Memory Erasability Amplification

Jan Camenisch[1(✉)], Robert R. Enderlein[1,2(✉)], and Ueli Maurer[2(✉)]

[1] IBM Research - Zurich, Rueschlikon, Switzerland
jca@zurich.ibm.com, scn2016@e7n.ch
[2] Department of Computer Science, ETH Zürich, Zurich, Switzerland
maurer@inf.ethz.ch

Abstract. Erasable memory is an important resource for designing practical cryptographic protocols that are secure against adaptive attacks. Many practical memory devices such as solid state drives, hard disks, or file systems are not perfectly erasable because a deletion operation leaves traces of the deleted data in the system. A number of methods for constructing a large erasable memory from a small one, e.g., using encryption, have been proposed. Despite the importance of erasable memory in cryptography, no formal model has been proposed that allows one to formally analyse such memory constructions or cryptographic protocols relying on erasable memory.

The contribution of this paper is three-fold. First, we provide a formal model of erasable memory. A memory device allows a user to store, retrieve, and delete data, and it is characterised by a leakage function defining the extent to which erased data is still accessible to an adversary.

Second, we investigate how the erasability of such memories can be amplified. We provide a number of constructions of memories with strong erasability guarantees from memories with weaker guarantees. One of these constructions of perfectly erasable memories from imperfectly erasable ones can be considered as the prototypical application of Canetti et al.'s All-or-Nothing Transform (AoNT). Motivated by this construction, we propose some new and better AoNTs that are either perfectly or computationally secure. These AoNTs are of possible independent interest.

Third, we show (in the constructive cryptography framework) how the construction of erasable memory and its use in cryptographic protocols (for example to achieve adaptive security) can naturally be composed to obtain provable security of the overall protocol.

Keywords: Secure memory erasure · Secure deletion · Adaptive corruption · Constructive cryptography · All-or-nothing-transforms (AoNT)

The first and second author were supported by the European Commission through the Seventh Framework Programme under the ERC grant #321310 (PERCY) and the third author was supported by the Zurich Information Security & Privacy Center (ZISC).

V. Zikas and R. De Prisco (Eds.): SCN 2016, LNCS 9841, pp. 104–125, 2016.
DOI: 10.1007/978-3-319-44618-9_6

1 Introduction

Persistent and erasable memory is a crucial ingredient of many practical cryptographic protocols that are secure against adaptive adversaries. However, for storage devices such as solid state disks, hard disks, and tapes it is rather difficult to truly erase information written on them. Therefore, constructions have been proposed that use a small amount of memory that is easier to erase (or at least harder for an attacker to tap into), such as smart cards and processor registers, to store a cryptographic key, and then to encrypt the data to be stored so that it no longer matters whether or not the ciphertext can be erased [9,11,17–21]. This approach is sometimes referred to as crypto paging. Surprisingly, no formal model of erasable memory has been proposed to date, despite of the importance of erasable memory for cryptographic protocol design and the cryptographic constructions for it.

1.1 Contributions of This Paper

In this paper we rectify this and first model erasable memory as a general resource in the constructive cryptography framework [14,15]. Our memory resource defines how a user, an adversary, and the environment can interact with the resource and to what extent stored data can be erased. In particular, different memory resources are characterized by what information about the stored data an adversary will be able to obtain when the environment allows it access to the memory resource. As we discuss, this allows one to model many different types of memory such as hard disks, solid state drives, RAM, and smart cards. Next, we study different constructions of erasable memory from one with weaker erasability properties or, in other words, constructions that amplify erasability. These constructions also show how memory resources can be used in protocol design and analysis. We then study the approach of crypto paging in our setting, i.e., constructions of a large erasable memory from a small one and a non-erasable memory. As it turns out, achieving the strongest possible type of erasable memory with this approach requires non-committing encryption and hence is only possible in the random oracle model (or requires additional communication between sender and receiver, which is not applicable here). We also show what kind of erasable memory can be achieved with this approach in the standard model.

One of our memory constructions employs All-or-Nothing Transforms (AoNT) [3] to obtain a perfectly erasable memory from one that leaks a constant fraction of the erased data. Motivated by this protocol, we study AoNTs and propose several new transforms that enjoy better parameters than previously known ones, a result that may be of independent interest. For example, we improve the standard construction of a perfectly-secure AoNT from a Linear Block Code (LBC), by observing that an LBC with a large minimum distance does not yield an AoNT with optimal privacy threshold. We propose the metric of *ramp minimum distance* and show that LBCs optimized for this metric yield perfectly secure AoNTs with better parameters than what can be achieved with

the standard construction. We further propose a computationally secure AoNT that operates over a large alphabet (large enough for one symbol to encode a cryptographic key) and that is optimal: the encoded data is just one symbol longer than the original data, and the transform is secure even if all but one of the symbols of the encoded data leak. We show that such an AoNT can be realized from a pseudo-random generator (PRG) with some specific properties.

1.2 Related Work

In most security frameworks, unlimited and perfectly erasable memory is available to protocols as part of the framework, with the exception of protocols that are proven to be adaptively secure in the non-erasure model, where no erasable memory is available. However, as mentioned already, no security framework explicitly models memory and consequently security proofs treat the adversary's access to the memory of a compromised party informally only. The only exception to this is the work by Canetti et al. and Lim [4,13], who model memory as special tapes of the parties' Turing machines and define how an adversary can access these special tapes. This very specific modelling therefore changes the machine model underlying the UC framework.

Hazay et al. [10] follow a different approach. They introduce the concept of *adaptive security with partial erasures*, where security holds if at least one party of a given protocol can successfully erase. Their model requires a special protocol design and has some restrictions regarding composition.

Both these approaches are rather limited. Indeed, if one wanted to consider different types of memory, one would have to change the modelling framework each time and potentially have to prove all composition theorems all over again. Moreover, these approaches do not allow one to analyse protocols that construct one type of memory from another type of memory, as we do in this paper. Indeed, one cannot analyse the security of protocols such as Yee's crypto-paging technique [20,21] and the constructions of Di Crescenzo et al. [6]. In contrast, we model memory as a resource (or ideal functionality) *within* the security framework (the constructive cryptography framework in our case) and thus do not suffer from these limitations.

2 Preliminaries

This section defines the notation used throughout this paper, presents the constructive cryptography model, and recalls various cryptographic building blocks and their security properties.

2.1 Notation

Let $\mathrm{GF}(q)$ denote the Galois field of q elements, where q is a prime power. If u is a vector or a list, let u_i or $u[i]$ denote the ith element of u. If $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_m)$ are lists, then (u, e) denotes the list $(u_1, \ldots, u_n, e)$ and (u, v)

denotes the list $(u_1, \ldots, u_n, v_1, \ldots, v_n)$; we write $e \in u$ to denote $(\exists i : e = u_i)$; we write $v = (u, \cdot)$ to denote that $(\forall i \in \{1, \ldots, n\} : u_i = v_i)$. If L is a set of positive integers, let $[u]_L$ denote the subvector of u taken at all positions in L. If S is a set, then 2^S denotes the powerset of S (the set of all subsets of S). Let $\boldsymbol{I}_r$ denote the identity matrix of size $r \times r$, and let $\mathbf{0}$ denote the zero matrix of appropriate size.

If A is a deterministic polynomial-time algorithm, then $y \leftarrow \mathsf{A}(x)$ denotes the assignment of variable y to the output of $\mathsf{A}(x)$. If A is a probabilistic polynomial-time (PPT) algorithm, then $y \xleftarrow{\$} \mathsf{A}(x)$ denotes the assignment of y to the output of $\mathsf{A}(x)$ when run with fresh random coins on input x. For a set $\mathbb{A}$: $x \xleftarrow{\$} \mathbb{A}$ denotes the assignment of x to a value chosen uniformly at random from $\mathbb{A}$. For a distribution $A(x)$, we denote the ensemble $\{A(x)\}_{x \in \{1^\eta | \eta \in \mathbb{N}, \eta > \eta_0\}}$ by the shorthand $\{A\}_{1^\eta}$.

Throughout this paper we denote the security parameter by $\eta \in \mathbb{N}$. Let 1^η denote the string consisting of η ones. Unless otherwise noted, all algorithms in this paper are PPT and take 1^η as extra (often implicit) input.

