# Authentication in the Bounded Storage Model

Yevgeniy Dodis*, Willy Quach**, and Daniel Wichs***

**Abstract.** We consider the streaming variant of the Bounded Storage Model (BSM), where the honest parties can stream large amounts of data to each other, while only maintaining a small memory of size $n$. The adversary also operates as a streaming algorithm, but has a much larger memory size $m \gg n$. The goal is to construct unconditionally secure cryptographic schemes in the BSM, and prior works did so for symmetric-key encryption, key agreement, oblivious transfer and multi-party computation. In this work, we construct *message authentication and signatures* in the BSM.

First, we consider the symmetric-key setting, where Alice and Bob share a small secret key. Alice can authenticate arbitrarily many messages to Bob by streaming long authentication tags of size $k \gg m$, while ensuring that the tags can be generated and verified using only $n$ bits of memory. We show a solution using local extractors (Vadhan; JoC '04), which allows for up to exponentially large adversarial memory $m = 2^{O(n)}$, and has tags of size $k = O(m)$.

Second, we consider the same setting as above, but now additionally require each individual tag to be small, of size $k \leq n$. We show a solution is still possible when the adversary's memory is $m = O(n^2)$, which is optimal. Our solution relies on a space lower bound for leaning parities (Raz; FOCS '16).

Third, we consider the public-key signature setting. A signer Alice initially streams a long verification key over an authentic channel, while only keeping a short signing key in her memory. A verifier Bob receives the streamed verification key and generates a short verification digest that he keeps in his memory. Later, Alice can sign arbitrarily many messages using her signing key by streaming large signatures to Bob, who can verify them using his verification digest. We show a solution for $m = O(n^2)$, which we show to be optimal. Our solution relies on a novel entropy lemma, of independent interest. We show that, if a sequence of blocks has sufficiently high min-entropy, then a large fraction of individual blocks must have high min-entropy. Naive versions of this lemma are false, but we show how to patch it to make it hold.

* NYU. E-mail:dodis@cs.nyu.edu Partially supported by gifts from VMware Labs and Google, and NSF grants 1619158, 1319051, 1314568.

** Northeastern University. Email:quach.w@northeastern.edu. Part of this work was done during an internship at NTT Research.

*** Northeastern and NTT Research. E-mail:wichs@ccs.neu.edu. Partially supported by NSF grants CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

## 1 Introduction

It is well known that there are strong restrictions on what cryptography can achieve without imposing any limits on the adversary's resources. In particular, Shannon showed that Alice and Bob cannot communicate secretly over an insecure channel, unless they share a secret key that is at least as large as the total size of the messages exchanged. Similarly, Alice and Bob cannot ensure the authenticity of their communication over an insecure channel, unless they share a secret key that is at least as large at the total number of messages exchanged. In either case, no security is possible in the public-key setting, when Alice and Bob have no shared secrets to begin with. Traditionally, cryptography has focused on overcoming the above limitations by restricting the adversary Eve to run in polynomial time. Unfortunately, the security of such schemes relies on unproven computational hardness assumptions, and at the very least, requires that $P \neq NP$.

The *Bounded Storage Model (BSM)* [23] offers an alternative by bounding the attacker's space instead of her run time. The goal is to construct unconditionally secure cryptographic schemes in the BSM, without relying on any unproven assumptions. A long series of prior works [3, 2, 4, 16, 1, 10, 22, 27, 5, 25, 21, 26, 12, 14, 9] showed how to achieve symmetric-key encryption, key agreement, oblivious transfer and multiparty computation in the BSM. In this work, we show how to achieve (symmetric-key) message authentication and (public-key) signatures in the BSM. We first begin by describing the BSM and survey what was done in prior work. We then turn to the problem of authentication in the BSM, and describe our results.

*The Bounded Storage Model (BSM).* In the bounded storage model (BSM), we restrict the storage capacity rather than the run-time of the adversary. Two distinct variants of the BSM exist in the literature, and we follow the terminology of [9] to distinguish between them.

The default variant for this work will be the *streaming* BSM [25, 14, 9]. In the streaming BSM, parties are modeled as streaming processes, that can send and receive large amounts of data between them by processing it one bit at a time, while only maintaining a small amount of memory. We denote the memory bound for the honest parties by $n$ and the memory bound for the adversary by $m$. Typically, we assume the adversary has much higher resources than the honest parties, namely $m \gg n$. Although the honest parties only have a small amount of memory, they can stream large amounts of communication $k \gg m$, so as to overwhelm the memory capacity of the adversary and prevent it from storing the communication between them in its entirety. Still, the honest parties are restricted to remembering even less about the exchanged communication than the adversary can remember! Surprisingly, as we will see, this suffices to construct many powerful cryptographic primitives in the BSM.

We contrast the streaming BSM with an earlier variant, referred to as the *traditional BSM* [23]. In the traditional BSM, a trusted third party (potentially, some natural process) broadcasts a long uniformly random string $X \in \{0,1\}^k$

referred to as a *randomizer string*. The honest parties can store a subset of $n$ physical locations of $X$, while the adversary can store any $m$ bis of information about $X$. After that, the string $X$ disappears, and the adversary gets unrestricted additional memory.

Most cryptographic schemes in the traditional BSM naturally translate to the streaming BSM by having one of the parties generate and stream the randomizer string $X$. However, looking forward, this will not be the case when it comes to authentication. The traditional BSM guarantees that the honest parties all get access to the same trusted randomizer string $X$, while in the streaming model, the adversary can tamper with any value $X$ sent by one honest party to another. This makes the authentication problem in the streaming BSM more natural, but potentially harder than in the traditional BSM, since the latter already implicitly assumes a mechanism for authenticating $X$.

On the other hand, the traditional BSM has additional features that we don't require in the streaming BSM: (a) in the traditional BSM, there is only one long randomizer string $X$ used in each cryptographic operation, whereas in the streaming BSM, the parties can stream many long messages, (b) in the traditional BSM, the long string $X$ is uniformly random, while in the streaming BSM, the streamed messages can have arbitrary structure, (c) in the traditional BSM, the honest parties can only access a small subset of locations in $X$, while in the streaming BSM, the parties read the entire communication while maintaining small local memory, (d) in the traditional BSM, the adversary is only limited to remembering $m$ bits of information about $X$ and gets unlimited memory otherwise, while in the streaming BSM, the adversary observes communication in a streaming manner and is limited to $m$ bits of memory overall. While some of our schemes do achieve some of these additional features, we view this as being of secondary importance, and focus on the streaming BSM as our default model.

