PASSIVE TIMESTAMPING IN THE BOUNDED STORAGE MODEL
SUBMISSION VERSION

Abstract. A timestamping scheme is *passive* if a stamper can stamp a document without communicating with any other player. The only communication done is at validation time. Passive timestamping has many advantages, such as information theoretic privacy and enhanced robustness. Passive timestamping, however, is not possible against polynomial time adversaries that have *unbounded storage* at their disposal. As a result, no passive timestamping schemes were constructed up to date.

In this paper we show passive timestamping is possible in the bounded storage model. I.e., if the adversary has *bounded storage*, and all players have access to a long random string. To the best of our knowledge, this is the first example of a cryptographic task that is possible in the bounded storage model, but is *impossible* in the "standard cryptographic setting", even assuming cryptographic assumptions.

We give an explicit construction that is secure against all bounded storage adversaries, and a significantly more efficient construction secure against all bounded storage adversaries that run in polynomial time.

Keywords: timestamping, bounded storage model, expander graphs, extractors

1. Introduction

The date on which a document was created is often a significant issue. Patents, contracts, wills and countless other legal documents critically depend on the date they were signed, drafted, etc. A timestamp for a document provides convincing proof that it existed at a certain time. For physical documents, many methods are known and widely used for timestamping: publication, witnessed signing and placing copies in escrow are among the most common. Techniques for timestamping digital documents, which are increasingly being used to replace their physical counterparts, have also become necessary.

Loosely speaking, a timestamping scheme consists of two mechanisms: A *stamping mechanism* which allows a user to stamp a document at some specific time t, and a *verification mechanism* which allows a recipient to verify at a later time $t' > t$ that the document was indeed stamped at time t.

1.1. Previous Work.

Digital timestamping systems were first introduced in Haber and Stornetta [15], where three timestamping systems are described. In the *naïve timestamping protocol*, the stamper sends the document to all the verifiers during timestamp generation. In the *linking* scheme, the stamper sends a one-way hash of the document to a trusted timestamping server. The server holds a *current hash*, which it updates by hashing it with the value sent by the stamper. This links the document to the previous documents and to the succeeding ones. In the *distributed trust* scheme, the document is used to select a subset of verifiers, to which the stamper sends a hash of the document. Bayer, Haber and Stornetta [3] improve upon the linking scheme, reducing the communication and storage requirements of the system and increasing its robustness, by replacing the linear list with a tree. Further work [16, 7, 6, 8, 5, 4] is mainly focused on additional improvements in terms of storage, robustness and reducing the trust required in the timestamping server(s).

One common feature of all the above protocols is that they require the stamper to send messages to a central authority (or a distributed set of servers) at timestamp generation.

1.2. Passive Timestamping.

We call a timestamping scheme *passive* if it does not require the stamper to send messages at timestamp generation. Passive timestamping schemes, if they exist, have a number of obvious advantages over active schemes. However, the notion of passive timestamping seems self-contradictory. How can we prevent an adversary from faking timestamps, if no action is taken at timestamp generation? More precisely, suppose that an adversary "learns" some document at time $t' > t$ and wants to convince a verifier that he stamped the document at time t. He can simulate the behavior of an "honest stamper" who signs the document at time t and generate a timestamp for the document. Note that the "honest stamper" does not need to send any messages before time t' and therefore the adversary will be able to convince a verifier that the document was stamped at time t.

A crucial point in the argument above is that in order to perform this simulation the adversary must store all the information available to the "honest stamper" at time t. We show that passive timestamping is possible in a scenario in which parties have bounded storage.

1.3. **The Bounded Storage Model.** In contrast to the usual approach in modern Cryptography, Maurer's *bounded storage model* [18, 19] bounds the *storage* (memory size) of dishonest players rather than their running time.

In a typical protocol in the bounded storage model a long random string r of length R is initially broadcast and the interaction between the polynomial-time participants is conducted based on storing small portions of r.[1] The security of such protocols should be guaranteed even against dishonest parties which have a lot of storage (much more than the honest parties) as long as they cannot store the whole string. We mention that most of the previous work on the bounded storage model concentrated on private key encryption [18, 10, 2, 1, 13, 14, 17, 25], Key Agreement [19, 10] and Oblivious Transfer [9, 11, 12]. In contrast, the notion of passive timestamping cannot be implemented in the "standard cryptographic setting". To the best of our knowledge this is the first example of a protocol in the bounded storage model which achieves a task that is impossible in the "standard cryptographic setting".

1.4. **Passive Timestamping In The Bounded Storage Model.** We now explain our setting for passive timestamping in the bounded storage model. We assume that there are ℓ rounds and at every round $1 \le t \le \ell$, a long random string r of length R is transmitted.[2]

The stamping mechanism: To stamp a document doc at time t, the scheme specifies a function $\mathrm{Stamp}(doc, r)$ whose output is short. To stamp the document doc, the stamper stores $\mathrm{Stamp}(doc, r)$. Intuitively, an adversary (who does not know doc at time t) is not able to store the relevant information and therefore is unable to stamp doc.

The verification mechanism: The verifier stores a short "sketch" of r (denoted by $\mathrm{Sketch}(r)$) for every time t. At a later time the stamper can send the timestamp $\mathrm{Stamp}(doc, r)$ and the verifier checks whether this timestamp is "consistent" with his sketch.

Efficiency of a timestamping scheme. We say that a timestamping scheme is (T, V) efficient if the stamper's algorithm runs online using space T and polynomial time and the verifier's algorithm runs online using space V and polynomial time. We want T and V to be small as functions of R.

1.5. **Our Notion Of Security.** Loosely speaking, we want to ensure that even an adversary with a lot of storage (say storage $M = \delta R$ for some constant $\delta < 1$) cannot forge a timestamp. Note, however, that a stamper with storage $M > T$ can easily stamp $k = M/T$ documents by running the stamping mechanism on some k documents and storing the generated timestamp (which is of

[1] One possible implementation is that r is broadcast at a very high rate by a trusted party. Another possibility is to have r transmitted from a satellite. Furthermore, the assumption that r is uniformly distributed can be relaxed and it is sufficient that r has high min-entropy.

[2] One can imagine that random bits are transmitted at high rate continuously by a trusted party, and that the string r consists of the bits transmitted between time t and time $t + 1$.

length at most T). We will therefore say that a scheme is secure if no adversary with space M can successfully stamp significantly more than M/T documents.

Security of a timestamping scheme. Given a (T,V)-efficient timestamping scheme. Let M_{max} be the bound on the storage of the most powerful adversary. The scheme is α-optimal if, for every $M \leq M_{max}$, no adversary with space M can successfully stamp more than $\alpha\frac{M}{T}$ documents.[3]

Notice that the definition above requires α–optimality for every $M \leq M_{max}$. Requiring α–optimality for M_{max} only, would have allowed adversaries with $M \ll M_{max}$ to produce $\frac{M_{max}}{T}$ stamped documents, contradicting the definition's spirit. The definition in its current form assures us that any adversary, weak or strong, with at most M_{max} memory, can honestly stamp *the same number of documents* if given slightly more resources (storage αM instead of M).

1.6. **Our Results.** In this paper we give two constructions of passive timestamping schemes in the bounded storage model. The first is secure in an information-theoretic sense (in the spirit of previous constructions in the bounded storage model). It requires no unproven assumptions and is secure against any adversary with arbitrary computational power as long as its storage capability is bounded. We now state this result (precise definitions appear in Section 3).

Theorem 1.1. *For every $\eta > 0$ and large enough R we construct a timestamping scheme which is $(T = O(R^{1/2+\eta}), V = O(R^{1/2+\eta}))$-efficient and $O(1)$-optimal. More precisely, every adversary with space $M^* \leq M^*_{max} = \Omega(R)$ has probability at most $2^{-R^{\Omega(1)}}$ to successfully stamp more than $O(M^*/T)$ documents. The timestamping scheme allows stamping documents of length $R^{\Omega(1)}$ and allows $R^{\Omega(1)}$ rounds.*

Our second system is more efficient. To achieve this efficiency it relies on cryptographic assumptions and is therefore secure only against adversaries that, in addition to being storage bounded, are required to run in polynomial time.

Theorem 1.2. *Assume that there exist collision resistant hash functions. For every $\eta > 0$ and large enough R there exists a timestamping scheme which is $(T = R^\eta, V = R^\eta)$-efficient and $O(\log R)$-optimal. More precisely, every adversary with space $M^* \leq M^*_{max} = \Omega(R)$ and running time polynomial in R has negligible probability to successfully stamp more than $O(\log R \cdot M^*/T)$ documents. The timestamping scheme allows stamping documents of length R and allows R rounds.*

We remark that Theorem 1.2 can be improved, and our method actually gives a scheme which is (T,V)-efficient for $T = V = 2^{(\log\log R)^{O(1)}} = R^{o(1)}$. We also remark that our technique can potentially reduce T and V to $\log^{O(1)} R$. This improvement will require an explicit construction of certain "expander graphs". More details appear in Appendix D.

