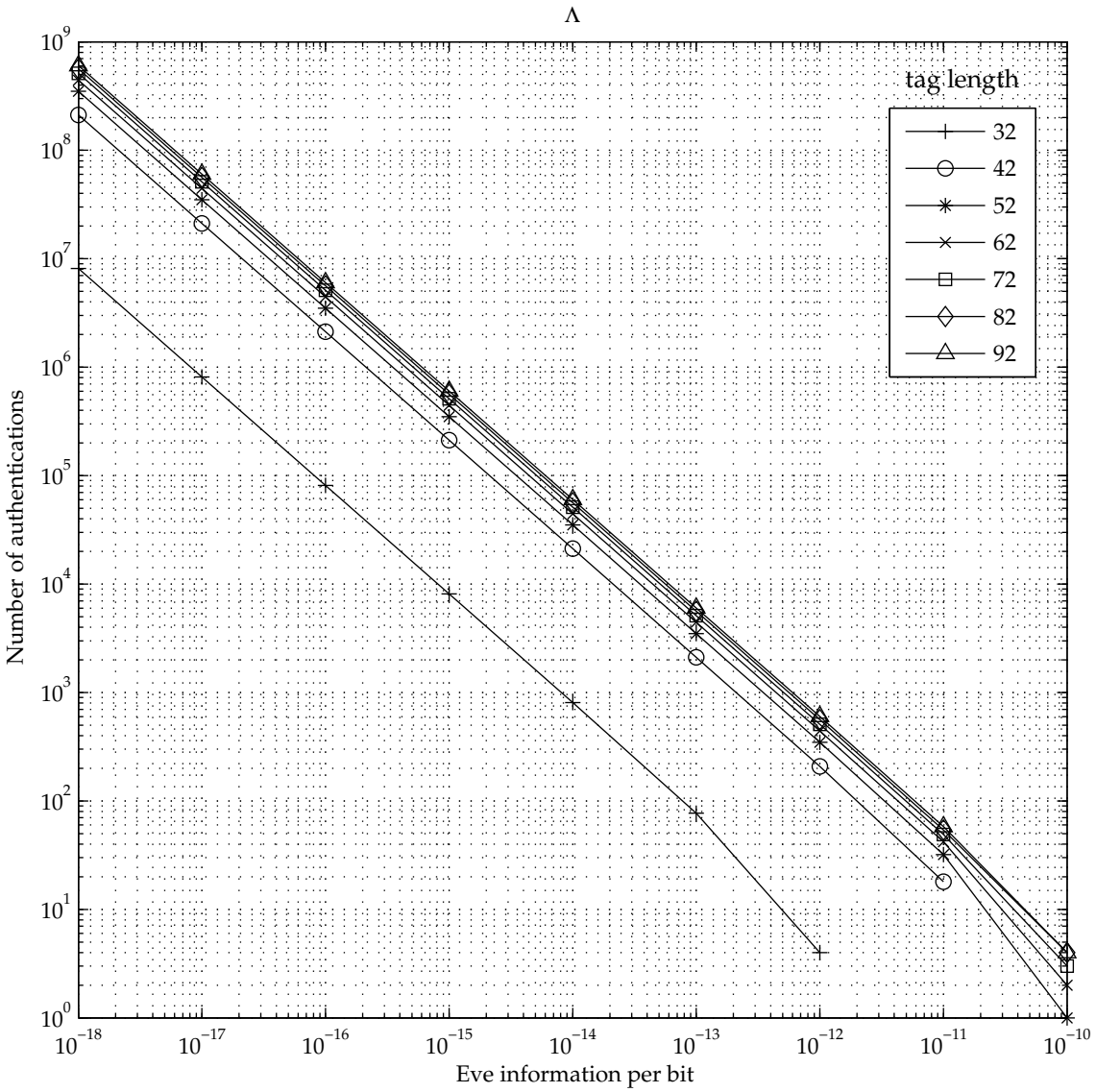
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.5: Lifetime for ε-ASU$_2$ Stinson hash function class.



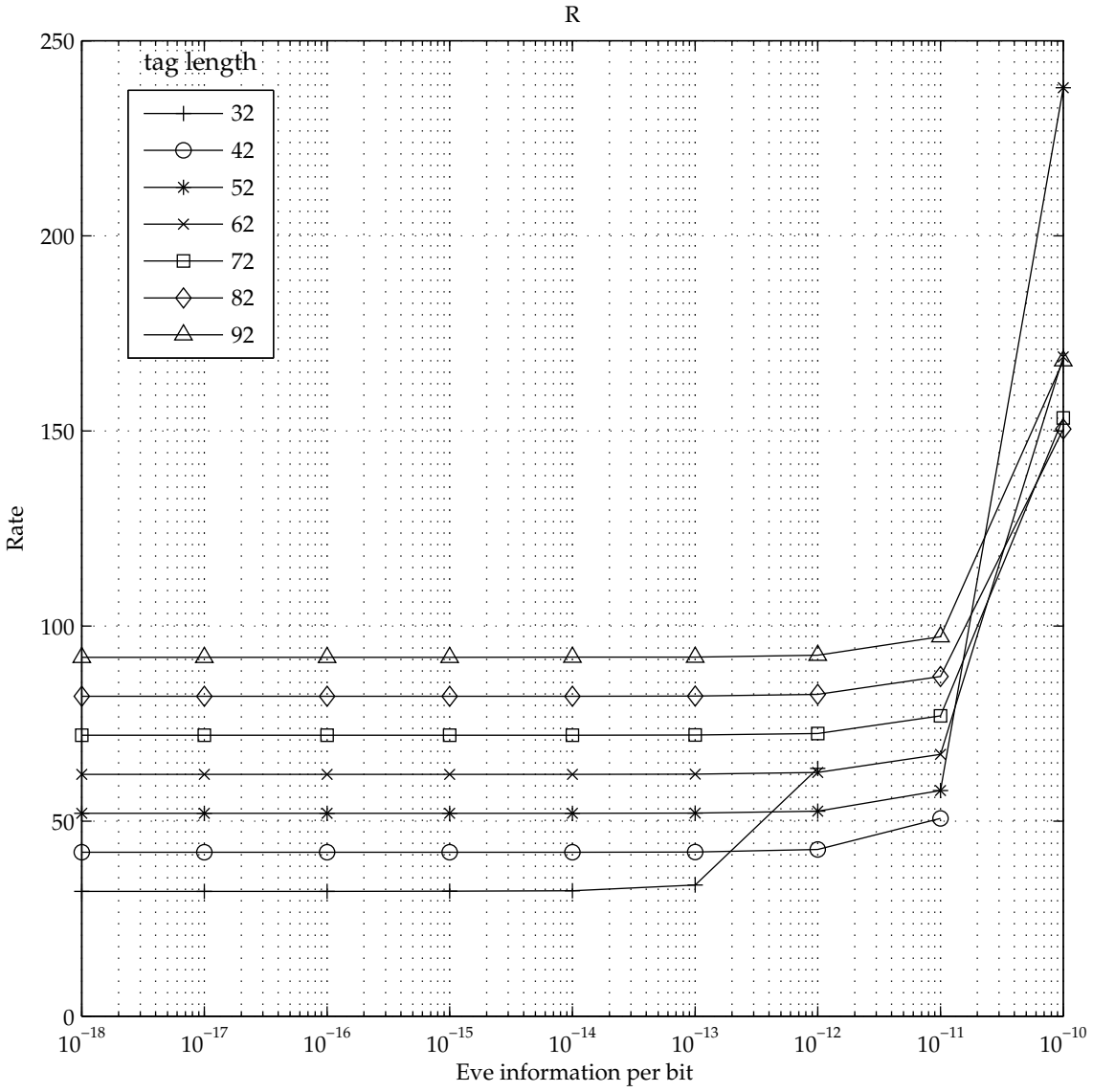
Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.6: Rate for ε-ASU$_2$ Stinson hash function class.



Parameters	
$P_{\max}$	2^{-30}
m	10^6

Figure 4.7: Lifetime for ε-ASU$_2$ hash function class obtained by Reed-Solomon codes.



Parameters	
$P_{\max}$	2^{-30}
m	10^6

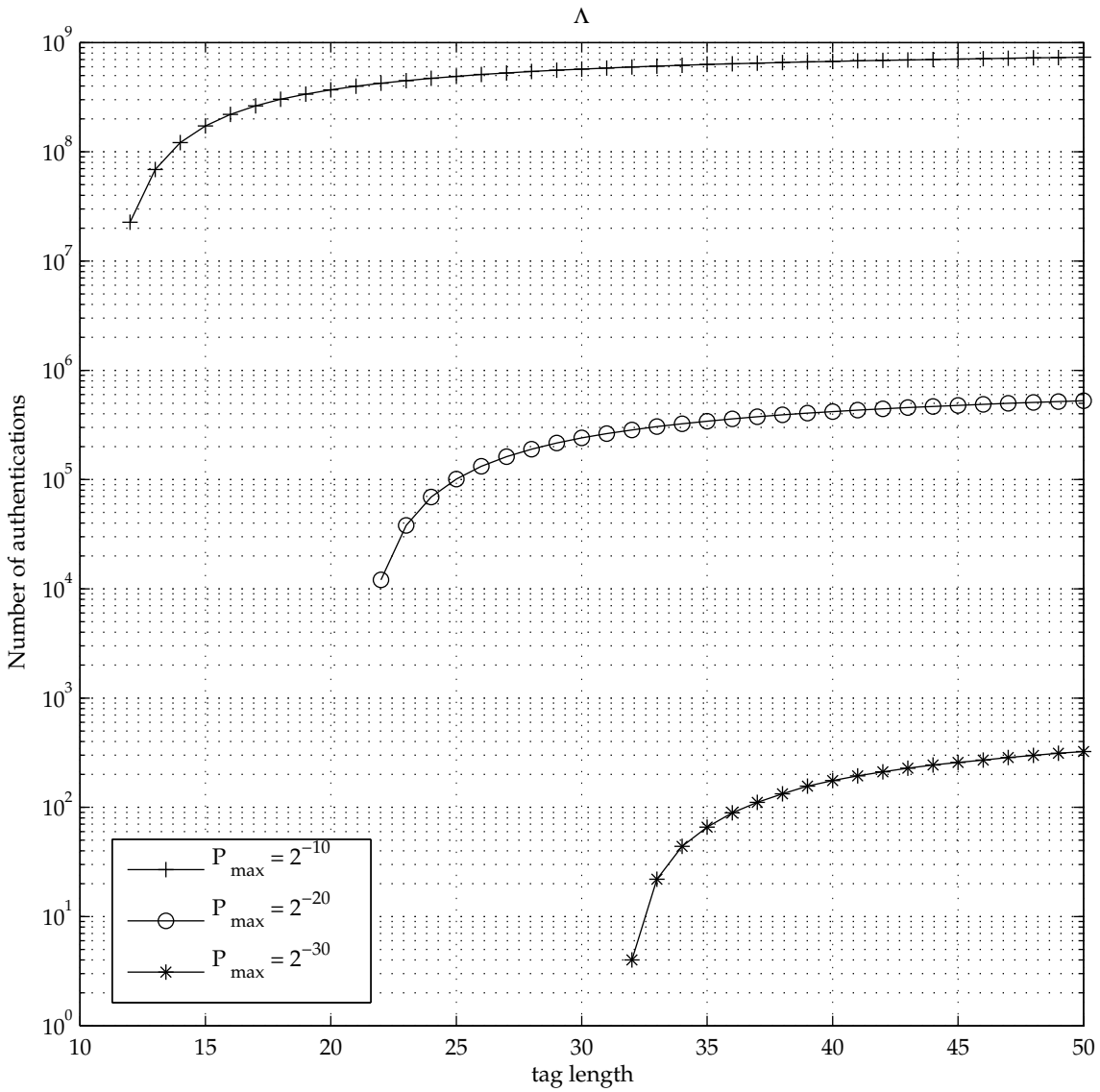
Figure 4.8: Rate for ε-ASU$_2$ hash function class obtained by Reed-Solomon codes.