*Prior Work in the BSM.* The seminal work of Maurer [23] introduced the traditional BSM. A series of papers [23, 6, 1, 10, 22, 27] showed how to achieve symmetric-key encryption in the traditional BSM, where the honest parties share a short secret key that they can use to encrypt arbitrarily many messages (i.e., CPA security). Each encryption relies on a fresh randomizer string $X$. The main technical tool in all these works was nicely abstracted in the work of Vadhan [27] as a locally computable (seeded) extractor, which can extract randomness from a long string $X$ by only reading a few locations of $X$. Overall, these works show that symmetric-key encryption is possible even if the adversarial memory $m = 2^{O(n)}$ can be up to exponentially larger than honest user memory $n$, as long as the ciphertext size (in which we include the size of the randomizer string) can be sufficiently large $k = O(m)$.[1] In particular, if we also want the ciphertext size to be polynomial, then we have to restrict the adversarial memory $m = \text{poly}(n)$ to be some arbitrarily large polynomial.

The works of [3, 2, 4, 16, 5] showed that it is also possible to achieve public key agreement, oblivious transfer, and general multi-party computation with

[1] Throughout the introduction, we ignore polynomial factors in the security parameter.

arbitrary corruptions thresholds in the traditional BSM when $m = O(n^2)$, and this was shown to be optimal by [11].

All of the above results in the traditional BSM also carry over to the streaming BSM, by having one of the honest parties generate the randomizer string $X$. However, it turns out that one can often do better in the streaming BSM. The work of Raz [25] proved memory lower bounds for learning parity, and used these to construct novel symmetric-key encryption in the streaming BSM with small ciphertexts of size $< n$, and with $m = O(n^2)$, which is tight. In particular, each individual ciphertext is small enough that it can even fully fit in honest user memory, but the adversary does not have enough memory to remember too many ciphertexts. Follow-up works [21, 26, 12] generalized this result and showed that, for up to exponentially large $m = 2^{O(n)}$, one can achieve ciphertext size $O(m/n)$, improving by a factor of $n$ over the prior results in the traditional BSM.[2] The work of [14] studied key agreement and oblivious transfer in the streaming BSM, and showed how to use Raz's lower-bound to construct new protocols that improve over the prior works in the traditional BSM by removing any correctness error and mildly reducing the number of rounds and the communication cost, while still requiring that the adversarial memory is bounded by $m = O(n^2)$. The recent work of [9] noticed that the lower bound of $m = O(n^2)$ from [11] does not hold in the streaming BSM, and showed how to achieve key agreement, oblivious transfer, and general multiparty computation in the streaming BSM, even when the adversarial memory $m = 2^{O(n)}$ can be up to exponentially larger than honest user memory $n$, at the cost of growing the round and communication complexities polynomially with $m$.

The work of [6] also considered symmetric-key authentication in the traditional BSM, where parties have a short shared secret key and can use it to authenticate a large number of messages, by using a fresh (authentic) randomizer string $X$ to authenticate each message. The construction is very simple – it uses a local extractor to extract a fresh secret key for a one-time message-authentication code (1-time MAC) from $X$, and then uses the 1-time MAC with this key to authenticate the message. Unfortunately, unlike all the previous results in the traditional BSM, this solution cannot immediately be ported to the streaming BSM: if one of the parties generates $X$ and streams it, then the adversary can tamper with it. Indeed, this would result in a completely insecure scheme in the streaming BSM.

### 1.1 Our Results

In this work, we study the problem of message authentication and signatures in the streaming BSM. We study three variants of the problem, as outlined below.

*Symmetric-Key Authentication.* We start with the symmetric-key setting, where Alice and Bob share a short secret key $\mathsf{sk}$, that they store in their $n$-bit memory.

[2] The above bound is for 1-bit messages and is optimal; if the adversary can store $> n$ ciphertexts under an $n$-bit key, then she can learn something about the encrypted messages via the Shannon lower bound.

To authenticate a message $\mu$, Alice streams a long authentication tag $\sigma$ to Bob who verifies the stream and either accepts or rejects.[3] Even though the tag size $k = |\sigma|$ can be very large, namely $k \gg m$, we ensure that generating and verifying the tag can be done in a streaming manner using only $n$ bits of memory. Our goal is to achieve unconditional security, where Alice and Bob can authenticate an arbitrarily large polynomial (or even sub-exponential) number of messages in this manner using their short secret key. We consider a streaming attacker Eve with $m$ bits of memory. Eve sits on the channel between Alice and Bob, and can observe and modify all communication between them, subject only to keeping at most $m$ bits of memory throughout this process. She performs a chosen-message attack, where she can adaptively choose many messages and observe the authentication tags honestly sent from Alice to Bob. Moreover, she can modify each of the messages and tags in transit and can observe whether Bob accepts or rejects. (We view the process of receiving and modifying the tags as a streaming algorithm, where Eve receives each tag as an input stream and produces a modified tag as an output stream. The streams need not be synchronized and Eve can wait to receive many/all of the bits of the input stream before outputting the first bit in the output stream.) Eve wins if she causes Bob to accept some arbitrary message that Alice did not authenticate.

We show how to solve this problem with up to exponentially large adversarial memory $m = 2^{O(n)}$ and tags of size $k = O(m)$. In particular, this means that if we also want the scheme to also be polynomially efficient (i.e., have polynomial-size tags), then we have to restrict the adversarial memory $m = \text{poly}(n)$ to be some arbitrarily large polynomial. We refer to Theorem 2 for a formal statement with parameters.

*Symmetric-Key Authentication with Short Tags.* Next, we consider the same problem as previously, but now also require that each individual tag $\sigma$ is "short", of size $< n$. Moreover, each tag can be fully generated, stored and verified by the honest parties in $n$ bits of memory, without needing to operate in the streaming model. We show how to solve this problem for $m = O(n^2)$, which is optimal.[4] This result relies on Raz's space lower bounds for learning parity [25]. We refer to Theorem 4 for a formal statement with parameters.

*Public-Key Signatures.* Lastly, we consider the public-key signature setting, where the honest parties do not have any shared secrets to begin with. As in the standard signature setting, we have an initialization stage where Alice generates a public verification key $\mathsf{vk}$ and a secret signing key $\mathsf{sk}$, and she broadcasts her verification key $\mathsf{vk}$ to other honest users over a public but authentic channel. In the BSM, we allow the verification key $\mathsf{vk}$ to be large, of size $> m$, and Alice

[3] We typically think of the messages $\mu$ as small relative to $n, m$. However, all of our results also extend to support long messages that Alice and Bob receive in a streaming manner.