In this submission version we concentrate on timestamping schemes with one round. (In such a scheme, only *one* long string r is transmitted and the stamper needs to convince the verifier that he "knew" the document before r was transmitted). We prove Theorems 1.1,1.2 for a single round. This simplifies the presentation. Nevertheless, our method easily translates to the case of many rounds and yields the results in the theorems above.

1.7. **Advantages Of Our Passive Timestamping Scheme.**

- No communication is made between the stamper and verifier at timestamp generation. The only communication made before the verification process is the transmission of the random string r.
- Everyone can stamp and everyone can verify and no central control or acquaintance between stamper and verifier is needed.

[3] We did not explicitly define what it means to "successfully stamp" a document at this point. Loosely speaking, we say that a stamper successfully stamps a document (after the transmission of some long string r) if the timestamp he generates will be accepted by the verifier. Precise definitions appear in Section 3.

- Information-theoretic privacy. The scheme hides the fact that timestamping occurred at all, e.g., an inventor can safeguard her inventions without revealing even the fact of their existence.
- Our schemes solve some of the robustness problems that plague active timestamping systems. In particular, it is much more difficult to mount a denial-of-service attack, since temporarily shutting down communications will not prevent the creation of new timestamps.

1.8. **Overview Of The First Construction.** The setup is the following: A string r of length R is transmitted and the stamper wants to convince a verifier that he "knew" a document prior to the transmission of this string.

Using the document to select indices. We implement the function Stamp(doc, r) as follows: Each document *doc* specifies some D indices that the stamper will remember from the long string. For that we use a bipartite graph where the left-hand vertices are all possible documents, the right-hand vertices are indices $1 \leq i \leq R$ and every left vertex has D neighbors. The indices selected by a document *doc* are the neighbors of *doc*. We want to force a stamper who would like to stamp k documents to store many indices. It turns out this is equivalent to the requirement that every k documents on the left have many different neighbors. This naturally leads to using an expander graph. (A bipartite graph is a (K, c)-expander if every $k < K$ vertices on the left have at least kc neighbors on the right.)[4]

To stamp a document *doc*, the stamper stores the content of the long string at the indices specified by *doc*. We use graphs with expansion $c \approx D$, therefore to correctly stamp k documents simultaneously a honest stamper must store roughly kD bits.

Using random sets for verification. The function Sketch(r) is implemented as follows. The verifier chooses a random subset of size $|\mathcal{H}| \approx R/D$ from the indices of r and stores the content of r at these indices. After the transmission of the random string r, a stamper may send a timestamp of a document *doc* (which consists of the content of r at the D indices defined by *doc*). By the birthday paradox, with high probability (over the choice of the verifier's random set) some of these indices were also stored by the verifier. The verifier checks that the content sent by the stamper is consistent with what he stored.

For a fixed string r and document *doc*, we say that a timestamp is "incorrect" if it differs from the "correct" timestamp of *doc* in many indices. The verification process we described guarantees that, with high probability, the verifier will reject an "incorrect" timestamp.

A sketch of the security proof. The basic intuition for the security proof is the following: Suppose that an adversary is able to successfully stamp some k documents. This means that he correctly stamped these k documents (as otherwise he is caught by the verifier). However, correctly stamping k documents requires storing kD indices, therefore if the storage of the adversary is $kD \leq M < (k+1)D$ he can successfully stamp at most k documents. This is the best we can hope for (by our notion of security) as he could have stamped k documents by simply running the "stamping mechanism" on any k documents.

However, the argument above is not sufficient. It does not rule out the possibility that the adversary can stamp many documents such that the identity of these documents *depend* on the random string r. Our security definition requires that for every adversary, with high probability (over the choice of r) there do not exist k documents which the adversary can successfully stamp. To prove the security of our scheme we use a "reconstruction argument" and show that any adversary

[4] We stress that we need to use *unbalanced graphs* (graphs which have many more vertices on the left than on the right-hand side). Such graphs were constructed in [24, 23]. However, we need graphs with somewhat different parameters. We construct such graphs by combining the constructions of [24] and a slight modification of [23, 21] (which in turn relies on explicit constructions of "randomness extractors" from [21, 22]).

which breaks the security guarantee can be used to compress the string r into a shorter string in a way that does not lose a lot of information. As the string r is random, we get a contradiction. The details are given in Section 4.

1.9. **Overview Of The Second Construction.** In the previous construction we chose $|\mathcal{H}| \approx R/D$ so that a random subset of size $|\mathcal{H}|$ in $[R]$ would intersect a subset of size D. We chose $|\mathcal{H}| = D \approx \sqrt{R}$, allowing both the honest stamper and the verifier to store only $\sqrt{R}$ bits. We now show how to increase the efficiency and reduce the storage of honest parties to only R^η bits for an arbitrary small constant $\eta > 0$.

We use the same index selection mechanism as before. However, this time we choose $D = R^\eta$. The verifier stores a short "hash" of the string r. When stamping a document the stamper also supplies a short "proof" that the indices he sent are consistent with the hashed value held by the verifier. We implement such a hashing scheme using *Merkle hash trees* [20]. We show that if collision resistant hash functions exist then a polynomial time adversary with bounded storage cannot produce an incorrect timestamp of a document. More precisely, we show that after the transmission of the random string r, no polynomial time adversary can *generate* many documents and stamp them correctly.

Hashing documents before stamping them. A bottleneck of our scheme is that when using expanders of degree D we can only handle documents of length D.[5] However, in a computational setting (as we have already assumed the existence of collision resistant hash functions) we can stamp longer documents by first hashing them to shorter strings and then stamping them.

1.10. **Outline of the Paper.** In section 2 we define the notations used throughout the paper and state the properties we require from the constructions used in our timestamp systems (unbalanced expander graphs, collision-resistant hash functions and Merkle hash trees). Section 3 contains a formal definition of the model and our notions security. Sections 4 and 5 describe our explicit constructions of timestamp schemes, secure in the information-theoretic and computationally bounded settings, respectively. We discuss our results in section 6. The appendices contain a construction for an explicit expander graph (needed by our timestamp systems), and the proofs for some of the more technical lemmas.

2. Preliminaries

2.1. **Notation.** The following conventions will be used throughout the paper. We refer to the random string as r, its length is denoted by R, and we think of it as composed of N blocks of length n denoted $r_1, \ldots, r_N$. For any subset $S \subseteq [N]$, the expression $r_{|S}$ will be taken to mean the string generated by concatenating the blocks r_i for all $i \in S$. The *Hamming Distance* between two strings r_1 and r_2 is the number of blocks on which the two strings differ. For a family of functions F, we denote by *Space*(F) the maximum space used by any function in F. We say a function f can be computed *online* with space s if there is an algorithm using space at most s which reads its input bits one by one and computes f in one pass.

2.2. **Unbalanced Expander Graphs.** A graph is expanding if every sufficiently small set has a lot of neighbors. Our timestamping scheme relies on *unbalanced expanders.*

Definition 2.1 (unbalanced expander graphs)**.** *A bipartite graph $G = (V_1, V_2, E)$ is (K_{max}, c)-*expanding *if, for any set $S \subset V_1$ of cardinality at most K_{max}, the set of its neighbors $\Gamma(S) \subseteq V_2$ is of size at least $c|S|$.*

[5] This is because in unbalanced expander graphs, the degree must be logarithmic in the number of left-hand vertices. Thus, shooting for degree D we can at most get that the left-hand set (which is the set of documents) is of size 2^D.

Note that we do not require that $|V_1| = |V_2|$. In fact, in our timestamping scheme we will use graphs in which $|V_1| \gg |V_2|$. In this paper we need unbalanced expanders with very specific requirements. Loosely speaking we want a $(K_{max}, \Omega(D))$-expanding graph with as small as possible degree D and right-hand side of size roughly $K_{max}D$. We use some existing constructions of unbalanced graphs [24] as well as a modification of [23] to prove the next Theorem.

Theorem 2.2. *There exists a fixed constant $\beta > 0$ such that for every $K_{max} \leq |V_1|$, there exists a bipartite graph $G = (V_1, V_2, E)$ with left degree D that is $(K_{max}, c = \beta D)$ expanding with $D = 2^{O(\log\log V_1 + (\log\log K_{max})^3)}$, and $|V_2| = 4\beta K_{max} D$. Furthermore, this graph is explicit in the sense that given a vertex $v \in V_1$ and an integer $1 \leq i \leq D$ one can compute the i'th neighbor of v in time polynomial in $\log|V_1| + \log D$.*

We sketch the proof of Theorem 2.2 in Appendix A.