Behaviour with different P_{max}

Figures 4.9 and 4.10 show the lifetime and the key rate of Reed-Solomon based classes for different values $p_{\max}$.

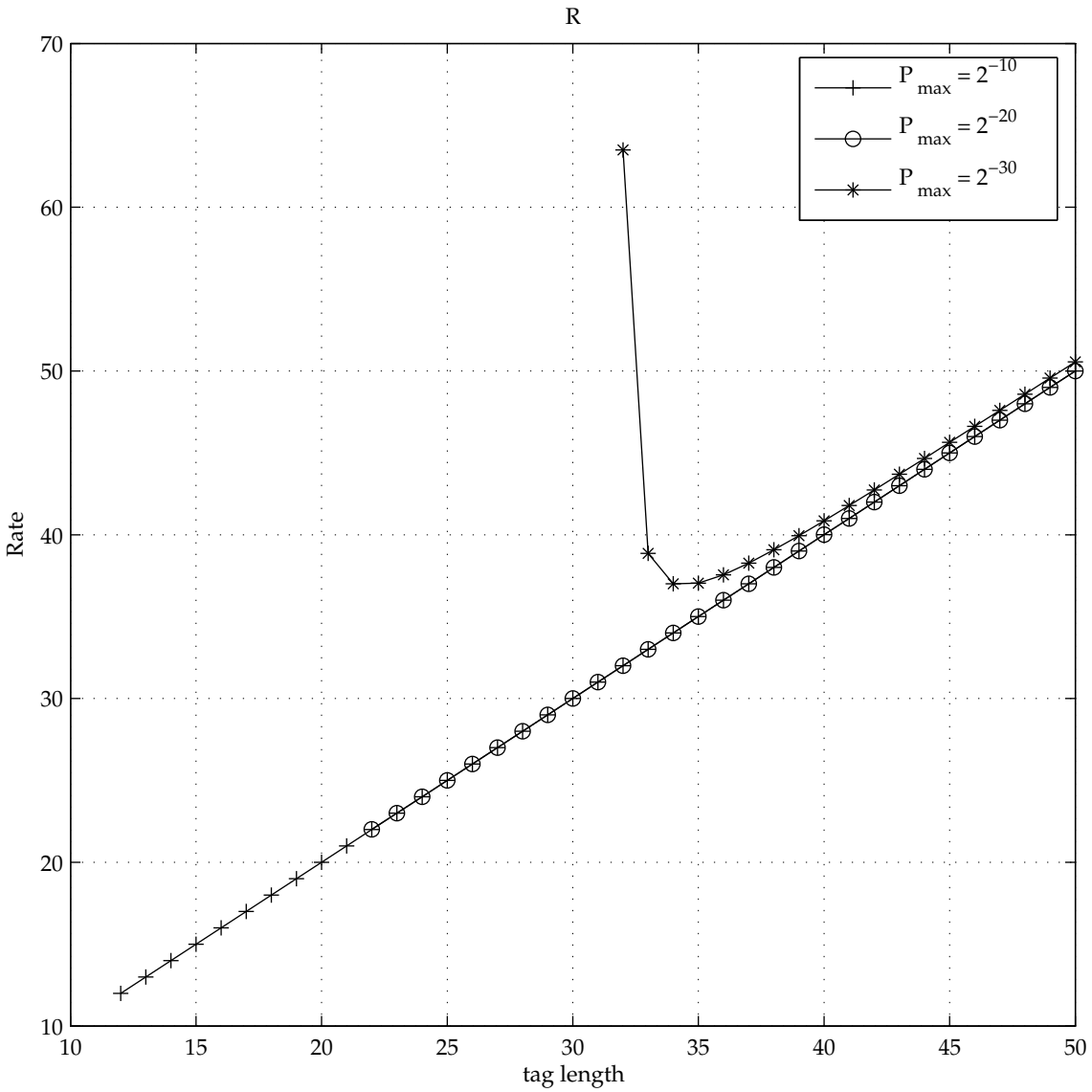
It is possible to note that all the lifetime lines of figure 4.9 tend to different asymptotical values.

In figure 4.10 it can be seen that, for long tags, the rate grows linearly with the tag length, while the lowest rate depends on the specific value of $P_{\max}$: in some cases ($P_{\max} = 2^{-10}, 2^{20}$) it coincides with the smallest possible tag length, while sometimes the shortest tag length has a short timelife that does not allow the cost of renewing k_h to be cushioned (this is the case of the peak for $P_{\max} = 2^{-30}$).



Parameters	
ℓ	10^{-12}
m	10^6

Figure 4.9: Lifetime for ε-ASU$_2$ hash function class obtained by Reed-Solomon codes for different values of $P_{\max}$.



Parameters	
ℓ	10^{-12}
m	10^6

Figure 4.10: Rate for ε-ASU$_2$ hash function class obtained by Reed-Solomon codes for different values of $P_{\max}$.

Comparison between different classes of hash functions

The rate R is strongly dependant on the size of the hash function class. Large classes, as classes based on Toeplitz matrices (see figure 4.11a), have long keys and, therefore, a higher initial entropy $H(k_h)$.

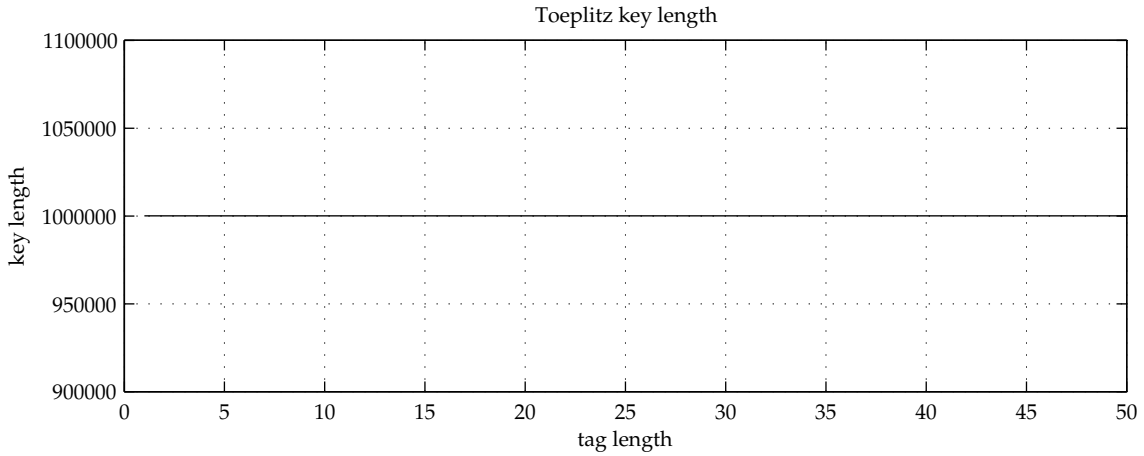
Small classes like Stinson construction (see figure 4.11b) and those based on Reed-Solomon codes (4.11c) have shorter keys and smaller initial entropy.

One could expect that classes with higher initial entropy would last longer, but, if the hash function key k_h has been generated by the QKD, the attacker knows a fixed rate (ℓ) of information on it, and this is enough to wipe out all the advantages given by a long key.

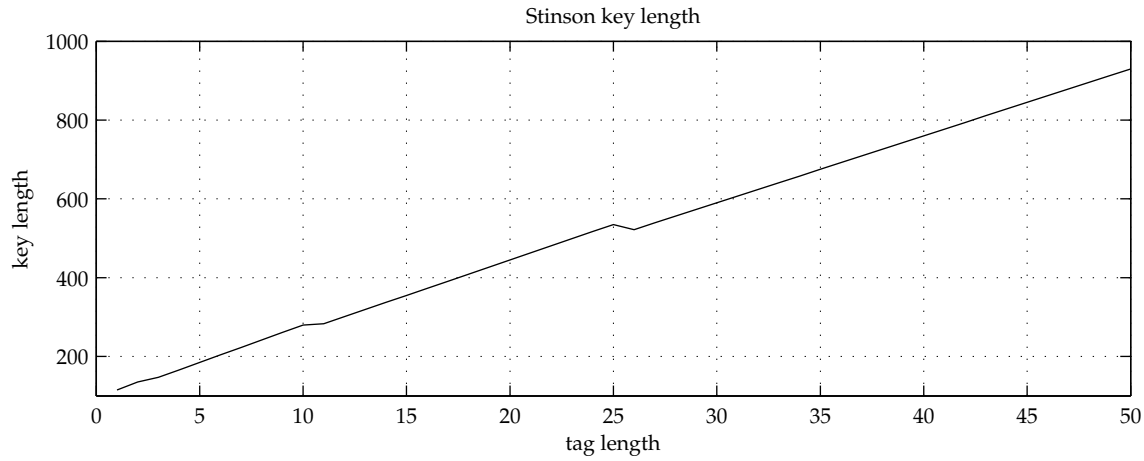
In fact, the rate required by large classes is higher than that of small ones, because the renewal of the selected hash function is much more expensive, and more frequent.