[4] Optimality follows from the standard authentication lower bound; if an adversary has enough memory to store $n$ tags (generated using an $n$ bit key) then it has enough information to forge tags of new messages.

generates and broadcasts $\mathsf{vk}$ in a streaming manner while maintaining $n$ bits of memory, which also bounds the size of the secret signing key $\mathsf{sk}$ that Alice retains in memory at the end of the initialization stage. An arbitrary verifier Bob receives and processes the streamed verification key $\mathsf{vk}$ using $n$ bits of memory, and derives a short *verification digest* $\mathsf{vd}$, which he keeps in his memory at the end of the initialization stage. Later, Alice can use her signing key $\mathsf{sk}$ to sign arbitrarily many messages by streaming large signatures over a fully insecure channel, and Bob can verify the streamed signatures using his verification digest $\mathsf{vd}$. The adversary Eve has $m$ bits of memory. During the initialization stage, Eve can observe (but not modify) the streamed verification key $\mathsf{vk}$. Afterwards, Eve performs a chosen-message attack, where she can adaptively choose many messages and observe the signatures sent from Alice to Bob. Moreover, she can modify each of the messages and signatures in transit and can observe whether Bob accepts or rejects. Eve wins if she can cause Bob to accept some arbitrary message that Alice did not sign.

We give a solution to the above problem when $m = O(n^2)$, which we show to be optimal. The size of the verification key and each of the signatures is $O(m)$. Our proof of security relies on a new "block entropy lemma", which we view as a result of independent interest, and discuss below.

*Block Entropy Lemma.* Consider a random variable $X = (X_1, \ldots, X_k) \in (\{0,1\}^b)^k$ consisting of $k$ blocks of $b$ bits each. Assume that $X$ has a high min-entropy rate with $\mathbf{H}_\infty(X) \geq \alpha \cdot (kb)$ for some parameter $\alpha$. We would intuitively like to say that this must imply that a large $\alpha_1$-fraction of the blocks $X_i$ must individually have a high min-entropy rates $\mathbf{H}_\infty(X_i) \geq \alpha_2 \cdot b$. For example, if $\alpha = \Omega(1)$, then we may intuitively hope that the above should hold with some $\alpha_1 = \alpha_2 = \Omega(1)$.

Indeed, if we were to consider Shannon entropy instead of min-entropy, then $\mathbf{H}(X) \leq \sum_i \mathbf{H}(X_i)$, and therefore the above holds with (e.g.,) $\alpha_1 = \alpha_2 = \alpha/2$. Unfortunately, when it comes to min-entropy, such statements are false for any reasonable parameters. As an example, consider the distribution on $X$ where we choose a random index $i^* \leftarrow [k]$, set $X_{i^*} = 0^b$, and for all other indices $i \neq i^*$ we choose $X_i \leftarrow \{0,1\}^b$ uniformly at random. Then $\mathbf{H}_\infty(X) \geq (k-1) \cdot b$, but for each individual index $i \in [k]$, we have $\mathbf{H}_\infty(X_i) \leq \log(k)$ since $\Pr[X_i = 0^b] \geq 1/k$.

While the above example shows that the statement is false in its basic form, it also highlights a potential way to augment the statement to make it hold while preserving its intuitive meaning. The example shows that for every fixed index $i$ chosen a-priori, $X_i$ may have low min-entropy. However, we may be able to find a large set of good indices $i$ after seeing $X$ (e.g., to avoid $i = i^*$ in the previous example), such that each such $X_i$ has high min-entropy even if we reveal that $i$ is in this good set. We formalize this by showing that for any $\alpha_1 \approx \alpha_2 \approx \alpha/2$, with overwhelming probability over $x \leftarrow X$ there is a large set of "good' indices $\mathcal{I}(x)$ of size $|\mathcal{I}(x)| \geq \alpha_1 \cdot k$, such that the min-entropy of the $i$'th block is high if we condition on $i \in \mathcal{I}(X)$ begin a good index: $\mathbf{H}_\infty(X_i | i \in \mathcal{I}(X)) \geq \alpha_2 \cdot k$.

Our block-entropy lemma is somewhat related to previous lemmas of [24, 27] showing that, if we (pseudo-)randomly sample a sufficiently large subset of locations $S \subseteq [k]$, then $X_S = \{X_i \ : \ i \in S\}$ has a high entropy rate, close

to the entropy rate of $X$. However, these previous lemmas do not allow us to say anything about the entropy rates of individual blocks $X_i$. Our lemma is also somewhat related to a lemma of [9], which shows that many individual *bits* of $X$ have some non-trivial entropy. However, that lemma does not give good parameters when extended to blocks, and indeed, the natural attempt at extending it leads to a false statement as discussed above.

### 1.2 Our Techniques

We now discuss the high-level technical ideas behind each of our results. To keep things simple, we will avoid getting bogged down with parameter details. Instead, we use "short/small" to denote values of size $< n$ that can fit in honest user memory, and "long/big" to denote values of size $> m$ that cannot even fit in adversary memory.

*Symmetric-Key Authentication.* Our basic result for symmetric-key authentication is fairly simple. We rely on two building blocks:

- A (strong) local extractor $\mathsf{Ext}$ that takes as input a long source $x$ and a small random $\mathsf{seed}$, and produces a small output $\mathsf{Ext}(x; \mathsf{seed})$ by only reading a small subset of the locations in $x$, where the locations are determined by $\mathsf{seed}$ [27]. In particular, it can be computed by reading $x$ in a streaming manner using small memory. The extracted output is statistically close to uniform even given $\mathsf{seed}$, as long as $x$ has sufficiently high entropy.
- A streaming one-time message-authentication code $\sigma = \mathsf{MAC}_{\mathsf{sk}}(\mu)$ that allows us to authenticate a single long message $\mu$ using a small secret key $\mathsf{sk}$, by reading $\mu$ in a streaming manner. The authentication tag is also small. This can be constructed easily using polynomial evaluation.

We use these building blocks to construct a symmetric-key authentication scheme in the BSM.

- The small shared secret key $\mathsf{sk} = (\hat{\mathsf{sk}}, \mathsf{seed})$ consists of key $\hat{\mathsf{sk}}$ for the streaming one-time MAC and $\mathsf{seed}$ for an extractor.
- To authenticate a message $\mu$, Alice then streams a long random string $x$ to Bob, and as she does so, also computes $r = \mathsf{Ext}(x; \mathsf{seed})$ and $\hat{\sigma} = \mathsf{MAC}(\hat{\mathsf{sk}}, (x \| \mu))$ in a streaming manner using small memory. She then appends the short value $\psi = r \oplus \hat{\sigma}$ as the final component of the long tag $\sigma = (x, \psi)$
- To verify the tag $\sigma = (x, \psi)$ for message $\mu$, Bob computes $r = \mathsf{Ext}(x; \mathsf{seed})$ and $\hat{\sigma} = \mathsf{MAC}(\hat{\mathsf{sk}}, (x, \mu))$ in a streaming manner using small memory. He then checks if $\psi = r \oplus \hat{\sigma}$ and, if so, accepts if $\hat{\sigma}$ is a valid tag for message $(x \| \mu)$ under $\mathsf{MAC}$ (using secret $\hat{\mathsf{sk}}$, and where the verification procedure is also performed in a streaming manner), and rejects otherwise.