2.3. **Collision Resistant Hash Functions And Merkle Hash Trees.** We sometimes need functions that are secure against adversaries that run in *super-polynomial* time, and therefore we use a parameter for the running time of the adversary:

Definition 2.3 (collision resistant hash functions)**.** *A family of functions $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$ is said to be $(\epsilon_H(n), \mu(n))$-collision resistant if, there is a polynomial time algorithm which given the description of $h \in H$ and an input $x \in \{0,1\}^n$ outputs $h(x)$. Furthermore, given a random function $h \in H$, no algorithm with running time bounded by $\mu(n)$ can find $x \neq y$ such that $h(x) = h(y)$ with probability greater than $\epsilon_H(n)$ where the probability is over the choice of h and the random coins of the algorithm and n is large enough.*

We use Merkle hash tree [20]. We give a detailed definition so that we can later use the notation in our construction.

Definition 2.4 (Merkle hash tree)**.** *Let $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$ be a $(\epsilon_H(n), \mu(n))$-collision resistant family of hash functions and $(r_1, \dots, r_N) \in \{0,1\}^n \times \dots \times \{0,1\}^n$ a vector of binary strings (assume N is a power of 2). Fix a key $h \in H$. The* Merkle tree *with key h of $r_1, \dots, r_N$, denoted $Tree^h(r_1, \dots, r_N)$, is a binary tree with N leaves, where each vertex is labelled in the following way:*

- *The i^{th} leaf is labelled with r_i.*
- *An internal vertex v with children labelled c_1 and c_2 is labelled with $h(c_1, c_2)$.*

$Root^h(r_1, \dots, r_N)$ denotes the label of the root vertex.

A Merkle path *from a leaf r_i is an ordered collection of all the nodes on the path from r_i to the root of $Tree^h(r_1, \dots, r_N)$, along with their siblings in the tree. Let $Path_i^h(r_1, \dots, r_N)$ denote the labelled Merkle-path from r_i where each node is labelled by the labelling of the Merkle hash tree.*

A labelled Merkle path P from a leaf r_i is valid *if P labels the root with $Root^h(r_1, \dots, r_N)$, and for every node v in the path from r_i to the root, P labels v by $h(c_1, c_2)$ where the c_1, c_2 are the labels P assigns to the children of v. (Note that the children of v appear in P). A labelled Merkle path P from a leaf r_i is* correct *for a tree $Tree^h(r_1, \dots, r_N)$ if its label for r_i is identical with the label of r_i in the tree.*

Clearly, every description of the form $Path_i^h(r_1, \dots, r_N)$ is both valid and correct for $Tree^h(r_1, \dots, r_N)$. The usefulness of Merkle hash trees stems from the fact that it is infeasible to find an incorrect valid path.

Definition 2.5. *A Merkle tree construction is* $(\epsilon_H, \mu(N, n))$-secure *if, for any set of leaves $(r_1, \dots, r_N)$ and any index $i \leq N$, no algorithm with running time bounded by $\mu(N, n)$ can produce a valid path from r_i that is incorrect for $Tree^h(r_1, \dots, r_N)$ with probability greater than ϵ_H (where the probability is over the choice of h and the random coins of the algorithm).*

The following standard Lemma observes that collision-resistant hash functions are sufficient to build Merkle Trees (we sometimes set N to be super-polynomial in n, and this is why we need very strong Collision Resistant Hash Functions):

Lemma 2.6. [20] *Let $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$ be a $(\epsilon_H(n), \mu(n))$-collision resistant family of hash functions. Then the Merkle tree construction based on H is $\left(\epsilon_H(n), O(\mu(n) - Nn - \log N \cdot n^{O(1)})\right)$-secure.*

Merkle Trees and storage: To check whether a path is valid requires a verifier to store h and $Root^h(r_1, \ldots, r_N)$, but not the rest of the tree. To prove that r_i is a leaf in $Tree^h(r_1, \ldots, r_N)$, the prover needs to store only $Path_i^h(r_1, \ldots, r_N)$, which is *logarithmic* in N. Note that calculating $Path_i^h(r_1, \ldots, r_N)$ can be done online (when $r_1, \ldots, r_N$ are given sequentially, and assuming the hash function can be computed for free) using space $O(n \log N)$. This is done by keeping in memory only the roots of the subtrees whose leaves have been seen so far (i.e. after we are given r_1 and r_2, we can compute $h(r_1, r_2)$ and forget the values of r_1 and r_2; in the same way after we have seen $r_1, \ldots, r_{2^k}$ we can remember only the label of the common ancestor for these leaves, forgetting all the nodes in the subtree).

3. One Round Timestamping: The Model

In this section we formally define our model for timestamping in the bounded storage model. The definitions are only for a single round.

A long random string r of length R is transmitted. The verifier takes a short sketch $\mathrm{Sketch}(r)$ of the random string and remembers it. An honest stamper, who wants to stamp a document $doc \in \mathcal{DOC}$, calculates $y = \mathrm{Stamp}(doc, r)$. When, at a later stage, the stamper wants to prove he knew the document doc at stamping time, he sends y to the verifier who computes $\mathrm{Verify}(\mathrm{Sketch}(r), doc, y)$ and decides whether to accept or reject. Thus,

Definition 3.1 (Passive timestamping scheme)**.** *A passive timestamping scheme consists of three functions:*

- *A stamping function* $\mathrm{Stamp}(doc, r)$*.*
- *A sketch function* $\mathrm{Sketch}(r)$ *(we allow* Sketch *to be a probabilistic function).*
- *A verification protocol* $\mathrm{Verify}(\mathrm{Sketch}(r), doc, y)$*.*

We require that for every string r and document doc, $\mathrm{Verify}(\mathrm{Sketch}(r), doc, \mathrm{Stamp}(doc, r))$ *accepts.*

We define efficiency:

Definition 3.2 (Efficiency)**.** *A passive timestamping scheme is* (T, V)-efficient *if* Stamp *can be computed online in space $T = T(R)$ and time polynomial in R, and* Sketch *can be computed online in space $V = V(R)$ and time polynomial in R.*

An honest stamper with space M can easily stamp M/T documents by running the function Stamp in parallel. We require that no adversary with memory M^* can successfully stamp significantly more than M^*/T documents. We first define our model for adversaries:

Definition 3.3 (adversary)**.** *An adversary consists of two functions:* $\mathrm{Store}^*(r)$*, which produces a short string b, and* $\mathrm{Stamp}^*(doc, b)$ *which, given a document doc and b, attempts to produce a timestamp for doc. The space M^* of an adversary is the maximal length of* $\mathrm{Store}^*(r)$*. An adversary* γ-successfully stamps *a document doc at (some fixed) r if*

$$\Pr[\mathrm{Verify}(\mathrm{Sketch}(r), doc, \mathrm{Stamp}^*(doc, \mathrm{Store}^*(r))) = \textit{Accept}] \geq \gamma$$

Note that this probability is over the coin tosses of Sketch *and the internal random coins of the adversary.*

We define security as:

Definition 3.4 (Security). *We say that a (T, V)-efficient timestamping scheme is α-optimal (for $\rho > 0$, $\alpha \geq 1$, $\gamma > 0$ and $M^*_{max} < R$) if for every $M^* \leq M^*_{max}$ and every adversary A with space M^*,*

$$\Pr_r \left[A \textit{ can } \gamma\textit{-successfully stamp } \alpha \frac{M^*}{T} \textit{ documents at } r \right] \leq \rho$$

Definition 3.4 is very strong. It guarantees that whenever the sketch size is small, no matter how powerful the adversary is, the number of documents the adversary can successfully stamp is very small. We remark that w.l.o.g. we can assume that the adversary is deterministic, as any probabilistic adversary is a distribution over deterministic adversaries, therefore there is always a deterministic adversary that is as successful as the best probabilistic adversary with that space.

3.1. **Security Against Feasibly Generated Documents.** Until now, we allowed the adversary to run in arbitrary time. When the adversary is time-bounded, we can imagine scenarios where Definition 3.4 does not hold, yet the system is secure because the adversary does not have the computational power to find the documents he can illegally stamp. It makes sense to require security only against "feasibly generated documents". We model feasibly generated documents by a probabilistic polynomial time machine $\mathrm{Generate}^*_{\mathrm{c}}$ which, on input r and an integer i, outputs a document.