See figures 4.12 and 4.13 for a comparison between the performances of Toeplitz, Stinson and Reed-Solomon based classes with the same parameters. Stinson and Reed-Solomon based classes always perform better than Toeplitz ones (and reach higher values of ℓ, see paragraph 4.3). In figure 4.13, which shows a closeup of 4.12, it is possible to appreciate the performance difference between Stinson and Reed-Solomon codes.

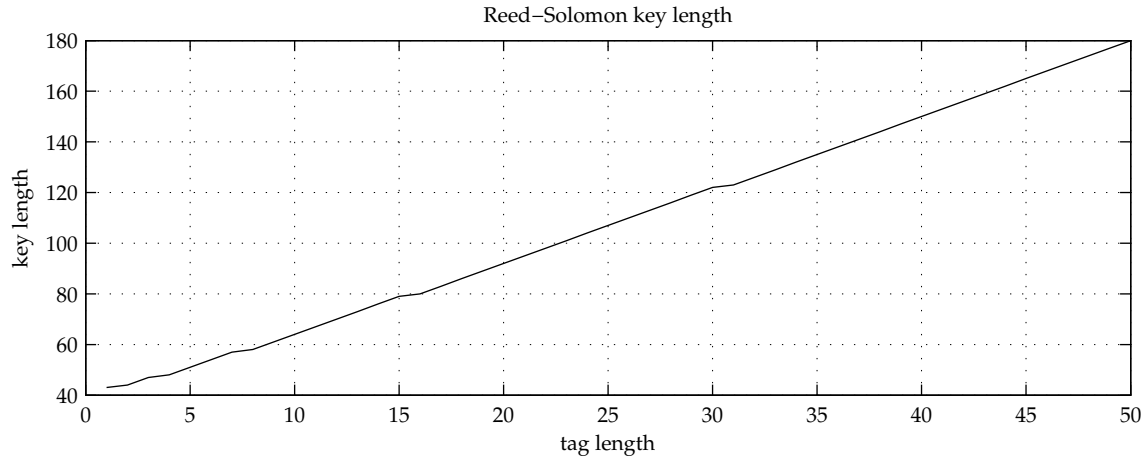
Figures 4.14 and 4.15 show that, for smaller values of l, the difference of performances between Toeplitz and Reed-Solomon classes gradually decrease. The performances of Stinson construction have not been plotted since would superimpose on Reed-Solomon ones.



(a) Key length for SU_2 class based on Toeplitz matrices.



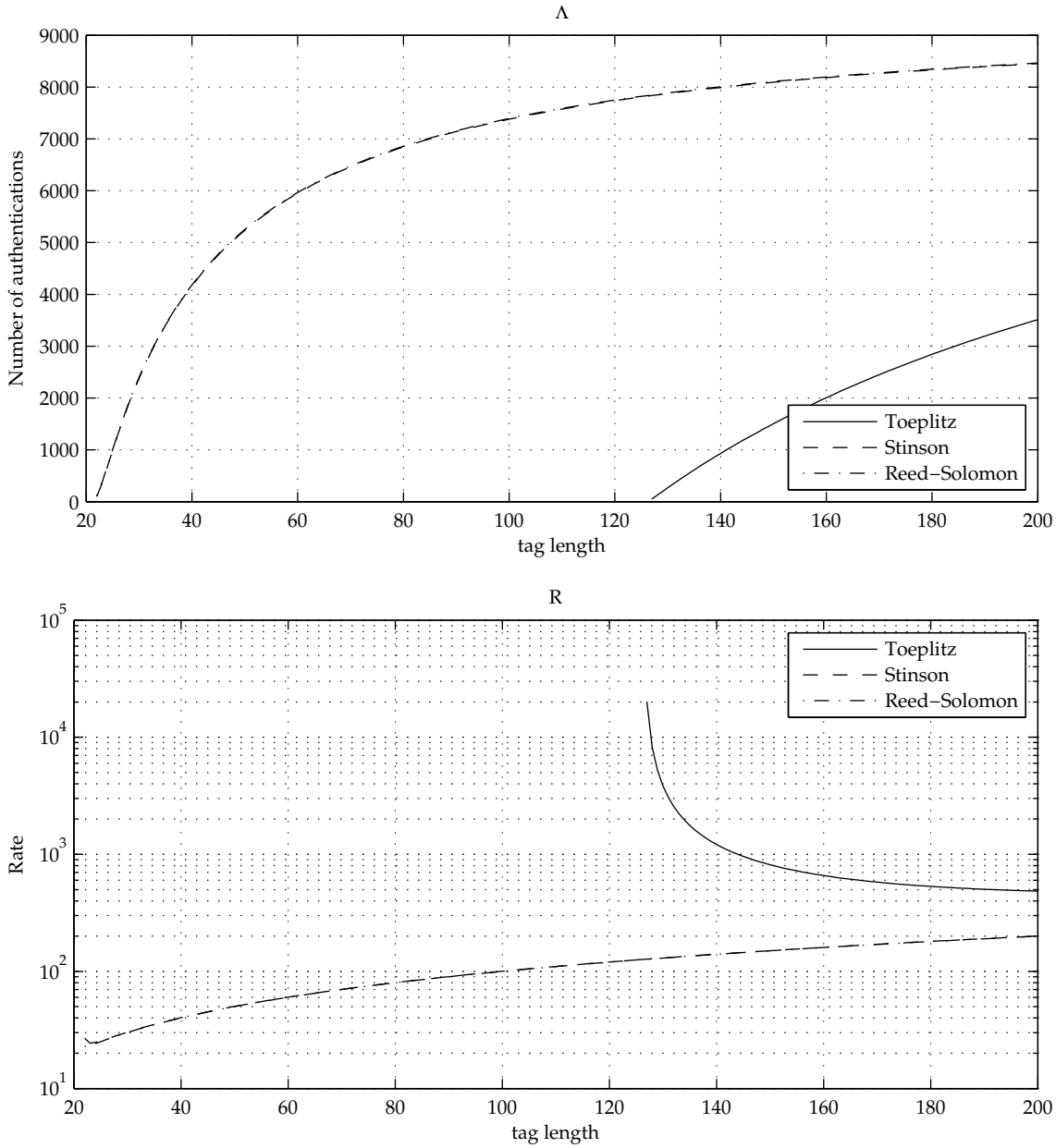
(b) Key length for ε-ASU_2 Stinson class.



(c) Key length for ε-ASU_2 class based on Reed-Solomon codes.

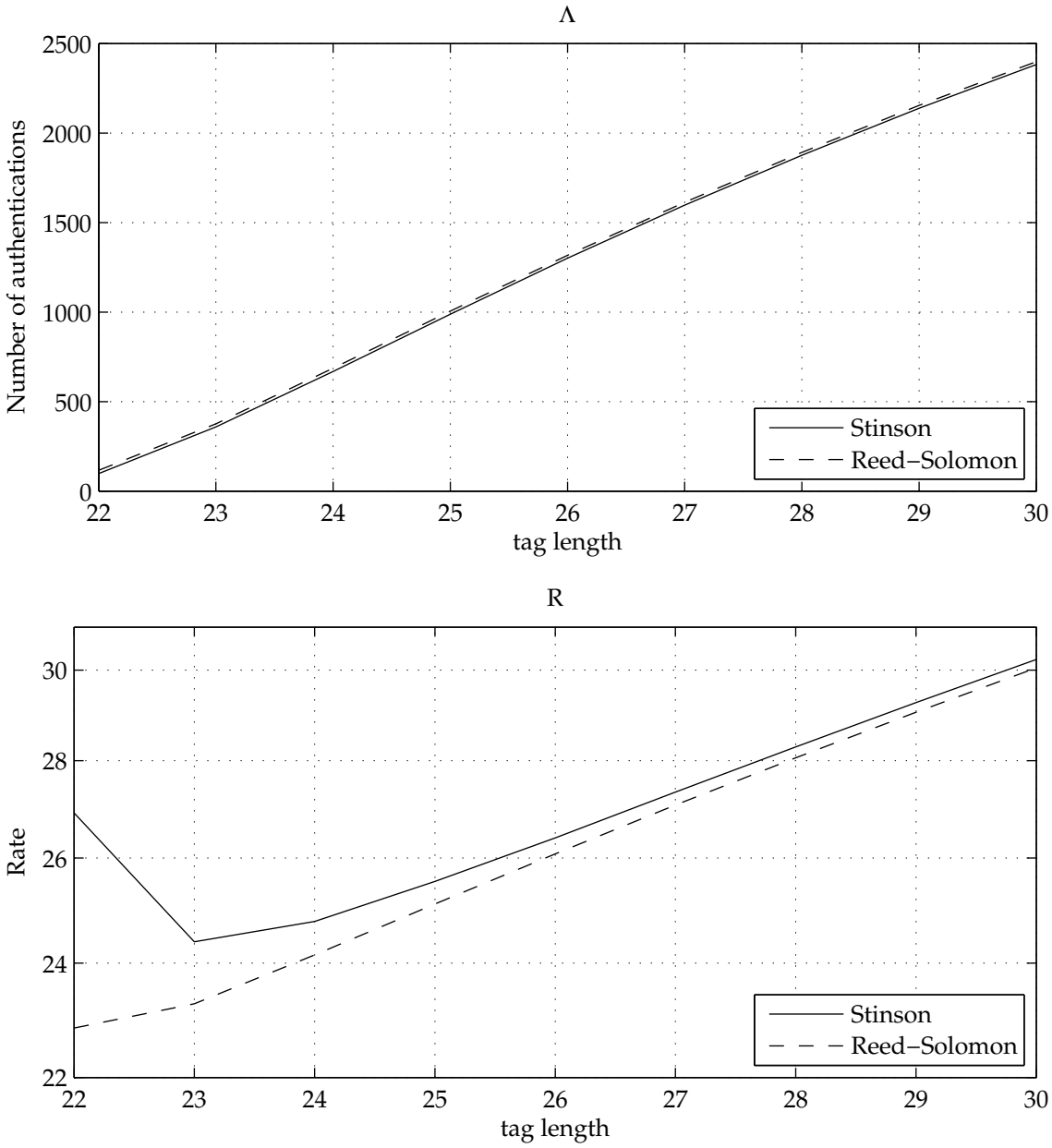
Parameters	
m	10^6

Figure 4.11: Key length for different classes of hash functions.