In essence, Alice is "encrypting" the one-time MAC tags using a symmetric-key encryption in the BSM, and then uses a one-time MAC to also authenticate the

BSM encryption randomness. Intuitively, the encryption ensures that even if the adversary sees many encrypted tags, she doesn't learn much about the secret key of the one-time MAC. However, formalizing this is somewhat subtle.

Consider an adversary Eve to first passively observes $q$ honestly generated tags $\sigma_1, \ldots, \sigma_q$ from Alice in a streaming manner, while maintaining $m$ bits of memory. Then Eve gets unlimited memory and observes an additional $(q+1)$'st tag $\sigma$ and outputs $\sigma'$. She wins if Bob accepts $\sigma'$ and $\sigma' \neq \sigma$. In our proof, we first switch the values $\psi_i$ inside the $q$ tags $\sigma_i$ to uniformly random. We rely on the security of the strong local extractor to argue that the change is indistinguishable, *even* if we later reveal the entire secret key $\mathsf{sk} = (\mathsf{seed}, \hat{\mathsf{sk}})$ and can therefore check if the adversary wins the game. Once we make this change, the one-time MAC key $\hat{\mathsf{sk}}$ is only used to generate the $(q+1)$'st tag $\sigma = (x, \psi)$ for message $\mu$ and verify $\sigma' = (x', \psi')$ for message $\mu'$. Therefore, we can rely on the security of the one-time MAC to argue that if $(\mu', \sigma') \neq (\mu, \sigma)$ then Bob rejects with overwhelming probability.

On a quick glance, it may seem that we could have applied the one-time MAC to just $\mu$ instead of the pair $(x, \mu)$, and still argued that Eve cannot cause Bob to accept any $\sigma' = (\mu', x', \psi')$ where $\mu' \neq \mu$, which should suffice. However, this turns out to be false and the scheme would be completely insecure with this modification. In particular, Eve would be able to learn something about the secret key by modifying the tags from Alice to Bob while keeping the message intact and seeing whether or not he accepts. For example, each time Alice generates a tag $\sigma_i = (\mu_i, x_i, \psi_i)$, Eve could flip a bit of $x_i$ and see if Bob accepts or rejects. This would eventually reveal the set of locations in $x$ read by $\mathsf{Ext}(x; \mathsf{seed})$. Once Eve learns this set, extractor security is lost, and indeed Eve can fully recover $\mathsf{sk}$ and break the scheme. In our proof, by showing that Eve cannot cause Bob to accept any $\sigma' \neq \sigma$, even for the same message $\mu$, we ensure that verification queries are useless. Our formal argument is more involved, and requires to carefully answer verification queries using low memory: our reduction cannot even store an entire tag.

Overall, we can get security against adversaries with memory of size $m$ where honest users only need memory of size $n = O(\log m)$ and where tags are of size $O(m)$. We refer to Section 5 for more details.

*Symmetric-Key Authentication with Short Tags.* Next, we turn to the same problem as in the previous paragraph, but additionally require that the authentication tags are small, of size $< n$. Furthermore, they can be fully generated, stored, and verified inside honest user memory, without relying on the streaming model. Since the secret key is also small, of size $< n$, if an adversary can simultaneously remember $n$ tags, then it can break security via the classical authentication lower bound. Therefore, the above setting necessitates bounding the adversary's memory to $m = O(n^2)$.

Constructing this type of authentication scheme implies some sort of space lower bound on learning. In particular, the adversary observes many outputs of the (potentially randomized) authentication function with a secret key $\mathsf{sk}$, but if its memory is bounded, it does not learn the function sufficiently well

to authenticate new messages. Unlike the previous setting where each function output individually was long and could not be stored in memory, in this setting each output is short, but the adversary cannot store too many outputs. Lower bounds in this setting are highly non-trivial, and the first lower bound of this form was shown only recently in the celebrated work of Raz [25], and subsequently extended by [21, 26, 12]. In particular, Raz proved a space lower bound for learning random parities. Here, we choose a uniformly random $\mathbf{x} \leftarrow \mathbb{F}_2^n$. The adversary can get many samples $(\mathbf{a}_i, b_i)$ where $\mathbf{a}_i \leftarrow \mathbb{F}_2^n$ and $b_i = \langle \mathbf{x}, \mathbf{a}_i \rangle$. If the adversary has $> n^2$ bits of memory and can remember $> n$ samples in full, then it can perform Gaussian elimination and recover $\mathbf{x}$. The work of Raz shows that, if the adversary's memory is sufficiently smaller than $n^2$ (and the number of samples is smaller than exponential), then the adversary cannot recover $\mathbf{x}$ or even distinguish the samples from uniformly random values.

Abstracting the above, we can think of Raz's result as showing that the inner-product function $f_{\mathbf{x}}(\mathbf{a}) = \langle \mathbf{x}, \mathbf{a} \rangle$ is a "weak pseudorandom function" (weak PRF) in the BSM, in the sense that an adversary with sufficiently small memory cannot distinguish the outputs of the function on random inputs $\mathbf{a}_i$ from uniformly random values. Unfortunately, it is not a-priori clear how to use a weak PRF in the BSM to solve the message authentication problem in the BSM. Indeed, even in the computational setting, we do not know of any "nice" construction of computational message authentication codes from a generic weak PRF (beyond just using the weak PRF to construct a PRG and applying [13] to go from a PRG to a PRF, but this approach does not translate to the BSM). Instead, we rely on the specifics of the inner product function, rather than just generically relying on it being a weak PRF in the BSM. Notice that the inner-product function is essentially the same as Learning Parity with Noise (LPN), just without noise! Moreover, there is a long body of work tying to construct simple/efficient message authentication schemes from LPN [17, 19, 20, 7]. In particular, the work of [7] abstracts out many of these ideas and shows how to construct message-authentication directly from any *key-homomorphic weak PRF*. We show that the same tricks there happen to carry over nicely to the BSM, and allow us to construct a message-authentication scheme with the desired parameters, by relying on the fact that the inner-product function is essentially a key-homomorphic weak PRF in the BSM via Raz's lower bound. Interestingly, we crucially rely on the linearity of the inner-product function in this construction, and do not know how to adapt subsequent space lower bounds for other functions (e.g., low-degree polynomials) [12] to get analogous results using them. Still, adapting the work of [7] to the bounded memory setting requires a great deal of care to ensure that all the reductions use small memory. We refer to Section 6 for more details.

*Public-Key Signatures.* In the public-key setting, we start with an initialization stage during which Alice streams a long verification key $\mathsf{vk}$ over an authentic channel. At the end of the initialization stage, Alice keeps a short signing key $\mathsf{sk}$, while a verifier Bob keeps a short verification digest $\mathsf{vd}$.