2.2 Constructive Cryptography

We present our results in the Constructive Cryptography framework [14,15]. The framework argues about *resources* and how to securely construct a resource from other resources using a protocol which consists of a set of *converters*. Resources and converters are systems that have a set of interfaces. Resources have an interface for each party considered (e.g., *Alice*), one for the adversary (the *Eve* interface), and one for the distinguisher (the *World* interface). The latter is an example of what Gaži et al. [8] introduce as a free interface and allows one to model the influence of the distinguisher (environment) on a resource, e.g., to define when a memory becomes readable by the adversary or to model adaptive adversarial behaviour. Converters have only two interfaces, an *inner interface* that connects to a party interface of a resource and an *outer interface* to which a party can connect. A *simulator* is a converter that attaches to the adversary interface of a resource. In this paper we consider only resources that have a single party interface, i.e., *Alice*. The *security* condition of Constructive Cryptography is as follows (we do not consider the *availability* condition in this paper) [15].

Definition 2.1. *A protocol (converter) π constructs resource S from resource R with respect to simulator σ, within ϵ, denoted*

$$\mathsf{R} \xrightarrow{\pi,\sigma,\epsilon} \mathsf{S},$$

if for all distinguishers D we have $\Delta^D(\pi^{Alice}R, \sigma^{Eve}S) \leq \epsilon(D)$, where Δ^D is the advantage of D in distinguishing the two systems [15].

The distinguisher D is a system itself and has access to all external interfaces of the composition of the resources and converters (cf. Fig. 1). With $\pi^{Alice}R$ we denote the system obtained by attaching the inner interface of π to the interface *Alice* of resource R, and likewise for σ^{Eve}. In this definition, ϵ is a function mapping distinguishers to positive real numbers. Informally, computational security

corresponds to π and σ being efficiently implementable and $\epsilon(D)$ being negligible for all efficiently implementable D. Constructions are composable, i.e., if $\mathsf{R} \xrightarrow{\pi_1,\sigma_1,\epsilon_1} \mathsf{S}$ and $\mathsf{S} \xrightarrow{\pi_2,\sigma_2,\epsilon_2} \mathsf{T}$, then $\mathsf{R} \xrightarrow{\pi_2\pi_1,\sigma_1\sigma_2,\epsilon_2+\epsilon_1} \mathsf{T}$.

2.3 Cryptographic Building Blocks

For our constructions, we require pseudo-random generators and exposure-resilient functions, the definitions of which we recall here for convenience.

Definition 2.2. *An ℓ-pseudo-random generator (PRG), i.e., $\mathsf{prg} : \{0,1\}^\eta \mapsto \{0,1\}^{\ell(\eta)}$, is secure if these ensembles are computationally indistinguishable: $\{\boldsymbol{b}\}_{1^\eta}$ for $\boldsymbol{b} \xleftarrow{\$} \{0,1\}^{\ell(\eta)}$; and $\{\mathsf{prg}(\boldsymbol{a})\}_{1^\eta}$ for $\boldsymbol{a} \xleftarrow{\$} \{0,1\}^\eta$* [12].

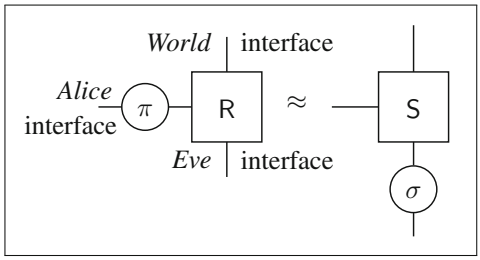

Fig. 1. The constructive statement for a resource with interfaces *Alice*, *Eve*, and *World*. Protocol π constructs S from R if there is a simulator σ such that R with π attached to its *Alice*-interface is indistinguishable from the resource S with σ attached to its *Eve*-interface (cf. Defintion 2.1).

Definition 2.3. *A d-exposure-resilient function (ERF) $\mathsf{erf} : \Phi^n \mapsto \Phi^k$, also denoted (Φ, n, d, k)-ERF, is ϵ-secure if for any set $L \subset \{1, \ldots n\}$ of size at most d, these distributions are ϵ-indistinguishable: $([\boldsymbol{b}]_L, \boldsymbol{x}_0)$ for $\boldsymbol{b} \xleftarrow{\$} \Phi^n, \boldsymbol{x}_0 \leftarrow \mathsf{erf}(\boldsymbol{b})$; and $([\boldsymbol{b}]_L, \boldsymbol{x}_1)$ for $\boldsymbol{b} \xleftarrow{\$} \Phi^n, \boldsymbol{x}_1 \xleftarrow{\$} \Phi^k$* [3].

2.4 All-or-Nothing Transform (AoNT)

An all-or-nothing transform (AoNT) [3] is similar to a secret-sharing scheme that requires all shares in order to reconstruct the secret. It consists of two algorithms aenc and adec.

Definition 2.4. *A d-AoNT with $\mathsf{aenc} : \Phi^k \overset{\$}{\mapsto} \Phi^n$ and $\mathsf{adec} : \Phi^n \mapsto \Phi^k$, also denoted (Φ, n, d, k)-AoNT, is ϵ-secure if: (Completeness) For all messages $\boldsymbol{a} \in \Phi^k$, $\boldsymbol{a} = \mathsf{adec}(\mathsf{aenc}(\boldsymbol{a}))$. (Privacy) For any set $L \subset \{1, \ldots n\}$ of size at most d, and for any two messages $\boldsymbol{a}_0, \boldsymbol{a}_1 \in \Phi^k$ the following two distributions are ϵ-indistinguishable: $(\boldsymbol{a}_0, \boldsymbol{a}_1, [\mathsf{aenc}(\boldsymbol{a}_0)]_L)$ and $(\boldsymbol{a}_0, \boldsymbol{a}_1, [\mathsf{aenc}(\boldsymbol{a}_1)]_L)$* [3].

Computational Security. In the context of computational security, the two functions aenc and adec take as additional input a (usually implicit) security parameter; Φ, n, k, and d may depend on that security parameter. For the privacy condition, it is required that the *ensembles* $\{(\boldsymbol{a}_0, \boldsymbol{a}_1, [\mathsf{aenc}(1^\eta, \boldsymbol{a}_0)]_L)\}_{1^\eta}$ and $\{(\boldsymbol{a}_0, \boldsymbol{a}_1, [\mathsf{aenc}(1^\eta, \boldsymbol{a}_1)]_L)\}_{1^\eta}$ be indistinguishable. In the sequel we also denote such computationally secure AoNTs as (Φ, n, d, k)-AoNTs, where the security parameter is implicit.

AoNT with Public Part. A $(\Phi, n+\nu, d, k)$-AoNT has a ν-public part, if in the privacy condition above the last ν symbols of $\mathsf{aenc}(\boldsymbol{a})$ are output in addition to $[\mathsf{aenc}(\boldsymbol{a})]_L$.

Realization from a Secret Sharing Scheme. It is easy to realize a perfect (Φ, n, d, k)-AoNT from any secret sharing scheme over alphabet Φ that outputs m shares, has a reconstruction threshold of n, a privacy threshold of d, and that encodes messages of size k shares, by simply ignoring all shares after the first n ones. This technique also works in the statistical and computational case.

Realization from an ERF. It is easy to realize an ϵ-secure $(\Phi, n+k, d, k)$-AoNT with a k-public part from any ϵ-secure (Φ, n, d, k)-ERF: $\mathsf{aenc}(\boldsymbol{a}) \overset{\$}{\mapsto} \boldsymbol{b}||(\mathsf{erf}(\boldsymbol{b})+\boldsymbol{a})$ where $\boldsymbol{b} \overset{\$}{\leftarrow} \Phi^n$; and $\mathsf{adec}(\boldsymbol{b}||\boldsymbol{x}) \mapsto \boldsymbol{x} - \mathsf{erf}(\boldsymbol{b})$ [3]. This technique also works in the computational case.

3 Modelling Imperfectly Erasable Memory

We now present our erasable memory resource. Recall that we aim to model memory that is used for persistent storage (such as hard disks, solid state drives, RAM, and smart cards), and not processor registers that store temporary values during computations. To this end, we define how the resource behaves upon inputs on the user (Alice), the adversary, and the world interfaces. It allows a user *Alice* to store a single data item *once*, retrieve it (many times), and erase it. The adversary can get access to the data only if such access is enabled on the *World*-interface. That is, the data stored is not initially available to her. Then, once access is enabled via a *weaken* input on the *World*-interface, the adversary can either *read* the data item stored (if the user has not yet deleted it) or *leak* the data, meaning that she will obtain as answer a function of the once stored data. This function determines the information that is still *leaked* although the data has been deleted. The adversary can influence the leakage by providing an additional input to the function (e.g., specify some bits that are leaked).