Definition 3.5 (Security against feasibly generated documents). *We say that a (T, V)-efficient timestamping scheme is α-optimal (for $\rho > 0$, $\alpha \geq 1$, $\gamma > 0$ and $M^*_{max} < R$) against feasibly generated documents, if for every $M^* \leq M^*_{max}$, every adversary A with space M^*, and every polynomial time machine* $\mathrm{Generate}^*_{\mathrm{c}}(r, i)$*:*

$$\Pr \left[A \textit{ can } \gamma\textit{-successfully stamp at } r \textit{ the } \alpha \frac{M^*}{T} \textit{ different documents } \{\mathrm{Generate}^*_{\mathrm{c}}(r, i)\}_{i=1}^{\alpha \frac{M^*}{T}} \right] \leq \rho$$

where the probability is over the choice of r *and the random coins of* $\mathrm{Generate}^*_{\mathrm{c}}$ *and* A*.*

4. A Scheme With Information Theoretic Security

In this section we describe a timestamping scheme which is information theoretically secure against arbitrary adversaries with small storage.

4.1. **The Stamping Scheme.** Let R, N and n be integers such that $R = N \cdot n$. Given a string $r \in \{0,1\}^R$, we partition it into N blocks of n bits. We use r_i to denote the i'th block of r. Let $\mathcal{DOC}$ denote the set of all documents which can be stamped. Let G be a $(K_{max}, \beta D)$ bipartite expander (V_1, V_2, E) with left degree D, where the "left" set V_1 is $\mathcal{DOC}$ and the "right" set V_2 is $[N]$. We define the three procedures Sketch, Stamp and Verify:

- $\mathrm{Sketch}(r) = \mathcal{H}, r_{|\mathcal{H}}$ where $\mathcal{H}$ has $|\mathcal{H}|$ elements selected at random from $[N]$.
- $\mathrm{Stamp}(doc, r) = r_{|\Gamma(doc)}$.
- $\mathrm{Verify}(\mathrm{Sketch}(r), doc, y) = \begin{cases} Accept & \mathrm{Sketch}(r)_{|\mathcal{H}\cap\Gamma(doc)} = y_{|\mathcal{H}\cap\Gamma(doc)} \\ Reject & otherwise \end{cases}$

Notice that $\mathrm{Sketch}(r)$ contains the restriction of r to the indices of $\mathcal{H}$, and therefore in particular contains the restriction of r to the indices of $\mathcal{H} \cap \Gamma(doc)$, and y contains the restriction of r to $\Gamma(doc)$ and therefore in particular contains the restriction of r to the indices of $\mathcal{H} \cap \Gamma(doc)$.

Theorem 4.1. *Let G be a $(K_{max}, \beta D)$-expanding graph, $\gamma > 0$ and $g = \frac{1}{5}$. Fix n large enough such that $n > \log N \geq \frac{1}{g \cdot \beta}$ and assume that $\log |\mathcal{DOC}| \leq g\beta Dn$. If $D|\mathcal{H}| \geq \frac{N}{g\beta} \ln(\frac{1}{\gamma})$ then the scheme is $(T = Dn, V = |\mathcal{H}|(n + \log N))$-efficient and α-optimal for $\alpha = \frac{1}{(1-4g)\cdot\beta}$, $\rho = 2^{-g\beta nD}$, γ and $M^*_{max} = (1-4g)\beta D n K_{max}$.*

Plugging in parameters, a corollary of this is:

Corollary 4.1.1. *For every $\eta > 0$ and large enough R we construct a timestamping scheme which is $(T = R^{1/2+\eta}, V = R^{1/2+\eta})$-efficient and $O(1)$-optimal with $\gamma = \rho = 2^{-R^{\Omega(1)}}$ and $M^*_{max} = \Omega(R)$. The timestamping scheme allows stamping documents of length $R^{\Omega(1)}$.*

We prove the corollary in the appendix (Section B). We remark that a probabilistic argument shows that there exist bipartite graphs of degree D which have expansion $(1 - o(1))D$ and using such non-explicit graphs in our construction (and setting $g = o(1)$) gives $\alpha = (1 + o(1))$ optimality (whereas the Theorem below only achieves $\alpha = O(1)$). In the remainder of the section we prove Theorem 4.1.

4.2. **Efficiency.** The verifier first chooses a random set $\mathcal{H}$ and stores it, and then stores $R_{\mathcal{H}}$. This can indeed be done online with space $V = |\mathcal{H}|(n + \log N))$. We now explain how the stamper can run online in space $T = Dn$. Observe that it can calculate the indices it will need to store *before* the random string goes by (since it knows *doc* before it sees the random string). As the indices take $D \log N < Dn$ space, it can work in place, replacing each index with the contents of the block as it goes by. We now turn to proving security.

4.3. **Security.** In Definition 3.3 we defined "successful stamping". Let $\mathcal{R}_{\text{successful}}(k) = \mathcal{R}^{\gamma}_{\text{successful}}(k)$ denote the set of random strings r on which the adversary γ-successfully stamps at least k documents. We would like to prove that $\mathcal{R}_{\text{successful}}$ has small probability. We first define a similar notion of "correct stamping":

Definition 4.2. *An adversary* correctly stamps *a document doc at r if* $\text{Stamp}^*(doc, \text{Store}^*(r)) = r_{|\Gamma(doc)}$. *An adversary* correctly stamps *a document doc at r with at most err errors, if the Hamming distance between* $\text{Stamp}^*(doc, \text{Store}^*(r))$ *and* $r_{|\Gamma(doc)}$ *is at most err.*

We let $\mathcal{R}_{\text{correct}}(k) = \mathcal{R}^{\text{err=g}\beta\text{D}}_{\text{correct}}(k)$ denote the set of random strings r for which there are at least k documents that the adversary correctly stamps with at most err errors.

The security proof has two parts.

Lemma 4.3. *Assume* $\log |\mathcal{DOC}| \leq g\beta Dn$, *$err = g\beta D$ and $n \geq \frac{1}{g \cdot \beta}$. For every $k \leq K_{max}$ and any adversary with space $M^* \leq (1 - 4g)\beta kDn$ we have* $\Pr_r[r \in \mathcal{R}_{\text{correct}}(k)] \leq 2^{-g\beta nDk}$.

We then relate $\mathcal{R}_{\text{correct}}$ and $\mathcal{R}_{\text{successful}}$:

Lemma 4.4. *Assume $D|\mathcal{H}| \geq \frac{N}{g\beta} \ln(\frac{1}{\gamma})$. For every k, $\mathcal{R}_{\text{successful}}(k) \subseteq \mathcal{R}_{\text{correct}}(k)$.*

Together,

Proof. (of Theorem 4.1) We need show that no adversary with space M^* can γ-successfully stamp more than $k = \alpha \frac{M^*}{Dn}$ documents. Notice that for $M^* \leq M^*_{max}$, $k = \frac{\alpha M^*}{Dn} \leq \frac{\alpha M^*_{max}}{Dn} = K_{max}$ and $M^* = \frac{kDn}{\alpha} = kDn(1 - 4g)\beta$. Hence, $\Pr_r[r \in \mathcal{R}_{\text{successful}}(k)] \leq \Pr_r[r \in \mathcal{R}_{\text{correct}}(k)] \leq 2^{-g\beta nDk} \leq 2^{-g\beta nD}$ where the first inequality follows by Lemma 4.4 and the second inequality follows by Lemma 4.3. The third inequality is because $k \geq 1$. □

4.4. **The Proof Of Lemma 4.4.**

Claim 4.4.1. *Fix an adversary, a string r and a document doc. If the adversary γ-successfully stamps doc at r then it correctly stamps doc at r with at most $\frac{N}{|\mathcal{H}|} \ln(\frac{1}{\gamma})$ errors.*

Proof. We prove the contrapositive. Suppose for some $doc \in \mathcal{DOC}$ and r, the timestamp provided by the adversary for doc has $err^* > err$ incorrect indices. Denote by $\mathcal{BAD}_{doc} \subset [N]$ the set of incorrect indices. The verifier catches the adversary iff $\mathcal{BAD}_{doc} \cap \mathcal{H} \neq \emptyset$, i.e. if one of the incorrect indices is in $\mathcal{H}$ (the set of indices stored by Sketch). For each index in $\mathcal{H}$, the probability that it hits $\mathcal{BAD}_{doc}$ is $\frac{err^*}{N}$, and the probability that none of them hits $\mathcal{BAD}_{doc}$ is $(1-\frac{err^*}{N})^{|\mathcal{H}|} \leq e^{-\frac{err^*|\mathcal{H}|}{N}}$ (assuming the set $\mathcal{H}$ is chosen with repetition). Hence, the adversary γ-successfully stamps doc with $\gamma \leq e^{-\frac{err|\mathcal{H}|}{N}}$. Turning that around, if the adversary γ-successfully stamps doc, then $err \leq \frac{N}{|\mathcal{H}|}\ln(\frac{1}{\gamma})$. □

In particular, for every r and doc for which the stamper is γ successful, $err \leq \frac{N}{|\mathcal{H}|}\ln(\frac{1}{\gamma}) \leq g\beta D$. Hence, the stamper correctly stamps doc at r with at most $err = g\beta D$ errors. It follows that $\mathcal{R}_{\text{successful}}(k) \subseteq \mathcal{R}_{\text{correct}}(k)$ as desired.