We first observe that if Alice and Bob could interact back-and-forth with each other during the initialization stage, we could solve the problem trivially

by having them run a key-agreement protocol in the BSM [3, 14, 9], at the end of which they would have a shared secret key. Later, they could use the secret key to authenticate messages via a symmetric-key authentication schemes in the BSM that we constructed earlier. However, all such key-agreement protocols require at least two back-and-forth rounds of interaction, while in our case we only allow one-way communication from Alice to Bob.[5] Unfortunately, it is easy to see that key agreement in one round is impossible, since nothing differentiates an honest Bob from the attacker Eve – if Bob knows Alice's key, so can Eve.

Nevertheless, we show that a variant of key agreement that we call *set key agreement* is possible in one round! In a set key agreement scheme, Alice streams a long message $\mathsf{vk}$ and outputs a set of keys $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_q)$, while Bob observes the stream and outputs some subset of Alice's keys $\mathsf{vd} = (T, (\mathsf{sk}_i)_{i \in T})$ for $T \subseteq [q]$. Security requires that there is some shared key $\mathsf{sk}_t$ for $t \in T$, meaning that the key is known to both Alice and Bob, such that $\mathsf{sk}_t$ looks uniformly random even given Eve's view of the protocol and all the other keys $\mathsf{sk}_j$ for $j \in [q]$, $j \neq t$. The index $t$ of this "good" shared key is a random variable that depends on the entire protocol execution and on Eve's view, and it is therefore not known to Alice or Bob.

We construct such set key agreement when the adversarial memory is sufficiently small $m = O(n^2)$, as follows. Alice streams a random string $x = (x_1, \ldots, x_k)$ consisting of $k$ blocks $x_i \in \{0,1\}^b$, where $b$ is very small (depending only on the security parameter). Alice chooses a small random subset $S_A \subseteq [k]$ of size $|S_A| = q$ and remember the blocks $x_i \ : \ i \in S_A$. Similarly, Bob chooses a small random subset $S_B \subseteq [k]$ of size $|S_B| = q$ and remember the blocks $x_i \ : \ i \in S_B$. We choose $k = O(m)$ and $q = O(\sqrt{m})$ carefully to ensure that Alice/Bob only use $n = \widetilde{O}(\sqrt{m})$ bits of memory and that with high probability their sets have non-trivial intersection $|S_A \cap S_B|$ of roughly security parameter size. After Alice finishes streaming $x$, she then also streams her set $S_A = \{i_1, \ldots, i_q\}$ along with $q$ extractor seeds $\mathsf{seed}_j$, and sets her keys to be the extracted values $\mathsf{sk}_j = \mathsf{Ext}(x_{i_j}, \mathsf{seed}_j)$ for $j \in [q]$. Bob computes the set $T = \{j \ : \ i_j \in S_B\}$ and sets $\mathsf{sk}_j = \mathsf{Ext}(x_{i_j}; \mathsf{seed}_j)$ for $j \in T$. We argue that security holds as long as Eve's memory is $\leq k/2$. Eve needs to decide what to remember about $x$ before she learns anything about $S_A, S_B$. We will use our new block entropy lemma to argue that there is a large fraction of locations $i$, such that the individual blocks $x_i$ have high min-entropy conditioned on what Eve remembered about $x$, and therefore it is likely that some such location $i^* = i_t$ appears in $S_A \cap S_B$. The corresponding key $\mathsf{sk}_t = \mathsf{Ext}(x_{i_t}; \mathsf{seed}_t)$ will then be the "good key" which looks uniform given Eve's view. As discussed in our description of the block entropy lemma, the set of locations $i$ such that $x_i$ has high min-entropy is not fixed, but

[5] We view this as a crucial component of our model. Alice can broadcast her verification key to the world, obliviously of who is listening and who will want to very her signed messages in the future. There may potentially even be a large number of different verifiers and Alice should not need to know about them or interact with them.

rather a random variable that depends on $x$ and on Eve's view. Therefore, also the index $t$ of the "good key" is such a random variable.

Once we have a set key agreement protocol, it is easy to build a signature scheme on top of it. To sign each message $\mu$, Alice uses each of the secret keys $\mathsf{sk}_i$ for $i \in [q]$ as a key for a symmetric-key authentication scheme in the BSM (from our first result), and sequentially authenticate the message $\mu$ using each of the $q$ keys individually. The verifier Bob verifies the authentication tags corresponding to the indices $j \in T$. To argue security, we rely on the security of the symmetric-key authentication scheme in the location $t$ corresponding to the "good" secret key. Note that, in the set key agreement protocol, we needed to choose $q = O(\sqrt{m})$ large to ensure that that $S_A$ and $S_B$ have a large overlap. This means that each of the keys $\mathsf{sk}_i$ needs to be very small, since Alice's storage is $n = q|\mathsf{sk}_i|$. However, this is not a problem since the symmetric-key authentication scheme in the BSM allowed us to make the keys as small as $O(\log m)$.[6]

We also give a lower bound to show that this type of public-key signatures in the BSM are impossible when $m > n^2$. We rely on a technical lemma due to Dziembowski and Maurer [11]. Translated to our context, the lemma addresses the scenario where Alice streams a long verification key $\mathsf{vk}$ to Bob, after which she stores some $n$ bit state $\mathsf{sk}$ and Bob stores some $n$ bit state $\mathsf{vd}$. The lemma says that there is some small value $E(\mathsf{vk})$ of size $m < n^2$ that Eve can store about $\mathsf{vk}$, such that, conditioned on $E(\mathsf{vk})$, the values $\mathsf{sk}$ and $\mathsf{vd}$ are statistically close to being independent. In particular, this means that if Eve samples a fresh $\mathsf{sk}'$ conditioned on $E(\mathsf{vk})$, then she can sign messages using $\mathsf{sk}'$ and Bob will accept the signatures with high probability. We refer to Section 7 for more details.

*Block Entropy Lemma.* Lastly, we give some intuition behind our block entropy lemma. We show that, for every fixed $x \in (\{0,1\}^b)^k$ there is some set $\mathcal{I}(x)$ such that, if $X$ has sufficiently high entropy $\mathbf{H}_\infty(X) \geq \alpha \cdot (bk)$, then:

- With overwhelming probability over $x \leftarrow X$, we have $|\mathcal{I}(x)| \geq \alpha_1 \cdot k$
- For all $x$ and $i \in \mathcal{I}(x)$: $\mathbf{H}_\infty(X_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{I}(X)) \geq \alpha_2 \cdot b$.

where the above holds for any $\alpha_1$ and $\alpha_2 = \alpha - \alpha_1 - \lambda/b$, where $\lambda$ is the security parameter (see Lemma 4 for a formal statement). We start by observing that the definition of min-entropy guarantees that for any $x$:

$$2^{-\alpha kb} \geq \Pr[X = x] = \prod_{i=1}^{k} \underbrace{\Pr[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]}_{2^{-e_i}} = 2^{-\sum_{i=1}^{k} e_i}.$$

By a simple averaging argument, there is an $\alpha_1$ fraction of "good" indices $i \in \mathcal{I}(x) \subseteq [k]$ for which $e_i \geq (\alpha - \alpha_1) \cdot b$. But the fact that $\Pr[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]$ is small does not guarantee that $\mathbf{H}_\infty(X_i | X_1 = x_1, \ldots, X_{i-1} =$

[6] This is also why we need to rely on the symmetric-key authentication scheme with long tags from our first result, rather than the one with short tags from our second result – in the latter, the secret keys would be of size $O(\sqrt{m})$ and hence Alice/Bob would need memory of size $O(m)$ exceeding that of Eve.

