



מכון ויצמן למדע  
WEIZMANN INSTITUTE OF SCIENCE

*Thesis for the degree  
Master of Science*

חבור לשם קבלת התואר  
מוסמך למדעים

*By  
Dror Eiger*

מאת  
דרור אייגר

*חלוקת סוד פרואקטיבית עם  
מחיקות חלקיות  
Proactive Secret Sharing  
with Partial Erasures*

*Advisor  
Prof. Shafi Goldwasser*

מנחה  
פרופ' שפי גולדווסר

*November 2007*

חשון תשס"ח

Submitted to the Scientific Council of the  
Weizmann Institute of Science  
Rehovot, Israel

מוגש למועצה המדעית של  
מכון ויצמן למדע  
רחובות, ישראל



## Abstract

Every cryptographic definition or construct requires that some part of the honest user's private memory is kept completely hidden from the adversary. Thus, no security is guaranteed if some malicious program is residing on a computer that is doing the cryptographic operations.

Instead of storing a secret (e.g. a secret key of an encryption scheme) centrally, *secret sharing schemes* distribute *shares* of the secret to several parties. In particular, in  $c$  out of  $p$  *threshold* schemes (Shamir, CACM '79; Blakely, Proc. of the AFIPS '79), no information on the secret will be gained by an adversary that corrupts up to  $c - 1$  parties over the entire life-time of the secret, whereas any  $c$  or more parties can jointly recover the secret.

*Proactive secret sharing* (Ostrovsky and Yung, PODC '91) take one more step from this and splits time into *time periods*, and consider a *mobile adversary*, which is one that can move around in different time periods, and at each time period can corrupt up to  $c - 1$  parties. Their proof of security requires that at every time period, each party can *perfectly erase* its memory, so that no information whatsoever will be left.

However, perfect erasures are hard to achieve in practice. Even if it were reasonable to assume perfect erasures, it is still an interesting question whether perfect erasures are necessary to achieve proactive secret sharing. On the other hand, without any form of erasures at all, proactive secret sharing is impossible.

Here we introduce the idea of using *partial erasures* as opposed to the two extremes of having perfect erasures on the one hand and no erasures on the other. We model partial erasures by an arbitrary length shrinking function, and give proactive secret sharing schemes with partial erasures for both the honest but curious and the malicious mobile adversary case. Thus, we show that partial erasures suffice for proactive secret sharing.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Secret Sharing	1
1.2	Proactive Secret Sharing	1
1.3	Our Result	2
1.4	Technique	4
1.5	Future Work	5
1.6	The Organization of the Rest of the Thesis	6
<b>2</b>	<b>Preliminaries and Definitions</b>	<b>6</b>
2.1	The Model of Mobile Adversary	6
2.2	Secret Sharing	7
2.3	The Partial Erasures Model	8
2.3.1	Storage Fraction vs. Time Periods	9
2.4	The Memory Model	9
2.5	Information Theory	10
2.5.1	Statistical Distance	10
2.5.2	Entropy	10
2.5.3	Other Definitions	12
2.6	Extractors	13
2.7	Universal Hash Functions and the Leftover Hash Lemma	14
2.8	Privacy and Robustness Definition in Partial Erasures Model	15
<b>3</b>	<b>Related Work</b>	<b>16</b>
3.1	The Bounded Storage Model (BSM)	16
3.1.1	Intrusion-Resilient Secret Sharing	17
3.2	Exposure-Resilient Functions (ERF)	17
3.3	Relation Between our Solution and Extractors	18
<b>4</b>	<b>Our Solution to the PSS with Partial Erasures</b>	<b>19</b>
4.1	The Scheme Modification Method	20
4.1.1	PESS (II)	21
4.2	Proactive $(p, p)$ Secret Sharing with Partial Erasures	22
4.3	Proactive $(c, p)$ Threshold Secret Sharing with Partial Erasures	23
4.4	Proofs	25

<b>5 Algorithms</b>	<b>34</b>
5.1 Notation . . . . .	34
5.2 Algorithms Used by the Modification Method . . . . .	35
5.3 Algorithms for the Proactive $(p, p)$ Secret Sharing Scheme with Partial Erasures	38
5.4 Algorithms for the Proactive $(c, p)$ Threshold Secret Sharing Scheme with Partial Erasures . . . . .	39
<b>6 Acknowledgments</b>	<b>41</b>

# 1 Introduction

## 1.1 Secret Sharing

A *secret sharing scheme*, introduced independently by Shamir [Sha79] and Blakley [Bla79], is a protocol that allows a dealer that holds a secret to share it with a number of parties. It consists of two phases: the *dealing phase* where the dealer splits the secret into several parts and delivers each part secretly to each party, and the *reconstruction phase* where some number of parties come together and reconstruct the secret.

Correctness requires that after dealing the secret is well defined and all subsets of the parties in what is called the *access structure* of the scheme can reconstruct the secret. Privacy requires that all subsets of the parties that are not in the access structure cannot get any information on the secret, even if they are coordinated by one adversary.

## 1.2 Proactive Secret Sharing

It has been suggested by Ostrovsky and Yung [OY91] that over the life-time of a system, it is unreasonable to assume that the adversary corrupts at most  $c$  parties. Instead, they consider splitting time into fixed size intervals, or *time periods*, and consider a *mobile adversary*, which is one that can move around in different time periods, and at each time period can corrupt up to a certain predetermined number of parties. Ostrovsky and Yung studied the question of general secure computation in the presence of a mobile adversary and presented, among other things, an information theoretic secret sharing against a mobile adversary. Their proof of security requires that at every time period, each party can *perfectly erase* its memory, so that no information whatsoever will be left.

Herzberg et al. [HJKY95] restricted their attention to threshold secret sharing schemes secure against the mobile adversary, which they name *proactive secret sharing*, and gave a detailed and efficient proactive secret sharing scheme. They also specialized this notion to one that is *robust*, which guarantees the correct reconstructibility of the secret at any time period. Following [OY91], they add a *refreshing phase* into a secret sharing scheme (in between each adjacent pair of time periods), during which each party first replaces its old share with a new one, and second, perfectly erases the old share and all the auxiliary information used in computing the new share (so as to destroy any dependencies between the shares in the different time periods). The refreshing is done so that at least  $c$  shares from the *same time period* are required to share the secret.

### 1.3 Our Result

We focus on the question of whether perfect erasures are necessary for proactive secret sharing. It is very easy to see that without *some form* of erasures, proactive secret sharing is impossible as follows: Suppose that an adversary eventually breaks into all players across many time periods, then (assuming no data was ever erased) the adversary can collect all the information held by all the players at a single time period in the past, say time period one, and use it to reconstruct the original secret<sup>1</sup>. Indeed, all the previous work in the mobile adversary model strongly relies on perfect erasure of past information (e.g. [CH94, HJKY95, OY91, SW99]).

In practice, perfect erasures are hard to achieve and are problematic as a security assumption, as pointed out by Jarecki and Lysyanskaya [JL00] in their study of adaptive adversaries versus static adversaries in the context of threshold secret sharing. Computers are designed and built to store information, making sure the possibly vital information will live through hardware failure and human error. The notion of erasing data in a non recoverable way contradicts this ground design rule. For example, when data is being moved, the general rule is to copy the whole data and only then to erase the old copy, so in case of power failure, a bad sector detection on the newly written data location, or a lack of storage, no data will be lost. Commercially used servers also take extra resiliency measures such as RAID systems (where the contents of the hard drive is written simultaneously on more than one drive, in order to withstand a hard drive failure), fail over computer clusters (where a group of separate yet tightly coupled computers work together and appear to the outside world as a single computer - any failure of a node is overcome by the redundant nodes), and backups. Thus, in order to perfectly erase data bookkeeping must be done in order to track each and every created copy. If the data is stored on a hard drive, as the storage devices are not designed to remove every trace of the written data, it is commonly traceable and at least partially reconstructable. Indeed, forensic methods can be used to extract data written over for a limited number of times. Even if the data is not meant to be written anywhere but the main Random Access Memory, many times the operating system duplicates it by itself to the hard drive, as part of the so-called virtual memory, and also to the fast memory cache, to speed up access and use. In view of the problematic nature of the assumption, regardless of whether perfect erasures can be ultimately achieved in practice at a reasonable cost or not (or whether it can be reasonably assumed or not), it is important to understand how essential for the security of cryptographic protocols is the ability to erase data fully and completely.

---

<sup>1</sup>This follows from the requirement that the secret must be constructable in each time period if enough players pull the information together.

The question we raise and study in this thesis is whether some other weaker form of erasures would suffice.

In particular, we propose to investigate *partial erasures* as opposed to the two extremes of having *perfect erasures* on the one hand (no information whatsoever is left after perfectly erasing the memory contents) and *no erasures* on the other (the memory is persistent). We propose to model partial erasures by a length shrinking function  $h : \{0, 1\}^* \rightarrow \{0, 1\}^*$ , that shrinks stored information by a fraction  $\phi$ , where  $0 \leq \phi \leq 1$ . For example, fraction 0 corresponds to the perfect erasures case, fraction 1 corresponds to the no erasures case, and fraction  $2/3$  corresponds to the case of only erasing a third of the string. That is, a shrinking function with  $\phi = 2/3$  is one that on input of size  $n$  bits, outputs an  $2n/3$  bit string. We call  $\phi$  a storage fraction. We do not even require that  $\phi$  will be a constant - it may be  $\frac{1}{\text{poly}(n)}$  close to 1 for the security parameter  $n$ . The shrinking function may then be chosen in the worst possible fashion by the adversary, who has complete knowledge of the secret sharing protocol, except that it must obey the storage fraction limitation.<sup>2</sup> In fact, the shrinking function need not be efficiently computable.

Our memory model thus will consist of general use random access memory, which can only be partially erased using a partial erasure function and a constant length register (for computation sake). We assume that the register is perfectly erasable. We remark that the existence of this perfectly erasable register is not only technologically reasonable, but does not provide a "back door" way to circumvent the lack of perfect erasure in main memory, since the register size is constant. We will have to pay special attention to our choice of schemes so that they are implementable using a constant size register for computations. We will further elaborate on the subject of algorithms that can be implemented in this model in subsection 2.4. We will call this memory model the bounded register memory model.

We show several constructions. First, we propose a new proactive secret sharing scheme where only partial erasures with storage fraction  $\phi$  by the honest parties is possible (where  $\phi$  can be  $\frac{1}{\text{poly}(n)}$  close to 1 for security parameter  $n$ ). We show that an honest but curious mobile adversary who corrupts up to  $p - 1$  of the  $p$  parties can gain only a negligible amount of information (in  $n$ ) about the secret. Second, we extend the scheme to a threshold proactive secret sharing scheme, which maintains correctness and privacy against malicious mobile adversary which corrupts up to a third of the parties. The extension requires adding a VSS scheme which can be implemented in our bounded register memory model. In particular, we use the VSS scheme defined by Feldman and Micali [FM88], which simplifies the scheme originally proposed by [BOGW88], and gives us the specified third fault resiliency bound.<sup>3</sup>

---

<sup>2</sup>Note that by modeling partial erasures this way, the case where the adversary can only choose to see the individual bits of the secret is included as a special case.

<sup>3</sup>We note that by implementing a different VSS scheme, such as the one defined by [RBO89], we can

More generally, we propose a method of transforming any proactive secret sharing schemes which obey certain conditions and require perfect erasures, into a proactive secret sharing scheme with the same fault tolerance which uses only partial erasures with storage fraction  $\phi$  (where again  $\phi$  can be  $\frac{1}{\text{poly}(n)}$  close to 1) and maintains correctness and privacy.

The conditions on the original proactive secret sharing to which our henceforward applies are:

1. The scheme consists of 3 phases - dealing, refreshing and reconstruction.
2. The scheme achieves privacy against a computationally unbounded adversary.
3. The shares are uniformly distributed over  $\text{GF}(2^n)$ .
4. All the computations can be done under the bounded register model.

The known PSS schemes of [CH94, HJKY95] obey these conditions.

Finally, in all of the schemes we construct, there is a dependence between the storage fraction  $\phi$  of the shrinking function  $h_\phi$  and the number of rounds for which the PSS system using  $h_\phi$  can be used while maintaining privacy. The dependency we achieve is as follows: Let  $p(n)$  be a polynomial. Fix a storage fraction  $\phi < 1 - \frac{1}{p(n)}$ . Then, the system can be safely used for any polynomial (in  $n$ ) number of rounds (where the polynomial is not fixed in advance).

## 1.4 Technique

The crux of the idea in all of the above constructions is to replace the representation of shares  $s_i \in \{0, 1\}^n$  (and refresh data sent between rounds) with so-called pseudo-shares  $(R_i, k_i)$  where the product  $R_i \cdot k_i = s_i$  ( $R_i \in \{0, 1\}^{n \times m}$  is a random matrix of size and  $k_i \in \{0, 1\}^m$  a random vector up-to the product whose dimensions are related to the storage fraction  $\phi$  used). Then, instead of perfectly erasing old shares (and old refresh data) it will suffice to apply the shrinking function  $h$  to the old  $k_i$ . What will remain in the memory for the adversary to use is  $(R_i, h(k_i))$ .

Informally, we prove in subsection 4.4 that given that:

1. All “chunks” of meaningful data are stored in the form of the tuples  $(R_i, k_i)$ .
2. All the  $k_i$ 's are partially erased as stated above, by applying  $h$  to the  $k_i$ 's before the adversary can access them.

---

tolerate  $\frac{t}{2}$  faults, but for this exposition we stick to the simpler [FM88] VSS.

3. The amount of these “chunks” is polynomial in the security parameter  $n$ .
4. All the computations performed on these “chunks” can be implemented in the fixed length register model (without the use of main memory to store meaningful information).

Then the amount of information that the adversary may gain on the secret is negligible.