In reality, there might be many different reason why an adversary gains access to the contents of a memory. This might be because the memory device is lost, the adversary controlling some malware on the computer that uses the memory, or the adversary running a cache-timing attack [1] on the computer, etc. Offering a *World*-interface via which it is determined what access is given to the adversary by the memory resource, models any such event. The UC and GNUC frameworks use a similar mechanism for corrupting parties, except that they (ab)use the party interfaces to do so. In UC, it is the adversary who corrupts and the environment is informed of the corruption through the party interfaces. In GNUC, the environment corrupts parties and the adversary is informed thereof.

There seem to be two natural extensions to our erasable memory resource which for simplicity we chose not to consider. First, we assume that inputs at the *World*-interface do not impact the user's ability to access the data, which might often not be the case. Although this would not be hard to model, it is not

The resource $\mathsf{M}\langle \Sigma, \psi, \rho, \kappa \rangle$:

Internal state and initial values: DATA $= \bot$, LDAT $= \bot$, HIST $= ()$.

Behavior:

- *Alice*(`store`, $\mu \in \Sigma$): if DATA $= \bot$: DATA $\leftarrow \mu$; LDAT $\xleftarrow{\$} \psi(\mu)$; *Alice* $\leftarrow ()$.
- *Alice*(`retrieve`): if "e" $\notin$ HIST: *Alice* $\leftarrow$ DATA.
- *Alice*(`erase`): if "e" $\notin$ HIST $\wedge$ DATA $\neq \bot$: HIST $\leftarrow$ (HIST, "e"); *Alice* $\leftarrow ()$.
- *Eve*(`gethist`): *Eve* $\leftarrow$ HIST.
- *Eve*(`read`): if ρ(HIST): *Eve* $\leftarrow$ DATA.
- *Eve*(`leak`, ξ): if κ(HIST, ξ): HIST $\leftarrow$ (HIST, "l"$||\xi$); *Eve* $\leftarrow \xi$(LDAT).
- *World*(`weaken`, w): if ("w"$||w$) $\notin$ HIST: HIST $\leftarrow$ (HIST, "w"$||w$); *World* $\leftarrow ()$.

Fig. 2. The general (imperfectly) erasable memory resource $\mathsf{M}\langle\cdot\rangle$.

important for the scope of this paper. Second, the user cannot change the stored data or store many different data items. Again, while it would not be hard to extend the resource to allow for that, we choose not to do that for simplicity. Also, this is not a serious restrictions as such requirements can also be addressed by using several instances of our memory resources.

3.1 Specification of the General Imperfectly Erasable Memory Resource $\mathsf{M}\langle\cdot\rangle$

We now present our formal specification of the general resource for imperfectly erasable memory $\mathsf{M}\langle \Sigma, \psi, \rho, \kappa \rangle$ that is given in Fig. 2 and then discuss in the next subsection a few instantiations of this general resource that match different types of memory. The resource maintains three variables DATA, LDAT, and HIST. The first one stores the data provided by the user, the second the data that can potentially be leaked to the adversary, and the third one logs the history of events, namely the erasure event, the parameter of each call on the *World* interface, and the input arguments of each successful leakage query. The resource is parametrized by an alphabet Σ, a conditional probability distribution ψ, and two predicates ρ and κ. The alphabet Σ is the set of possible values that can be stored. The conditional distribution ψ operates on the data and determines what information could potentially leak to the adversary by outputting LDAT. This models the extent to which the resource is able to erase the data. The predicate ρ takes as input the history of the resource and determines whether or not the adversary is allowed to read the memory. Finally, the predicate κ takes as input the history of the resource and the deterministic function ξ submitted by the adversary and determines whether or not the adversary obtains the leakage ξ(LDAT).

Most of the commands that can be submitted to the resource and its behaviour should now be clear from Fig. 2, however, a few details merit explanation. First, the data that is potentially leaked, LDAT, is determined using ψ already when the data is stored in the resource. This is without loss of generalty but is

here useful because, depending on the predicate κ, the adversary may query the resource multiple times with the `leak` command and the answers to these commands need to be consistent. Second, when the adversary queries the resource with a leak command, she can input a parameter ξ that may influence the leakage she obtains. This models the process of an adversary reading the erased data from a memory device, e.g., an adversary might try to read the data bit by bit, each time influencing the remaining bits in the memory. Third, the adversary is allowed to obtain the history from the resource at any time. This is necessary so that a simulator has enough information to properly simulate a construction. Finally, the *World*-interface accepts any value w for an external event, because these depend on the particular resource that is modelled and possibly on how it is constructed. This will become clear later when we discuss constructions of one type of memory from other types in Sect. 4.

3.2 Instantiations of $\mathsf{M}\langle\Sigma, \psi, \rho, \kappa\rangle$

We now describe special cases of the $\mathsf{M}\langle\Sigma, \psi, \rho, \kappa\rangle$ resource that correspond to memory devices appearing in the real world. We start by describing non-erasable memory, i.e., memory that becomes readable by the adversary once access is enabled by the *World*-interface. This models what happens in a typical file system: files that are unlinked are not actually erased and can often be completely recovered with specialized tools (at least until the blocks are re-used). We then describe perfectly erasable memory. Such a memory could be implemented by specialized hardware, such as smartcards, but often will have only limited capacity. Large perfectly erasable memories are often not directly available in reality. We are thus interested in the construction of such memories from resources with lesser guarantees. Each of the latter can be influenced through *World*-events separately, hence we will describe both a variant of the perfectly erasable memory that accepts a single type of *World*-event (easier to describe) and a variant that accepts an arbitrary number of events. Finally we describe imperfectly erasable memories, i.e., memories with security guarantees between the two extremes just discussed. Such memories leak partial information if the adversary is granted access by *World after* an erasure. In reality, often not all the data is actually removed during an erasure: on magnetic storage, overwritten data can still be partially recovered with specialized equipment [9]. Similarly, often parts of the data were copied to a different medium (swap space, backup, file system journal, etc.) before the erasure and the copies were not fully erased themselves. One can thus easily imagine that the adversary can deduce a constant number of bits that were stored, or obtains a noisy version of the data that was stored. For simplicity, we consider imperfectly erasable memories which ignore the parameter of `weaken` (only a single *World*-event can be modelled), and only leak once (no adaptive leakage). We now describe these categories of memory in detail. Table 1 provides an overview of these and further specialization of them that we consider in the following sections.

Table 1. Different specializations of $\mathsf{M}\langle\Sigma, \psi, \rho, \kappa\rangle$ that allow one to erase data.

Perfectly erasable memory	Influence of the *World* interface
$\mathsf{PM}\langle\Sigma\rangle$	single world event makes memory readable by adversary
$\mathsf{PMW}\langle\Sigma\rangle$	multiple world events are modelled
$\mathsf{PMWa}\langle\Sigma\rangle$	specific version of $\mathsf{PMW}\langle\Sigma\rangle$ (Fig. 4)
$\mathsf{PMWb}\langle\Sigma\rangle$	specific version of $\mathsf{PMW}\langle\Sigma\rangle$ (Fig. 4)
$\mathsf{PMWc}\langle\Sigma\rangle$	specific version of $\mathsf{PMW}\langle\Sigma\rangle$ (Fig. 4)
Imperfectly erasable memory	Information adversary obtains on deleted data
$\mathsf{IM}\langle\Sigma, \psi, \Xi\rangle$	reveals $\xi(\text{LDAT})$ if $\Xi(\xi) = 1$
$\mathsf{IMD}\langle\Phi, n, d\rangle$	reveals d symbols to adversary
$\mathsf{IMDP}\langle\Phi, s_1, s_2, d\rangle$	reveals d symbols of first part, all symbols of second part
$\mathsf{IMI}\langle\Phi, n, d\rangle$	each symbol revealed independently with probability p
$\mathsf{IMN}\langle\Phi, n, d\rangle$	reveals through noisy channel
$\mathsf{IML}\langle\Sigma, v\rangle$	reveals through a length shrinking function
$\mathsf{IMLP}\langle\Phi, n, a+k, v\rangle$	reveals a length shrinking function on first part, full second part

Non-erasable Memory. To model non-erasable memory, we let ρ return true if `weaken` was called irrespective of `erase`. (In fact, the `erase` command could be dropped entirely.) The memory does not leak, hence κ always returns false and ψ is irrelevant. The only relevant parameter is the alphabet Σ and thus we denote this resource by $\mathsf{NM}\langle\Sigma\rangle$.