$x_{i-1}$) is large, since the concrete outcome $x_i$ that we got may not have been the maximally likely one. However, let us additionally condition on the event that $i \in \mathcal{I}(X)$ is a good index. We show that, for each $i$, either (I) the conditional min-entropy is high $\mathbf{H}_\infty(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{I}(X)) \geq \alpha - \alpha_1 - \lambda/b$ or (II) the probability $\Pr[i \in \mathcal{I}(X) | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 2^{-\lambda}$ is low. By removing indices $i$ for which (II) holds from $\mathcal{I}(x)$, we get our lemma. For a random $x \leftarrow X$, the probability that any index was removed is negligible.

## 2 Preliminaries

*Notation.* When $X$ is a distribution, or a random variable following this distribution, we let $x \leftarrow X$ denote the process of sampling $x$ according to the distribution $X$. If $X$ is a set, we let $x \leftarrow X$ denote sampling $x$ uniformly at random from $X$. We use the notation $[k] = \{1, \ldots, k\}$. If $x \in \{0,1\}^k$ and $i \in [k]$ then we let $x[i]$ denote the $i$'th bit of $x$. If $s \subseteq [k]$, we let $x[s]$ denote the list of values $x[i]$ for $i \in s$.

We refer to the full version for more definitions and facts about information theory, including the local extractors of [27], which we call BSM-extractors in this work. We will use the following construction of BSM-extractors.

**Theorem 1 ( [27]).** *For any $m \geq \ell$,$\lambda$, there is a $(n, m, \varepsilon)$-BSM extractor $\mathsf{BSMExt} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^\ell$ with $n = O(\ell + \lambda + \log m)$, $\varepsilon = 2^{-\Omega(\lambda)}$, $k = O(m + \lambda \log(\lambda))$, $d = O(\log m + \lambda)$.*

We will use the following lemmas.

**Lemma 1 ( [8]).** *For any random variables $X, Y$,for every $\varepsilon > 0$ we have*

$$\Pr_{y \leftarrow Y}[\mathbf{H}_\infty(X|Y = y) \geq \mathbf{H}_\infty(X|Y) - \log(1/\varepsilon)] \geq 1 - \varepsilon.$$

**Lemma 2 ( [8]).** *For any random variables $X, Y, Z$ where $Y$ is supported over a set of size $T$ we have $\mathbf{H}_\infty(X|Y, Z) \leq \mathbf{H}_\infty(X|Z) - \log T$.*

**Lemma 3 ( [15, 4] ).** *Let $k$ be an integer, and $S_A$, $S_B \subset [k]$ be two uniformly and independently sampled subset of $[k]$ of size $q$ where $q \geq 2\sqrt{\lambda \cdot k}$. Then*

$$\Pr[|S_A \cap S_B| < \lambda] < e^{-\lambda/4}.$$

## 3 Definitions

In this section, we define message authentication codes (MACs) in Section 3.1 and signatures in Section 3.2 in the BSM. We discuss how we model honest parties and adversaries in these settings.

### 3.1 Message Authentication Codes

A message-authentication code (MAC) over a message space $\mathcal{M}$ is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$: on input a security parameter $\lambda$, output a key $\mathsf{sk}$.
- $\mathsf{MAC}(\mathsf{sk}, \mu)$: on input a key $\mathsf{sk}$ and a message $\mu \in \mathcal{M}$, output a tag $\sigma$.
- $\mathsf{Verify}(\mathsf{sk}, \mu, \sigma)$: on input a key $\mathsf{sk}$, a message $\mu \in \mathcal{M}$ and a tag $\sigma$, output a bit $b$.

We define the following properties.

**Definition 1 (Correctness).** *We say that a MAC is* correct *if for all message $\mu \in \mathcal{M}$:*

$$\Pr[\mathsf{Verify}(\mathsf{sk}, \mu, \sigma) = 1 \,:\, \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda), \sigma \leftarrow \mathsf{MAC}(\mathsf{sk}, \mu)] \geq 1 - \mathrm{negl}(\lambda).$$

**Definition 2 (Streaming MAC).** *For an integer $n$, we say that a MAC is a* streaming MAC with memory $n$ *if* $\mathsf{MAC}$ *and* $\mathsf{Verify}$ *can respectively be computed by streaming algorithms with memory $n$, given streaming access to $\mu$ and $(\mu, \sigma)$ respectively, and if* $\mathsf{KeyGen}$ *can be computed (in place) with memory $n$. In particular, MAC keys* $\mathsf{sk}$ *have size at most $n$.*

**Definition 3 ((Selective) unforgeability under chosen message (and verification) attacks).** *For an algorithm* $\mathsf{Adv}$*, consider the following experiment:*

*Experiment* $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$*:*

1. *Sample* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. *Compute* $(\mu^*, \sigma^*) \leftarrow \mathsf{Adv}^{\mathsf{MAC}(\mathsf{sk},\cdot),\mathsf{Verify}(\mathsf{sk},\cdot,\cdot)}$.
3. *Output* $1$ *if* $\mathsf{Verify}(\mathsf{sk}, \mu^*, \sigma^*) = 1$ *output* $0$ *otherwise.*

*We say that an adversary* $\mathsf{Adv}$ *is* admissible *if* $\mathsf{Adv}$ *did not query* $\mathsf{MAC}(\mathsf{sk}, \cdot)$ *on input $\mu^*$. In the following, all the adversaries are assumed admissible.*

*We say that a MAC is $(\varepsilon, \mathcal{A})$-unforgeable under chosen message and chosen verification queries attacks (*$\mathsf{uf\text{-}cmva}$*) against a class $\mathcal{A}$ of adversaries, if for all adversary* $\mathsf{Adv} \in \mathcal{A}$*:*

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv}) = 1] \leq \varepsilon.$$

*We simply say that a MAC is $\mathcal{A}$-*$\mathsf{uf\text{-}cmva}$*-secure if it is $(\varepsilon, \mathcal{A})$-*$\mathsf{uf\text{-}cmva}$*-secure for some $\varepsilon = \mathrm{negl}(\lambda)$.*