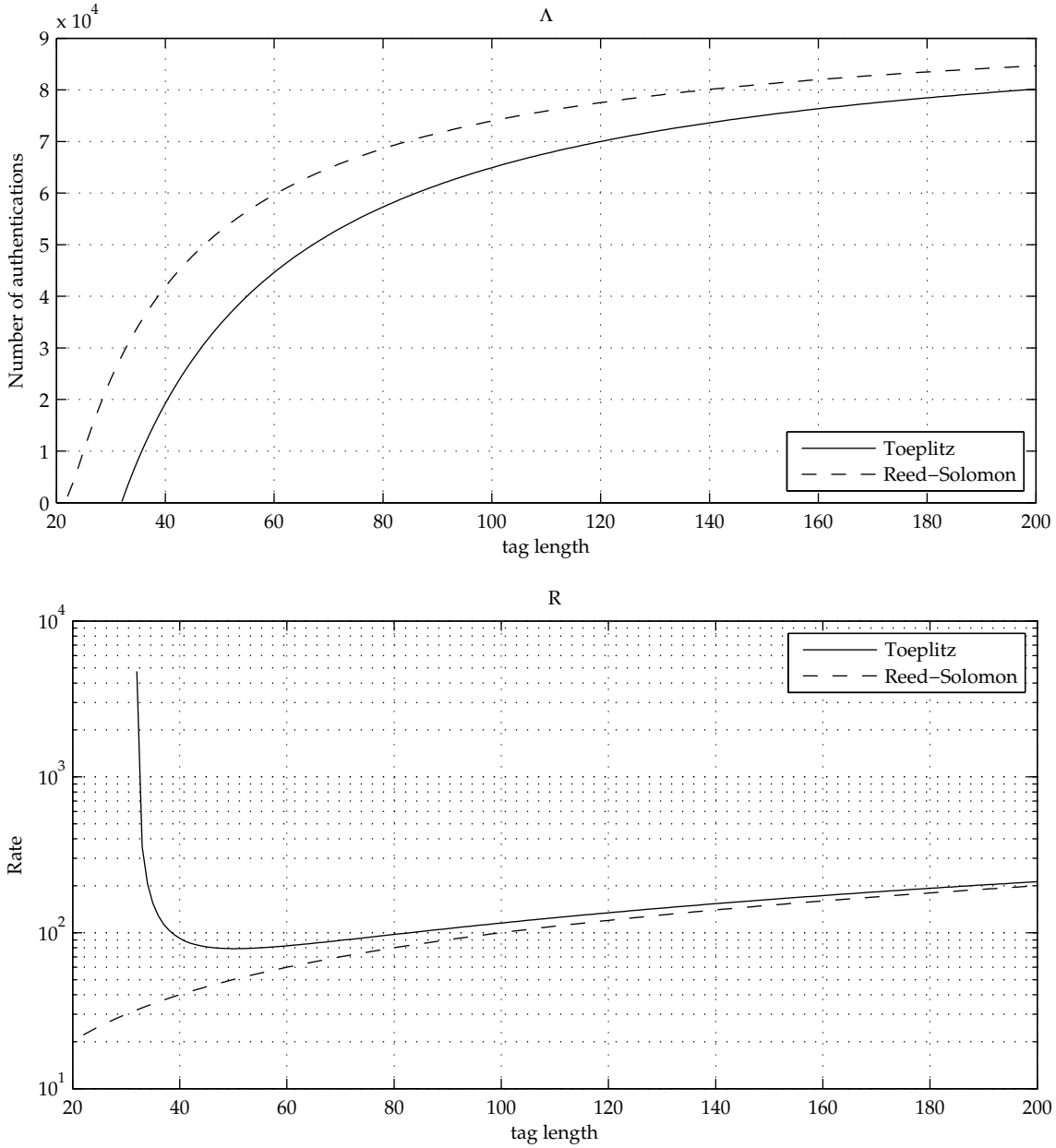
Parameters	
m	10^6
$P_{\max}$	2^{-20}
ℓ	10^{-10}

Figure 4.12: Lifetime and rate for different classes of hash functions.



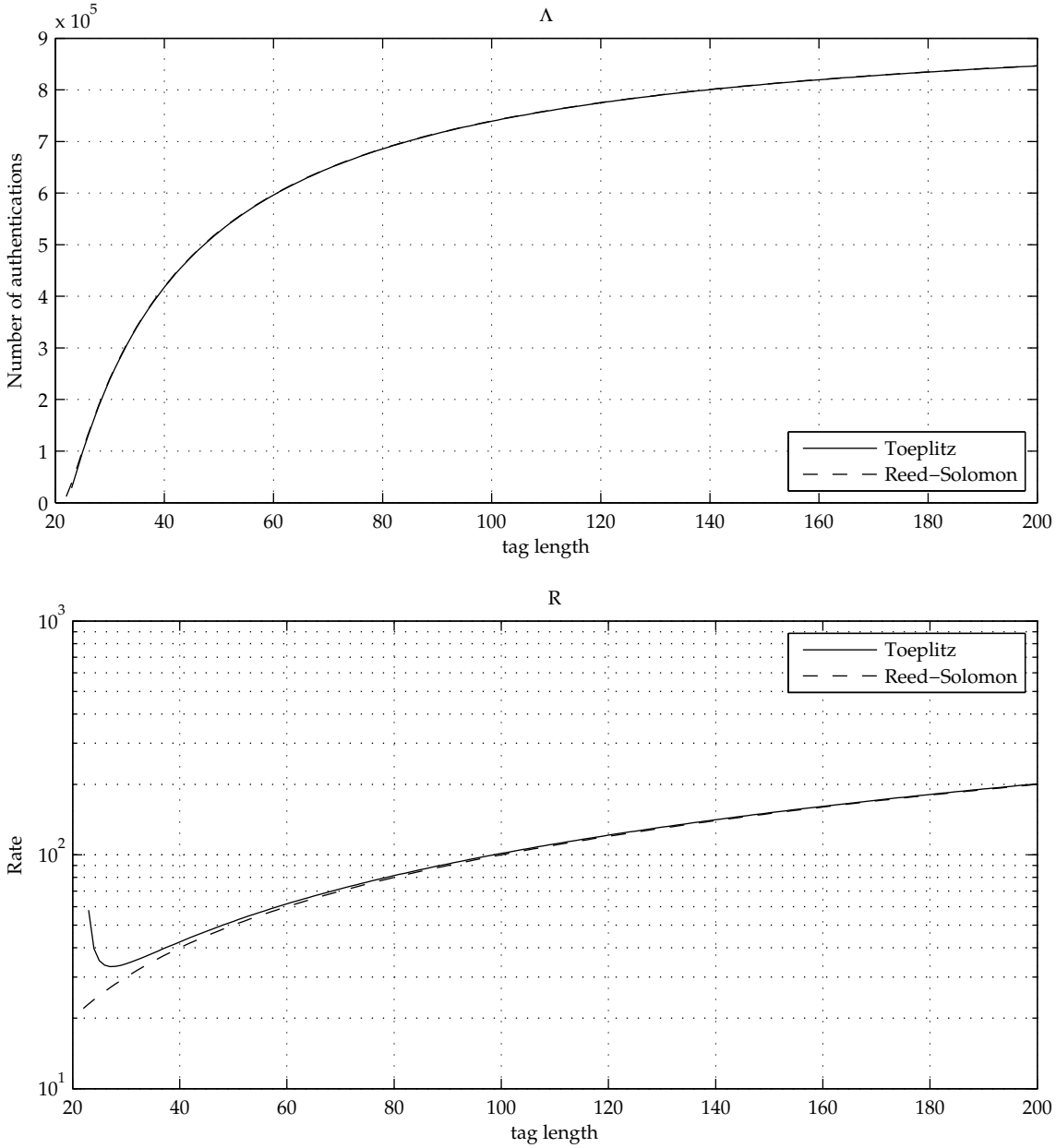
Parameters	
m	10^6
$P_{\max}$	2^{-20}
ℓ	10^{-10}

Figure 4.13: Comparison of lifetime and rate between Stinson and Reed-Solomon classes of hash functions.



Parameters	
m	10^6
$P_{\max}$	2^{-20}
ℓ	10^{-11}

Figure 4.14: Lifetime and rate for different classes of hash functions.



Parameters	
m	10^6
$P_{\max}$	2^{-20}
ℓ	10^{-12}

Figure 4.15: Lifetime and rate for different classes of hash functions.

Net key rate for QuAKE

To evaluate the effects of authentication in a real application, we consider the case of the QuAKE experiment (see [CBC+11]). During a single round of key processing (that is the sum of sifting, reconciliation, and privacy amplification) two authentications are required (one by Alice and one by Bob) and an amount of 20000 bits (N_{rec}) is processed on average.[1] The data transmitted on the public channel is between 500 kb and 1 Mb in each way, so we consider the maximum message length to be authenticated as 10^6 bits. The chosen reconciliation protocol is Winnow, and the average number of revealed bits is $N_{\text{rev}} = 0.29 N_{\text{rec}}$.