Perfectly Erasable Memory. To model perfectly erasable memory, we let ρ return true only if `weaken` was called (perhaps multiple times with specific parameters) before `erase` was called.[1] This memory does not leak, hence κ always returns false and ψ is irrelevant. We describe two versions of the resource: $\mathsf{PM}\langle\Sigma\rangle$ fixes ρ to return true if `weaken` appears in the history earlier than or without `erase`, hence only a single *World*-event can be modelled. $\mathsf{PMW}\langle\Sigma, \rho\rangle$ lets one specify a custom ρ, allowing the modelling of many *World*-events. Figure 4 in the next section shows examples of ρ in the case where there are two relevant *World*-events.

[1] In this paper, we chose to consider monotone ρ's. We chose to model the memory resource in such a way that it only responds on the same interface it was activated, hence it is not possible for the adversary to be notified of an event that causes the memory to become readable. To simplify the modelling of simulators, we consider the adversary to be eager and trying to read the memory as soon as possible and then placing the resulting data in an "intermediate buffer" that can then be collected through the *Eve*-interface at a later point.

Imperfectly Erasable Memory. To model imperfectly erasable memory, we fix ρ and split κ into two predicates, a fixed predicate that checks only the history and a freely specifiable predicate Ξ that checks only the adversary's choice ξ. The other parameters Σ and ψ can be freely specified. We denote this resource by $\mathsf{IM}\langle\Sigma, \psi, \Xi\rangle$. We consider only resources allowing for a single *World*-event. The predicate ρ returns true only if the first recorded event in the history is a `weaken` command (as opposed to an `erase` command). The fixed predicate returns true if the first two recorded events in the history are an `erase` command followed by a `weaken` command (if `weaken` was called first, the adversary should call `read` and not `leak`), and no `leak` query succeeded previously. Thus, we consider only resources allowing for a single *World*-event. The predicate κ returns true if the fixed predicate does so and Ξ accepts ξ. In the next section, when we discuss erasability amplification, we further specialize this resource.

4 Constructing Better Memory Resources

In this section we consider constructions of memory resources with stronger security properties from memory resources with weaker ones. We start by showing how to use our memory resources in protocol constructions and then explain the issues that arise when doing so. Thereafter, we describe several specializations of the imperfectly erasable memory resource $\mathsf{IM}\langle\cdot\rangle$ presented in the previous section and then show how to construct memory resources with stronger properties from ones with weaker properties. For example, we show how to construct perfectly erasable memories from memories that leak a certain number of bits. Finally, we consider the construction of a large perfectly erasable memory from a small one plus a large non-erasable memory.

4.1 Admissible Converters for Constructions Using Erasable Memory

As stated previously, one of our reasons to model memory is to be able to analyse cryptographic protocols where the adversary at some point obtains access to the memory. This means that one needs to restrict converters to use only our memory resources for storage. Assuming that an adversary in a real environment may typically not be able to get access to processor registers, we still allow a converter to store temporary values locally and use a memory resource only for persistent storage. Let us now formalize the distinction between persistent and temporary storage and the restrictions we put on converters.

The computation done by a converter is divided in *computation phases*. A phase starts when a converter is activated outside of a computation phase. Informally, a phase ends as soon as the converter responds to that activation or makes a request that is not guaranteed to be answered immediately, i.e., where there is a chance that the adversary is activated before the request completes. For example, a computation phase ends if the converter makes a request that goes

over an unreliable communication network, but does not end if the converter asks to store or retrieve data from a memory resource.

In this paper, all our resources always respond on the same interface they were activated. It is then easy to define a computation phase of a converter: the phase starts as soon as the converter's outer interface is activated, and stops as soon as the converter writes on its outer interface. That is, activations of the inner interface do not interrupt the phase. However, in a more general setting, resources may respond on a different interface than the one they were activated on, and thereby activate a different party or the adversary. The definition of computation phase of converters must therefore be adjusted to take this into account.

State that is discarded at the end of a computation phase is *temporary*. State that must persist between two or more computation phases is *persistent*. (Converters must keep all persistent state in memory resources.) This distinction ensures that whenever the adversary has control, the entire internal state of a protocol is inside memory resources, and thus subject to attack.

Discussion. Other models, notably Canetti et al. and Lim [4,13], also make a distinction between storage needed during computation and persistent storage. However they do it in a way that does not cleanly separate the various layers of abstraction: they assume the existence of a constant number of "processor registers" that are perfectly erasable and place no restriction on the amount of time that data can remain in such a register. For example, their model therefore does not exclude reserving a part of the CPU registers to permanently store a cryptographic key, and use a crypto paging technique [20,21] to have as much (computationally secure) perfectly erasable memory as required. Thus, to ensure a meaningful analysis, a similar restriction would have to be used in their approach.

4.2 Memory Erasability Amplification

We now describe several variants of imperfectly erasable memory that are relevant for practice, namely memory that leaks a constant number of bits, memory that leaks bits with a certain probability, memory that leaks a noisy version of the data, and memory that leaks the output of a length-shrinking function of the data. We then show how to construct memories which leak less information from each of these variants, in other words, we show how to amplify the erasability of each variant.

4.2.1 Amplifying Memory Leaking Exactly d Symbols

On many file systems, unlinked files are not necessarily immediately erased in their entirety. For instance, on most SSDs, deleted data persists until the flash translation layer flashes the corresponding erase block. Furthermore, data may survive erasure if it was copied to another medium, such as a cache, the swap space or backups. An adversary could therefore potentially recover parts of data

that were believed to be erased. In full generality, the adversary may not obtain the entire data but still have an influence on which parts of the data she obtains in an attack, e.g., because she can steal just one backup tape, because of the cost of the attack or time constrains forcing her to choose the most juicy parts of the data, or because the adversary could influence the system beforehand to some degree and ensure that the parts of the data she is interested in were backed-up/swapped/cached.

To model such a scenario, we define the memory resource $\mathsf{IMD}\langle \Phi, n, d\rangle$ storing n symbols of an alphabet Φ, and where the adversary can obtain exactly d symbols of his choice when the memory leaks. This resource is a specialization of $\mathsf{IM}\langle \Sigma, \psi, \Xi\rangle$, where $\Sigma = \Phi^n$, ψ is the identity function, and Ξ accepts any function that reads at most d symbols from LDAT.

In a real setting, and depending on the nature of the attack, the adversary may obtain less than d symbols or might not have full control over which symbols she obtains. A memory resource in such a setting can be perfectly constructed from $\mathsf{IMD}\langle \Phi, n, d\rangle$ with the identity converter. (A memory resource where the adversary can obtain more than d symbols with a small probability ϵ can also be constructed from $\mathsf{IMD}\langle \Phi, n, d\rangle$, albeit with an error probability equal to ϵ; see, e.g., Sect. 4.2.2.)

The converter I2P shown in Fig. 3 constructs $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMD}\langle \Phi, n, d\rangle$. This converter is parametrized by an AoNT (cf. Sect. 2.4). In a nutshell, I2P just applies the AoNT encoding algorithm $\mathsf{aenc}(\cdot)$ to the incoming data before storing it in $\mathsf{IMD}\langle\cdot\rangle$; and decodes the encoded data stored in $\mathsf{IMD}\langle\cdot\rangle$ using $\mathsf{adec}(\cdot)$ before outputting it. The erasure command is transmitted to $\mathsf{IMD}\langle\cdot\rangle$ directly. The privacy property of the AoNT guarantees that if the adversary obtains d symbols of the encoded data, she obtains no meaningful information about the original data. Thus, we obtain the following theorem.

The converter $\mathsf{I2P}\langle \Phi, k, \mathsf{aenc}, \mathsf{adec}\rangle$:

Behavior:

- *Outer*(`store`, $\mu \in \Phi^k$): *Inner* $\xleftarrow{\$}$ (`store`, $\mathsf{aenc}(\mu)$). *Inner* $\rightarrow$ (). *Outer* $\leftarrow$ ().
- *Outer*(`retrieve`): *Inner* $\leftarrow$ (`retrieve`). *Inner* $\rightarrow \phi$.
  If $\phi \neq$ (): *Outer* $\leftarrow \mathsf{adec}(\phi)$. Else: *Outer* $\leftarrow$ ().
- *Outer*(`erase`): *Inner* $\leftarrow$ (`erase`). *Inner* $\rightarrow$ (). *Outer* $\leftarrow$ ().