4.5. **The Proof Of Lemma 4.3.** We first define a compression function $\text{Com}(r)$ for $r \in \mathcal{R}_{\text{correct}}(k)$. Let $r \in \mathcal{R}_{\text{correct}}(k)$. Suppose $doc_1, \ldots, doc_k$ are the documents that the adversary correctly stamps at r with at most err errors. Denote $\Gamma = \cup_{1\leq i\leq k}\Gamma(doc_i)$, that is the set of all indices which are selected by one of the k documents. Denote $\mathcal{BAD} = \cup_{1\leq i\leq k}\mathcal{BAD}(doc_i)$, that is the set of all indices which are bad for at least one of the k documents. We call an index $j \in \Gamma \setminus \mathcal{BAD}$ useful. We choose $\text{Com}(r)$ to be:

$$\text{Com}(r) = (doc_1, \ldots, doc_k; \text{Store}^*(r); r_{|\bar{\Gamma}}; \mathcal{BAD}; r_{|\mathcal{BAD}})$$

We define a "decompression" function $\text{Dec}(a)$ that gets as input $\text{Com}(r)$ and tries to recover r. Let r be a string from $\mathcal{R}_{\text{correct}}$, i.e., a string on which the stamper correctly stamps k-documents with at most err errors. From $doc_1, \ldots, doc_k$, that appear in $\text{Com}(r)$, we recover the set Γ, and from $\text{Com}(r)$ we learn which indices are in the subset $\mathcal{BAD} \subset \Gamma$. Now, for every $1 \leq j \leq N$ we recover r_j as follows:

- If $j \notin \Gamma$ then we use the information in $r_{|\bar{\Gamma}}$ to find r_j.
- If $j \in \mathcal{BAD}$ then we use the information in $r_{|\mathcal{BAD}}$ to find r_j.
- If $j \in \Gamma \setminus \mathcal{BAD}$ then we find an i such that $j \in \Gamma(doc_i)$. We run $\text{Stamp}^*(doc_i, \text{Store}^*(r))$ and take r_j from its output.

The only case where we do not take the value of r_j directly from $\text{Com}(r)$ is for $j \in \Gamma \setminus \mathcal{BAD}$. However, all such indices j are useful, and therefore we correctly decode them. Therefore, for every $r \in \mathcal{R}_{\text{correct}}$ we have $\text{Dec}(\text{Com}(r)) = r$.

We now analyze the output length of the compression function Com. The documents $doc_1, \ldots, doc_k$ take $k \log |\mathcal{DOC}|$ bits space. $|\text{Store}^*(r)| \leq M^*$, by definition. As G is expanding and $k \leq K_{max}$, $|\Gamma| \geq \beta kD$ and therefore $r_{|\bar{\Gamma}} \leq R - \beta kDn$. We represent $\mathcal{BAD} \subseteq \Gamma$ by a binary vector of length $|\Gamma| \leq kD$ which has a "one" for indices in $\Gamma \cap \mathcal{BAD}$ and a "zero" for indices in $\Gamma \setminus \mathcal{BAD}$. Each of the k documents is correctly stamped at r with at most err errors, and therefore for every such document doc_i we have $|\mathcal{BAD}_{doc_i}| \leq err$ and $|\mathcal{BAD}| \leq k \cdot err$. The representation of $r_{|\mathcal{BAD}}$ is therefore bounded by $k \cdot err \cdot n$. We conclude that the total length of the output of Com is at most $k \log |\mathcal{DOC}| + M^* + R - \beta kDn + kD + k \cdot err \cdot n$. We denote this quantity $R - \Delta$.

As every $r \in \mathcal{R}_{\text{correct}}$ has a small description (of length $R - \Delta$) we have $|\mathcal{R}_{\text{correct}}| \leq 2^{R-\Delta}$ and therefore $Pr[r \in \mathcal{R}_{\text{correct}}] \leq 2^{-\Delta}$. We have $kD \leq g\beta kDn$ (for large enough n). We also have $err = g\beta D$ and by our assumption $\log |\mathcal{DOC}| \leq g\beta nD$. Altogether, $R-\Delta \leq R-\beta Dkn[1-3g]+M^*$. We get that $\Delta \geq (1-3g)\beta Dkn - M^*$. As $M^* \leq (1-4g)\beta kDn$ we get $\Delta \geq g\beta kDn$ as desired.

5. An Efficient Scheme Secure Against Polynomial Time Adversaries

The scheme suggested in Section 4 requires the honest parties (stamper and verifier) to store many bits, namely $TV > R$ where T is the stamp size, V the sketch size and R the random string length. In other words, if the stamp size is small (say $T = R^\eta$) then the sketch size V is almost all of the random string. Our second scheme has a very small sketch and stamp size. This is achieved by using the previous stamping scheme with a small T and using a different verification method that allows the verifier to use much less storage. This verification method is valid only against computationally bounded adversaries and takes advantage of the bounded computational capabilities of the cheating party.

5.1. **The Stamping Scheme.** Let H be a family of collision resistant hash functions and $R = \log H + Nn$. We partition a string $r \in \{0,1\}^R$ into $N+1$ blocks, denoted $r_0, r_1, \ldots, r_N$ where r_0 is of length $\log H$ and for $i > 0$, r_i is of length n. The string r_0 (which didn't appear in the previous scheme) serves as a "key" to the "hash function". We use the same "index selection" mechanism as in Section 4; G is a bipartite graph with left degree D, where the left set is the set $\mathcal{DOC}$ and the right set is the set $[N]$. We now describe the stamp, sketch and verify procedures:

Sketch$_c$**:** The verifier stores r_0 and the root of a Merkle hash tree over $r_1, \ldots, r_N$ with hash function r_0. That is $\text{Sketch}_c(r) = r_0, Root^{r_0}(r_1, \ldots, r_N)$. Note that Sketch$_c$ is *deterministic* (unlike the case of the previous section where Sketch is probabilistic).

Stamp$_c$**:** Given a document $doc \in \mathcal{DOC}$ we define $\text{Stamp}_c(doc, r) = \left(Path_j^{r_0}(r_1, \ldots, r_N)\right)_{j \in \Gamma(doc)}$. I.e., the stamper uses the function Stamp of the previous section, and for every $j \in \Gamma(doc)$ adds a "Merkle proof" $\left(Path_j^{r_0}(r_1, \ldots, r_N)\right)$ that he stamped it correctly.

Verify$_c$**:** Given a document doc, a "root" a and a stamp y composed of D Merkle-paths $p_1, \ldots, p_D$, the function $\text{Verify}_c(a, doc, y)$ accepts iff all paths are valid.

Theorem 5.1. *Let H be a $(\epsilon_H(n), \mu(n))$-collision resistant hash family which can be computed in space $O(Dn \log N)$. Assume the adversary has running time at most $\frac{\mu(n)}{poly(R,k,\log N)}$. Let G be a $(K_{max}, \beta D)$-expanding graph. Let $g = \frac{1}{5}$. Assume $\log|\mathcal{DOC}| \le g\beta Dn$ and $n > \log N \ge \frac{1}{g \cdot \beta}$. The scheme is $(T = O(Dn \log N), V = O(Dn \log N))$-efficient and $O(\alpha \cdot \log N)$-optimal against feasibly generated documents for $\alpha = \frac{1}{(1-4g)\cdot\beta}$, $\rho = 2^{-g\beta nD} + \epsilon_H$ and $M^*_{max} = (1-4g)\beta DnK_{max}$.*[6]

Plugging in the expander construction of Theorem 2.2 and collision resistant hash functions we obtain a scheme with efficiency R^η for arbitrary $\eta > 0$.

Theorem 5.2. *Let H be a $(\epsilon_H(n), \mu(n))$-collision resistant hash family with $\mu(n) = \frac{1}{\epsilon_H(n)} > p(n)$ for any polynomial p, Assume the adversary has running time polynomial in R. For every $\eta > 0$, there exists a scheme that is $(T = R^\eta, V = R^\eta)$-efficient and $O(\log R)$-optimal against feasibly generated documents for $\rho = 2^{-R^{\Omega(1)}}$ and $M^*_{max} = \Omega(R)$. The scheme allows stamping documents of length $R^{\Omega(1)}$.*

We prove Theorem 5.2 in the appendix (Section C). A more careful computation would have given $T = V = 2^{(\log\log R)^{O(1)}} = R^{o(1)}$. It is possible to get an even more efficient scheme with $T = V = (\log R)^{O(1)}$. However, this result requires a better graph than the one constructed in Theorem 2.2. It is folklore that such graphs exist by a probabilistic argument. In Appendix D we survey the parameters that can be achieved if such graphs are explicitly constructed. In the remainder of the section we prove Theorem 5.1.