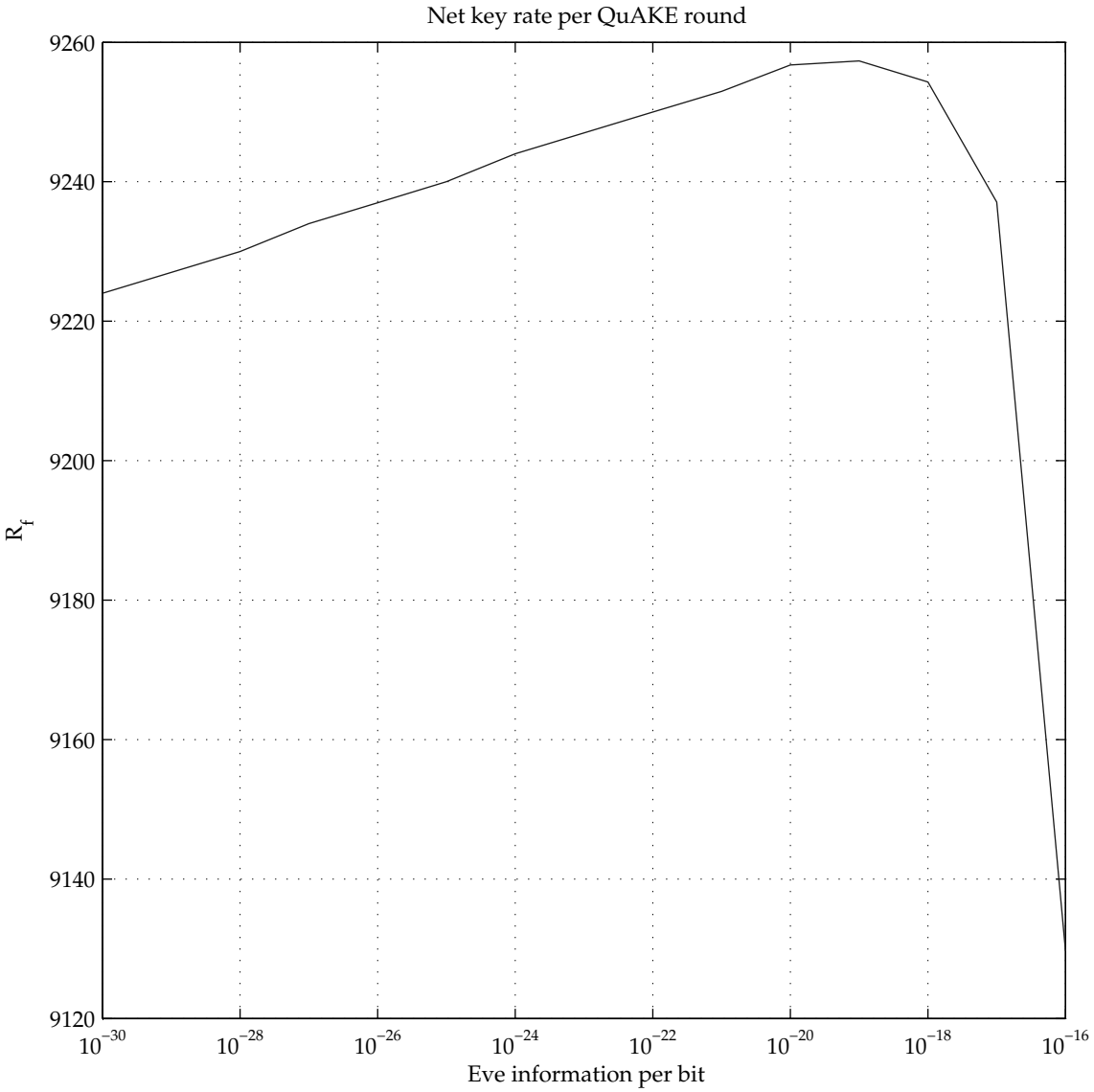
Quake average values	
m	10^6
N_{rec}	20000
N_{rev}	5800

The net key rates for QKD round for probabilities of success of Eve's attack $P_D^{\max} = 2^{-50} \approx 10^{-15}$ and $P_D^{\max} = 2^{-100} \approx 10^{-30}$ have been plotted in figures 4.16 and 4.17 as functions of Eve's average information per bit (ℓ). The authentication scheme used is a Reed-Solomon based ε-ASU$_2$ with one-time pad.

To make a comparison, the N_{sec} required to leave Eve no more than 1 bit of overall knowledge on the final key is 9732.

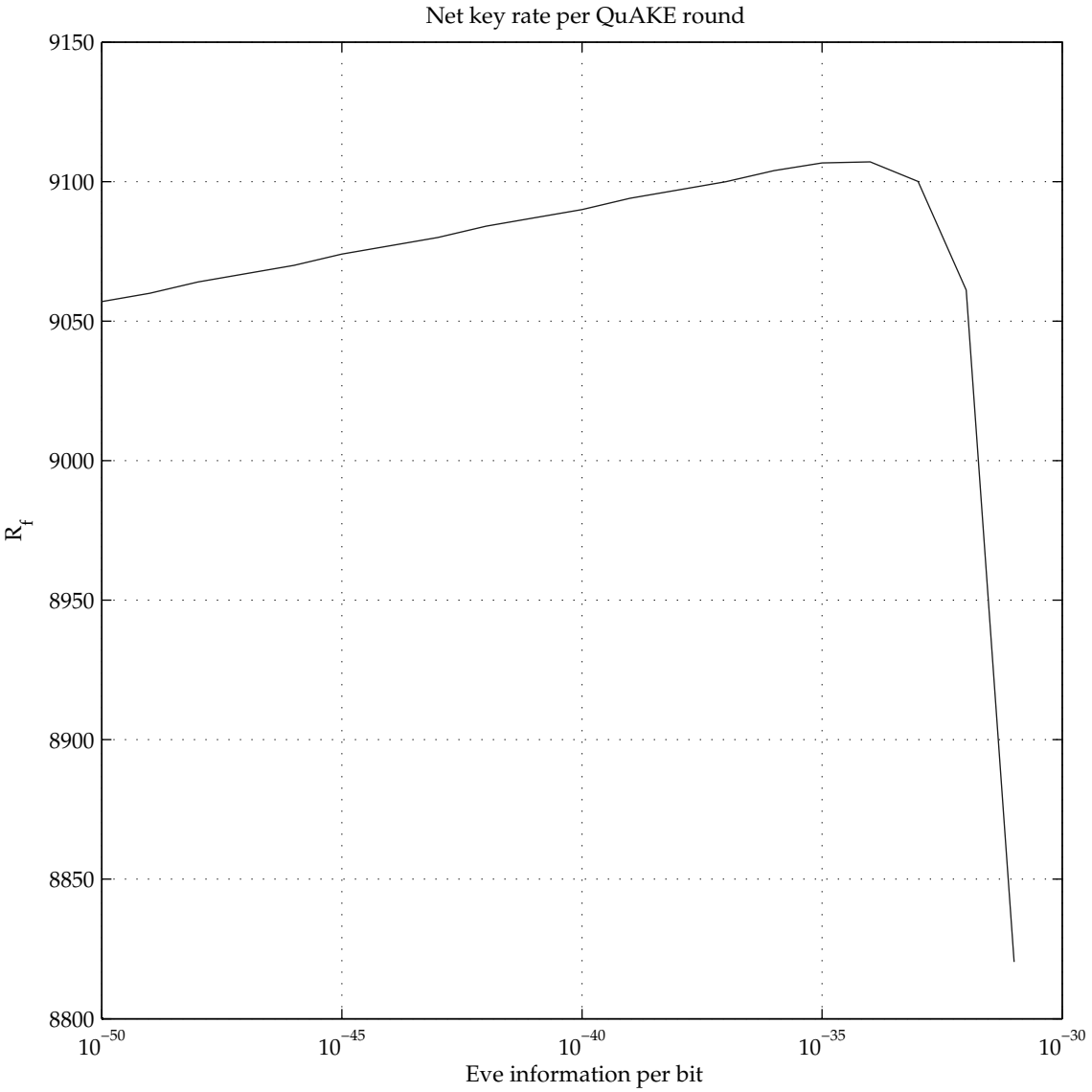
The net key rate as a function of the maximum probability success for Eve's attack ($P_D^{\max}$) is drawn in figure 4.18. Note how the probability of success of the attack decreases exponentially with the rate reduction.

[1] 20000 bits correspond to the average length of sifted key for 50 packets, see [CBC+11] for details.



Optimum values	
ℓ	10^{-19}
R	53.338
n	52
Λ	139

Figure 4.16: Net QKD output rate per round as a function of ℓ for $P_D^{\max} = 2^{-50} \approx 10^{-15}$.



Optimum values	
ℓ	10^{-34}
R	103.465
n	102
Λ	228

Figure 4.17: Net QKD output rate per round as a function of ℓ for $P_D^{\max} = 2^{-100} \approx 10^{-30}$.

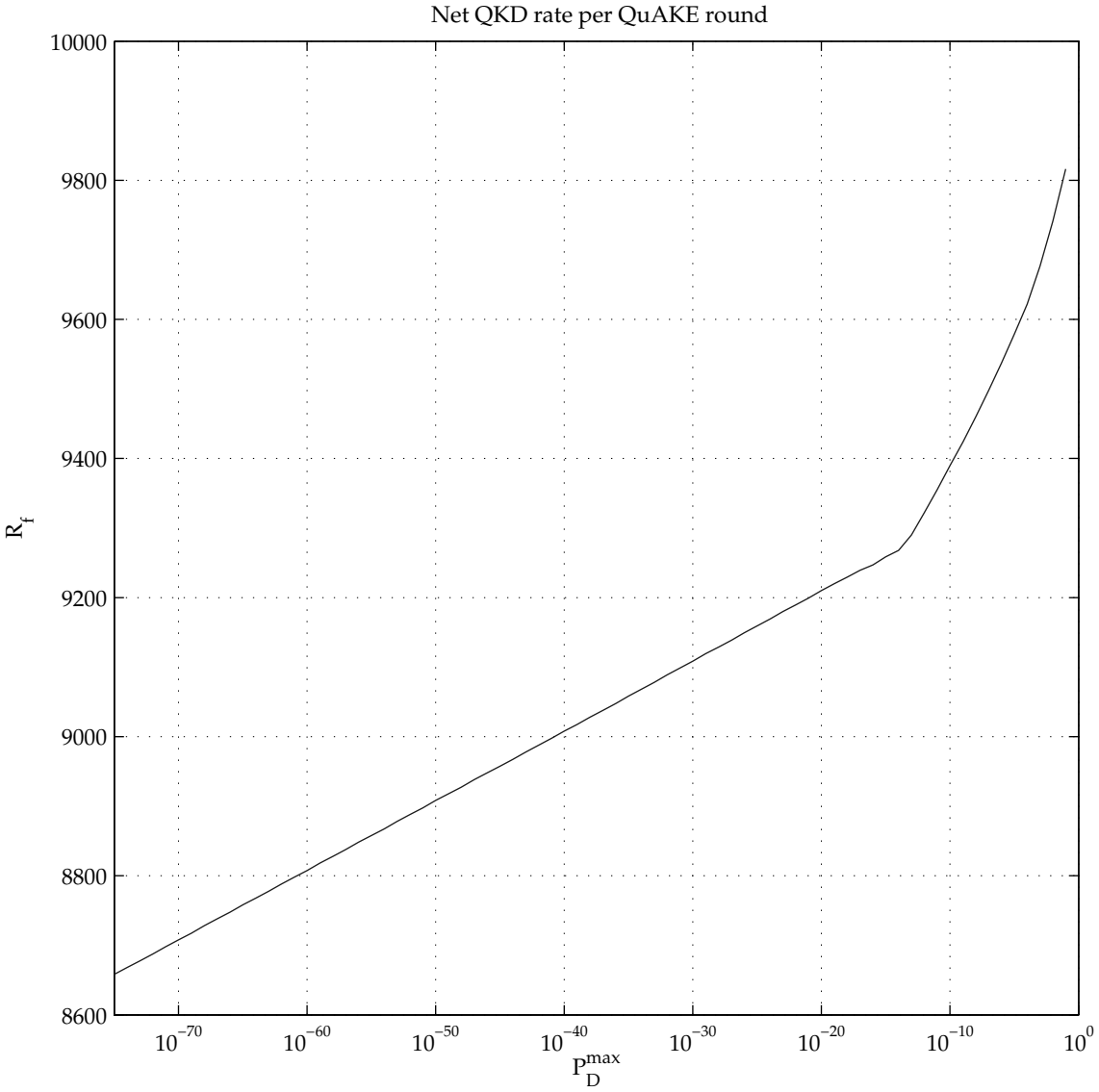


Figure 4.18: Net QKD output rate as a function of P_D.