*Remark 1.1.* The function  $h$  defined in subsection 2.3 can be applied to any arbitrary length input, as long as the output of the function obeys the bound of  $\phi$ . Even if the secret data is huge, as long as it is still polynomial in length in the security parameter, the results will stand. It will, however, necessitate the creation of huge  $(k, R)$  tuples for representing data  $s = Rk$ . Another option could be, in a case of a very long string of data, to divide it into separate blocks, and erase each one individually. This theoretically corresponds to a reduction in the strength of the adversary, as the division into blocks can be seen as a limitation on the shrinking functions available to the adversary, to only include functions whose input is from a single block, without relation to the other blocks. For practical reasons, however, it may be reasonable to divide huge secrets into smaller ones, create and store (and partially erase) a much smaller tuple for each. That said, in this thesis we treat the secrets unbroken, as the result is clearly stronger in this case.

*Remark 1.2.* Repeated application of the erasure function  $h$  is not allowed. This is due to a number of reasons:

1. The type of erasures available to a computer in the OS level are not of concern to the protocol which runs on the application level. As far as the honest protocol is concerned the data is fully erased once  $h$  has been applied to it.
2. Application of  $h$  repeatedly within the same time period will make the shrinking fraction definition pointless.
3. Planning on re-erasing data (which has been erased before) in a subsequent round, cannot help an honest party, since the adversary has the power to switch into this party at any subsequent round of his choice.

Besides, the result is only stronger by not allowing such behavior.

## 1.5 Future Work

The represented technique in this thesis may be exploited to replace perfect erasures with partial erasures for many other, unrelated tasks.

Subsequent to our work, the technique has been used by [CL08] to convert any circuit relying on perfect erasures into an circuit requiring only partial erasures. For example, they achieve *forward security*, as defined in [And97, DvOW92, Gün89]. Here, every time period the secret key is changed, in a way that if the adversary corrupts the receiver at time period  $i$ , it cannot decrypt secrets which have been encrypted in previous time periods. Clearly, without some sort of erasure, this cannot be done. However, with the represented technique, even partial erasure would suffice.

## 1.6 The Organization of the Rest of the Thesis

In Section 2 we define (and prove where needed) tools and terms to be used in this thesis. Section 3 is where we will elaborate on other work related to ours. The related work is discussed after the definitions, since many of the definitions are required by it. In section 4 we present our solution to the question at hand, and section 5 is an implementation of the algorithms, which are required by our construction.

# 2 Preliminaries and Definitions

## 2.1 The Model of Mobile Adversary

The following model of mobile adversary is adapted from [HJKY95, OY91]. We start their basic model, which assumes honest but curious adversaries and does not include robustness against malicious adversary. We formally define the model below. The formal definition of a secret sharing scheme will be provided in subsection 2.2.

We assume a system of  $p$  parties  $P_1, \dots, P_p$  that will proactively share a secret value  $s$  through a threshold secret sharing scheme, tolerating corruptions. We assume that the system is securely and properly initialized. The goal of the scheme is to prevent the adversary from learning the secret  $s$ .

**Parties and Communication Model.** We assume that each party has a local, completely hidden source of randomness, and that either each pair of parties share a completely private channel between them, or may achieve such a channel by public key cryptography.

**Time Periods and Refresh Phases.** Time is divided into *time periods* which are determined by the common global clock. At the beginning of each time period the parties engage in an interactive *refresh* protocol. At the end of a refresh phase the parties hold new shares of the (same) secret  $s$ .

**The Mobile Adversary Model.** We assume that the adversary corrupts less than  $c$  out of  $p$  parties in each time period. The adversary can corrupt parties at any moment during a time period. If a party is corrupted during a refresh phase, we consider the party as corrupted during both time periods adjacent to the refresh phase.

As explained in [HJKY95], the reason behind this way of counting corrupted parties is that it is very hard to analyze what happens if we differentiated between an adversary who moves from one party to another during the refresh phase and the adversary who just stays in both parties throughout. It is also not a realistic concern in our setting, where the refresh phase is very short when compared to the length of a time period.

## 2.2 Secret Sharing

**Definition 2.1** (Secret Sharing). We define  $p$  party secret sharing by a tuple  $\Pi = (S, (S_1, \dots, S_p), R, D)$ , where  $S$  is a finite *secret domain*, each  $S_i$  is a finite *share domain* of party  $P_i$  from which its share  $\mathbf{sh}_i$  is picked,  $R$  is a probability distribution from which the dealer's random input is picked, and  $D$  is a share distribution function mapping a secret  $s \in S$  and a random input  $r \in R$  to a  $p$ -tuple of shares  $D(s, r) = (\mathbf{sh}_1 \times \dots \times \mathbf{sh}_p) \in (S_1 \times \dots \times S_p)$ , and then privately communicating each share  $\mathbf{sh}_i$  to party  $P_i$ . We say that  $\Pi$  *realizes* an access structure  $\Gamma \subset 2^{[p]}$  if it satisfies the following: 1. Correctness. For any qualified set  $Q = \{i_1, \dots, i_k\} \in \Gamma$ , there exists a *reconstruction function*  $\text{rec}_Q : S_{i_1} \times \dots \times S_{i_k} \rightarrow S$  such that for every secret  $s \in S$ ,  $\Pr[\text{rec}_Q(D(s, R)_Q) = s] = 1$ , where  $D(s, R)_Q$  denotes a restriction of  $D(s, R)$  to its  $Q$ -entries; 2. Privacy. For any unqualified set  $U \notin \Gamma$  and secrets  $s, s' \in S$  the random variables  $D(s, R)_U$  and  $D(s', R)_U$  are indistinguishable.

Depending on the notion of indistinguishability used in the above definition, the privacy could be computational, statistical, or information theoretical.

In this work we refer to the following specific secret sharing schemes.

**Definition 2.2** ( $(c, p)$  Threshold Secret Sharing). A  $(c, p)$  threshold secret sharing scheme is a secret sharing scheme  $\Pi$  that realizes the  $(c, p)$  threshold access structure  $\Gamma = \cup S$ , over all  $S \in 2^{[p]}$  s.t.  $|S| \geq c$ .

**Definition 2.3** (Proactive  $(c, p)$  Threshold Secret Sharing). A proactive  $(c, p)$  threshold secret sharing scheme is a threshold secret sharing scheme that realizes the  $(c, p)$  threshold access structure against a mobile adversary.

**Definition 2.4** (Robustness of a Proactive  $(c, p)$  Threshold Secret Sharing). A proactive  $(c, p)$  threshold secret sharing scheme is robust, if privacy and correctness are maintained in the presence of less than  $c$  malicious parties.

As a mobile adversary may break over time into at least  $c$  parties, without any erasures the adversary can eventually discover  $c$  shares of the secret that were used at the same time period. Thus some sort of erasure is essential to the proactive threshold schemes. Indeed all previous proactive secret sharing schemes require perfect erasures of past shares and refresh data at the end of each refresh phase.

All existing proactive secret sharing schemes [CH94, HJKY95] consist of 3 phases: Dealing, Refreshing and Reconstruction. During the Refresh phase, parties send other parties refresh data. Moreover, in all existing schemes the data a party  $i$  sends in the Refresh phase to the other parties, can be generated for one party following the other, using the data being sent to the previous parties and randomness. That is, first generate the data to send to the first party, then to the second (using the data sent to the first), then to the third (using the data sent to the first two), and so on. We will assume any proactive secret sharing scheme conforms to this format.

Extension to malicious Adversaries: [HJKY95] also gives a proactive secret sharing for malicious adversary, by applying a VSS scheme over an honest but curious proactive threshold secret sharing scheme. In this thesis, we will address both models of adversaries: (1) honest but curious and (2) malicious. Unless we specify so explicitly, an honest but curious adversary is assumed.

## 2.3 The Partial Erasures Model

For the partial erasure model, we define a *storage fraction*  $0 \leq \phi < 1$ , which defines the upper bound on the fraction of the data which remains after the erasure is applied. For our result we allow  $\phi$  to be even  $\frac{1}{poly(n)}$  close to 1. We think of  $\phi$  as chosen by the adversary. Namely, it can choose any polynomial function of  $n$  to be close to 1. It is easy to see that we cannot have  $\phi = 1$ , as it stands for no erasure at all, and without any sort of erasure security cannot be maintained in these schemes.

We model partial erasures by an arbitrary length shrinking function  $h : \{0, 1\}^* \rightarrow \{0, 1\}^*$ , s.t. for all inputs  $x$  to the function,  $\frac{|h(x)|}{|x|} \leq \phi$ . By *partially erasing* some information  $x$ , we mean exactly the operation of applying such an  $h$  to  $x$ . So when a party partially erases  $x$ , the result is  $h(x)$ , and an adversary who breaks into the party after the partial erasure will see only  $h(x)$ , as opposed to nothing under perfect erasures, or  $x$  if no erasures were applied. The ability to choose a priori what data will remain and see later on the remaining data is the strength of the adversary.

Our shrinking function may be set by the adversary whenever it corrupts some party (that is, on the worst case, just in the previous time period). It may even change with time.

However, we do limit the adaptivity of the function, as we do not allow the output of the function to depend on data in memory other than the erased data itself. Therefore, our function is defined to have limited adaptivity.

We will call *perase* ( $x$ ) the application of the erasure function  $h$  on some string  $x$ .

### 2.3.1 Storage Fraction vs. Time Periods

Naturally, as our system has multiple fixed length time periods, or rounds, in each of which the adversary may learn more information, there might be a dependency between the required erased fraction and the number of played rounds. A central goal would be to limit such a dependency to the minimum. As we will see in section 4, for our solution the required erased data is *logarithmic* in the number of played rounds. Namely, let the security parameter be  $n$ . Then, for storage fraction  $\phi < 1 - \frac{1}{poly(n)}$ , the information gained on the secret in every round grows by at most a  $\frac{1}{exponential(n)}$  amount. Thus, as long as the PSS is utilized for a  $poly(n)$  rounds,<sup>4</sup> the information gained on the secret is still bounded by an amount negligible in  $n$ .

We say that  $h$  is a *partial erasure function with storage fraction  $\phi$* .

## 2.4 The Memory Model

For our work, we will assume a computer model that has a small fixed length register, and general use random access memory. We will assume the register is perfectly erasable, whereas for the other memory, we only know that it can be partially erased by the erasure function stated in subsection 2.3.

This is a reasonable assumption, as the modern computer contains many types of storage memory, such as Registers, Cache, Random Access Memory (RAM), Hard Drives etc, where each type of memory has different properties, in terms of space, and longevity of erased data.

We note that the existence of a constant size perfectly erasable register is not only reasonable, but does not somehow provide a "back door" way to circumvent the lack of perfect erasure in main memory, since the size of the secret is growing asymptotically with a security parameter, whereas the register size is *fixed*. Thus, at no time can the register hold any non-negligible part of the secret.

It is not immediately clear which algorithms can be computed in this very restricted model of fixed length register. The length of the register limits us to operations which may be calculated locally, bit by bit. For example, modular calculations are harder to perform in this model. However, subsequent work by [CL08], suggests to bypass the problem by breaking a complex circuit into many simple ones, each of which is can be easily calculated

---

<sup>4</sup>The maximal number of rounds, or at least their polynomial function of  $n$ , must be fixed in advance.

under the register model, and generate a storage tuple for any intermediate step. Clearly, each intermediate tuple will add to the adversary's knowledge, yet as long as the total amount of tuples is polynomial in the security parameter  $n$ , the amount of data the adversary may gain will remain negligible.

We stress that our memory model is different from the locally computable model, proposed in the extractor literature in [Vad04]. In the locally computable model, every output bit must depend on a fixed number of source input bits, whereas in our model, the limitation is only that at any given stage of the computation, only a constant number of bits are stored in the memory.

## 2.5 Information Theory

We will define and prove some facts in information theory, which will later be used in the thesis.

### 2.5.1 Statistical Distance

We state and prove some useful facts about the statistical distance. In this subsection, all the capital letters are random variables (r.v.s), and their corresponding scripts the possible values they can take on.

**Definition 2.5** (Statistical Distance). Suppose that  $A$  and  $B$  are two distributions over the same set  $\mathcal{Z}$ . The statistical distance between  $A$  and  $B$  is defined as  $\Delta(A, B) := \frac{1}{2} \sum_{z \in \mathcal{Z}} |\Pr_A[z] - \Pr_B[z]|$ .

**Definition 2.6** (Statistical Distance from the Uniform). Let  $d(A) := \Delta(A, U_{\mathcal{A}})$ . Also define  $d(A|B) := \sum_b d(A|B=b) \cdot \Pr_B[b] = \sum_b \Pr_B[b] \frac{1}{2} \sum_a |\Pr_{A|B=b}[a] - \frac{1}{|\mathcal{A}|}|$ .

We say that a random variable  $A$  is  $\epsilon$ -close to the uniform given  $B$  to mean that  $d(A|B) \leq \epsilon$ .

Note that  $0 \leq \Delta(A, B) \leq 1$ , and thus also  $0 \leq d(A|B) \leq 1$ .

### 2.5.2 Entropy

In this subsection we give the definition of entropies, which will be used in the proof of the main theorems.

**Definition 2.7** (Shannon Entropy). Let  $X$  be a random variable distributed over some set  $\mathcal{V}$ , then:

$$H(X) = - \sum_{v \in \mathcal{V}} \Pr[X = v] \log \Pr[X = v]$$

We also define the conditional entropy as  $H(Y|X) = \sum_{v \in \mathcal{V}} \Pr[X = v] H(Y|X = v)$ , just as we did with the conditional statistical distance. Using Bayes' theorem we easily get that  $H(Y|X) = H(Y, X) - H(X)$ .

**Definition 2.8** (Min Entropy). Let  $X$  be a random variable distributed over some set  $\mathcal{V}$ , then:

$$H_\infty(X) = -\log \max_{v \in \mathcal{V}} \Pr[X = v]$$

$X$  is called a  $k$ -source if  $H_\infty(X) \geq k$ .