*We alternatively say that a MAC is* $(\varepsilon, \mathcal{A})$-selectively *unforgeable under chosen message and chosen verification queries attack (*$\mathsf{suf\text{-}cmva}$*) if any adversary* $\mathsf{Adv} \in \mathcal{A}$ *that declare $\mu^*$ before the beginning of experiment makes the associated experiment output* $1$ *with probability at most $\varepsilon$.*

*Next, we say that a MAC is $(\varepsilon, \mathcal{A})$-(selectively) unforgeable under chosen message attacks (*$(\mathsf{s})\mathsf{uf\text{-}cma}$*) if any adversary* $\mathsf{Adv} \in \mathcal{A}$ *that only make tagging*

*queries* $\mathsf{MAC}(\mathsf{sk},\cdot)$ *makes the associated experiment output* 1 *with probability at most* $\varepsilon$*.*

*Last, we say that a MAC is* $\mathcal{A}$-(($\mathsf{s}$)$\mathsf{uf}$-$\mathsf{cm}$($\mathsf{v}$)$\mathsf{a}$) *if it is* $(\varepsilon,\mathcal{A})$-(($\mathsf{s}$)$\mathsf{uf}$-$\mathsf{cm}$($\mathsf{v}$)$\mathsf{a}$) *for some* $\varepsilon = \mathrm{negl}(\lambda)$*.*

**Definition 4 (Indistinguishability under chosen message attacks).** *We say that a MAC satisfies* $(\varepsilon,\mathcal{A})$*-indistinguishability under chosen-message attacks (*$\mathsf{ind}$-$\mathsf{cma}$*) if for all* $\mathsf{Adv} \in \mathcal{A}$*:*

$$\left|\Pr[\mathsf{Adv}^{\mathsf{MAC}(\mathsf{sk},\cdot)} = 1] - \Pr[\mathsf{Adv}^{\mathsf{MAC}(\mathsf{sk},0)} = 1]\right| \leq \varepsilon$$

*where the probabilities are over the randomness of* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*

*We say that a MAC is* $\mathcal{A}$-$\mathsf{ind}$-$\mathsf{cma}$ *if it is* $(\varepsilon,\mathcal{A})$-$\mathsf{ind}$-$\mathsf{cma}$ *for some* $\varepsilon = \mathrm{negl}(\lambda)$*.*

Next, we define the class of adversaries we consider in this work.

**Definition 5 (Streaming Adversaries for MACs).** *We say that an adversary* $\mathsf{Adv}$ *in any of the security experiments above is* streaming *with memory* $m$ *if it is a streaming algorithm that accesses oracles* $\mathsf{MAC}$ *and* $\mathsf{Verify}$ *(whenever applicable) in a streaming manner using memory at most* $m$*. Namely, it provides the inputs to oracles* $\mathsf{MAC}$ *and* $\mathsf{Verify}$ *as streams of bits and receives the outputs of* $\mathsf{MAC}$ *as streams. The calls to* $\mathsf{MAC}$ *and* $\mathsf{Verify}$ *are not required to be synchronous: for instance* $\mathsf{Adv}$ *can start a oracle call to* $\mathsf{Verify}$ *while streaming the input to* $\mathsf{MAC}$ *or receiving a tag in a streaming manner.*

*We add the restriction that there is at most one current call to* $\mathsf{MAC}$ *and at most one current call to* $\mathsf{Verify}$ *at any given point in time during the experiment, namely, the calls to* $\mathsf{MAC}$ *are done in a sequential manner, and similarly for* $\mathsf{Verify}$*. For technical reasons, we only allow any given oracle call to* $\mathsf{Verify}$ *to be concurrent with at most one call to* $\mathsf{MAC}$*: we disallow a single verification query to span several tagging queries.*

*We will usually refer to* $Q$ *(resp.* $Q_T, Q_V$*) as the number of total (resp. tagging, verification) queries made by* $\mathsf{Adv}$*.*

The class of adversaries above notably covers man-in-the-middle adversaries who have the ability alter a tag being currently streamed, and observe either the output of the verifier, or use it as a final forgery.

We argue that the restriction to the concurrency of oracle calls above is reasonable. Notably, in a setting where an authenticator streams tags to a single verifier and where a man-in-the-middle possibly tampers the communication between them, we believe it is reasonable to assume that the verifier only authenticates one streamed tag at a time. In such a setting, adversaries would satisfy the conditions of Definition 5.

*Remark 1 (Adversaries with Unbounded Temporary Memory).* In the traditional BSM, adversaries have access to unbounded temporary memory as long as they eventually compress the information given to them down to some bounded amount $m$. We can similarly define a class of adversaries stronger than Definition 5, who, given a streamed tag output by $\mathsf{MAC}(\mathsf{sk},\cdot)$, can process any (blocks

of) streamed bits using unbounded temporary memory as long as they compress their state down to some $m$-bit state before the next bit (or block) is streamed. In particular, such adversaries can compute bounded-size non-uniform advice on their own, before any query is made.

We will use the following MAC secure against unbounded adversaries, as long as only one tag is given out. We refer to the full version for a construction and a proof.

**Claim 1 (One-Time Information-Theoretic MACs).** *Let $\mathcal{A}$ be the set of (potentially unbounded) algorithm that make at most 1 tagging query and at most $Q_V$ verification queries. Then for all integer $k$ and constant $c > 0$, there exists a streaming MAC with memory $n = O(\lambda + \log k)$ and message space $\{0,1\}^k$ which is $(k \cdot Q_V/2^{c\lambda}, \mathcal{A})$-$\mathsf{uf}$-$\mathsf{cmva}$-secure. Furthermore, the MAC produces tags of size $n$.*

### 3.2 Signatures

A *streaming* signature scheme over a message space $\mathcal{M}$ is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$: on input a security parameter $\lambda$, stream a verification key $\mathsf{vk}$ and store signing key $\mathsf{sk}$.
- $\mathsf{KeyReceive}(1^\lambda, \mathsf{vk})$: on input a security parameter $\lambda$ and a streamed verification key $\mathsf{vk}$, output a verification state $\mathsf{vd}$.
- $\mathsf{Sign}(\mathsf{sk}, \mu)$: on input a signing key $\mathsf{sk}$ and a (potentially streamed) message $\mu \in \mathcal{M}$, output a (potentially streamed) signature $\sigma$.
- $\mathsf{Verify}(\mathsf{vd}, \mu, \sigma)$: on input a verification state $\mathsf{vd}$, a (potentially streamed) message $\mu \in \mathcal{M}$ and a (potentially streamed) signature $\sigma$, output a bit $b \in \{0,1\}$.