Chapter

5

The communication protocol

5.1 General description

We studied a network protocol to implement all the communication steps needed to perform the processing of the keys exchanged through the quantum channel.

We chose to build our model of network interaction as an application layer protocol on top of UDP in order to keep it as light as possible and to have a high level of control over each transmitted packet. We decided not to use either TCP or other more complex transport protocols so as not to be exposed to possible attacks that could exploit peculiar weaknesses of communication-oriented protocols, e.g. those oriented to the sliding window mechanism, or a TCP reset attack.

Our main goal is to be able to fully process a key that has been transmitted through the quantum channel and to be reasonably sure that nobody interfered with our messages. We should keep in mind that we are communicating through a public channel, so we cannot prevent a possible eavesdropper to sniff all of our packets. Similarly, it would be naïve to try to forestall all possible DoS attacks, because it only takes cutting the wires we are using for the processing, or, if we were using a free-air quantum channel, to put a physical obstacle between Alice and Bob to completely disrupt our transmission.

The structure of the protocol is highly asymmetrical, since the roles of the two communicating parties are very focused on the specific and different tasks that Alice and Bob are expected to perform during the quantum key distribution; therefore it has been unavoidable to study two different transition models and even different states.

5.2 Our environment

In our setup the raw and the sifted keys are available in the form of indexed files as soon as they have been successfully sent (by Alice) or received (by Bob).

Before the first round, Alice and Bob must share a pool of secret keys that have been previously exchanged through a secret channel (possibly from a previous QKD as well). These keys will be used to generate the hash functions to authenticate the messages and to calculate the one-time-pad of the authentication tags.

When the system starts up, both Alice and Bob are in *Start* state and the hash function is not set up yet.

We assume that Alice sends the last packet in each round. If this is not the case, it is sufficient to authenticate the last message of the round sent by Bob and the first message of the following round sent by Alice.

5.3 Packets

Every message of our application layer protocol can be fragmented among many UDP packets that will be reassembled by the receiver using the information contained in the packet headers.

These packets are characterised by a header that describes the content of the remaining data of the datagram, and an optional payload, whose structure depends on the specific packet type.

Header structure

The structure of the header is described in table 5.1.

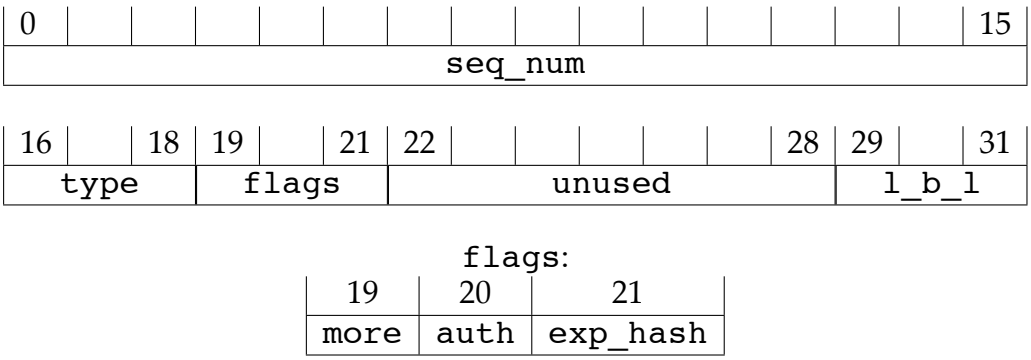


Table 5.1: Packet header

`seq_num` A 16-bit sequence number.

`type` Identifies the type of packet. The currently allowed packet types are listed in table 5.2.

`more` Tells whether the message has been divided into more packets. If it is set to 1 it means that the next packet carries the continuation of the same message. If the

message is contained in a single message, or if the packet holds the final part of the message, it is set to 0.

auth Tells if the current packet contains the authenticating tag, which is appended to the message.

exp_hash Signals that the sender's hash function has expired.

unused Unused bits, to make the header length a multiple of 8 bits. They can contain a random value to increase the header's entropy.

l_b_l (Last block length) Tells the number of bits in the last byte of the packet that contain data of the message, because as a rule we could have any length of the data.

type	
START	0
NEWHASH	1
SIFT	2
PROCESS	3
ABORT	7

Table 5.2: Packet types

Packet types

START

The START packet contains its header only. It may signal that the hash function has expired with the exp_auth flag set to 1.

SIFT

If Bob has any key files to process, he sends an authenticated message with the data needed to start processing the key. If Bob has not got any key files but during the previous round a key has been generated, he sends an authenticated SIFT packet containing only the header and the authenticating tag in the payload. In both cases, the authentication tag is calculated for the concatenation of all the packets Bob sent from the last authenticated SIFT message, or, if the previous round produced no key, for the current SIFT message by itself.

If Bob has no key file and no key was generated during the previous round, he sends an unauthenticated packet containing only the header.

PROCESS

This packet contains all the data needed to perform the key processing: parameters for key reconciliation and privacy amplification. The exact format of the data depends on the specific method chosen to reconciliate the keys. The last PROCESS packet sent by Alice during a key generation round contains a tag that authenticates all the messages sent starting from the reception of the authenticated SIFT packet.

ABORT

The name of this packet is self-explaining: it means that something wrong has happened and it signals that a reset of the connection is needed. Every partial result that has not been confirmed before is wiped out.

5.4 State transition model

The state transition diagram for the protocol is shown in figure 5.1. Each box represents a state, while each link represents a transition and is labelled by a description of the events that are related to it. The events or the received packets that trigger the transition are written before the slash character, while the messages that are consequently sent follow it.

In each state, Alice (or Bob) waits for a message of the expected type and with the expected sequence number. If such a valid message is received, Alice (Bob) moves to the next state according to the transition labelled with the message type. If an invalid message is received or after a given amount of time expires and no valid message has been received, an ABORT message is sent and the *STARTING* state is selected. All ABORT transitions describe an error event and are not explicitly drawn in figure 5.1.

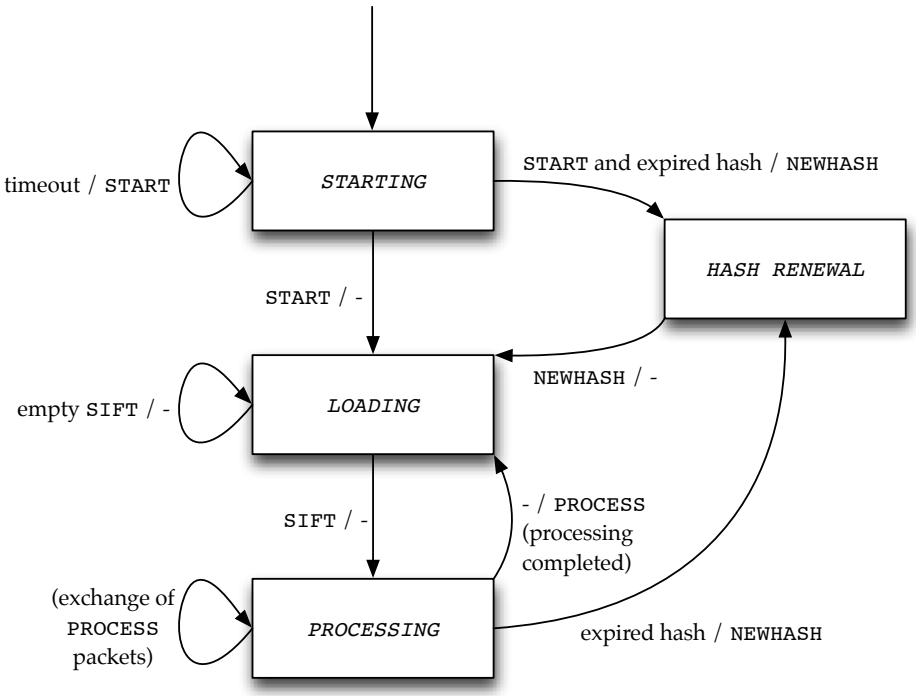
5.5 State descriptions

STARTING

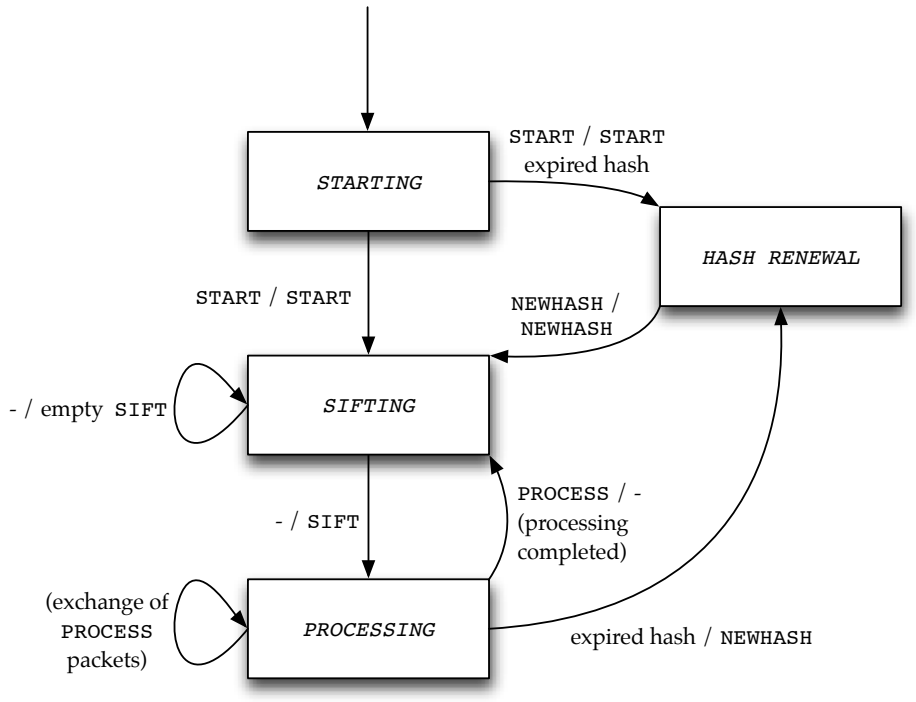
Description This is the state in which both Alice and Bob are when the system starts. This state means that there is no active or valid connection to the other entity.