**Definition 2.9** (Rényi Entropy). Let  $X$  be a random variable distributed over some set  $\mathcal{V}$ , then:

$$H_{\mathfrak{R}}(X) = -\log \sum_{v \in \mathcal{V}} (\Pr[X = v])^2$$

The Rényi entropy indicates the collision probability: If  $X, X'$  are independent identically distributed random variables, then  $H_{\mathfrak{R}}(X) \geq k$  iff  $\Pr[X = X'] \leq 2^{-k}$ .

**Theorem 2.10** (Chain rule). Let  $X_1, X_2, \dots, X_n$  be random variables. Then:

$$H(X_1, X_2, \dots, X_n) = \sum_{i=1}^n H(X_i | X_1, X_2, \dots, X_{i-1})$$

*Proof.* By repeatedly applying  $H(Y, X) = H(X) + H(Y|X)$  we get that:

$$\begin{aligned} H(X_1, X_2) &= H(X_1) + H(X_2 | X_1) \\ H(X_1, X_2, X_3) &= H(X_1) + H(X_2, X_3 | X_1) \\ &= H(X_1) + H(X_2 | X_1) + H(X_3 | X_1, X_2) \\ &\vdots \\ H(X_1, X_2, \dots, X_n) &= H(X_1) + H(X_2 | X_1) + \dots + H(X_n | X_1, X_2, \dots, X_{n-1}) \\ &= \sum_{i=1}^n H(X_i | X_1, X_2, \dots, X_{i-1}) \end{aligned}$$

□

### 2.5.3 Other Definitions

**Definition 2.11** (Mutual Information). Let  $X$  and  $Y$  be random variables. We define their mutual information as:

$$\begin{aligned} I(X : Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X, Y) \end{aligned}$$

Another standard representation is:

$$I(X : Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

**Lemma 2.12.** *Conditional mutual information is interchangeable:*

$$I(Y : X) - I(Y : X | Z) = I(X : Z) - I(X : Z | Y)$$

*Proof.* We invoke the definition above:

$$\begin{aligned} I(Y : X) - I(Y : X | Z) &= (H(X) - H(X | Y)) - (H(X | Z) - H(X | Y, Z)) \\ &= (H(X) - H(X | Z)) - (H(X | Y) - H(X | Y, Z)) \\ &= I(X : Z) - I(X : Z | Y) \end{aligned}$$

□

**Theorem 2.13** (Chain rule for Mutual Information). *Let  $X_1, X_2, \dots, X_n$  be random variables. Then:*

$$I(X_1, X_2, \dots, X_n : Y) = \sum_{i=1}^n I(X_i : Y | X_1, X_2, \dots, X_{i-1})$$

*Proof.* By applying theorem 2.10 we get that:

$$\begin{aligned} I(X_1, X_2, \dots, X_n : Y) &= H(X_1, X_2, \dots, X_n) - H(X_1, X_2, \dots, X_n | Y) \\ &= \sum_{i=1}^n H(X_i | X_1, X_2, \dots, X_{i-1}) - \sum_{i=1}^n H(X_i | X_1, X_2, \dots, X_{i-1}, Y) \\ &= \sum_{i=1}^n I(X_i : Y | X_1, X_2, \dots, X_{i-1}) \end{aligned}$$

□

**Definition 2.14** (Kullback-Leibler Divergence). Let  $X$  and  $Y$  be random variables over  $\mathcal{V}$ . We define the Kullback-Leibler divergence as:

$$D(X \parallel Y) = \sum_{v \in \mathcal{V}} \Pr[X = v] \log \frac{\Pr[X = v]}{\Pr[Y = v]}$$

We also easily get from the second representation of mutual information that:

$$I(X : Y) = D(p(x, y) \parallel p(x)p(y))$$

It is also clear that:

$$D(X \parallel U_n) = n - H(X)$$

**Lemma 2.15** (Proved by [CT06], theorem 12.6.1). *Let  $P_1$  and  $P_2$  be two distributions. Then:*

$$D(P_1 \parallel P_2) \geq \frac{1}{2 \ln 2} \Delta(P_1, P_2)^2$$

**Lemma 2.16.** *Let  $X \in \{0, 1\}^n$  be a random variable. Then:*

$$d(X) \leq \sqrt{2 \ln 2 (n - H(X))} \tag{1}$$

*Proof.* By definition 2.14 and lemma 2.15 we get that:

$$\frac{1}{2 \ln 2} d(X)^2 \leq D(X \parallel U_n) = n - H(X)$$

Therefore also:

$$d(X) \leq \sqrt{2 \ln 2 (n - H(X))}$$

□

**Theorem 2.17** (Proved by [CT06], theorem 16.3.2). *Let  $X \in \{0, 1\}^n$  be a random variable such that  $d(X) \leq \frac{1}{2}$ . Then:*

$$n - H(X) \leq -d(X) \log \frac{d(X)}{2^n} \tag{2}$$

## 2.6 Extractors

In this subsection we give the definition of extractors, which have much similarity to our solution.

**Definition 2.18** (Extractor).  $\text{Ext} : \{0, 1\}^m \times \{0, 1\}^d \rightarrow \{0, 1\}^n$  is a  $(\alpha, \epsilon)$ -extractor, if for every  $\alpha$ -source  $X$  over  $\{0, 1\}^m$ ,  $\text{Ext}(X, U_d)$  is  $\epsilon$ -close to  $U_n$ .

**Definition 2.19** (Strong Extractor [NZ96]).  $\text{Ext} : \{0, 1\}^m \times \{0, 1\}^d \rightarrow \{0, 1\}^n$  is a strong  $(\alpha, \epsilon)$ -extractor, if for every  $\alpha$ -source  $X$  over  $\{0, 1\}^m$ ,  $U_d \circ \text{Ext}(X, U_d)$  is  $\epsilon$ -close to  $U_d \times U_n$ .

**Definition 2.20** (Locally Computable Extractor).  $\text{Ext} : \{0, 1\}^m \times \{0, 1\}^d \rightarrow \{0, 1\}^n$  is a  $t$ -locally computable (or  $t$ -local)  $(\alpha, \epsilon)$ -extractor, if for every  $r \in \{0, 1\}^d$ ,  $\text{Ext}(x, r)$  depends on only  $t$  bits of  $x$ .

## 2.7 Universal Hash Functions and the Leftover Hash Lemma

In this subsection we give the definition of universal hash functions and the leftover hash lemma, which will be used in the proof of the main theorems.

**Definition 2.21** (Universal Hash Function). Let  $h : \{0, 1\}^\ell \times \{0, 1\}^n \rightarrow \{0, 1\}^m$  be an efficiently computable function. For a fixed  $y \in \{0, 1\}^\ell$ , view  $h(y, x)$  as a function  $h_y(x)$  of  $x$ . Let  $Y \in_R \{0, 1\}^\ell$ . Then,  $h$  is a pairwise independent universal hash function if for every  $x \neq x' \in \{0, 1\}^n$  and every  $a, a' \in \{0, 1\}^m$ , we have that:

$$\Pr_Y [h_Y(x) = a \wedge h_Y(x') = a'] = 2^{-2m}$$

**Lemma 2.22** (The Leftover Hash-Lemma, as defined and proved by [HILL99]). *Let  $D : \{0, 1\}^n$  be a probability ensemble with Rényi entropy  $H_{\mathfrak{X}}(D) \geq m$ , and let  $e$  be a positive integer. Let  $h : \{0, 1\}^\ell \times \{0, 1\}^n \rightarrow \{0, 1\}^{m-2e}$  be a universal hash function. Let  $X \in_D \{0, 1\}^n$ ,  $Y \in_R \{0, 1\}^\ell$ . Then  $\Delta(\langle h_Y(X), Y \rangle, \langle U_{m-2e}, Y \rangle) \leq 2^{-(e+1)}$ , and therefore also  $d(h_Y(X) | Y) \leq 2^{-(e+1)}$ .*

*Proof.* Let  $s = m - 2e$ , and for all  $x \in \{0, 1\}^n$ ,  $y \in \{0, 1\}^\ell$ ,  $a \in \{0, 1\}^s$  let  $\chi(x, y, a)$  be the indicator of  $h_y(x) = a$ . We want to show that:

$$\mathbb{E}_Y \left[ \sum_{a \in \{0, 1\}^s} |\mathbb{E}_X [\chi(X, Y, a)] - 2^{-s}| \right] / 2 \leq 2^{-(e+1)}$$

We will show that for all  $a \in \{0, 1\}^s$ ,

$$\mathbb{E}_Y [|\mathbb{E}_X [\chi(X, Y, a)] - 2^{-s}|] \leq 2^{-(e+s)}$$

and the proof follows from it.

For any random variable  $Z$ ,  $E[|Z|^2] \geq E[|Z|]^2$ , by Jensen's Inequality. Therefore, it is sufficient to show that for all  $a \in \{0, 1\}^s$ ,

$$E_Y \left[ \left( E_X [\chi(X, Y, a)] - 2^{-s} \right)^2 \right] \leq 2^{-2(e+s)}$$

Let  $X' \in_D \{0, 1\}^n$ . Then, we can write the above as:

$$E_{X, X'} \left[ E_Y \left[ (\chi(X, Y, a) - 2^{-s}) (\chi(X', Y, a) - 2^{-s}) \right] \right]$$

For each fixed pair of values  $x, x'$  for  $X, X'$ , if  $x \neq x'$ , the expectation with respect to  $Y$  is 0 because of the pairwise independence property of universal hash-functions. If  $x = x'$ , then:

$$E_Y \left[ (\chi(x, Y, a) - 2^{-s})^2 \right] = 2^{-s} (1 - 2^{-s}) \leq 2^{-s}$$

Because  $H_{\mathfrak{R}}(D) \geq m$ , by definition of Rényi entropy it follows that  $\Pr[X = X'] \leq 2^{-m}$ . Thus, the sum is at most  $2^{-(m+s)} = 2^{-2(e+s)}$  by definition of  $s$   $\square$

## 2.8 Privacy and Robustness Definition in Partial Erasures Model

In this subsection we give the privacy definition of a proactive  $(c, p)$  threshold secret sharing scheme with partial erasures.

**Definition 2.23** (View of the Adversary). Let  $VIEW^T$  denote the view of the adversary  $E$  up to time period  $T$ , i.e. the concatenation of  $VIEW^{T-1}$  and all the public information that  $E$  sees as well as the information seen when breaking into at most  $c - 1$  parties in time  $T$ .  $VIEW^0$  is defined to be the empty set.

**Definition 2.24** (Privacy of a proactive  $(c, p)$  threshold secret sharing scheme with partial erasures). We say that a proactive  $(c, p)$  threshold secret sharing scheme with partial erasures  $\Pi = (S, (S_1, \dots, S_p), R, D)$  is  $(\phi, \tau)$ -secure, if for all time periods  $T \leq \tau$ , for all adversaries  $E$  that breaks into at most  $c - 1$  parties per time period, for all partial erasing functions  $h$  with storage fraction  $\phi$ , and for all  $secret \in S$ ,  $d(secret|VIEW^T)$  is negligible in the security parameter  $m$ .

**Definition 2.25** (Robustness of a proactive  $(c, p)$  threshold secret sharing scheme with partial erasures). Like the perfect erasure scheme, we say a proactive  $(c, p)$  threshold secret sharing scheme with partial erasures  $\Pi = (S, (S_1, \dots, S_p), R, D)$  is robust, if privacy and correctness are maintained in the presence of less than  $c$  malicious parties.

## 3 Related Work

### 3.1 The Bounded Storage Model (BSM)

Much of modern cryptography assumes that the adversary is computationally bounded. The Bounded Storage Model (BSM) proposed by Maurer [Mau90, Mau92], makes in contrast assumptions on the storage capabilities of the adversary. These assumptions enables novel approaches to the secure communication problem as follows.

As usual communicating parties A and B begin with a short initial secret key  $k$ . They use this key  $k$  and access to a very long public random string  $R$  in order to derive a longer key  $X$  which will enable them to communicate privately over the public channel by simply using  $X$  as a one-time pad. The key (or one-time pad) derivation protocol takes place in two phases.

Phase I: During this phase all parties (A, B, the adversary) have access to the long random string  $R$ . A and B (non-interactively) apply a public *key deriving function*  $f$  to  $(k, R)$  to derive the long key  $X$  that they can use as a one-time pad for the purpose of encryption. The adversary which has access to  $R$  is space bounded, and cannot store all of  $R$ . This is formalized by modeling the adversary's storage with a length shrinking *storage function*  $h_\phi$ , i.e. a function  $h_\phi : \{0, 1\}^* \rightarrow \{0, 1\}^*$  s.t.  $\forall x, \frac{|h_\phi(x)|}{|x|} \leq \phi$ , where  $\phi$  is a constant  $0 \leq \phi < 1$  which is called the *storage fraction* of the adversary (or of function  $h_\phi$ ). The best the adversary can do at this phase is to store  $h_\phi(R)$ .

Phase II: The common random string disappears. In this phase, the honest parties use their derived key  $X$  to communicate. The question is, how much information can the adversary find about  $X$ ? A sequence of works including [AR99, DM02, Mau92] culminated in showing that, for  $\phi$  arbitrarily close to 1, there exists an explicit key deriving function  $f$  such that,  $X = f(k, R)$  is  $\epsilon$ -close to uniform given  $k, h_\phi(R)$ . Namely, even if the adversary is given the initial secret key  $k$  at this point it cannot distinguish  $X$  apart from random with significant probability.

The ideas emerging from the BSM research inspire a way to capture some weak form of erasures. In particular, the information about  $R$  (the long common random string) stored by the bounded-space adversary in the BSM model was captured by computing an arbitrary length shrinking function applied to  $R$ . In the PSS setting we will use the same kind of length shrinking function to capture the act of partially erasing old shares of a secret. Whereas in BSM one wants to exhibit a key derivation function  $f$  such that the derived key  $f(k, R)$  is still essentially indistinguishable from random, in the PSS setting we will introduce the idea of pseudo-shares. Pseudo shares are representation of ordinary PSS secret shares such that ordinary shares can be derives from pseudo shares upon reconstruction; and yet pseudo-

shares can be partially erased by applying the length shrinking function at the end of every round, so that the corresponding share will essentially be indistinguishable from random. This will make it impossible for the (mobile) adversary attacking the PSS to essentially distinguish the secret itself from random.