5.2. **Efficiency.** The efficiency parameters are derived from the fact that it is possible to compute a path of a Merkle tree online in space $O(n \log N)$, when not taking the complexity of the hash function into account. The hash function can be computed, by our assumption, in space $O(Dn \log N)$ and therefore the verifier (who stores only the root) can run online in space

[6] Recall that the verifier is *deterministic*. Thus, γ is unimportant, and we therefore omit it in this section. To be strictly formal all the propositions in this section apply for any $\gamma > 0$.

$O(n \log N) + O(Dn \log N) = O(Dn \log N)$. The stamper calculates the indices it will need before it sees r. It computes the Merkle-paths of these leaves, and stores the labels for the paths it will need for stamping as it goes along. There are D paths where each is of length $O(n \log N)$. By using additional $O(Dn \log N)$ space to compute the hash function, the stamper can run online in space $O(Dn \log N)$. This proves the efficiency of the scheme. We now turn to proving security.

5.3. **Security.** We follow the outline of the correctness proof of the information-theoretic version of Section 4, except that now we work with security for generated documents. We show that if the adversary *successfully stamps* many documents then he *correctly stamps* many documents which is impossible by the "reconstruction argument" of the previous section.

Fix some adversary with memory M^* and running time to be specified later. We use *coins* to denote the concatenation of the random coins used by $Stamp_c^*$ and $Generate_c^*$. We define $\mathcal{R}^{\text{comp}}_{\text{correct}}(k)$ to be the set of pairs $(r, coins)$ such that, for every $1 \leq i \leq k$, the Merkle paths output by $Stamp_c^*(Store_c^*(r), Generate_c^*(r,i))$ are correct.

We now want to define the computational analogue of $\mathcal{R}_{\text{successful}}$ and relate it to $\mathcal{R}^{\text{comp}}_{\text{correct}}$. We define $\mathcal{R}^{\text{comp}}_{\text{successful}}(k) = \mathcal{R}^{\text{comp},\gamma=1}_{\text{successful}}(k)$ to be the set of all pairs $(r, coins)$ for which the adversary 1-successfully stamps the k documents $\{\text{Generate}_c^*(r,i)\}$ (i.e. all the Merkle paths output by $Stamp_c^*(Store_c^*(r), Generate_c^*(r,i))$ are valid). This definition of success corresponds to the notion of security in Definition 3.5. We prove:

Lemma 5.3. *Assume the adversary has running time at most* $\frac{\mu(n)}{poly(R,k,\log N)}$. *Then,*

$$\Pr_{r,coins}\left[(r, coins) \in \mathcal{R}^{\text{comp}}_{\text{successful}}(k) \setminus \mathcal{R}^{\text{comp}}_{\text{correct}}(k)\right] \leq \epsilon_H$$

Thus, we have that $\Pr[\mathcal{R}^{\text{comp}}_{\text{successful}}(k)] \leq \Pr[\mathcal{R}^{\text{comp}}_{\text{correct}}(k)] + \epsilon_H$. We then show (using Lemma 4.3):

Lemma 5.4. *Assume* $\log |\mathcal{DOC}| \leq g\beta Dn$, $err = g\beta D$ *and* $n \geq \frac{1}{g \cdot \beta}$. *For any adversary with space* $M^* \leq (1-4g)\beta k D n$ *and* $k \leq K_{max}$ *we have* $\Pr_{r,coins}[(r, coins) \in \mathcal{R}^{\text{comp}}_{\text{correct}}(k)] \leq 2^{-g\beta n D k}$.

We prove Lemmas 5.3 and 5.4 in the appendix (Section E). Together, Lemmas 5.3 and 5.4 imply that $\Pr_{r,coins}[(r, coins) \in \mathcal{R}^{\text{comp}}_{\text{successful}}(k)] \leq 2^{-g\beta n D k} + \epsilon_H$. Let $T = cDn \log N$ (where c is the constant hidden in the $O(\cdot)$ notation in the assumption of Theorem 5.1). Then for $k = c\alpha \log N \frac{M^*}{T} \leq \alpha \log N \frac{M^*_{max}}{T} = K_{max}$, the probability that an adversary can generate and stamp more than k documents is less than $2^{-g\beta n D k} + \epsilon_H \leq 2^{-g\beta n D} + \epsilon_H = \rho$, which proves Theorem 5.1.

5.4. **Stamping Longer Documents.** The timestamping system described in this section has good security parameters but only works if $\log |\mathcal{DOC}| = R^a$ for a small constant a. This is somewhat small. (Recall that honest parties run in time polynomial in R). However, using the standard technique of collision resistant hashing we can convert any timestamping system which works for short documents into a system which works for longer documents by first hashing the documents and then stamping them. The exact details are given in appendix F. When applying this to the scheme of Theorem 5.2 we get a scheme which allows stamping even documents of length R. This requires the same assumption on collision resistant hash functions which was used to derive Theorem 5.2.

6. Discussion And Open Problems

Adversaries need not run online. Our passive timestamping schemes are secure against adversaries with bounded storage. Note that we do not require such adversaries to run online. The adversary may store an *arbitrary function* which maps the string r into few bits. No restrictions are placed on the complexity of this function in our first scheme. In our second scheme we require that this function is computable in polynomial time.

Online versus locally-computable. The strategies for the honest players are efficient in the sense that they work online using small space and polynomial time. A stronger notion of efficiency called "locally-computable" was suggested in [25]. It requires the honest players to store a small substring of the string r. More precisely, the players need to choose a subset $S \subseteq [R]$ before the random string is transmitted and only store $r|_S$. We point out that our first scheme has this additional property, whereas our second scheme does not.[7] Natural open problems are whether our first scheme can be improved to yield better parameters, and whether our second scheme can be improved to run with strategies that are locally computable.

Explicit expander graphs. In Appendix D we show that our method can give a scheme with poly-logarithmic efficiency if certain expander graphs are explicitly constructed. This is an additional motivation to the vast effort spent in recent years to construct expander graphs with optimal parameters.

References

[1] Y. Aumann, Y.Z. Ding, and M. O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48, 2002.

[2] Y. Aumann and M. O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Crypology — CRYPTO '99*, volume 1666, pages 65–79, 1999.

[3] D. Bayer, S. Haber, and W. S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In R. M. Capocelli et al., editor, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. Springer-Verlag, Berlin Germany , New York, 1992.

[4] J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991.

[5] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. *Lecture Notes in Computer Science*, 765:274.

[6] Ahto Buldas and Peeter Laud. New linking schemes for digital time-stamping. In *Information Security and Cryptology*, pages 3–13, 1998.

[7] Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-stamping with Binary Linking Schemes. In Hugo Krawczyk, editor, *Advances on Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 486–501, Santa Barbara, USA, August 1998. Springer-Verlag.

[8] Ahto Buldas, Helger Lipmaa, and Berry Schoenmakers. Optimally efficient accountable time-stamping. In *Public Key Cryptography*, pages 293–305, 2000.

[9] Christian Cachin, Claude Crepeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *IEEE Symposium on Foundations of Computer Science*, pages 493–502, 1998.

[10] Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, 1997.

[11] Yan Zong Ding. Oblivious transfer in the bounded storage model. *Lecture Notes in Computer Science*, 2139:155, 2001.

[12] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *Theory of Cryptography — TCC '04*, volume 2951, Cambridge, MA, USA, February 2004. Springer-Verlag. To appear.

[13] Y.Z. Ding and M.O. Rabin. Hyper-encryption and everlasting security. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 1–26, 2002.

[14] Stefan Dziembowski and Ueli Maurer. Tight security proofs for the bounded-storage model. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 341–350. ACM, May 2002.

[15] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Lecture Notes in Computer Science*, 537:437.

[16] Stuart Haber and W. Scott Stornetta. Secure names for bit-strings. In *ACM Conference on Computer and Communications Security*, pages 28–35, 1997.

[17] C. Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In *Advances in Cryptology — CRYPTO '02*, volume 2442, pages 257–271. Springer, 2002.

[7] In our second scheme, both stamper and verifier read blocks of the string r online, and need to "hash them" quickly before reading the next incoming blocks. Thus, to apply this our second scheme, one needs to use hash functions which can be computed very efficiently.

[18] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.
[19] U. Maurer. Secret key agreement by public discussion. *IEEE Transaction on Information Theory*, 39(3):733–742, 1993.
[20] Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, pages 218–238. Springer-Verlag New York, Inc., 1989.
[21] R. Raz, O. Reingold, and S. Vadhan. Error reduction for extractor. In *40th IEEE Symposium on Foundations of Computer Science*, pages 191–201, 1999.
[22] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28:1433–1459, 1999.
[23] Amnon Ta-Shma. Storing information with extractors. *Information Processing Letters*, 83(5):267–274, September 2002.
[24] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 143–152, New York, NY, USA, 2001. ACM Press.
[25] S.P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In *Advances in Cryptology — CRYPTO '03*. Springer, 2003.