Fig. 3. The converter I2P constructing $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMD}\langle \Phi, n, d\rangle$. The converter is parametrized by a (Φ, n, d, k)-AoNT $(\mathsf{aenc}, \mathsf{adec})$.

Theorem 4.1. *If* $(\mathsf{aenc}, \mathsf{adec})$ *is an* ϵ*-secure* (Φ, n, d, k)*-AoNT, then*

$$\mathsf{IMD}\langle \Phi, n, d\rangle \xrightarrow{\pi,\sigma,\epsilon} \mathsf{PM}\langle \Phi^k\rangle,$$

where $\pi = \mathsf{I2P}\langle \Phi, k, \mathsf{aenc}, \mathsf{adec}\rangle$ *and* σ *is the simulator provided in the proof.*

The proof this theorem is found in the full version of this paper. A similar theorem can be stated for the computational case.

Multi-part Leakage. It is sometimes the case that the memory is segmented into multiple independent parts, e.g., over two different file systems on different partitions of the same physical disc and that each part reacts differently to an attack.

We define a multi-part memory resource $\mathsf{IMDP}\langle \Phi, s_1, s_2, d\rangle$ storing data in $\Phi^{s_1+s_2}$. The memory is divided in two parts, the first part consisting of the first s_1 symbols and the second of the other s_2 symbols. The first part of the memory leaks similarly to $\mathsf{IMD}\langle \Phi, s_1, d\rangle$, while the second one leaks the entire data. When attacking the memory, the adversary must submit the choice of leakage for the first part *before* obtaining the leakage of the second part. We get the following theorem, the proof of which is similar to the one of Theorem 4.1 and is omitted.

Theorem 4.2. *If* $(\mathsf{aenc}, \mathsf{adec})$ *is an* ϵ*-secure* $(\Phi, n+\nu, d, k)$*-AoNT with public part* ν*, then*

$$\mathsf{IMDP}\langle \Phi, n, \nu, d\rangle \xrightarrow{\pi,\sigma,\epsilon} \mathsf{PM}\langle \Phi^k\rangle,$$

where $\pi = \mathsf{I2P}\langle \Phi, k, \mathsf{aenc}, \mathsf{adec}\rangle$ *and* σ *is the simulator provided in the proof.*

A similar theorem can be stated for the computational case.

Choice of Alphabet. The most suitable choice of Φ depends on the application. Possible values are GF(2) when bits can be leaked independently, e.g., because the adversary must read them one by one from the surface of a disc; $\mathrm{GF}(2^{512\cdot 8})$ to $\mathrm{GF}(2^{4096\cdot 8})$ when the smallest leakable unit is a file system block; or even $\mathrm{GF}(2^{128\cdot 1024\cdot 8})$ to $\mathrm{GF}(2^{8192\cdot 1024\cdot 8})$ when the smallest leakable unit is an erase blocks of an SSD. In the latter two cases, it is also possible to design the system in such a way that only parts of a block are written to before proceeding with the next one, thereby reducing the alphabet size and limiting the amount of exposure per leaked block.

4.2.2 Amplifying Memory Leaking Symbols with Probability p

Above, we modelled an adversary who chooses which symbols leak from the imperfect memory. In practice, the adversary may not have this much power: for example, some parts of a deleted file might still be present in the journal, but the adversary has no control over which ones. To model this, let us now consider an adversary who obtains each symbol of the data uniformly and independently at random with a certain probability p during a leakage. We denote a memory with such a behaviour by $\mathsf{IMI}\langle \Phi, n, d\rangle$. This resource is a specialization of $\mathsf{IM}\langle \Sigma, \psi, \Xi\rangle$ where $\Sigma = \Phi^n$, ψ acts like an erasure channel with erasure probability $(1-p)$ (i.e., each symbol of the data is transmitted correctly with probability p and otherwise is replaced with "$\perp$"), and Ξ accepts only the identity function.

One can treat $\mathsf{IMI}\langle\cdot\rangle$ similarly to $\mathsf{IMD}\langle\cdot\rangle$ in constructions with just a small statistical error, as the following observation shows. Constructing $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMI}\langle \Phi, n, p\rangle$ directly without first constructing $\mathsf{IMD}\langle \Phi, n, d\rangle$ might be more efficient (better parameters, less statistical error), but such a direct construction is

out of the scope of this paper. The proof of the following observation is found in the full version of this paper.

Observation 4.3. *For all* $(n, d) \in \mathbb{N}^2$, $p \in [0, 1]$, *and fields* Φ *we have that* $\mathsf{IMI}\langle \Phi, n, p\rangle \xrightarrow{\mathsf{id},\sigma,\epsilon} \mathsf{IMD}\langle \Phi, n, d\rangle$, *where* id *is the identity converter,* σ *is the simulator provided in the proof, and* $\epsilon = (1 - \mathrm{BinomialCDF}(d; n, p)) = \sum_{i=d+1}^{n} \binom{n}{i} p^i \cdot (1-p)^{n-i}$.

4.2.3 Amplifying Memory with Noisy Leakage

Another possible setting is that the data is written to and erased from magnetic storage, and the adversary, who has physical access to the storage medium, must make an educated guess for each bit of the data [9]. One can model this as if the data was transmitted through a noisy binary symmetric channel. We denote such a memory by $\mathsf{IMN}\langle \Phi, n, d\rangle$. This resource is a specialization of $\mathsf{IM}\langle \Sigma, \psi, \Xi\rangle$ where $\Sigma = \Phi^n$, ψ acts like a noisy $|\Phi|$-ary channel with crossover probability $(1-p)/|\Phi|$ (i.e., each symbol of the data is transmitted correctly with probability p and otherwise is replaced with a symbol drawn uniformly at random from Φ), and Ξ accepts only the identity function.

Observation 4.4. *For all* $(n, d) \in \mathbb{N}^2$, $p \in [0, 1]$, *and fields* Φ *we have that* $\mathsf{IMN}\langle \Phi, n, p\rangle \xrightarrow{\pi,\sigma,0} \mathsf{IMI}\langle \Phi, n, p\rangle$, *where* $\pi = \mathsf{id}$ *is the identity converter and* σ *is the simulator that replaces all erased symbols in the leakage by random symbols.*

4.2.4 Amplifying Memory with Limited Leakage Output Domain

Another possible setting is that the adversary does not obtain individual symbols of the data but rather a function of the data. For example, with a cache-timing attack [1], she might deduce some information about the data without recovering it completely. In general, one can consider an adversary that obtains any *length-shrinking* function of the contents of the memory. We denote such a memory by $\mathsf{IML}\langle \Sigma, v\rangle$. This resource is a specialization of $\mathsf{IM}\langle \Sigma, \psi, \Xi\rangle$, where ψ is the identity function and Ξ accepts only functions that have at most v different output values.

For any non-trivial parameters, it is not possible to construct a perfectly erasable memory from $\mathsf{IML}\langle\cdot\rangle$, because the adversary can submit a leakage function $\xi \in \Xi$ that runs the decoding logic of the converter. The reason for this is as follows. Let $v \geq 2$, $|\Sigma'| \geq 2$, $|\Sigma| \geq 2$, and let π be a converter that constructs $\mathsf{PM}\langle \Sigma'\rangle$ from $\mathsf{IML}\langle \Sigma, v\rangle$. We now show that this construction has a statistical error of at least $\frac{1}{2}$. The distinguisher chooses two distinct messages $a_0, a_1 \in \Sigma'$, flips a coin $b \xleftarrow{\$} \{0, 1\}$, and stores a_b. He then makes the memory weak by setting the relevant flags on the *World*-interface and submits a leakage function ξ that returns 0 iff a_0 was encoded in $\mathsf{IML}\langle\cdot\rangle$ by using the decoding logic of π—recall that the distinguisher may depend on π. The distinguisher then outputs 1 iff ξ outputs b. No simulator will be able to properly simulate that scenario with probability more than $\frac{1}{2}$ as it does not know if the distinguisher stored a_0 or a_1.