### 3.1.1 Intrusion-Resilient Secret Sharing

Independently, Dziembowski has addressed a very similar question in [Dzi06a, Dzi06b], and secret sharing in particular in [DP07], called *intrusion resiliency in the Bounded Storage Model*. His work addresses secret sharing within the BSM model. He puts a limitation on the adversary in the amount of data it can retrieve from parties it corrupts, so if the secret keys are too large, and created properly, the adversary cannot learn meaningful data from it, yet the honest parties can reconstruct the secret with very small data transfer. Our approach is different, as we focus on a method to withstand partial erasures of data by honest parties, instead of restricting the intrusion into the corrupted parties. He assumes the *adversary* is *bounded*, whereas we assume weak *honest parties*.

Due to this fundamental difference, our shares do not have such a size requirement. For obvious reasons, our shares are larger than in the perfect erasure model, as the extra data is used in the partial erasure model to “confuse” the adversary. However, since the percentage of data is guaranteed to be erased, we do not need to go up to the sizes required by the BSM model.

## 3.2 Exposure-Resilient Functions (ERF)

Exposure-Resilient Functions, or ERFs, were introduced by [CDH<sup>+</sup>00, Dod00]. An  $\ell$ -ERF is a threshold function with a random input, which if the adversary learns all but  $\ell$  bits of the input, cannot distinguish the output of the function from random. Formally, a function  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$  is  $\ell$ -ERF if for any choice of  $m - \ell$  bits and for  $k \in_R \{0, 1\}^m$  and  $r \in_R \{0, 1\}^n$ , given the  $m - \ell$  bits of  $k$ ,  $f(k)$  and  $r$  are indistinguishable, either perfectly, statistically or computationally. For the perfect model, it has been proven that it is impossible to construct an  $\ell$ -ERF for  $\ell < \frac{m}{2}$  if  $n > \log m$ . For the statistical model, it is intuitive (and has been proven), that a lower bound of  $\ell \geq n$  still exists, and indeed, [Dod00] has shown a Statistical ERF which essentially meets this lower bound. This lower bound is further removed in the case of computational ERF, by using a pseudo-random generator.

The ERF objectives seem quite related to our PESS (Partial Erasure Secret Sharing), when defining  $\phi = \frac{m-\ell}{m}$ . However, there are a few critical differences between the two:

1. The ERF limits the adversary's knowledge to specific bits, whereas our shrinking function  $h(\cdot)$  may be any arbitrary function which is applied to the whole random string.
2. In the PSS setting, new (refresh) random data is injected into the process at each stage which essentially effects the timing of the adversary's knowledge.

In ERF:

- (a) There is a fixed known function  $f$ .
- (b) The adversary chooses up to  $m - \ell$  bits of  $k$ .
- (c) The adversary sees his choice bits of  $k$ .
- (d) The adversary attempts to guess  $f(k)$ .

In PESS:

- (a) There is a fixed known function  $f$ .
- (b) The adversary chooses a shrinking function  $h(\cdot)$  to apply on  $k$ , leaving up to  $m - \ell$  bits.
- (c) New refresh data  $R$  is independently chosen.
- (d) The adversary given  $R$  and  $h(k)$  attempts to guess  $f(k, R)$ .

Viewing  $R$  as part of the choice of  $f$ , in PESS the adversary lacks complete knowledge of  $f$  when she makes her choice of shrinking function, whereas when she makes her choice of bits to see in ERF she has complete knowledge of  $f$ .

*Note 3.1.* As our result is in the statistical model, and not in the perfect model, the lower bound of the ERF which may apply here is  $\ell \geq n$ . Indeed, the bound on  $\phi$  in theorem 4.6 implies the same thing: If we define  $\phi = \frac{m-\ell}{m} = 1 - \frac{\ell}{m}$ , then  $\ell \geq n$  implies  $\phi \leq 1 - \frac{n}{m}$ , and as we define  $m = 3n^c$ , for some constant  $c$ , we get  $\phi \leq 1 - \frac{1}{3n^{c-1}}$ . Our bound is even stronger,  $\phi \leq 1 - \frac{1}{n^{c-1}}$ , as we need to handle the adversary's added power by choosing a function  $h$  of the bits and not only a selection of bits.

### 3.3 Relation Between our Solution and Extractors

Our solution involves generating two random variables,  $k$  and  $R$ , that given full knowledge of  $R$  and only partial knowledge of  $k$ , one cannot distinguish the output of some function  $f(k, R)$  from random. The unknown randomness in  $k$  can be seen as entropy, making the function  $f$  a strong extractor (as defined in definition 2.19).

One would think that the problem may be solved by using any strong extractor, in the following way: Say we have a secret  $s \in \{0, 1\}^n$ . Let  $k \in_R \{0, 1\}^m$  and  $R \in_R \{0, 1\}^d$ , and let  $\text{Ext} : \{0, 1\}^m \times \{0, 1\}^d \rightarrow \{0, 1\}^n$  be a strong  $(\alpha, \epsilon)$ -extractor. Then by storing  $k$ ,  $R$ , and  $s \oplus \text{Ext}(k, R)$ , the honest party can easily retrieve  $s$ , whereas after partially erasing  $k$ , the adversary will not be able to learn anything about  $s$  - she will not be able to distinguish  $s \oplus \text{Ext}(k, R)$  from uniform. However, this is not the case, as the extractor must be computable in our memory model.

One could then think that any locally computable strong extractor would fit here. However, we next show that such a solution is not achievable.

Say  $\text{Ext}$  is a  $t$ -local strong  $(\alpha, \epsilon)$ -extractor, and say the adversary chooses  $h$  to be equally distributed - for example, it chooses to keep only the first  $\phi m$  bits of  $k$ . Clearly, in that case  $H_\infty(k | h(k)) = (1 - \phi) \cdot m$  (as defined in definition 2.8). Thus, we need  $\text{Ext}$  to be a strong  $((1 - \phi) \cdot m, \epsilon)$ -extractor. According to corollary 9.2 of [Vad04], if  $\text{Ext}$  is a  $t$ -local strong  $(\delta m, \epsilon)$ -extractor, then  $t \geq (1 - \epsilon - 2^{-n}) \cdot (1/\delta) \cdot n$ . In our case,  $\delta = 1 - \phi$ , so it is required that  $t \geq (1 - \epsilon - 2^{-n}) \cdot (1/(1 - \phi)) \cdot n$ . Even if  $\phi$  were a constant,  $t$  would still be greater than some constant  $c$  times  $n$ , contrary to our demand of a fixed length  $t$ .

Thus, locally computable strong extractors may not be used here, specifically we must use another form of strong extractor, which is computable in our memory model. For more information on our memory model, see subsection 2.4.

## 4 Our Solution to the PSS with Partial Erasures

In the proactive  $(c, p)$  threshold secret sharing, the goal is to proactively share the secret, against an adversary that is mobile and can corrupt up to  $c - 1$  parties in each time period. To maintain correctness and privacy, in between time periods the parties enter a *refreshing phase*. At the end they get new shares. If the adversary breaks into a party for the first time at time  $t$ , then we want to say that whatever information the adversary can see will not allow it to infer the old shares, those that the party holds for time 1 to  $t - 1$ . Perfect erasures is one way to guarantee that. As we will show, weaker forms of erasures are also good enough.

In subsection 4.1 we present a method to modify any existing proactive secret sharing scheme that requires perfect erasures under general conditions, into a new scheme that requires only partial erasures, and will maintain every other property of the original scheme.

The conditions are:

1. The scheme consists of 3 phases:

- (a) Dealing - when the parties receive their shares.
  - (b) Refreshing - when the parties negotiate to create new shares, after every time period.
  - (c) Reconstruction - when some group of parties reconstruct the secret.
2. The shares, refresh data, and secret are uniformly distributed over  $\text{GF}(2^n)$ . (we remark that it is possible to relax the condition on the secret to come from any distribution over  $\text{GF}(2^n)$ .)
  3. The refresh data can be computed sequentially, one party following the other.
  4. All the computations can be done under our memory model, defined in subsection 2.4.

The proactive secret sharing schemes of [CH94, HJKY95] satisfy these conditions (under some minor modifications to the schemes). We will define there some tools which will be used to prove the correctness of the modification method, as well as the theorems in the following subsections. For concreteness, in subsection 4.2 we present a particular proactive  $(p, p)$  secret sharing scheme, in the honest but curious mobile adversary setting. In subsection 4.3 a proactive  $(c, p)$  threshold secret sharing scheme, for  $c < \frac{1}{3}p$  robust against malicious mobile adversary is presented. The proofs of the stated theorems are in subsection 4.4.

## 4.1 The Scheme Modification Method

We give a high level design of the modified protocol:

Say  $\Pi$  is a  $(c, p)$  proactive secret sharing scheme which requires perfect erasures, and whose shares are uniformly chosen from  $\text{GF}(2^n)$ , and meets all the other conditions stated above. We think of  $n$  as the security parameter. We define hereby the modified proactive secret sharing scheme PESS ( $\Pi$ ).

Notation:

1. WLOG let  $\Pi$  call a subroutine to generate the refresh data, namely  $s_{i,jt}^t = \text{refresh}(i, j, \{s_{ij'}^t\}_{1 \leq j' < j})$  – accepts as arguments the source party  $i$ , the target party  $j$ , and the third argument is the refresh data  $\{s_{ij'}^t\}$  generated for the other parties so far (recall that as we specified before the refresh data can be generated for one party following the other).
2. WLOG let  $\Pi$  call a subroutine which uses the refresh data to update the shares, namely  $\text{update}(s_i^t, s_{ji}^t, \dots)$  (accepts as arguments the old shares  $s_i^t$ , and all the refresh data party  $i$  received). PESS ( $\Pi$ ) will use altered versions of these two subroutines,  $\text{refresh}'$  and  $\text{update}'$ .

The difference is that:

1. *refresh'* and *update'* take as input the so-called pseudo-shares (defined below) instead of the shares, but essentially do the same computation.
2. Our altered subroutines will do their computation locally (that is, by using only the fixed length register to store valuable data) bit by bit, i.e. for any bit number  $\alpha \in \{0 \dots m\}$ , *refresh'* <sub>$\alpha$</sub>  will compute the  $\alpha$ 's bit of the refresh data to be sent by party  $i$  to party  $j$ , according to  $\Pi$ , and likewise *update'* <sub>$\alpha$</sub> . By conditions 3 and 4, this is possible.

Our protocol calls subroutines named  $gen_{\Pi}^1()$ ,  $gen_{\Pi}^2()$  and  $gen_{\Pi}^3()$  in order to generate the share parts. The subroutines are called in the dealing and refreshing phases. Detailed code for  $gen_{\Pi}^1()$ ,  $gen_{\Pi}^2()$  and  $gen_{\Pi}^3()$  can be found in algorithms 1, 2 and 3, respectively, in subsection 5.2).

We are finally ready to describe the transformation of  $\Pi$ .

#### 4.1.1 PESS ( $\Pi$ )

**Dealing** In original scheme  $\Pi$ , the dealer privately sends each party  $i$  its share  $s_i^1$ . In the modified scheme PESS ( $\Pi$ ), for every share  $s_i^1 \in \{0, 1\}^n$ , the dealer will generate random  $k_i^1 \in \{0, 1\}^m$  and  $R_i^1 \in \{0, 1\}^{n \times m}$ , such that  $s_i^1 = R_i^1 k_i^1$  using subroutine  $gen_{\Pi}^1(s_i^1)$ . She will then privately send  $(k_i^1, R_i^1)$  to the party instead. (We will be calling  $(k_i^1, R_i^1)$  a *pseudo-share*.)

**Refreshing** In  $\Pi$ , at the end of every time period  $t$ , in order to refresh their shares, party  $i$  may send party  $j$  refresh data,  $s_{ij}^t$ .

In PESS ( $\Pi$ ), the sending party  $i$  will instead generate a random matrix  $\rho_{ij}^t$  and a random vector  $\kappa_{ij}^t$ , such that  $s_{ij}^t = \rho_{ij}^t \kappa_{ij}^t$ , using subroutine  $gen_{\Pi}^2()$ . This subroutine calls the scheme-specific *refresh'* <sub>$\alpha$</sub> . To prevent confusion we will be calling  $(\kappa_{ij}^t, \rho_{ij}^t)$  a *pseudo-refresh data*.

Say party  $i$  accepts the pseudo-refresh data from all the other parties. Then party  $i$  generates a random pseudo-share  $(k_i^{t+1}, R_i^{t+1})$  subject to the product  $R_i^{t+1} k_i^{t+1} = update' \left( (R_i^t, k_i^t), \{(\rho_{ji}^t, \kappa_{ji}^t)\}_{1 \leq j \leq p} \right)$ , using subroutine  $gen_{\Pi}^3 \left( (R_i^t, k_i^t), \{(\rho_{ji}^t, \kappa_{ji}^t)\}_{1 \leq j \leq p} \right)$ . This subroutine calls the scheme-specific *update'* <sub>$\alpha$</sub> .

Finally, each party applies *perase*  $(k_i^t)$  and *perase*  $(\kappa_{ij}^t)$  for all  $i, j$ , i.e. only  $h(k_i^t)$  and  $h(\kappa_{ij}^t)$  remain from  $k_i^t$  and  $\kappa_{ij}^t$ , respectively.

**Reconstruction** Any group of  $c$  parties, which could reconstruct the secret in the original scheme, may in the modified scheme compute  $R_i^t k_i^t$ , and use it to compute the secret.