Appendix A. An Explicit Expander Construction

The expander construction of Theorem 2.2 is a combination of the construction of [24] and a modification of the construction of [23] (which in turn relies on a construction of "randomness extractors" from [21]). In this Appendix we give a sketch of this argument. We assume the reader has some background in the area of "randomness extractors". More precise details will appear in the final version.

It is convenient to work with the notion of universal extractors.

Definition A.1 (universal extractor)**.** [21, 23]
We say $F : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^{m(k_{\max})}$ is a strong $(k_{\max}, k \mapsto m(k), \epsilon)$ **universal** *extractor if for every $k' \le k_{\max}$ there is a restriction $F' : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m(k')}$ of F which is a strong (k', ϵ)–extractor. If $m(k) = k - \Delta$, we say F has Δ entropy loss.*

We translate a universal extractor $F : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^{m(k_{\max})}$ to a bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = N = 2^n$, $|V_2| = D \cdot 2^{m(k_{max})}$, left degree $D = 2^d$ where we put an edge $(v_1, (y, v_2)) \in E$ iff $F(v_1, y) = v_2$. If F is a strong $(k_{\max}, k \mapsto m(k), \epsilon)$ universal extractor with $\Delta = O(1)$ entropy loss, then G is $(K_{max} = 2^{k_{max}}, \beta)$ expanding, with $\beta = \frac{1}{2}\frac{1}{2^\Delta}(1-\epsilon)$ [24]. We prove:

Lemma A.2. *(based on [21, 24]) For every $k_{\max} \le n$ and every $\epsilon > 0$ there exists an explicit strong $(k_{\max}, k \mapsto k - \Delta, \epsilon)$ universal extractor $F : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^{k_{\max}-\Delta}$ with $d = O(\log n + \log^3(\frac{k_{\max}}{\epsilon}))$ and $\Delta = O(1)$.*

Choosing, say, $\epsilon = \frac{1}{2}$ gives us the desired graph with some constant $\beta > 0$, and $|V_2| = D \cdot 2^{m(k_{max})} = D \cdot 2^{k_{max}-\Delta} = D \cdot \frac{K_{max}}{2^\Delta} = 4DK_{max}\beta$.

Proof. (Of Lemma A.2) We first note that if E_1, E_2 are strong $(k_{\max}, k \mapsto m_i(k), \epsilon_i)$ universal extractors, then their concatenation $E_1 \circ E_2$ (where $E_1 \circ E_2(x; y_1, y_2) = E_1(x, y_1) \circ E_2(x, y_2)$) is a strong $(k_{\max}, k \mapsto m_1(k) + m_2(k - m_1(k)), \epsilon_1 + \epsilon_2)$ universal extractor (see, e.g., [23]).

Raz, Reingold and Vadhan [21] explicitly construct a strong extractor with optimal entropy loss and $d = O(\log^3(\frac{n}{\epsilon}))$, by concatenating many calls to a strong extractor E_{RRV} that extracts half the entropy, and an extractor E_{SZ} of Srinivasan and Zuckerman [22]. [21] show the extractor E_{RRV} is universal [8]. We soon show that E_{SZ} is also universal, and therefore their whole construction is universal.

[8]An informal statement of this, with a brief explanation, appears in the conclusions section of [21]. We briefly sketch how to get this result from the [21] paper. [21] define weak designs, and also discuss a variant of it where

This gives the desired strong universal extractor, with $d = O(\log^3(\frac{n}{\epsilon}))$. Having that, we observe that by first condensing the source using the lossless condenser of Ta-Shma, Umans and Zuckerman [24], we can further reduce d to $d = O(\log n + \log^3(\frac{k_{\max}}{\epsilon}))$, as desired.

To see that E_{SZ} is universal, we go back to their proof. Srinivasan and Zuckerman show that if $H = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is ϵ^2 close to uniform in the ℓ_1 norm, meaning that for every $x_1 \neq x_2$, $H(x_1) \circ H(x_2)$ is ϵ^2 close to uniform, then for every source X with min-entropy $k \geq m + 2\log(1/\epsilon) + O(1)$, $(H, H(X))$ is ϵ close to uniform. We now show a prefix property of the extractor E_{SZ}. For $k \leq k_{max}$, we look at a prefix $h' : \{0,1\}^n \to \{0,1\}^{m'}$ of h, where $m' = k - 2\log(1/\epsilon) - O(1) \leq k_{max} - 2\log(1/\epsilon) - O(1)$, and $H' = \left\{h' : \{0,1\}^n \to \{0,1\}^{m'}\right\}$. Clearly, H' is ϵ^2 close to uniform in the ℓ_1 norm, and so $(H, H'(X))$, which is a prefix of $(H, H(X))$, is ϵ close to uniform, as desired. ☐

Appendix B. Proof Of Corollary 4.1.1

We plug into Theorem 4.1 the $(K_{max}, \beta D)$-expanding graph of Theorem 2.2 to get a concrete result, where β is the constant guaranteed in Theorem 2.2.

Theorem B.1. *Let $g = \frac{1}{5}$, and let n and N be large enough such that $n > \log N \geq \frac{1}{\beta g}$ and $N^\eta \geq \frac{1}{g\beta}$ for a given constant $\eta > 0$. The timestamping scheme with the expander of Theorem 2.2 is $(T = V = N^{1/2+\eta}n)$-efficient and is α-optimal for $\alpha = \frac{1}{(1-4g)\cdot\beta}$, $\rho = \gamma = 2^{-N^\eta}$, and $M^*_{max} = \frac{1-4g}{4}nN$. The timestamping scheme allows stamping documents of length $N^{\Omega(1)}$.*

Proof. Let $\eta > 0$ be some constant. We set $|\mathcal{H}| = D = N^{1/2+\eta}$ and $\gamma = 2^{-N^\eta}$. Thus, $D|\mathcal{H}| = N^{1+2\eta} \geq \frac{N}{g\beta}\ln(\frac{1}{\gamma})$, where the inequality follows because $N^\eta \geq \frac{1}{g\beta}$. We use the explicit expander of Theorem 2.2 where:

- We set $K_{max} = \frac{N}{4D\beta}$,
- and we let $V_1 = \mathcal{DOC}$ be the largest integer so that the expander of Theorem 2.2 gives an expander with degree D.

We first show that $\log|\mathcal{DOC}| \geq D^{\Omega(1)}$. By Theorem 2.2 we have that:

$$D \leq 2^{a[\log\log\mathcal{DOC} + (\log\log K_{max})^3]} \tag{1}$$

for some constant a. Setting $\log|\mathcal{DOC}| = D^{\frac{1}{2a}}$ we can get a graph with degree bounded by $\sqrt{D}\cdot 2^{a(\log\log K_{max})^3} \leq \sqrt{D}2^{a(\log\log N)^3} \leq D$ as required (where the two last inequalities follow by $K_{max} \leq N$ and the fact that $D \geq \sqrt{N}$).

We now apply Theorem 4.1. Note that by our choice of parameters $\log|\mathcal{DOC}| \leq D \leq Dn\beta g$ where the last inequality follows as $n \geq \frac{1}{g\cdot\beta}$. We have that the scheme is $(T = V = N^{1/2+\eta}n)$-efficient and α-optimal for $\alpha = \frac{1}{(1-4g)\cdot\beta}$, $\rho = 2^{-g\beta nD} \leq 2^{-D}$, and $M^*_{max} = (1-4g)\beta DnK_{max} = \frac{1-4g}{4}nN$ ☐

Corollary 4.1.1 is now a simple special case.

Appendix C. Proof of Theorem 5.2

Let $\eta > 0$ be a constant. We use the explicit expander of Theorem 2.2, with left-hand set $\mathcal{DOC}$, right-hand set of size N, left degree D, and which is $(K_{max}, \beta D)$-expanding, where we set

- $K_{max} = \frac{N}{4\beta D}$ and

$\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho(i-1)$. We need the variant, and we name it prefix-weak design. [21] study Trevisan's extractor with prefix-weak design and show that for every $k' \leq k$ if we look at just the first $k'/2$ output bits we get a strong extractor. This gives a strong universal extractor with $d = O(\log^2(\frac{n}{\epsilon}))$ and $m(k') = \frac{k'}{2}$.

- $|V_1| = |\mathcal{DOC}| = 2^{D^{1/2a}}$, where the constant $a > 0$ is the constant hidden in the $O(\cdot)$ in Theorem 2.2.