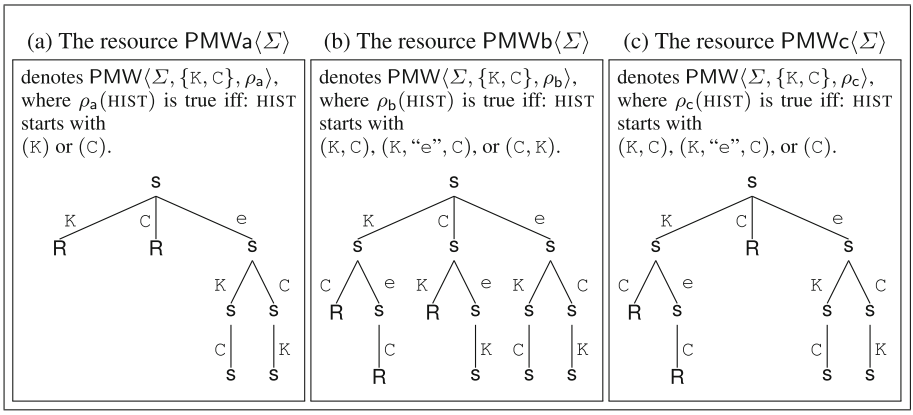

Fig. 4. Several variants of a perfectly erasable memory resource with two *World*-flags. The prefix decision trees visualize whether the adversary has read access to the memory depending on the event history HIST. A branch labelled "e" represents an erasure event, and branches labelled "K" (key) or "C" (ciphertext) represent the setting of the corresponding flags on the *World*-interface. An "R" node means that the memory is readable (and allows the adversary to collect the data at any time from then on), and an "s" (secure) node means that it does not.

The converter XPM⟨ℓ, prg⟩:

Behavior:

- *Outer*(store, $\mu \in \mathrm{GF}(2^{\ell(\eta)})$): $sk \xleftarrow{\$} \mathrm{GF}(2^\eta)$. $\delta \leftarrow \mathsf{prg}(sk) + \mu$. *Inner* ← (PM, store, sk). *Inner* → (). *Inner* ← (NM, store, δ). *Inner* → (). *Outer* ← ().
- *Outer*(retrieve): upon error in the following, abort with *Outer* ← ().
  Inner ← (PM, retrieve). *Inner* → $sk \in \mathrm{GF}(2^\eta)$.
  Inner ← (NM, retrieve). *Inner* → δ. $\mu \leftarrow \delta - \mathsf{prg}(sk)$. *Outer* ← μ.
- *Outer*(erase): *Inner* ← (PM, erase); *Inner* → (). *Outer* ← ().

Fig. 5. The converter XPM constructing a large perfectly erasable memory $\mathsf{PMWa}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ or $\mathsf{PMWc}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ using a small perfectly erasable memory $\mathsf{PM}\langle \mathrm{GF}(2^\eta)\rangle$ and a large non-erasable memory $\mathsf{NM}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$. The converter is parametrized by an ℓ-PRG prg, and the implicit security parameter η.

However, one can obtain a meaningful construction by starting from a memory resource with multi-part leakage. Let $\mathsf{IMLP}\langle \Phi, s_1, s_2, v\rangle$ be analogous to $\mathsf{IMDP}\langle \Phi, s_1, s_2, d\rangle$ defined previously, except that the first part leaks similarly to $\mathsf{IML}\langle \Phi^{s_1}, v\rangle$. Here it is crucial to note that the function ξ submitted by the adversary can read only the first part of the memory. In particular, given a universal hash function $\mathsf{h} : \Phi^a \times \Phi^n \mapsto \Phi^k$, one can construct the resource $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMLP}\langle \Phi, n, a+k, v\rangle$, by using I2P with an AoNT obtained from a universal hash function (see full paper for details). The construction is $(2v2^{(k-n)/2})$-secure [3,5]. This construction is essentially the one proposed by Canetti et al. [4] and Lim [13].

4.3 Constructing a Large Perfectly Erasable Memory from a Small One

We now discuss how a small perfectly erasable memory can be used together with a large, possibly non-erasable memory to construct a large perfectly erasable memory. The basic idea underlying this construction is that of Yee et al.'s crypto paging [20,21]: one stores a cryptographic key in the small perfectly erasable memory, encrypts the data with that key, and stores the resulting ciphertext in the large, possibly non-erasable memory. The resulting resource $\mathsf{PMWa}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ will allow the adversary to read the stored data if the resource is weakened by the environment before the user erases the key. The specification of this resource is given in Fig. 4a and the protocol XPM for the construction is provided in Fig. 5.

The resource just constructed allows the adversary to read the stored data if either the small erasable *or* the large non-erasable memory become weak before the user erases the key. Thus, this resource is weaker than what one would expect, i.e., it should be the case that the adversary can only read the data if *both* underlying resources become weak before the user erases the key. The corresponding resource $\mathsf{PMWb}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ is depicted in Fig. 4b. Unfortunately, the realization of this resource would require a non-committing and non-interactive encryption scheme, which can only be constructed in the random oracle model but not in the standard model.

However, it is possible to construct the somewhat better resource $\mathsf{PMWc}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$, shown in Fig. 4c. Here the adversary can read the stored data if the memory storing the ciphertext becomes weak before the user calls delete. It is not hard to see that $\mathsf{PMWc}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ implies $\mathsf{PMWa}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$, essentially the simulator attached to the Eve interface of $\mathsf{PMWa}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ has to hold back the leaked data until the non-erasable memory becomes leakable. In summary, we get the following theorem, the proof of which is found in the full paper.

Theorem 4.5. *If* prg *is a secure* ℓ*-PRG, then*

$$\left[\mathsf{PM}\langle \mathrm{GF}(2^{\eta})\rangle, \mathsf{NM}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle\right] \xrightarrow{\pi,\sigma,\epsilon} \mathsf{PMWc}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle,$$

where $\pi = \mathsf{XPM}\langle \ell, \mathsf{prg}\rangle$*,* σ *is given in the full version, and* ϵ *is a negligible function.*

As stated above, XPM also constructs $\mathsf{PMWa}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ from the same resources. Furthermore, in the random oracle model, a protocol that is identical to XPM except that calls to prg are replaced by calls to the random oracle, constructs $\mathsf{PMWb}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ from the same resources.

Let us discuss our the memory resources just discussed in light of some secure memory constructions in the literature. As mentioned, Yee et al. introduce crypto paging [20,21] to let a secure co-processor encrypt its virtual memory before paging it out to its host's physical memory or hard disk. Translated to our

setting, this means that the non-erasable memory is weak from the beginning. Therefore, to get meaningful guarantees, only the resource $\mathsf{PMWb}\langle \mathrm{GF}(2^{\ell(\eta)})\rangle$ can be used in their setting, the other two would allow the adversary to always read the data. Thus, to realize their system, Yee et al. require a non-committing and non-interactive encryption scheme (and hence random oracles).

Di Crescenzo et al. [6] consider a memory resource that allows one to update the stored data such that when the resource becomes weak the adversary can only read the data stored last. They then provide a construction for a large such resource from a small one and a large non-erasable memory. Again they assume that for both resources the data can be updated and that the non-erasable one leaks all data ever stored in it. None of our resources does allow for such updates but, as already discussed, resources that allow this can be constructed by using several of our respective resources in parallel. Thus, their security definition and construction can be indeed modelled and analysed with the memory resources we define, however, doing this is out of scope of this extended abstract.

5 New Realizations of All-or-Nothing Transforms

In Sect. 4 we saw the importance of AoNTs for constructing perfectly erasable memory from certain types of imperfectly erasable ones. In this section we present several novel AoNTs. We start by showing the dual of the $\mathsf{I2P}$ protocol: any protocol that constructs $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMD}\langle \Phi, n, d\rangle$ can be used to realize a (Φ, n, k, d)-AoNT. We then present a perfect AoNT with better parameters than what is found in the literature, based on the novel concept of *ramp minimum distance* of a matrix. We then show that one can combine several AoNTs to achieve an AoNT over a small field but with a large message space and a good privacy threshold d. Finally, we provide a computationally-secure AoNT over a large field that has a very large privacy threshold.

5.1 AoNT from a Protocol that Constructs $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMD}\langle \Phi, n, d\rangle$

Sect. 4.2.1 described the protocol $\mathsf{I2P}$, parametrized by an AoNT, that constructs a perfectly erasable memory $\mathsf{PM}\langle \Phi^k\rangle$ from an imperfectly erasable one $\mathsf{IMD}\langle \Phi, n, d\rangle$. As the following theorem states, any protocol π (not necessarily one based on an AoNT) that constructs $\mathsf{PM}\langle \Phi^k\rangle$ from $\mathsf{IMD}\langle \Phi, n, d\rangle$ can be used to construct an AoNT using the algorithm $\mathsf{C2A}$ (given in Fig. 6), albeit one where adec is a probabilistic algorithm and where decoding might fail with a small probability.