*Remark 4.1.* The size of  $m$  is a parameter. It will be chosen to be a polynomial function of  $n$  where the polynomial is related to a lower bound on the storage fraction of the partial erasure function. Say  $m = 3n^c$  for constant  $c > 1$ , where  $\phi < 1 - \frac{1}{n^{c-1}}$ .

*Remark 4.2.* The computation of the new shares is done, as mentioned above, bit by bit using a fully erasable register of constant length, without storing any full length intermediate values, as described in detail in subsection 5.2. Even though it might not be the most efficient way to compute new shares, the adversary learns no data from the computation itself.

*Remark 4.3.* If robustness against a malicious dealer or malicious parties is required, a VSS scheme may be applied to the pseudo-shares sent by the dealer, and to the refresh data sent by each of the parties at the beginning of each time period. Indeed, a VSS scheme was used to achieve robustness in [HJKY95] in this precise way. In our context, one must ensure that computations done in the VSS must be implementable using the fixed length register model. It is unclear whether the VSS scheme used by [HJKY95] can be implemented in the fixed register model. Instead, we show how to do this for the scheme defined by [BOGW88], or rather its simplified version, defined by [FM88]. We will elaborate on this in subsection 4.3.

*Remark 4.4.* We assume that the function  $h$  (to be used by party  $i$ ) is adversary chosen in the worst case manner, and may change from time period to time period. More precisely, an adversary may choose  $h$  in an arbitrary manner, differently for every participating party, and anew before each time period (i.e. prior to refreshing).

*Remark 4.5.* If PESS ( $\Pi$ ) is used in the perfect erasure model, the matrix  $R$  (or  $\rho$ ) part of any pseudo-share and pseudo-refresh data would also be erased. Note that the matrix  $R$  (and correspondingly  $\rho$ ) need not be erased. This only adds strength to our results.

**Theorem 4.6.** *Let  $c > 1$  and let  $m = \text{poly}(n) = 3n^c$  in the method above. Then PESS( $\Pi$ ) described above is  $(\phi, \tau)$ -secure as per definition 2.24, where  $\tau$  is sub-exponential in  $m$  and  $\phi$  is such that  $\phi < 1 - \frac{1}{n^{c-1}}$ .*

## 4.2 Proactive $(p, p)$ Secret Sharing with Partial Erasures

**Dealing** On input  $(s, r)$  where  $s$  is the secret of length  $n$  and  $r$  random string of length  $\text{poly}(n, p)$ :

1. Generate from  $r$  random  $(k_1^1, \dots, k_p^1) \in \{0, 1\}^m$  and  $(R_1^1, \dots, R_p^1) \in \{0, 1\}^{n \times m}$ , such that  $\bigoplus_{i=1}^p R_i^1 k_i^1 = s$ , using  $\text{gen}^1$ .
2. Send  $(k_i^1, R_i^1)$  to  $P_i$  privately.

Note: in the original scheme, the shares  $s_i$  are uniform, and therefore so are the pseudo-shares  $(k_i^1, R_i^1)$ .

**Refreshing** In between each adjacent time periods  $t$  and  $t + 1$ , each party  $P_i$  does the following, to refresh its old pseudo-share  $(k_i^t, R_i^t)$  into a new one  $(k_i^{t+1}, R_i^{t+1})$ .

1. Generate random  $(\kappa_{i1}^t, \dots, \kappa_{ip}^t) \in \{0, 1\}^m$  and  $(\rho_{i1}^t, \dots, \rho_{ip}^t) \in \{0, 1\}^{n \times m}$ , such that  $\bigoplus_{j=1}^p \rho_{ij}^t \kappa_{ij}^t = 0$ , using  $gen^2$ .
2. Privately send  $(\kappa_{ij}^t, \rho_{ij}^t)$  to party  $P_j$ , for all  $j \in [1, p]$ .
3. Privately receive  $(\kappa_{ji}^t, \rho_{ji}^t)$  from party  $P_j$ , for all  $j \in [1, p]$ .
4. Generate random  $k_i^{t+1}$  and  $R_i^{t+1}$ , such that  $R_i^{t+1} k_i^{t+1} = R_i^t k_i^t \oplus \bigoplus_{j=1}^p \rho_{ji}^t \kappa_{ji}^t$ , using  $gen^3$ .
5. Apply *perase*  $(k_i^t)$  and *perase*  $(\kappa_{ij}^t)$ , i.e.  $h(k_i^t)$  and  $h(\kappa_{ij}^t)$  remain from  $k_i^t$  and  $\kappa_{ij}^t$ , respectively.

*Remark 4.7.* Note that in  $gen^2$  and  $gen^3$  we call  $refresh'_\alpha$  and  $update'_\alpha$ , respectively, which are described in detail in subsection 5.3.

**Reconstruction** All parties pull together their current pseudo-share  $(k_i^t, R_i^t)$ , and compute the secret  $s = \bigoplus_{i=1}^p R_i^t k_i^t$ .

**Theorem 4.8.** *Let  $c > 1$  and let  $m = poly(n) = 3n^c$  in the protocol above. Then the proactive  $(p, p)$  secret sharing scheme described above is  $(\phi, \tau)$ -secure as per definition 2.24, where  $\tau$  is sub-exponential in  $n$  and  $\phi$  is such that  $\phi < 1 - \frac{1}{n^{c-1}}$ .*

### 4.3 Proactive $(c, p)$ Threshold Secret Sharing with Partial Erasures

Next, we'll modify the efficient  $(c, p)$  threshold proactive secret sharing protocol, secure against a malicious adversary, as defined in [HJKY95] to one which uses only partial erasures. According to their protocol, which enhances Shamir's secret sharing [Sha79], each party holds the value of a random  $c$  degree polynomial in  $\mathbb{Z}_q$  at the party's id, and the polynomial's value at 0 is the secret  $s$ . The shares are refreshed, by each party sending its neighbors their value of a random polynomial whose 0 coefficient is 0 (that is, their value at 0 is 0). Each party adds the received values to its current secret and deletes the old one, as well as the update values.

The scheme of [HJKY95] is robust against malicious adversary (which controls a minority of players) under the discrete log assumption. Essentially [HJKY95] uses the VSS of Feldman [Fel87] on top of their proactive secret sharing scheme.

In this thesis, instead of [Fel87] VSS, we will use the VSS scheme defined by [FM88], which does not rely on any unproved computational assumption such the intractability of discrete log (but rather the existence of secure physical channels between pairs of users). More importantly, contrary to [Fel87], the scheme of [FM88] is easy to fit into the register memory model.

As commonly done in VSS, we assume broadcast channels are available.

**Dealing** On input  $(s, r)$  where  $s$  is the secret of length  $n$  and  $r$  random string of length  $poly(n, p)$ :

1. Generate a random  $c$  degree polynomial  $f(\cdot)$  in  $\text{GF}(2^n)$  (as nothing in the original protocol limits us to  $\mathbb{Z}_q$  in particular), whose value in 0 is  $s$  (that is, the free coefficient is  $s$ ).
2. Generate from  $r$  random  $(k_1^1, \dots, k_p^1) \in \{0, 1\}^m$  and  $(R_1^1, \dots, R_p^1) \in \{0, 1\}^{n \times m}$ , such that  $f(i) = R_i^1 k_i^1$ , using  $gen^1$ .
3. Send  $(k_i^1, R_i^1)$  to  $P_i$  privately.
4. **Verifying:** For robustness, the following will be used:
  - (a) Generate  $\ell = poly(n)$  more  $c$  degree polynomials  $g^{(1)}(\cdot), \dots, g^{(\ell)}(\cdot)$ . For every  $j \in \{1 \dots \ell\}$ , generate random  $(\bar{k}_1^{1(j)}, \dots, \bar{k}_p^{1(j)}) \in \{0, 1\}^m$  and  $(\bar{R}_1^{1(j)}, \dots, \bar{R}_p^{1(j)}) \in \{0, 1\}^{n \times m}$ , such that  $g^{(j)}(i) = \bar{R}_i^{1(j)} \bar{k}_i^{1(j)}$ , using  $gen^1$ .
  - (b) For each polynomial  $g^{(j)}$ , send each pseudo-share  $(\bar{k}_i^{1(j)}, \bar{R}_i^{1(j)})$  to  $P_i$  privately.
  - (c) Each party  $i$  picks  $Q_{\lceil \frac{\ell(i-1)}{p} \rceil + 1}, \dots, Q_{\lceil \frac{\ell i}{p} \rceil} \in_R \{0, 1\}$  and broadcasts them.
  - (d) For every  $j$ , if  $Q_j = 0$ , broadcast  $g^{(j)} = g^{(j)}$ . Otherwise, broadcast  $g^{(j)} = g^{(j)} + f$ .
  - (e) Each party  $i$  will verify for every  $j$  that his values are valid (and declare if so), as defined in algorithm 8, in subsection 5.4.

Note: in the original scheme, the shares  $f(i)$  are uniform, and therefore so are the pseudo-shares  $(k_i^1, R_i^1)$ . Likewise for  $g^{(j)}(i)$ .

**Refreshing** In between each adjacent time periods  $t$  and  $t + 1$ , each party  $P_i$  does the following, to refresh its old pseudo-share  $(k_i^t, R_i^t)$  into a new one  $(k_i^{t+1}, R_i^{t+1})$ .

1. Generate random  $(\kappa_{i1}^t, \dots, \kappa_{ip}^t) \in \{0, 1\}^m$  and  $(\rho_{i1}^t, \dots, \rho_{ip}^t) \in \{0, 1\}^{n \times m}$ , such that the  $\rho_{ij}^t \kappa_{ij}^t$ s will define a  $c$  degree polynomial  $f_i^t$  whose value in 0 is 0 (that is, the free coefficient is 0), using  $gen^2$ .

2. Privately send  $(\kappa_{ij}^t, \rho_{ij}^t)$  to party  $P_j$ , for all  $j \in [1, p]$ .
3. Privately receive  $(\kappa_{ji}^t, \rho_{ji}^t)$  from party  $P_j$ , for all  $j \in [1, p]$ .
4. **Verifying:** For robustness, follow practically the same method used above by the dealer, in order to allow validation of the refresh data:
  - (a) For every  $l \in \{1 \dots \ell\}$ , generate random  $(\bar{\kappa}_{i1}^{t(l)}, \dots, \bar{\kappa}_{ip}^{t(l)}) \in \{0, 1\}^m$  and  $(\bar{\rho}_{i1}^{t(l)}, \dots, \bar{\rho}_{ip}^{t(l)}) \in \{0, 1\}^{n \times m}$ , such that the  $\bar{\rho}_{ij}^{t(l)} \bar{\kappa}_{ij}^{t(l)}$ s will define a  $c$  degree polynomial  $g_i^{t(l)}$  with free coefficient 0, using  $gen^2$ .
  - (b) For every  $l \in \{1 \dots \ell\}$ , privately send  $(\bar{\kappa}_{ij}^{t(l)}, \bar{\rho}_{ij}^{t(l)})$  to party  $P_j$ , for all  $j \in [1, p]$ .
  - (c) Each party  $j$  picks  $Q_{\lceil \frac{\ell(j-1)}{p} \rceil + 1}, \dots, Q_{\lceil \frac{\ell j}{p} \rceil} \in_R \{0, 1\}$  and broadcasts them.
  - (d) For every  $l$ , if  $Q_l = 0$ , calculate and broadcast  $g_i^{nt(l)} = g_i^{t(l)}$ . Otherwise, calculate and broadcast  $g_i^{nt(l)} = g_i^{t(l)} + f_i^t$ .
  - (e) Each party  $j$  will verify for every  $l$  that  $g_i^{nt(l)}(0) = 0$  (in main memory), and that his values are valid (and declare if so), as defined in algorithm 8, in subsection 5.4.
5. Generate random  $k_i^{t+1}$  and  $R_i^{t+1}$ , such that  $R_i^{t+1} k_i^{t+1} = R_i^t k_i^t + \sum_{j=1}^p \rho_{ji}^t \kappa_{ji}^t$ , using  $gen^3$ .
6. Apply *perase* ( $k_i^t$ ) and *perase* ( $\kappa_{ij}^t$ ) for all  $i, j$ .

*Remark 4.9.* Note that in  $gen^2$  and  $gen^3$  we call *refresh'* $_{\alpha}$  and *update'* $_{\alpha}$ , respectively, which are described in detail in subsection 5.4.

**Reconstruction** Any  $c$  of the parties pull together their current pseudo-share  $(k_i^t, R_i^t)$ , and compute the polynomial and by that the secret.

**Theorem 4.10.** *Let  $c > 1$ , and let  $m = poly(n) = 3n^c$  and  $\ell = poly(n)$  in the protocol above. Then the proactive  $(c, p)$  secret sharing scheme described above is robust and  $(\phi, \tau)$ -secure as per definition 2.24, where  $\tau$  is sub-exponential in  $n$  and  $\phi$  is such that  $\phi < 1 - \frac{1}{n^{c-1}}$ .*

## 4.4 Proofs

We will first define some notation and lemmas, which will later on aid us in our proofs.

**Definition 4.11** (Partially Erased Tuple). Let  $c > 1$  and let  $m = poly(n) = 3n^c$ . Let  $k \in \{0, 1\}^m$  and let  $R \in \{0, 1\}^{n \times m}$ . We say that  $(k, R)$  is a partially erased tuple, if (1) a partial erasure function  $h$  was applied on  $k$  (where  $h$  is defined as in subsection 2.3, with

storage fraction  $\phi$  where  $\phi < 1 - \frac{1}{ne-1}$ ; and (2)  $R$  and  $k$  are random, up-to the product  $Rk$ .<sup>5</sup> That is, the partially erased tuple  $(k, R)$  consists of  $(h(k) \in \{0, 1\}^{\phi \cdot m}, R)$ .