The degree of this graph is

$$2^{a[\log (D^{1/2a})+(\log\log K_{max})^3]} = \sqrt{D}\cdot 2^{a(\log\log K_{max})^3} \leq \sqrt{D}2^{a(\log\log N)^3} \leq D$$

where the inequalities follow by $K_{max} \leq N$, and by choosing $D = N^e$ where $e > 0$ is some constant to be chosen later [9]. We choose $n = N^{e'}$ where $e' > 0$ is a constant to be chosen later. We observe that $\mu(n)$ and $\frac{1}{\epsilon_H(n)}$ are greater than any polynomial in N and that $N = \frac{R}{n} \geq \sqrt{R}$. We choose the constant e' as a function of the constant e (which still needs to be determined). We choose e' small enough so that the space needed to compute the hash function (which is some fixed polynomial in $n = N^{e'}$) is at most $D = N^e$. We conclude that the space of the hash function is bounded by $D \leq Dn\log N$ as required. We meet all the requirements of Theorem 5.1 and conclude that the scheme is (T, V)-efficient for $T = O(Dn\log N) \leq N^{3e} \leq R^\eta$ for sufficiently small e. We also have that $V = O(n\log N) + n^{O(1)} \leq O(N^{2e}) + n^{O(1)} \leq R^\eta$. Furthermore, $M^*_{max} = \Omega(K_{max}Dn) = \Omega(Nn) = \Omega(R)$ as required, and the scheme is $O(\log R)$-optimal for $\rho = 2^{-D} \leq 2^{-R^{\Omega(1)}}$. The scheme can stamp documents of length $D^{\Omega(1)} \geq R^{\Omega(1)}$.

Appendix D. A Scheme Using a Non-Explicit Expander Graph

It is folklore (and easy to verify) that for some constant $\beta > 0$ there exists a $(K_{max}, \beta D)$ expanding bipartite graph G with left degree D, left-hand side having $2^{D^{\Omega(1)}}$ vertices and right-hand side having $N = K_{max}D$ vertices. (Actually, a probabilistic argument gives even stronger results.) We show that an explicit construction of such a graph implies much better passive time-stamping schemes. Specifically,

Theorem D.1. *Assume a graph G as above can be explicitly constructed. Let H be a $(\epsilon_H(n), \mu(n))$-collision resistant hash family with $\epsilon_H(n) = 2^{-n^{\Omega(1)}}$ and $\mu(n) = 2^{n^{\Omega(1)}}$. Assume the adversary has running time polynomial in R. For every $c > 0$, there exists a scheme that is $(T = (\log R)^{O(1)}, V = (\log R)^{O(1)})$-efficient and $O(\log R)$-optimal against feasibly generated documents for $\rho = 2^{-(\log^c R)}$, $M^*_{max} = \Omega(R)$. The scheme allows stamping documents of length $\log^c R$.*

Proof. We now use non-explicit expanders. We choose n to be large enough so that $\mu(n)$ and $1/\epsilon_H(n)$ are greater than $N^{\log^c N}$. We can meet this requirement with $n = (\log N)^{O(1)}$. This makes $N = R/n \geq \sqrt{R}$. Let d be a constant to be chosen later. We choose $D = \log^d R$. We meet all the requirements of Theorem 5.1 and conclude that the scheme is $((\log R)^{O(1)}, (\log R)^{O(1)})$-efficient for $M^*_{max} = \Omega(K_{max}Dn) = \Omega(Nn) = \Omega(R)$ as required and that it is $O(\log R)$-optimal for $\rho = 2^{-D} \leq 2^{-\log^c R}$ for sufficiently large d. The scheme can stamp documents of length $D^{\Omega(1)} \geq \log^c R$ for large enough d. ☐

We can now stamp much longer documents (even up to length R) using the techniques of Section 5.4.

Appendix E. Proving Security Against Bounded Adversaries

E.1. **Proof Of Lemma 5.3.** Assume not. Then, given r and a random h we can run the adversary k times to find a valid, incorrect path for $Tree^h(r_1, \ldots, r_N)$ with probability ϵ_H (for each run we check if any of the D paths output by the adversary is valid but incorrect). This algorithm runs in time less than $\mu(n)/poly(R)$, contradicting the security promised by Lemma 2.6.

[9] In fact, we can even work with $D = 2^{poly\log\log(N)}$, but for simplicity we work with larger D's

E.2. **Proof Of Lemma 5.4.** By an averaging argument there exists some fixed value $coins'$ of $coins$, such that

$$\Pr_r[(r, coins') \in \mathcal{R}^{\text{comp}}_{\text{correct}}(k)] \geq \Pr_{r,coins}[(r, coins) \in \mathcal{R}^{\text{comp}}_{\text{correct}}(k)]$$

Let $\mathcal{R}_{\text{correct}}(k) = \{r|(r, coins') \in \mathcal{R}^{\text{comp}}_{\text{correct}}(k)\}$. Note that $\mathcal{R}_{\text{correct}}(k) \subseteq \mathcal{R}^{\text{err=0}}_{\text{correct}}(k)$ where the latter is the set defined in Section 4 with zero errors. This follows because the adversary with coins fixed to $coins'$ outputs correct paths on k different documents, and every such path contains the correct leaf. Thus, if there exists an adversary A for which $Pr_{r,coins}\ [(r, coins) \in \mathcal{R}^{\text{comp}}_{\text{correct}}(k)]\ > 2^{-g\beta nDk}$, then there exists a deterministic algorithm A' (which may be non-uniform), for which $Pr_r[r \in \mathcal{R}_{\text{correct}}(k)] > 2^{-g\beta nDk}$. By Lemma 4.3, for any adversary with space $M^* \leq (1-4g)\beta kDn$ and $k \leq K_{max}$,

$$\Pr_r[r \in \mathcal{R}_{\text{correct}}(k)] \quad \leq \quad 2^{-g\beta nDk}$$

(In fact the parameters are slightly better because we have $err = 0$ and therefore $\mathcal{BAD}$ is empty). Note that Lemma 4.3 is true for *any* adversary – it makes no assumptions about its running time or uniformity – therefore our use of A' instead of A is allowed.

Appendix F. Hashing Documents Before Stamping Them

We explain how to take a scheme which stamps only short documents and extend it to a scheme which stamps longer documents. Let Stamp, Sketch and Verify be the stamping and verification functions for the original scheme, which stamps documents of length $\log|\mathcal{DOC}|$. Assume there exists $H : \{0,1\}^R \mapsto \{0,1\}^n$ that is an $(\epsilon_H(n), \mu(n))$-collision resistant hash family. We will use this hash function with $n = \log|\mathcal{DOC}|$.

Given r, we partition it into h, r', where $|h| = \log H$ will be used to select a function from H. We define the new procedures $\text{Sketch}_{\text{long}}, \text{Stamp}_{\text{long}}$ and $\text{Verify}_{\text{long}}$.

- $\text{Sketch}_{\text{long}}(r) = h, \text{Sketch}(r')$
- $\text{Stamp}_{\text{long}}(doc, r) = \text{Stamp}(h(doc), r')$.
- $\text{Verify}_{\text{long}}(\text{Sketch}_{\text{long}}(r), doc, y) = \text{Verify}(\text{Sketch}(r'), h(doc), y)$.

Theorem F.1. *Suppose the original system is a (T, V)-efficient timestamp system that is α-optimal against feasibly generated documents with parameters ρ and γ for adversaries with running time polynomial in R. Assume $\mu(\log|\mathcal{DOC}|)$ is larger than any polynomial in R and that $Space(H) \leq T$. Then the derived scheme allows stamping documents of length R. The scheme is also $(T, V + \log H)$-efficient and α-optimal for $\rho' = \rho + \epsilon_H(\log|\mathcal{DOC}|)$ and $\gamma' = \gamma$ against polynomial time adversaries.*

Proof. Suppose for a particular r and random coins the adversary can generate and γ-successfully stamp $\alpha\frac{M^*}{T}$ documents in the new system. If $|\{h(\text{Generate}^*_{\text{c}}(r, i))\}| > \alpha\frac{M^*}{T}$ this adversary can be used to γ-successfully stamp $\alpha\frac{M^*}{T}$ documents in the original system. Therefore the probability that this event occurs is at most ρ (the adversary against the original system runs the new adversary and then hashes each document it outputs). If $|\{h(\text{Generate}^*_{\text{c}}(r, i))\}| \leq \alpha\frac{M^*}{T}$, then by the pigeonhole principle there must be $i \neq j$ for which $h(\text{Generate}^*_{\text{c}}(r, i)) = h(\text{Generate}^*_{\text{c}}(r, j))$, thus we can find a collision for h in time polynomial in R. Therefore the probability that this event occurs is at most $\epsilon_H(\log|\mathcal{DOC}|)$. The total probability of success is thus bounded by $\rho + \epsilon_H(\log|\mathcal{DOC}|)$. □