Interaction

Alice Alice periodically sends a START packet to Bob's IP address. If Alice does not hold a valid hash function (i.e. at startup there is no shared hash function, or the current hash function has expired), the exp_hash is set to 1. When Alice receives Bob's reply, she switches to *LOADING* state or, if an expired hash function has been signalled by one of the two sides, to *HASH RENEWAL*.



(a) Alice transitions



(b) Bob transitions

Figure 5.1: Diagram of state transitions for Alice (a) and Bob (b).

Bob Bob waits for a START packet from Alice. When a START packet has been received, he replies with another START packet. If a valid hash function is not available (see Alice case), the exp_hash is set to 1. Then, Bob switches to *SIFTING* state or, if an expired hash function has been signalled by one of the two sides, to *HASH RENEWAL*.

HASH RENEWAL

Description This state is reached when one of the two parties has no valid hash function anymore and has successfully signalled this event in a previous START, PROCESS, or SIFT packet. In this state Alice chooses the keys that will be used to generate the hash function and transmits their IDs to Bob.

Interaction

Alice
1. Alice gets the needed seed bits from the key database and generates a new hash function. Then she sends a NEWHASH packet containing the addresses of the used keys; the whole packet is authenticated by the new hash function.
2. Alice waits for Bob's reply. If the reply is a NEWHASH packet with a valid authentication, then the new hash function has been correctly validated, Alice switches to *LOADING* state, otherwise an ABORT packet is sent and proceeds to *STARTING* state.

Bob Bob waits for a NEWHASH packet from Alice. When a valid packet has been received, he fetches the keys identified by the enclosed IDs from the key database and generates the new hash function. Bob now checks that the packet has been correctly authenticated by the new hash function. If this is the case, he sends an empty validated NEWHASH packet to Alice and then switches to *SIFTING* state, otherwise an ABORT packet is sent and proceeds to *STARTING* state.

SIFTING

Description This state is only used by Bob. In this step, Bob selects a sifted key from the pool of available sifted key files and starts its processing.

Interaction

Bob Bob selects one of the sifted keys not yet processed and sends an authenticated SIFT packet containing the file index, the sifting information and, possibly, information for the reconciliation and privacy amplification phases. Then Bob switches to *PROCESSING* state.

LOADING

Description This state is only used by Alice. In this step, Alice loads the raw key corresponding to the sifted key selected by Bob.

Interaction

Alice Alice receives a `SIFT` packet from Bob:

Authentication and sifting (Bob has a sifted key to process.) The `SIFT` packet received is authenticated and its payload is not empty. Alice checks the packet authentication. The final key generated during the previous round, if there is any, is stored into the key database. The raw keys identified by the indexes contained in the packets are loaded. Alice performs the sifting and, possibly, some other actions of the reconciliation and privacy amplification phases. Then Alice sends the first `PROCESS` packet to Alice; if this is the only `PROCESS` packet that will be sent during the current round, we authenticate the concatenation of the packet and the generated final key[1], then Alice stays in *LOADING* state. Otherwise, an `ABORT` packet is sent and Alice proceeds to *STARTING* state.

Authentication only (Bob has no sifted key to process but there still is a final key to acknowledge.) The `SIFT` packet received is authenticated, but its payload is empty. Alice checks the packet authentication. The final key generated during the previous round is stored into the key database. No key index has been sent, so Bob has not got any sifted key left. Alice stays in *LOADING* state and waits for the next `SIFT` packet. Otherwise, an `ABORT` packet is sent and Alice proceeds to *STARTING* state.

Empty `SIFT` packet (Bob has no sifted key to process and there is no final key to acknowledge.) The `SIFT` packet received is not authenticated and its payload is empty. No key index has been sent, so Bob has no sifted key left. Alice stays in *LOADING* state and waits for the next `SIFT` packet.

If no valid packet is received within a timeout, an `ABORT` packet is sent and Alice proceeds to *STARTING* state.

PROCESSING

Description In this state Alice and Bob exchange messages to accomplish the correct generation of the final key.

[1] At the last `PROCESS` packet the current round terminates, so Alice holds all the information needed to generate the final key.

Interaction

Alice sends and receives `PROCESS` packets.

Last packet The last `PROCESS` packet sent by Alice will be the last packet of the current round. It contains the last information needed to generate the final key and the authentication of the concatenation of *all the packets sent by Alice during the `PROCESSING` state of the current round and the final key*. Then Alice switches to *`LOADING`* state. On errors, an `ABORT` packet is sent and Alice proceeds to *`STARTING`* state.

Bob sends and receives `PROCESS` packets.

Last packet The last `PROCESS` packet sent by Alice contains all the necessary information to generate the final key and must be authenticated as described above. Bob generates the final key and verifies the authentication of *all the packets received by Bob during the `PROCESSING` state of the current round and the final key*. If the authentication is valid, Bob stores the final key into the key database and switches to *`SIFTING`* state. On errors, an `ABORT` packet is sent and Bob proceeds to *`STARTING`* state.

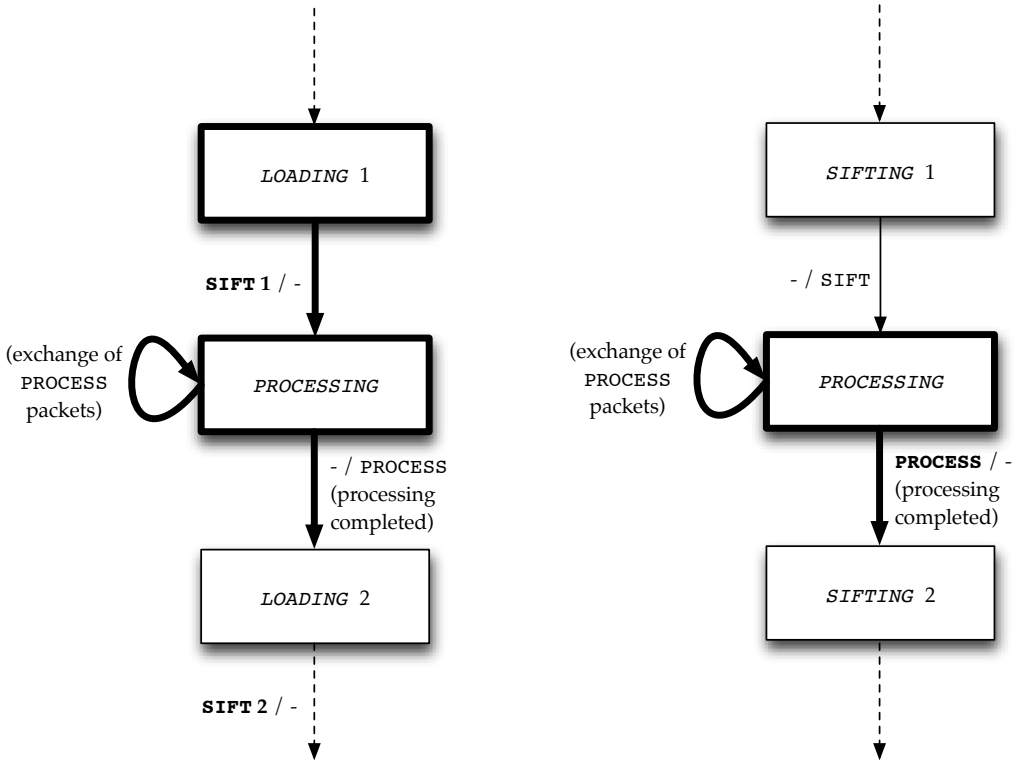
5.6 Security of the protocol

This protocol guarantees that every key that has been generated and confirmed by an authenticated reply, as explained above, is genuine and that it has not been tampered with by a third party. In fact we can observe that there is only one way that leads from the *`STARTING`* state to the generation and storing of a final key into the key database: namely, both Alice and Bob have to go through a series of states that lead to an authenticated reply. This reply contains a tag that authenticates all the outgoing messages that carried information needed to generate the key. Therefore, we can state that all the paths that can be chosen, according to the state transition diagrams in figure 5.1, must comply with one of the cases in figure 5.2.

The hash function used to generate and check the correctness of the authentication tags is guaranteed to be valid, because, if it had expired, only the *`HASH RENEWAL`* state would have been reachable.

Possible errors

Since it could happen that a mismatch between Alice's and Bob's reconciled keys went undetected, there is some probability for the two final keys to be different after privacy amplification. This is why Alice is needed to authenticate the final key along with the sent messages: we are using the authentication function for the further purpose of ensuring that the reconciliation phase has been successful.



(a) The Message SIFT 1 authenticates itself and the PROCESS messages received during the previous round, if any. The Message SIFT 2 authenticates all PROCESS messages received during the current round.

(b) The last PROCESS message authenticates all the messages received during current round, besides the final key.

Figure 5.2: Possible state transitions that lead to the generation of a final key for Alice (a) and Bob (b). Transitions marked with thick lines are the authenticated ones. Packets marked in bold are the ones containing the related authenticating tags.

In our protocol model, if one or more mismatches in the reconciliation phase have not been corrected, we cannot distinguish this case from the case in which an attacker has been trying to forge one of the messages.

As an alternative, one could use a protocol in which the messages and the final key are authenticated separately, at the cost of adding an authentication step and, therefore, shortening the number of rounds a single hash function could have been used. We chose instead to use only one authentication and handle an invalid authentication as a tampering attempt performed by an attacker.