**Lemma 4.12.** *Let  $s \in \{0, 1\}^n$  such that  $d(s) = 0$ . Let  $(k, R)$  be a partially erased tuple such that  $k \in \{0, 1\}^m$ ,  $R \in \{0, 1\}^{n \times m}$  and  $h(k) \in \{0, 1\}^{\phi \cdot m}$ . Let  $f$  be a public 1-to-1 and onto function, such that  $s = f(Rk)$ . Then:*

$$d(s \mid (h(k), R)) \leq 2^{-(1-\phi) \cdot \frac{m}{2}} + 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)}$$

*Proof.* Lets see what information is available on  $Rk$ , given  $R$  and  $h(k)$ . Let  $y = h(k)$ , and  $D$  the distribution of  $y$ . As  $|h(k)| \leq \phi \cdot m$  and  $k \in_R \{0, 1\}^m$ , by counting argument we get that  $\Pr_k [ |h^{-1}(h(k))| < 2^{(1-\phi) \cdot \frac{m}{2}} ] < 2^{-(1-\phi) \cdot \frac{m}{2}}$ , or:

$$\Pr_{y \sim D} [ |\{k \mid h(k) = y\}| < 2^{(1-\phi) \cdot \frac{m}{2}} \mid y ] < 2^{-(1-\phi) \cdot \frac{m}{2}}$$

Let  $\mathcal{A}$  denote the set of  $y$ 's for which  $|\{k \mid h(k) = y\}| < 2^{(1-\phi) \cdot \frac{m}{2}}$ .

By definition, for all  $h(k) \notin \mathcal{A}$  we get that the Rényi entropy is  $H_{\mathfrak{R}}(k \mid h(k)) \geq (1-\phi) \cdot \frac{m}{2}$ . As the random  $R$  is unrelated to the definition of  $h$  which is applied on  $k$ ,  $R$  may be seen as a universal hash function. Thus, the leftover hash lemma [HILL99, ILL89] may be used here: Our hash function is  $\{0, 1\}^{n \times m} \times \{0, 1\}^m \rightarrow \{0, 1\}^n$ , so for lemma 2.22, we have  $2e = (1-\phi) \cdot \frac{m}{2} - n$ , and from the application of the lemma, we get that for all  $h(k) \notin \mathcal{A}$ ,  $d(Rk \mid h(k), R) \leq 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)}$ .

Thus, we get that:

$$\begin{aligned} d(Rk \mid h(k), R) &= d(Rk \mid h(k) \in \mathcal{A}, R) \Pr[h(k) \in \mathcal{A}] \\ &\quad + d(Rk \mid h(k) \notin \mathcal{A}, R) \Pr[h(k) \notin \mathcal{A}] \\ &\leq \Pr[h(k) \in \mathcal{A}] + d(Rk \mid h(k) \notin \mathcal{A}, R) \\ &\leq 2^{-(1-\phi) \cdot \frac{m}{2}} + 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)} \end{aligned}$$

Since  $f$  is 1-to-1 and onto, any information we have of  $Rk$  we clearly have of  $s$  and viceversa. Therefore, also:

$$d(s \mid h(k), R) \leq 2^{-(1-\phi) \cdot \frac{m}{2}} + 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)}$$

□

---

<sup>5</sup>Namely, let  $v = Rk$ . Then  $(k, R)$  are chosen at random from  $\{(k', R') \text{ s.t. } v = R'k' \mid k' \in \{0, 1\}^m, R' \in \{0, 1\}^{n \times m}\}$ .

**Lemma 4.13.** *Let  $m = 3n^c$ . Let  $s \in \{0, 1\}^n$  such that  $d(s) = 0$ . Let  $\ell$  be a sub-exponential number in  $m$ . For each  $i \in \{1, \dots, \ell\}$ , let  $(k_i, R_i)$  be a partially erased tuple such that  $k_i \in \{0, 1\}^m$ ,  $R_i \in \{0, 1\}^{n \times m}$  and  $h(k_i) \in \{0, 1\}^{\phi m}$ , and let  $f_i$  be a public 1-to-1 and onto function, such that  $s = f_i(R_i k_i)$ . Then:*

$$d(s \mid (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) < \text{negli}(m)$$

*Proof.* Since we know from lemma 4.12 that  $d(s \mid (h(k_i), R_i)) \leq 2^{-(1-\phi) \cdot \frac{m}{2}} + 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)} < \frac{1}{2}$ , we get according to theorem 2.17:

$$H(s \mid (h(k_i), R_i)) \geq n - \left( -2^{-(1-\phi) \cdot \frac{m}{2}} - 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)} \right) \log \left( 2^{-(1-\phi) \cdot \frac{m}{2} - n} + 2^{-((1-\phi) \cdot \frac{m}{4} + \frac{n}{2} + 1)} \right)$$

For simplicity, we will define:

$$\epsilon(m) = \left( -2^{-(1-\phi) \cdot \frac{m}{2}} - 2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)} \right) \log \left( 2^{-(1-\phi) \cdot \frac{m}{2} - n} + 2^{-((1-\phi) \cdot \frac{m}{4} + \frac{n}{2} + 1)} \right)$$

Taking  $\phi < 1 - \frac{1}{n^{c-1}}$  and as  $m = 3n^c$ , we get that  $\phi < 1 - \frac{3n}{m}$  and  $(1 - \phi) \frac{m}{6} > \frac{n}{2}$ , and  $2^{-((1-\phi) \cdot \frac{m}{4} - \frac{n}{2} + 1)} < 2^{-(1-\phi) \cdot \frac{m}{12}}$ . Therefore,  $\epsilon(m)$  is positive and negligible in  $m$ , and we get that:

$$H(s \mid (h(k_i), R_i)) \geq n - \epsilon(m)$$

Thus, the mutual information of  $s$  and  $(h(k_i), R_i)$  is:

$$I(s : (h(k_i), R_i)) = H(s) - H(s \mid (h(k_i), R_i)) \leq n - (n - \epsilon(m)) = \epsilon(m) \quad (3)$$

By applying theorem 2.13 (the chain rule), we get that:

$$I(s : (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) = \sum_{i=1}^{\ell} I(s : (h(k_i), R_i) \mid (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) \quad (4)$$

We wish to bound  $I(s : (h(k_1), R_1), \dots, (h(k_\ell), R_\ell))$ , so we need to bound each term in the above summation  $I(s : (h(k_i), R_i) \mid (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1}))$ .

According to lemma 2.12,  $I(Y : X) - I(Y : X \mid Z) = I(X : Z) - I(X : Z \mid Y)$ , so if we define  $Y = s$ ,  $X = (h(k_i), R_i)$  and  $Z = (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})$  we get that:

$$I(s : (h(k_i), R_i)) - I(s : (h(k_i), R_i) \mid (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) = \quad (5)$$

$$I((h(k_i), R_i) : (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) - \quad (6)$$

$$I((h(k_i), R_i) : (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1}) \mid s) \quad (7)$$

Since we chose  $(k_i, R_i)$  random up-to the product, the only relevant data in  $(h(k_i), R_i)$  is the product  $R_i k_i$  - any thing else in  $(h(k_i), R_i)$  is random, and unrelated to any data in any other tuple. Since we have a 1-to-1 and onto function  $f_i$  such that  $s = f_i(R_i k_i)$ , knowing  $s$  we also know the product  $R_i k_i$ . Thus, by definition of mutual information, given  $s$ , the tuples  $(h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})$  have no mutual information with  $(h(k_i), R_i)$ , and we have that:

$$I((h(k_i), R_i) : (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1}) \mid s) = 0$$

So, by 5 we get that:

$$\begin{aligned} I(s : (h(k_i), R_i)) - I(s : (h(k_i), R_i) \mid (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) &= \\ I((h(k_i), R_i) : (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) &\geq 0 \end{aligned}$$

And therefore:

$$I(s : (h(k_i), R_i) \mid (h(k_1), R_1), \dots, (h(k_{i-1}), R_{i-1})) \leq I(s : (h(k_i), R_i))$$

Therefore, according to 4 and 3:

$$I(s : (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) \leq \sum_{i=1}^{\ell} I(s : (h(k_i), R_i)) \leq \ell \epsilon(m)$$

And by definition of mutual information:

$$H(s \mid (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) = H(s) - I(s : (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) \geq n - \ell \epsilon(m)$$

Therefore, by lemma 2.16, we get that:

$$d((s \mid (h(k_1), R_1), \dots, (h(k_\ell), R_\ell)) \leq \sqrt{2 \ln 2 \ell \epsilon(m)} < \text{negli}(m)$$

□

**Definition 4.14** (View Extension). Let  $\Pi$  be a proactive  $(c, p)$  threshold secret sharing scheme, running for  $\tau$  time periods. Let  $E$  be a mobile adversary. Let  $VIEW_E^\tau$  be the view of  $E$  after the  $\tau$  time periods in the perfect erasure model, and let  $\widehat{VIEW}_E^\tau$  be the view of  $E$  in the partial erasure model. We will define the view extension  $\widetilde{VIEW}_E^\tau$  as the additional information the adversary sees in the partial erasure model. That is,  $\widetilde{VIEW}_E^\tau = \widehat{VIEW}_E^\tau \setminus VIEW_E^\tau$ .

Clearly, for the modification method defined in subsection 4.1, and for the schemes defined

in subsections 4.2 and 4.3,  $\widetilde{VIEW}_E^T$  will include only partially erased tuples.

Now we can finally go on to proving the theorems:

*Proof of Theorem 4.6.* Correctness is easy to see, since by construction, the product of any pair  $(R_i^t, k_i^t)$  is the share in  $\Pi$ , and the product of any pair  $(\rho_i^t, \kappa_i^t)$  is the original refresh data in  $\Pi$ .

As for privacy, let  $E$  be a mobile adversary, as defined in subsection 2.1.

According to the definition of  $\Pi$ , and thus also for PESS ( $\Pi$ ),  $E$  may corrupt at most  $c-1$  parties at any time period. Without loss of generality, consider the case where  $E$  corrupts exactly  $c-1$  parties in each time period, since by corrupting less she can only gain less information.

By definition, on the perfect erasure model, the adversary  $E$  has no information about the secret  $s$  ( $d(s | VIEW_E^T) = 0$ ). Lets see what information  $E$  may gain from the partial erasures - what is  $d(s | \widetilde{VIEW}_E^T)$ : There are two types of information, which is perfectly erased in the perfect erasure model, and only partially erased here: pseudo-shares and pseudo-refresh data.

Pseudo-shares:

At any time period  $t$ , the adversary  $E$  has exactly  $c-1$  pseudo-shares and misses  $p-c+1$  pseudo-shares, which are then partially erased. The partially erased pseudo-shares are generated by  $gen_{\Pi}^1()$  and  $gen_{\Pi}^3()$  as random up to the product of the pseudo-share parts. Therefore, according to definition 4.11, these are partially erased tuples. Furthermore, by definition of a threshold secret sharing scheme, for any of these pseudo-shares  $(k_i^t, R_i^t)$ , if  $E$  knew the product  $R_i^t k_i^t$ , she would be able to derive  $s$ . That is, she knows a function  $f_i^t$ , such that  $s = f_i^t(R_i^t k_i^t)$ . Thus we have that  $d(s) = 0$ ,  $d(s | R_i^t k_i^t) = 1$  and  $s$  and  $R_i^t k_i^t$  are both in  $\{0,1\}^n$ . Therefore,  $f_i^t$  must be 1-to-1 and onto, for if 2 possible values of  $R_i^t k_i^t$  would give the same  $s$ , there would be some value of  $s$  which isn't in image of the known  $f$ , contrary to  $d(s) = 0$ . Thus, at any time period  $t$ ,  $E$  has  $p-c+1$  partially erased tuples for each of which she knows a 1-to-1 and onto function which maps the product of the tuple to  $s$ .

Pseudo-refresh data:

Between any pair of adjacent time periods  $t$  and  $t+1$ ,  $E$  may switch between parties. We will define the following subgroups of the  $p$  parties:

$S_r$  is the group of parties in which  $E$  remains - these parties are corrupted in both time periods.

$S_l$  is the group of parties which  $E$  leaves - these parties are corrupted in time period  $t$ , and not in time period  $t+1$ .

$S_a$  is the group of parties to which  $E$  - these parties are not corrupted in time period  $t$ , yet they are corrupted in time period  $t+1$ . Clearly,  $|S_a| = |S_l|$ .

$S_o$  is the group of other parties - these parties are corrupted in neither time period.

If  $E$  stays in the same  $c - 1$  parties,  $|S_r| = c - 1$  and  $S_l = S_a = \emptyset$ . If  $E$  switches all the parties,  $S_r = \emptyset$  and  $|S_a| = |S_l| = c - 1$

Lets look at the pseudo-refresh data used by each of the sets:

$S_r$ :  $E$  has the pseudo-refresh data either sent or received by any of the parties in this set between the two time periods, so clearly the partial erasure of them may add no new information.

$S_l, S_a, S_o$ : Pseudo-refresh data which were both sent by and received by parties in these sets, are not available to  $E$  in the perfect erasure model, so  $E$  may possibly gain information from them when they are only partially erased. The partially erased pseudo-refresh data is generated by  $gen_{\Pi}^2()$  as random up to the product of the data parts. Therefore, according to definition 4.11, these are partially erased tuples.

$E$  may gain information from the partially erased pseudo-refresh data. We will over estimate the adversary knowledge, by assuming that not only can she deduce  $s$  from knowledge of the products of a cluster of tuples, but can deduce  $s$  from the product of each of these tuples alone. That is, for each tuple  $(\kappa_{ji}^t, \rho_{ji}^t)$ , the adversary knows a function  $f_{ji}^t$  such that  $s = f_{ji}^t(\rho_{ji}^t \kappa_{ji}^t)$ . As before, since  $s$  and  $\rho_{ji}^t \kappa_{ji}^t$  are both in  $\{0, 1\}^n$ , we have that  $f_{ji}^t$  must be 1-to-1 and onto.