Theorem 5.1. *If* (π, σ, ϵ) *are such that* $\mathsf{IMD}\langle \Phi, n, d\rangle \xrightarrow{\pi,\sigma,\epsilon} \mathsf{PM}\langle \Phi^k\rangle$*, then the algorithm* $\mathsf{C2A}\langle \Phi, n, k, \pi\rangle$ *is a* 6ϵ*-secure* (Φ, n, d, k)*-AoNT with a probabilistic* adec *and where decoding may fail with probability less than* 2ϵ*.*

The proof of this theorem is found in the full version of this paper. One can make an analogous statement in the computational case.

The algorithm $\mathsf{C2A}\langle \Phi, n, k, \pi \rangle$:

Behavior:

- $\mathsf{aenc}(\mu \in \Phi^k)$: $\pi.\mathit{Outer} \leftarrow (\texttt{store}, \mu)$.
  While true:
  If $\pi.\mathit{Inner} \rightarrow (\texttt{store}, \phi \in \Phi^n)$: return ϕ.
  Else if anything is sent by $\pi.\mathit{Inner}$: $\pi.\mathit{Inner} \leftarrow ()$.
  Else: abort by returning $\perp$.
- $\mathsf{adec}(\phi \in \Phi^n)$: $\pi.\mathit{Outer} \leftarrow (\texttt{retrieve})$.
  While true:
  If $\pi.\mathit{Inner} \rightarrow (\texttt{retrieve})$: $\pi.\mathit{Inner} \leftarrow \phi$.
  Else if $\pi.\mathit{Outer} \rightarrow \mu \in \Phi^k$: return μ.
  Else if $\pi.\mathit{Inner} \rightarrow (\texttt{erase})$: abort by returning $\perp$.
  Else if anything is sent by $\pi.\mathit{Inner}$: $\pi.\mathit{Inner} \leftarrow ()$.
  Else: abort by returning $\perp$.

Fig. 6. The algorithm C2A that realizes a (Φ, n, d, k)-AoNT from a converter π, where π constructs $\mathsf{PM}\langle \Phi^k \rangle$ from $\mathsf{IMD}\langle \Phi, n, d \rangle$.

5.2 Perfectly Secure AoNT Based on Matrices with Ramp Minimum Distance

This subsection shows how one can improve the standard realization of AoNTs based on linear block codes of Canetti et al. [3] by using our novel concept of *ramp minimum distance*.

The Standard Realization. Let $\boldsymbol{G}$ be the $k \times n$ generator matrix with elements in $\mathrm{GF}(q)$ of a linear block code with minimum distance d. The encoding function of the perfectly secure $(\mathrm{GF}(q), (n+k), d, k)$-AoNT is as follows:

$$\mathsf{aenc}(\boldsymbol{a} \in \mathrm{GF}(q)^k) : \boldsymbol{b} \xleftarrow{\$} \mathrm{GF}(q)^n; \boldsymbol{y} \leftarrow \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ \boldsymbol{G} & \boldsymbol{I}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{a} \end{bmatrix}; \text{ return } \boldsymbol{y}.$$

Further details are given in the full paper.

Let us now show how to use the concept of ramp minimal distance to construct better AoNTs.

Definition 5.2. *A $k \times n$ matrix $\boldsymbol{G}$ with elements in* $\mathrm{GF}(q)$ *has* ramp minimum distance *d if for every $r \in \{1, \ldots, k\}$, every $r \times (n-(d-r))$ submatrix of $\boldsymbol{G}$ has rank r.*

Note that the concept of (regular) minimum distance comes from coding theory, and requires that all $k \times (n-(d-1))$ sub-matrices of G have rank k (which is equivalent to saying that for every $r \in \{1, \ldots, k\}$, all $r \times (n-(d-1))$ sub-matrices of $\boldsymbol{G}$ have rank r), where $\boldsymbol{G}$ is the generator matrix of a linear block code. A matrix with minimum distance d also has a *ramp* minimum distance d (the converse is obviously not true).

Now for the generator matrix with ramp minimum distance, we can construct an AoNT and thus obtain the following theorem, the proof of which is found in the full version of this paper.

Theorem 5.3. *The standard realization of an AoNT (sketched above and detailed in the full paper), parametrized by a $k \times n$ matrix $\boldsymbol{G}$ with elements in* $\mathrm{GF}(q)$ *with* ramp *(instead of regular)* minimum distance d*, is a perfectly secure* $(\mathrm{GF}(q), (n+k), d, k)$*-AoNT.*

It remains to find a matrix with a desired ramp minimum distance. One way is to chose a random matrix, as shown by the following theorem that we prove in the full paper.

Theorem 5.4. *For all $(n, k, d) \in \mathbb{N}^3$, and all prime powers q, a $k \times n$ matrix where all elements were chosen independently and uniformly at random over* $\mathrm{GF}(q)$*, has ramp minimum distance d with probability at least*

$$1 - \sum_{i=1}^{k} \binom{k}{i} (q-1)^i q^{\left(H_q\left(\frac{d-i}{n}\right)-1\right)n},$$

where $H_q(x) := \begin{cases} 0 \text{ if } x = 0 \text{ or } x = 1; \\ x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x) \text{ if } 0 < x < 1. \end{cases}$

Unfortunately, we do not know of any efficient method to check whether a random matrix has a given ramp minimum distance. For practical parameters, however, it is feasible to generate and test such matrices with small values of k and d (e.g., less than 20).

Better AoNTs Using our Realization. Given a fixed size, it is sometimes possible to find matrices with a given ramp minimum distance but no matrix with the same (regular) minimum distance. Hence AoNTs based on matrices with a ramp minimum distance can achieve better parameters than previously known realizations. We now illustrate this fact with a numerical example. Let us determine the best message length k that a perfect AoNT with fixed parameters $n = 30$, $d = 12$, and $q = 2$ can achieve with both our realization and the standard realization. Both realizations will require a matrix with $(30 - k)$ rows and (ramp or regular, respectively) minimum distance $d = 12$. First, observe that there exists a 6×24 matrix over $\mathrm{GF}(2)$ with ramp minimum distance 12 (see the full paper). Hence using our realization, we can achieve $k = 6$. Plotkin [16] showed that a binary code with block length $2d$ and distance d can have at most $4d$ codewords. Hence there cannot exist a 6×24 matrix with (regular) minimum distance $d = 12$ (as it would generate a code with $2^6 = 64$ codewords, which is more than $4d = 48$). The best AoNT one can hope for using the standard realization thus has $k = 5$.

Statistical Security. Theorem 5.4 stated that by choosing a random generator matrix, one can achieve a certain ramp minimum distance with a certain probability $(1 - \epsilon)$. If one uses our realization, but without checking that the matrix

actually has the required ramp minimum distance, then the resulting AoNT will be perfectly secure with probability $(1 - \epsilon)$. (Note that this is different from saying that the AoNT is ϵ-secure, as the randomness used to generate the AoNT is not part of the distinguishing experiment.) In practice, one can make ϵ very small, e.g., $\epsilon < 2^{-\eta}$, and it might be acceptable to chose a random matrix and not check its properties to realize an AoNT.

5.3 Realizing a Perfectly Secure AoNT over a Small Field by Combining AoNTs

Designing perfectly-secure AoNTs over very small fields, e.g., GF(2), is hard. The previous realization does not scale well to large message lengths k and large privacy thresholds d; and realizations based on Shamir's secret sharing scheme are always over large fields—using such a $(\mathrm{GF}(2^a), n, d, k)$-AoNT unmodified over GF(2) instead would result in a $(\mathrm{GF}(2), an, d, ak)$-AoNT with a poor privacy threshold d. The leakage of any GF(2) element means that the entire original $\mathrm{GF}(2^a)$ element is compromised. We now show how to combine the two approaches to realize a perfectly secure AoNT over a small field but with large k and d.

Our realization requires two AoNTs, a "fine-grained" one and a "coarse-grained" one, operating over a small field S and a large field L, respectively. We require that the number of elements of L be a power of that of S and that $k^s = \log(|L|)/\log(|S|)$ be true. We need to interpret a string of $k^\ell k^s$ elements from S as a string of k^ℓ elements of L, an operation we denote by S$\triangleright$L. The converse operation is denoted L$\triangleright$S.