Possible attacks

It would be very difficult for a tampering attack to go undetected due to the bounds on the probability of success imposed by the authentication step (see paragraph 2.1). In the overwhelming majority of the cases, such an attack could only lead to the failure of the whole process due to the unsuccessful validation of the forged tag.

The only attack that could be carried out with significant results against this setup would be to continuously intercept some packet during each round, thus preventing each partial key from being processed completely. In fart, any lost, or corrupted, packet would bring to the round being aborted. As every authentication shortens the hash function lifetime, and uses one-time pad to encrypt the hash value, sooner or later both Alice and Bob will reach the point where no shared keys are available anymore, and quantum key distribution is no longer possible.

The same effect could be obtained by sending an uninterrupted sequence of `ABORT` and `START` packets with the `exp_hash` flag set to 1: this would cause the legitimate party to spoil every valid key in the attempt of generating new hash functions that would be continuously replaced by new ones.

Chapter

6

Conclusions

In this work we have considered the problem of providing unconditionally secure authentication for the public transmission in Quantum Key Distribution (QKD). In particular we have studied specific classes of hash functions, called ε-Almost Strongly Universal (ε-ASU$_2$), that allow to build authentication codes whose robustness is not based on the attack computational complexity. These hash functions generate a tag t for each message x to be authenticated. Even if a valid couple (x, t) is intercepted by an attacker on the public channel, a ε-ASU$_2$ class ensures that the success probability of both impersonation and substitution attacks are upper bounded by a function of the tag length n and the total number F of hash functions.

Since the hash functions are used in the context of QKD, it is important to reduce the rate of key bits consumed by the authentication code, because this rate will be subtracted from the actual key generation rate. Wegman and Carter proposed to use a single hash function more than once by encrypting the hash value with a one-time pad, but Cederlöf and Larsson pointed out how it is possible to accomplish a successful attack if the opponent has partial information on the keys used for the one-time pad, as is the case with QKD.

In this work we have introduced a general upper bound to the probability P_D of success for the attacker in the case of a Cederlöf-Larsson like attack as a function of both the number of authentications performed with a same hash function, and the average information that the opponent has on each key bit. This is, at the best of our knowledge, the first upper bound on P_D for this kind of attack, and it allows to calculate the maximum number of times a single hash function can be used (see equation (4.30)), and the minimum key rate needed for unconditionally secure authentication (equation (4.32)). Optimisation of these parameters is of utmost importance to attain the highest possible output rate in QKD. In paragraph 4.3 the optimisation of the final key rate for the QuAKE experiment (a B92 QKD system developed by the QuantumFuture project of the University of Padova) is described, and it is shown that the probability of success of

attack decreases exponentially with the rate loss.

Many constructions for ε-ASU$_2$ hash function classes have been reviewed and compared. The best authentication method seen in this work is Reed-Solomon based ε-ASU$_2$ hash function class. Besides being a small class (its size is $\sim 2^{3n+2(\log_2 m - \log_2 n)}$, with m being the maximum message length and n being the tag length), it is also flexible, since there exists a class for each values of maximum message length m and tag length n. Even if Reed-Solomon is not always the smallest sized class available[1], its cardinality has a constant ratio <2 with respect to the lower bound (2.12) (see figure 3.1).

Finally, the communication protocol specifically designed for QuAKE has been described, with details on the packet structure, on the interaction model of the two parties and on the different phases of the key processing. The authentication related features have been particularly stressed.

6.1 Future work

Since during QKD all the transmitted data is continuously authenticated, it is important to correctly estimate the amount of time that it takes to initialise a new hash function, as, during this time, no data can be authenticated. It is important, hence, to choose the combination of tag length, privacy amplification parameters and, as a consequence, maximum lifetime of a single hash function, which allows to attain the maximum output key rate.

It could be appropriate to generate final keys with adjustable ℓ (that is Eve's average information per bit on final keys), because the requirements on ℓ for authentication keys could be far stricter than those of different applications. In this case it would be sufficient to decide beforehand the purpose of the keys produced during each round and then set the privacy amplification parameter consequently.

[1] See, for example, authentication codes based on geometrical codes like those described in [Bie97a], which have the drawback of existing only for some values of m and n.

Bibliography

[AL11] Aysajan Abidin and Jan-Åke Larsson. Security of Authentication with a Fixed Key in Quantum Key Distribution. September 2011.

[AS96] Mustafa Atici and Douglas R. Stinson. Universal hashing and multiple authentication. In *Advances in Cryptology, CRYPTO '96*, pages 16–30. Springer, 1996.

[BB84] C. H. Bennett and G. Brassard. *Quantum cryptography: Public key distribution and coin tossing*, volume 175, pages 175–179. Bangalore, India, 1984.

[BBCM95] C.H. Bennett, G. Brassard, C. Crepeau, and U.M. Maurer. Generalized privacy amplification. *Information Theory, IEEE Transactions on*, 41(6):1915 –1923, nov 1995.

[Ben92] Charles H. Bennett. Quantum cryptography using any two nonorthogonal states. *Phys. Rev. Lett.*, 68:3121–3124, May 1992.

[Bie96] Jürgen Bierbrauer. Construction of orthogonal arrays. *Journal of Statistical Planning and Inference*, 56(1):39–47, 1996.

[Bie97a] Jürgen Bierbrauer. Universal hashing and geometric codes. *Designs, Codes and Cryptography*, 221:207–221, 1997.

[Bie97b] Jürgen Bierbrauer. Universal hashing and geometric codes. *Designs, Codes and Cryptography*, 11:207–221, 1997. 10.1023/A:1008226810363.

[BJKS93] Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, and Ben J. M. Smeets. On families of hash functions via geometric codes and concatenation.pdf. In *Advances in Cryptology, CRYPTO '93*, pages 331–342, 1993.

[BLT+03] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. H. Nickel, C. H. Donahue, and Charles G. Peterson. Fast, efficient error reconciliation for quantum cryptography. *Physical Review A*, 67(5):052303, May 2003.

[Bod07] Marco Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In Claude Carlet and Berk Sunar, editors, *WAIFI 2007 proceedings*, volume 4547 of *LNCS*, pages 116–133. Springer, June 2007. `http://bodrato.it/papers/#WAIFI2007`.

[BS93] Gilles Brassard and Louis Salvail. Secret-Key Reconciliation by Public Discussion. *International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology, EUROCRYPT*, pages 410–423, 1993.

[CBC+11] Matteo Canale, Davide Bacco, Simon Calimani, Francesco Renna, Nicola Laurenti, Giuseppe Vallone, and Paolo Villoresi. A prototype of a free-space QKD scheme based on the B92 protocol. In *International Symposium on Applied Sciences in Biomedical and Communication Technologies, ISABEL*, Barcelona, October 2011.

[CL08] Jörgen Cederlöf and Jan-Åke Larsson. Security Aspects of the Authentication Used in Quantum Cryptography. *IEEE Transactions on Information Theory*, 54(4):1735–1741, April 2008.

[CW77] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (Extended Abstract). *ACM Symposium on Theory of Computing, STOC*, pages 106–112, 1977.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644 – 654, nov 1976.

[ELAB09] David Elkouss, Anthony Leverrier, Romain Alléaume, and Joseph J. Boutros. Efficient reconciliation protocol for discrete-variable quantum key distribution. In *IEEE International Symposium on Information Theory, ISIT*, pages 1879–1883. IEEE, 2009.

[GN93] P. Gemmell and M. Naor. Codes for interactive authentication. In *Advances in Cryptology, CRYPTO '93*, pages 355–367. Springer, 1993.

[GRTZ02] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Rev. Mod. Phys*, 74:145–195, 2002.

[JKS93] T. Johansson, G. Kabatianskii, and Ben J. M. Smeets. On the relation between A-codes and codes correcting independent errors. In *Advances in Cryptology, EUROCRYPT '93*, pages 1–11, 1993.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[Kra94] Hugo Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology, CRYPTO '94*, volume 10598, pages 129–139, 1994.

[Kra95] Hugo Krawczyk. New Hash Functions for Message Authentication. *International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology, EUROCRYPT '95*, 921:301–310, 1995.

[KSJ96] G.A. Kabatianskii, Ben Smeets, and Thomas Johansson. On the Cardinality of Systematic Authentication Codes Via Error-Correcting Codes. *Information Theory, IEEE Transactions on*, 42(2):566–578, 1996.

[Mau00] U.M. Maurer. Authentication theory and hypothesis testing. *Information Theory, IEEE Transactions on*, 46(4):1350 –1356, jul 2000.

[MDMD10] Fabio Mesiti, Maria Delgado, Marina Mondin, and Fred Daneshgaran. Sparse-graph codes for information reconciliation in QKD applications. In *International Symposium on Applied Sciences in Biomedical and Communication Technologies, ISABEL*, pages 1–5. IEEE, November 2010.

[MNT90] Y. Mansour, Noam Nisan, and P. Tiwari. The computational complexity of universal hashing. In *Annual ACM symposium on Theory of computing, STOC*, pages 235–243, New York, New York, USA, 1990. ACM Press.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing - STOC '90*, pages 213–223, New York, New York, USA, 1990. ACM Press.

[NR10] L.H. Nguyen and A.W. Roscoe. New combinatorial bounds for universal hash functions. *University Computing*, pages 1–14, 2010.

[PB45] R.L. Plackett and J.P. Burman. The design of optimum multi-factorial experiments. *Biometrika*, 33:305–325, 1945.

[Rog95] Phillip Rogaway. Bucket hashing and its application to fast message authentication. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 29–42, London, UK, 1995. Springer-Verlag.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Sar80] Dilip V. Sarwate. A note on universal classes of hash functions. *Information Processing Letters*, 10(1):41–45, 1980.

[Sha49] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, October 1949.

[Sti88] D. R. Stinson. Some constructions and bounds for authentication codes. *J. Cryptol.*, 1:37–51, March 1988.

[Sti94a] D. R. Stinson. Combinatorial techniques for universal hashing. *J. Comput. Syst. Sci.*, 48(2):337–346, 1994.

[Sti94b] D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 4(3):369–380, July 1994.

[Sti96] D. R. Stinson. On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes. In *Congressum Numerantium 114*, pages 7–27, 1996.

[TDG04] Zhihui Tang, Ramani Duraiswami, and Nail Gumerov. Fast algorithms to compute matrix-vector products for pascal matrices. Technical Report CS-TR-4563 UMIACS; UMIACS-TR-2004-08, University of Maryland Computer Science Department, March 2004.

[WC81] 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, June 1981.