Thus, after any time period  $t$ ,  $E$  learns at most  $p^2$  partially erased tuples of pseudo-refresh data, for each of which she knows a 1-to-1 and onto function which maps the product of the tuple to  $s$ .

To conclude, all the adversary  $E$  may learn about the secret  $s$  is at most  $\tau(p^2 + p - c + 1)$  partially erased tuples, from each of which she knows a 1-to-1 and onto function to  $s$ , so according to lemma 4.13 (to which we pass  $s \mid VIEW_E^\tau$  instead of  $s$ ), given that  $\tau$  is sub-exponential in  $m$ , the amount of information that the adversary may learn about the secret  $s$  is negligible:

$$d\left(s \mid \widehat{VIEW}_E^\tau\right) = d\left(s \mid VIEW_E^\tau, \{(h(\kappa_i^t), R_i^t)\}, \{(h(\kappa_{ij}^t), \rho_{ij}^t)\}\right) \leq \text{negli}(m)$$

Thus PESS(II) described in subsection 4.1 is  $(\phi, \tau)$ -secure as per definition 2.24.  $\square$

*Proof of Theorem 4.8.* This is a direct application of the modification method on the following scheme, so all that remains is to prove the correctness and privacy of the following scheme in the perfect erasure model:

**Dealing** On input  $(s, r)$  where  $s$  is the secret of length  $n$  and  $r$  random string of length  $\text{poly}(n, p)$ :

1. Generate  $s_1^1, \dots, s_{p-1}^1 \in_R \{0, 1\}^n$ , and set  $s_p^1 = s \oplus \bigoplus_{i=1}^{p-1} s_i^1$  (so  $s = \bigoplus_{i=1}^p s_i^1$  and all the shares are random).
2. Send  $s_i^1$  to  $P_i$  privately.

**Refreshing** In between each adjacent time periods  $t$  and  $t + 1$ , each party  $P_i$  does the following, to refresh its old share  $s_i^t$  into a new one  $s_i^{t+1}$ .

1. Generate  $s_{i1}^t, \dots, s_{i,p-1}^t \in_R \{0, 1\}^n$ , and set  $s_{ip}^t = \bigoplus_{j=1}^{p-1} s_{ij}^t$  (so  $\bigoplus_{j=1}^p s_{ij}^t = 0$ , and the refresh data are random).
2. Privately send  $s_{ij}^t$  to party  $P_j$ , for all  $j \in [1, p]$ ; Privately receive  $s_{ji}^t$  from party  $P_j$ , for all  $j \in [1, p]$ .
3. Compute  $s_i^{t+1} = s_i^t \oplus \bigoplus_{j=1}^p s_{ji}^t$ .
4. Perfectly erase  $s_i^t$  and all the  $s_{ij}^t$  and  $s_{ji}^t$ .

**Reconstruction** All parties pull together their current share  $s_i^t$ , and compute the secret

$$s = \bigoplus_{i=1}^p s_i^t.$$

**Correctness:**

Clearly,  $s = \bigoplus_{i=1}^p s_i^1$ . Assume now that  $s = \bigoplus_{i=1}^p s_i^t$ . Then:

$$\begin{aligned} \bigoplus_{i=1}^p s_i^{t+1} &= \bigoplus_{i=1}^p \left( s_i^t \oplus \bigoplus_{j=1}^p s_{ji}^t \right) \\ &= \left( \bigoplus_{i=1}^p s_i^t \right) \oplus \bigoplus_{j=1}^p \bigoplus_{i=1}^p s_{ji}^t \\ &= s \oplus \bigoplus_{j=1}^p 0 \\ &= s \end{aligned}$$

Therefore, by induction, for any  $t$   $s = \bigoplus_{i=1}^p s_i^t$ .

**Privacy:**

Clearly, at time period 1, privacy is maintained. Assume now that it is maintained until time period  $t$ . Assume also that on time period  $t$ , the adversary corrupt the maximum  $p - 1$  parties (as she clearly cannot learn more by corrupting less), WLOG  $P_1, \dots, P_{p-1}$ . Then, between time periods  $t$  and  $t + 1$ , the adversary can either remain in the same  $p - 1$  parties, or she can switch.

If she remains, the adversary can easily learn all the refresh data being sent between these

two time periods (even the data  $P_p$  sends to itself, as  $s_{pp}^t = \bigoplus_{j=1}^{p-1} s_{pj}^t$ ). However, as the refresh data is XORed with  $s_p^t$ , which she did not have, she still does not have  $s_p^{t+1}$ , and privacy is maintained.

If she switched, WLOG to  $P_2, \dots, P_p$ , then she has  $s_p^{t+1}$ . However, she does not have  $s_1^{t+1}$ .  $s_1^{t+1} = s_1^t \oplus \bigoplus_{j=1}^p s_{j1}^t$ , but since she does not have  $s_{11}^t$  and  $s_{p1}^t$ , she cannot learn anything about it. Likewise,  $s_p^t = s_p^{t+1} \oplus \bigoplus_{j=1}^p s_{jp}^t$ , but since she does not have  $s_{1p}^t$  and  $s_{pp}^t$ , she cannot learn anything about it. She does have  $s_{1p}^t \oplus s_{11}^t$  and  $s_{p1}^t \oplus s_{pp}^t$ , however it simply doesn't help her, as for any random  $s$ , there are equally probable possible values of  $s_{11}^t$ ,  $s_{1p}^t$ ,  $s_{p1}^t$  and  $s_{pp}^t$ .

Therefore, by induction, privacy is maintained for any time period  $t$ .  $\square$

*Proof of Theorem 4.10. Correctness:*

Let  $\Pi$  be the algorithm defined by [HJKY95]. By construction, the product of any pair  $(R_i^t, k_i^t)$  is the original share in  $\Pi$ , and the product of any pair  $(\rho_i^t, \kappa_i^t)$  is the original refresh data in  $\Pi$ .

Resistance against malicious adversaries has been proven in [FM88], and as the verification polynomials of the refresh data have free coefficient 0, receiving parties can also easily verify the value of the polynomial in 0 is 0 with probability  $1 - 2^{-\ell}$ .

### Privacy:

Let  $E$  be a mobile adversary, as defined in subsection 2.1.  $E$  may corrupt at most  $c - 1$  parties at any time period. Without loss of generality, consider the case where  $E$  corrupts exactly  $c - 1$  parties in each time period, since by corrupting less she can only gain less information.

By [FM88, HJKY95], privacy is being maintained in the perfect erasure model, and the adversary  $E$  has no information about the secret  $s$  ( $d(s | VIEW_E^T) = 0$ ). Lets see what information she may learn from the partial erasures - what is  $d(s | \widehat{VIEW}_E^T)$ : There are three types of information, which is perfectly erased in the perfect erasure model, and only partially erased here: pseudo-shares, pseudo-refresh data and verification data.

### Pseudo-shares:

At any time period  $t$ , the adversary  $E$  misses  $p - c + 1$  pseudo-shares, which are then partially erased. The pseudo-shares are random up to the product of the pseudo-share parts, and therefore are partially erased tuples, according to definition 4.11. Furthermore, by definition of a threshold secret sharing scheme, for any of these pseudo-shares  $(k_i^t, R_i^t)$ , if  $E$  knew the product  $R_i^t k_i^t$ , she would be able to derive  $s$ . That is, she knows a function  $f_i^t$ , such that  $s = f_i^t(R_i^t k_i^t)$ . Thus we have that  $d(s) = 0$ ,  $d(s | R_i^t k_i^t) = 1$  and  $s$  and  $R_i^t k_i^t$  are both in  $\{0, 1\}^n$ . Therefore,  $f_i^t$  must be 1-to-1 and onto, for if 2 possible values of  $R_i^t k_i^t$  would give the same  $s$ , there would be some value of  $s$  which isn't in image of the known  $f$ , contrary

to  $d(s) = 0$ . Thus, at any time period  $t$ ,  $E$  has  $p - c + 1$  partially erased tuples for each of which she knows a 1-to-1 and onto function which maps the product of the tuple to  $s$ .

Pseudo-refresh data:

Between any pair of adjacent time periods  $t$  and  $t + 1$ ,  $E$  may switch between parties. Let  $S_r$  be the group of parties in which  $E$  remains - these parties are corrupted in both time periods.  $E$  has the pseudo-refresh data either sent or received by any of the parties in  $S_r$  between the two time periods, so clearly the partial erasure of them may add no new information. Pseudo-refresh data which were both sent by and received by parties not in  $S_r$  are not available to  $E$  in the perfect erasure model, so  $E$  may possibly gain information from them when they are only partially erased. The pseudo-refresh data is random up to the product of the pseudo-refresh data parts, and therefore are partially erased tuples, according to definition 4.11.

$E$  may gain information from the partially erased pseudo-refresh data. We will over estimate the adversary knowledge, by assuming that not only can she deduce  $s$  from knowledge of the products of a cluster of tuples, but can deduce  $s$  from the product of each of these tuples alone. That is, for each tuple  $(\kappa_{ji}^t, \rho_{ji}^t)$ , the adversary knows a function  $f_{ji}^t$  such that  $s = f_{ji}^t(\rho_{ji}^t \kappa_{ji}^t)$ . As before, since  $s$  and  $\rho_{ji}^t \kappa_{ji}^t$  are both in  $\{0, 1\}^n$ , we have that  $f_{ji}^t$  must be 1-to-1 and onto.

Thus, after any time period  $t$ ,  $E$  learns at most  $p^2$  partially erased tuples of pseudo-refresh data, for each of which she knows a 1-to-1 and onto function which maps the product of the tuple to  $s$ .

Verification data:

The VSS may give the adversary  $E$  more data, in the form of partially erased verification tuples  $(\bar{\kappa}_{ij}^{t(l)}, \bar{\rho}_{ij}^{t(l)})$ , sent in step 4e (of the refresh phase, or step 4e in the dealing phase). If  $g_i^{t(l)} = g_i^{t(l)} + f_i^t$  was broadcasted in step 4d, the adversary knows a 1-to-1 and onto function  $f_{ij}^{t(l)}$  such that  $s = f_{ij}^{t(l)}(\bar{\rho}_{ij}^{t(l)} \bar{\kappa}_{ij}^{t(l)})$ . On the worst case, these functions exist for all the verification tuples. Thus, at any time period  $t$ ,  $E$  has at most  $\ell p$  partially erased tuples of verification data for each of which she knows a 1-to-1 and onto function which maps the product of the tuple to  $s$ .

To conclude, all the adversary  $E$  may learn about the secret  $s$  is at most  $\tau(\ell p + p^2 + p - c + 1)$  partially erased tuples, from each of which she knows a 1-to-1 and onto function to  $s$ , so according to lemma 4.13 (to which we pass  $s \mid \text{VIEW}_E^\tau$  instead of  $s$ ), given that  $\tau$  is sub-exponential in  $m$ , the amount of information that the adversary may learn about the secret  $s$  is negligible:

$$d\left(s \mid \widehat{\text{VIEW}}_E^\tau\right) = d\left(s \mid \text{VIEW}_E^\tau, \left\{ (h(k_i^t), R_i^t) \right\}, \left\{ (h(\kappa_{ij}^t), \rho_{ij}^t) \right\}, \left\{ \left( h\left(\bar{\kappa}_{ij}^{t(l)}\right), \bar{\rho}_{ij}^{t(l)} \right) \right\} \right) \leq \text{negli}(m)$$

Thus the scheme described in subsection 4.3 is  $(\phi, \tau)$ -secure as per definition 2.24.  $\square$

## 5 Algorithms

These algorithms are used to prove the possibility of calculating as required in section 4, under the register model defined in subsection 2.4

### 5.1 Notation

The following algorithms will follow this notation:

1. The braces  $\{ \}$  signify a comment
2.  $Reg$  is the fixed length register. all other variables are in the main memory.
3.  $Var[\alpha]$  in the code signify the alpha's bit of the variable  $Var$ . For instance,  $Reg[0]$  is the first cell of the register. For a matrix such two parameters are used, for instance  $R_i^1[\alpha, \beta]$ .
4.  $[Val]_\alpha$  in the comments signify the alpha's coordinate (bit) of the computable value  $Val$ . A different notation is used, since the whole value isn't being computed but only the specified bit.