The encoding function of our combined AoNT then works as follows. One first applies the coarse-grained AoNT to the whole data vector and then applies the fine-grained AoNT to each element of the result:

$$\mathsf{aenc}(\boldsymbol{a} \in S^{k^s k^\ell}):$$
$$\boldsymbol{x} \xleftarrow{\$} \mathsf{aenc}^\ell(\mathrm{S}\triangleright\mathrm{L}(\boldsymbol{a})); \forall j \in \{1, \ldots, n^\ell\} : \boldsymbol{b}[j] \xleftarrow{\$} \mathsf{aenc}^s(\mathrm{L}\triangleright\mathrm{S}(\boldsymbol{x}[j])); \text{ return } \boldsymbol{b}.$$

It's easy to see how the decoding function adec of the combined AoNT works and it is thus omitted. We have the following theorem, the proof of which is found in the full version of this paper.

Theorem 5.5. *Given a perfectly secure* (S, n^s, d^s, k^s)*-AoNT* $(\mathsf{aenc}^s, \mathsf{adec}^s)$ *and a perfectly secure* $(L, n^\ell, d^\ell, k^\ell)$*-AoNT* $(\mathsf{aenc}^\ell, \mathsf{adec}^\ell)$ *such that* $k^s = \log(|L|)/\log(|S|)$*, the AoNT* $(\mathsf{aenc}, \mathsf{adec})$ *described above is a perfectly secure* $(S, n^s n^\ell, (d^s + 1)(d^\ell + 1) - 1, k^s k^\ell)$*-AoNT.*

Numerical Example. Let us suppose that we are interested in a perfect AoNT that operates over $S = \mathrm{GF}(2)$ and that can store a cryptographic key of size $k = 256$ bits using at most $n = 8192$ bits (a kilobyte) of memory.

If we use a $(\mathrm{GF}(2^{10}), 819, 793, 26)$-AoNT built according to Franklin and Yung [7] unmodified over the field GF(2), we get a $(\mathrm{GF}(2), 8190, 793, 260)$-AoNT. This AoNT has a privacy threshold d of only 793 bits.

By combining a $(\mathrm{GF}(2), 32, 11, 8)$-AoNT (which can be found by exhaustive search) with a $(\mathrm{GF}(2^8), 255, 223, 32)$-AoNT built according to Franklin and Yung [7], one gets a $(\mathrm{GF}(2), 8160, 2687, 256)$-AoNT. This AoNT has a much better privacy threshold d of 2687, i.e., 2687 arbitrary bits may leak to the adversary.

5.4 Computationally Secure AoNT over a Large Field from a PRG

We now present a realization of a computationally secure AoNTs over a large field $\mathrm{GF}(2^\eta)$, where η is the security parameter. Our realization is optimal in the sense that it achieves both an optimal message length $k = n - 1$ (thus an optimal rate $(n-1)/n$) and an optimal privacy threshold $d = n-1$. That is, the AoNT needs just a single additional element to encode a message and remains private even if the adversary obtains all but any one element.

Definition 5.6. *An ℓ-PRG where the output length is a multiple of the input length, i.e.,* $\mathsf{prg} : \mathrm{GF}(2^\eta) \mapsto \mathrm{GF}(2^\eta)^{\ell(\eta)/\eta}$*, is* KD-secure*, if for all $i = 1, \ldots, \ell(\eta)/\eta$, these ensembles are computationally indistinguishable:*

- $\{(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_{\ell(\eta)/\eta})\}_{1^\eta}$ *where* $sk \xleftarrow{\$} \mathrm{GF}(2^\eta)$, $\boldsymbol{x} \leftarrow \mathsf{prg}(sk)$, *and* $x'_i \leftarrow x_i + sk$.
- $\{\boldsymbol{x}\}_{1^\eta}$ *where* $\boldsymbol{x} \xleftarrow{\$} \mathrm{GF}(2^\eta)^{\ell(\eta)/\eta}$.

Note that this property is somewhat reminiscent of the KDM-CCA2 security of encryption functions [2].

Our realization, somewhat reminiscent of the OAEP realization of Canetti et al. [3], is as follows:

$$\begin{aligned}
&\mathsf{aenc}(\boldsymbol{m} \in \mathrm{GF}(2^\eta)^{\ell(\eta)/\eta}): && sk \xleftarrow{\$} \mathrm{GF}(2^\eta); \boldsymbol{x} \leftarrow \mathsf{prg}(sk);\ \boldsymbol{y} \leftarrow \boldsymbol{x} + \boldsymbol{m};\\
& && \text{return } \boldsymbol{y} || \big(sk + \textstyle\sum_{i=1}^{\ell(\eta)/\eta} y_i\big).\\
&\mathsf{adec}(\boldsymbol{y}||z): && \text{return } \boldsymbol{y} - \mathsf{prg}(z - \textstyle\sum_{i=1}^{\ell(\eta)/\eta} y_i).
\end{aligned}$$

Theorem 5.7. *Given an ℓ-PRG that is both secure and KD-secure, the realization above yields a secure $(\mathrm{GF}(2^\eta), 1 + \ell(\eta)/\eta, \ell(\eta)/\eta, \ell(\eta)/\eta)$-AoNT.*

The proof of this theorem is found in the full version of this paper. There we further observe that Canetti et al.'s [3] computationally-secure AoNT built by combining an exposure resilient function (ERF) with a pseudo-random generator (PRG) can have an essentially arbitrarily high message length k and message rate k/n, but cannot achieve a very high privacy threshold d.

References

1. Bernstein, D.J.: Cache-timing attacks on AES. Manuscript, April 2005. https://cr.yp.to/antiforgery/cachetiming-20050414.pdf

2. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
3. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
4. Canetti, R., Eiger, D., Goldwasser, S., Lim, D.-Y.: How to protect yourself without perfect shredding. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 511–523. Springer, Heidelberg (2008)
5. Canetti, R., Eiger, D., Goldwasser, S., Lim, D.-Y.: How to protect yourself without perfect shredding. Cryptology ePrint Archive, Report 2008/291 (2008)
6. Di Crescenzo, G., Ferguson, N., Impagliazzo, R., Jakobsson, M.: How to forget a secret. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 500–509. Springer, Heidelberg (1999)
7. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: 24th ACM STOC, pp. 699–710. ACM Press, May 1992
8. Gaži, P., Maurer, U., Tackmann, B.: Manuscript. (available from the authors)
9. Gutmann, P.: Secure deletion of data from magnetic and solid-state memory. In: Proceedings of the Sixth USENIX Security Symposium, vol. 14, San Jose, CA (1996)
10. Hazay, C., Lindell, Y., Patra, A.: Adaptively secure computation with partial erasures. Cryptology ePrint Archive, Report 2015/450 (2015)
11. Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: introducing concurrency, removing erasures (extended abstract). In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 221–242. Springer, Heidelberg (2000)
12. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, Boca Raton (2015)
13. Lim, D.-Y.: The paradigm of partial erasures. Ph.D. thesis, Massachusetts Institute of Technology (2008)
14. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012)
15. Maurer, U., Renner, R.: Abstract cryptography. In: ICS 2011, pp. 1–21. Tsinghua University Press, January 2011
16. Plotkin, M.: Binary codes with specified minimum distance. IRE Trans. Inf. Theor. **6**(4), 445–450 (1960)
17. Reardon, J., Basin, D.A., Capkun, S.: SoK: secure data deletion. In: 2013 IEEE Symposium on Security and Privacy, pp. 301–315. IEEE Computer Society Press, May 2013
18. Reardon, J., Capkun, S., Basin, D.: Data node encrypted file system: efficient secure deletion for flashmemory. In: Proceedings of the 21st USENIX Conference on Security Symposium, pp. 17–17. USENIX Association (2012)
19. Reardon, J., Ritzdorf, H., Basin, D.A., Capkun, S.: Secure data deletion from persistent media. In: ACM CCS 2013, pp. 271–284. ACM Press, November 2013
20. Yee, B.: Using secure coprocessors. Ph.D. thesis, CMU (1994)
21. Yee, B., Tygar, J.D.: Secure coprocessors in electronic commerce applications. In: Proceedings of The First USENIX Workshop on Electronic Commerce, New York (1995)