## 5.2 Algorithms Used by the Modification Method

---

**Algorithm 1**  $gen_{\Pi}^1(s_i^1)$

---

$k_i^1 \leftarrow_R \{0, 1\}^m$

**for**  $\alpha \leftarrow 1, \dots, n$  **do**

$Reg[0] \leftarrow 0$

**for**  $\beta \leftarrow 1, \dots, m$  **do**

**if**  $k_i^1[\beta] = 1$  and  $\forall l \in \{\beta + 1, \dots, m\}. k_i^1[l] = 0$  **then**

$Reg[0] \leftarrow Reg[0] \oplus s_i^1[\alpha]$

$R_i^1[\alpha, \beta] \leftarrow Reg[0]$

**else**

$R_i^1[\alpha, \beta] \leftarrow_R \{0, 1\}$

**if**  $k_i^1[\beta] = 1$  **then**

$Reg[0] \leftarrow Reg[0] \oplus R_i^1[\alpha, \beta]$

**end if**

**end if**

**end for**

**end for**

**return**  $k_i^1, R_i^1$

---

---

**Algorithm 2**  $gen_{\Pi}^2()$ 

---

```
for all  $j$  in  $i$ 's target do  
   $\kappa_{ij}^t \leftarrow_R \{0, 1\}^m$   
  if  $j$  is in the first  $l - 1$  targeted parties then  
     $\rho_{ij}^t \leftarrow_R \{0, 1\}^{n \times m}$   
  else  
    for  $\alpha \leftarrow 1, \dots, n$  do  
       $Reg[0] \leftarrow 0$   
      for  $\beta \leftarrow 1, \dots, m$  do  
        if  $\kappa_{ij}^t[\beta] = 1$  and  $\forall l \in \{\beta + 1, \dots, m\}. \kappa_{ij}^t[l] = 0$  then  
           $Reg[0] \leftarrow Reg[0] \oplus refresh'_\alpha(i, j, \{\rho_{ij'}^t, \kappa_{ij'}^t\}_{1 \leq j' < j})$   
           $\rho_{ij}^t[\alpha, \beta] \leftarrow Reg[0]$   
        else  
           $\rho_{ij}^t[\alpha, \beta] \leftarrow_R \{0, 1\}$   
          if  $\kappa_{ij}^t[\beta] = 1$  then  
             $Reg[0] \leftarrow Reg[0] \oplus \rho_{ij}^t[\alpha, \beta]$   
          end if  
        end if  
      end for  
    end for  
  end if  
end for  
return  $\{\rho_{ij}^t, \kappa_{ij}^t\}_{1 \leq j \leq p}$ 
```

---

---

**Algorithm 3**  $gen_{\Pi}^3 \left( (R_i^t, k_i^t), \{(\rho_{ji}^t, \kappa_{ji}^t)\}_{1 \leq j \leq p} \right)$ 

---

 $k_i^{t+1} \leftarrow_R \{0, 1\}^m$ **for**  $\alpha \leftarrow 1, \dots, n$  **do** $Reg[0] \leftarrow 0$ **for**  $\beta \leftarrow 1, \dots, m$  **do****if**  $k_i^{t+1}[\beta] = 1$  and  $\forall l \in \{\beta + 1, \dots, m\}. k_i^{t+1}[l] = 0$  **then** $Reg[0] \leftarrow Reg[0] \oplus update'_{\alpha} \left( (R_i^t, k_i^t), \{(\rho_{ji}^t, \kappa_{ji}^t)\}_{1 \leq j \leq p} \right)$  $R_i^{t+1}[\alpha, \beta] \leftarrow Reg[0]$ **else** $R_i^{t+1}[\alpha, \beta] \leftarrow_R \{0, 1\}$ **if**  $k_i^{t+1}[\beta] = 1$  **then** $Reg[0] \leftarrow Reg[0] \oplus R_i^{t+1}[\alpha, \beta]$ **end if****end if****end for****end for****return**  $k_i^{t+1}, R_i^{t+1}$ 

---

### 5.3 Algorithms for the Proactive $(p, p)$ Secret Sharing Scheme with Partial Erasures

---

**Algorithm 4**  $refresh'_\alpha$

---

```

 $Reg[1] \leftarrow 0$ 
if  $j \neq p$  then
     $Reg[1] \leftarrow_R \{0, 1\}$ 
else
    for  $j' \leftarrow 1, \dots, p-1$  do
        {Add  $[\rho_{ij'}^t \kappa_{ij'}^t]_\alpha$  to the register}
        for  $\beta \leftarrow 1, \dots, m$  do
            if  $\rho_{ij'}^t[\alpha, \beta] = 1$  and  $\kappa_{ij'}^t[\beta] = 1$  then
                 $Reg[1] \leftarrow Reg[1] \oplus 1$ 
            end if
        end for
    end for
end if
return  $Reg[1]$ 

```

---



---

**Algorithm 5**  $update'_\alpha$

---

```

 $Reg[1] \leftarrow 0$ 
{Calculate  $[R_i^t k_i^t]_\alpha$  into the register}
for  $\beta \leftarrow 1, \dots, m$  do
    if  $R_i^t[\alpha, \beta] = 1$  and  $k_i^t[\beta] = 1$  then
         $Reg[1] \leftarrow Reg[1] \oplus 1$ 
    end if
end for
for  $j \leftarrow 1, \dots, p$  do
    {Add  $[\rho_{ji}^t \kappa_{ji}^t]_\alpha$  to the register}
    for  $\beta \leftarrow 1, \dots, m$  do
        if  $\rho_{ji}^t[\alpha, \beta] = 1$  and  $\kappa_{ji}^t[\beta] = 1$  then
             $Reg[1] \leftarrow Reg[1] \oplus 1$ 
        end if
    end for
end for
return  $Reg[1]$ 

```

---

## 5.4 Algorithms for the Proactive $(c, p)$ Threshold Secret Sharing Scheme with Partial Erasures

---

**Algorithm 6** *refresh'* <sub>$\alpha$</sub>

---

$Reg[1] \leftarrow 0$

**if**  $j < c$  **then**

$Reg[1] \leftarrow_R \{0, 1\}$

**else**

    {According to Lagrange interpolation  $s_{ij}^t = \sum_{j'=1}^{c-1} s_{ij'}^t \cdot L_{j'}(j)$ , where  $L_{j'}(j) = \prod_{l \neq j'} \frac{j-l}{j'-l}$ }

$L_{j'}^\gamma(j) \leftarrow 2^\gamma \prod_{l \neq j'} \frac{j-l}{j'-l}$  {These are constants with no relation to the secrets, so they can be computed and stored in main memory, without revealing anything to the adversary}

**for**  $j' \leftarrow 1, \dots, c-1$  **do**

    {Add  $[s_{ij'}^t \cdot L_{j'}(j)]_\alpha$  to the register by going bit by bit of  $s_{ij'}^t$ }

**for**  $\gamma \leftarrow 1, \dots, m$  **do**

$Reg[2] \leftarrow 0$

**for**  $\beta \leftarrow 1, \dots, m$  **do**

            {Compute  $[s_{ij'}^t]_\gamma$  into the register}

**if**  $\rho_{ij'}^t[\gamma, \beta] = 1$  and  $\kappa_{ij'}^t[\beta] = 1$  **then**

$Reg[2] \leftarrow Reg[2] \oplus 1$

**end if**

**end for**

**if**  $Reg[2] = 1$  **then**

$Reg[1] \leftarrow Reg[1] \oplus L_{j'}^\gamma(j)[\alpha]$

**end if**

**end for**

**end for**

**end if**

**return**  $Reg[1]$

---

---

**Algorithm 7**  $update'_\alpha$ 

---

$Reg[1] \leftarrow 0$

{Calculate  $[R_i^t k_i^t]_\alpha$  into the register}

**for**  $\beta \leftarrow 1, \dots, m$  **do**

**if**  $R_i^t[\alpha, \beta] = 1$  and  $k_i^t[\beta] = 1$  **then**

$Reg[1] \leftarrow Reg[1] \oplus 1$

**end if**

**end for**

**for**  $j \leftarrow 1, \dots, p$  **do**

  {Add  $[\rho_{ji}^t \kappa_{ji}^t]_\alpha$  to the register}

**for**  $\beta \leftarrow 1, \dots, m$  **do**

**if**  $\rho_{ji}^t[\alpha, \beta] = 1$  and  $\kappa_{ji}^t[\beta] = 1$  **then**

$Reg[1] \leftarrow Reg[1] \oplus 1$

**end if**

**end for**

**end for**

**return**  $Reg[1]$

---

---

**Algorithm 8** Party  $i$ 's verification that for a given  $j$  his share is valid

---

Calculate  $g^{(j)}(i)$  in main memory {Since it is public, no new information may be revealed from the computation}  
{Check for every bit  $\alpha$ }  
**for**  $\alpha \leftarrow 1, \dots, n$  **do**  
     $Reg[0] \leftarrow 0$   
    {Compute  $\left[ \bar{R}_i^{t(j)} \bar{k}_i^{t(j)} \right]_\alpha$  into the register}  
    **for**  $\beta \leftarrow 1, \dots, m$  **do**  
        **if**  $\bar{R}_i^{t(j)}[\alpha, \beta] = 1$  and  $\bar{k}_i^{t(j)}[\beta] = 1$  **then**  
             $Reg[0] \leftarrow Reg[0] \oplus 1$   
        **end if**  
    **end for**  
    **if**  $Q_j = 1$  **then**  
        {Add  $[R_i^t k_i^t]_\alpha$  to the register}  
        **for**  $\beta \leftarrow 1, \dots, m$  **do**  
            **if**  $R_i^t[\alpha, \beta] = 1$  and  $k_i^t[\beta] = 1$  **then**  
                 $Reg[0] \leftarrow Reg[0] \oplus 1$   
            **end if**  
        **end for**  
    **end if**  
    **if**  $[g^{(j)}(i)]_\alpha \neq Reg[0]$  **then**  
        **return** *invalid*  
    **end if**  
**end for**  
**return** *valid*

---

## 6 Acknowledgments

I wish to thank my advisor, Shafi Goldwasser, for her helpful ideas and suggestions, including the main idea of this thesis. For her useful comments and endless patience, and for helping me transform the rough ideas into a formal and readable thesis. It was a privilege to work under her.

I would also like to thank my friends Gillat Kol and Or Meir, for listening to my numerous conjectures, and for their thoughts and ideas.

## References

- [And97] R. Anderson, *Two remarks on public key cryptology*, 1997.
- [AR99] Yonatan Aumann and Michael O. Rabin, *Information theoretically secure communication in the limited storage space model*, CRYPTO (Michael J. Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer, 1999, pp. 65–79.
- [Bla79] G.R. Blakley, *Safeguarding cryptographic keys*, Proceedings of AFIPS 1979 National Computer Conference **48** (1979), no. 313-317, 390.
- [BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)*, in *STOC* [DBL88], pp. 1–10.
- [CDH<sup>+</sup>00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai, *Exposure-resilient functions and all-or-nothing transforms*, in Preneel [Pre00], pp. 453–469.
- [CH94] Ran Canetti and Amir Herzberg, *Maintaining security in the presence of transient faults*, CRYPTO (Yvo Desmedt, ed.), Lecture Notes in Computer Science, vol. 839, Springer, 1994, pp. 425–438.
- [CL08] Ran Canetti and Dah-Yoh Lim, *How to Deal with Imperfect Erasures*, Submitted to EUROCRYPT’08 (2008).
- [CT06] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley-Interscience New York, 2006.
- [DBL88] *Proceedings of the twentieth annual acm symposium on theory of computing, 2-4 may 1988, chicago, illinois, usa*, ACM, 1988.
- [DBL89] *Proceedings of the twenty-first annual acm symposium on theory of computing, 15-17 may 1989, seattle, washington, usa*, ACM, 1989.
- [DM02] Stefan Dziembowski and Ueli M. Maurer, *Tight security proofs for the bounded-storage model*, STOC, 2002, pp. 341–350.
- [Dod00] Yevgeniy Dodis, *Exposure-Resilient Cryptography*, Ph.D. thesis, Massachusetts Institute of Technology, 2000.

- [DP07] Stefan Dziembowski and Krzysztof Pietrzak, *Intrusion-resilient secret sharing*, FOCS, IEEE Computer Society, 2007.
- [DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener, *Authentication and authenticated key exchanges*, Des. Codes Cryptography **2** (1992), no. 2, 107–125.
- [Dzi06a] Stefan Dziembowski, *Intrusion-resilience via the bounded-storage model*, TCC (Shai Halevi and Tal Rabin, eds.), Lecture Notes in Computer Science, vol. 3876, Springer, 2006, pp. 207–224.
- [Dzi06b] ———, *On forward-secure storage*, CRYPTO (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer, 2006, pp. 251–270.
- [Fel87] Paul Feldman, *A practical scheme for non-interactive verifiable secret sharing*, FOCS, IEEE, 1987, pp. 427–437.
- [FM88] Paul Feldman and Silvio Micali, *Optimal algorithms for byzantine agreement*, in *STOC* [DBL88], pp. 148–161.
- [Gün89] Christoph G. Günther, *An identity-based key-exchange protocol*, EUROCRYPT, 1989, pp. 29–37.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby, *A pseudorandom generator from any one-way function*, SIAM J. Comput. **28** (1999), no. 4, 1364–1396.
- [HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung, *Proactive secret sharing or: How to cope with perpetual leakage*, CRYPTO (Don Coppersmith, ed.), Lecture Notes in Computer Science, vol. 963, Springer, 1995, pp. 339–352.
- [ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby, *Pseudo-random generation from one-way functions (extended abstracts)*, in *STOC* [DBL89], pp. 12–24.
- [JL00] Stanislaw Jarecki and Anna Lysyanskaya, *Adaptively secure threshold cryptography: Introducing concurrency, removing erasures*, in Preneel [Pre00], pp. 221–242.
- [Mau90] Ueli M. Maurer, *A provably-secure strongly-randomized cipher*, EUROCRYPT, 1990, pp. 361–373.

- [Mau92] ———, *Conditionally-perfect secrecy and a provably-secure randomized cipher*, Journal of Cryptology **5** (1992), no. 1, 53–66.
- [NZ96] Noam Nisan and David Zuckerman, *Randomness is linear in space*, Journal of Computer and System Sciences **52** (1996), no. 1, 43–52.
- [OY91] Rafail Ostrovsky and Moti Yung, *How to withstand mobile virus attacks (extended abstract)*, PODC, ACM, 1991, pp. 51–59.
- [Pre00] Bart Preneel (ed.), *Advances in cryptology - eurocrypt 2000, international conference on the theory and application of cryptographic techniques, bruges, belgium, may 14-18, 2000, proceeding*, Lecture Notes in Computer Science, vol. 1807, Springer, 2000.
- [RBO89] Tal Rabin and Michael Ben-Or, *Verifiable secret sharing and multiparty protocols with honest majority (extended abstract)*, in *STOC [DBL89]*, pp. 73–85.
- [Sha79] Adi Shamir, *How to share a secret*, Commun. ACM **22** (1979), no. 11, 612–613.
- [SW99] Douglas R. Stinson and Ruizhong Wei, *Unconditionally secure proactive secret sharing scheme with combinatorial structures*, Selected Areas in Cryptography (Howard M. Heys and Carlisle M. Adams, eds.), Lecture Notes in Computer Science, vol. 1758, Springer, 1999, pp. 200–214.
- [Vad04] Salil P. Vadhan, *Constructing locally computable extractors and cryptosystems in the bounded-storage model*, Journal of Cryptology **17** (2004), no. 1, 43–77.