**Definition 6 (Correctness).** *We say that a signature scheme is* correct *if for all message $\mu \in \mathcal{M}$:*

$$\Pr[\mathsf{Verify}(\mathsf{vd}, \mu, \sigma) = 1] \geq 1 - \mathrm{negl}(\lambda),$$

*where* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda), \mathsf{vd} \leftarrow \mathsf{KeyReceive}(1^\lambda, \mathsf{vk}), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$.

**Definition 7 (Streaming Signature).** *For an integer $n$, we say that a streaming signature scheme has memory $n$ if* $\mathsf{KeyGen}$, $\mathsf{KeyReceive}$, $\mathsf{Sign}$ *and* $\mathsf{Verify}$ *can respectively be computed by streaming algorithms with memory $n$, given streaming access to $\mu$ and $(\mu, z)$ respectively. We additionally require that* $\mathsf{sk}$ *and* $\mathsf{vd}$ *have size at most $n$.*

**Definition 8 (Unforgeability under chosen message attacks).** *For an algorithm* $\mathsf{Adv}$*, consider the following experiment:*

*Experiment* $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$*:*

1. *Sample* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*,* $\mathsf{vd} \leftarrow \mathsf{KeyReceive}(1^\lambda, \mathsf{vk})$*, and compute* $\mathsf{view}_{\mathsf{Adv}} \leftarrow \mathsf{Adv}(\mathsf{vk})$*.*
2. *Compute* $(\mu^*, \sigma^*) \leftarrow \mathsf{Adv}^{\mathsf{Sign}(\mathsf{sk},\cdot),\mathsf{Verify}(\mathsf{vd},\cdot,\cdot)}(\mathsf{view}_{\mathsf{Adv}})$*.*
3. *Output* $1$ *if* $\mathsf{Verify}(\mathsf{vd}, \mu^*, \sigma^*) = 1$ *output* $0$ *otherwise.*

*We say that an adversary* Adv *is* admissible *if* Adv *did not query* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *on input* $\mu^*$*. In the following, all the adversaries are assumed admissible.*

*We say that a signature scheme is* $(\varepsilon, \mathcal{A})$*-unforgeable under chosen message attacks (*uf-cma*) against a class* $\mathcal{A}$ *of adversaries, if for all adversary* $\mathsf{Adv} \in \mathcal{A}$*:*

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv}) = 1] \leq \varepsilon.$$

*We simply say that a signature scheme is* $\mathcal{A}$*-*uf-cma*-secure if it is* $(\varepsilon, \mathcal{A})$*-*uf-cmva*-secure for some* $\varepsilon = \mathrm{negl}(\lambda)$*.*

*We define variants of unforgeability, namely selective and/or without verification queries (*((s)uf-cm(v)a)*-security), in a similar way than in Section 3.1.*

We consider a similar class of streaming adversaries for signatures, as in Definition 5.

**Definition 9 (Streaming Adversaries for Signatures).** *We say that an adversary* Adv *in any of the security experiments above is* streaming *with memory* $m$ *if it is a streaming algorithm that accesses oracles* KeyGen*,* MAC *and* Verify *(whenever applicable) in a streaming manner using memory at most* $m$*. Namely, such adversaries first observe the stream* vk *produced by* KeyGen *(but not* sk*). Then, it provides the inputs to oracles* Sign *and* Verify *as streams of bits and receives the outputs of* Sign *as streams. The calls to* Sign *and* Verify *are not required to be synchronous: for instance* Adv *can start a oracle call to* Verify *while streaming an input to* Sign *or receiving a tag in a streaming manner. We add the restriction that there are at most one concurrent call to* Sign *and at most one concurrent call to* Verify *at any given point in time during the experiment, and any give oracle call to* Verify *can be concurrent with at most one call to* Sign*.*

*We will refer to* $Q$ *(resp.* $Q_S, Q_V$*) as the number of total (resp. signing, verification) queries made by* Adv*.*

As in Remark 1, one can consider adversaries with temporary unbounded memory for signatures.

## 4 Block Entropy Lemma

In this section, we describe and prove our block entropy lemma (Lemma 4), which we will use to build signature schemes in the streaming BSM Section 7. In the full version, we state a corollary that might be of independent interest.

**Lemma 4.** *Let* $X = (X_1, \ldots, X_k) \in \{0,1\}^{kb}$ *be a random variable with blocks* $X_i \in \{0,1\}^b$*, such that* $\mathbf{H}_\infty(X) \geq \alpha \cdot (kb)$ *for some* $\alpha > 0$*. Then, there are some parameters* $\alpha_1, \alpha_2, \varepsilon$ *instantiated below and some set* $\mathsf{BAD} \subseteq \{0,1\}^{kb}$ *such that the following holds:*

1. $\Pr[X \in \mathsf{BAD}] \leq \varepsilon$.
2. *For all* $x = (x_1, \ldots, x_k)$ *there is a set* $\mathcal{I}(x) \subseteq [k]$ *such that:*
   (a) *if* $x \notin \mathsf{BAD}$ *then* $|\mathcal{I}(x)| \geq \alpha_1 \cdot k$,
   (b) *for all* $i \in \mathcal{I}(x)$: $\mathbf{H}_\infty(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{I}(X)) \geq \alpha_2 \cdot b$.

   *Furthermore, the values* $x_1, \ldots, x_i$ *fully determine whether or not* $i \in \mathcal{I}(x)$.

*The parameters* $\alpha_1, \alpha_2, \varepsilon$ *can be set according to either of the following two options:*

(i) *For any* $\alpha_1 > 0, \gamma > 0$, *we can set* $\alpha_2 = (\alpha - \alpha_1 - \log(1/\gamma)/b), \varepsilon = k \cdot \gamma$.
*For example, for any* $\rho > 0$, *we can set* $\alpha_1 = \alpha_2 = \alpha/2 - \rho$ *and* $\varepsilon = k \cdot 2^{-2\rho b}$.

(ii) *For any* $\beta > 0, \gamma > 0, \delta \in [0,1]$, *we can set:* $\alpha_1 = \beta - (1+\delta)\gamma, \alpha_2 = (\alpha - \beta - \log(1/\gamma)/b), \varepsilon = \exp\left(-\frac{\delta^2}{3}\gamma k\right)$. *For example, for any* $\rho > 0$ *such that* $b \geq \log(1/(2\rho))/(2\rho)$, *we can set:*
$\alpha_1 = \alpha_2 = \alpha/2 - \rho$ *and* $\varepsilon = e^{-\frac{2}{3}\rho k}$.

We refer to the full version for a proof for getting parameter choice (ii) in Lemma 4; parameter choice (i) will suffice in our applications.

*Proof.* Fix any $x = (x_1, \ldots, x_k)$. Then, by the definition of min-entropy, we have:

$$2^{-\alpha \cdot (kb)} \geq \Pr[X = x] = \prod_{i=1}^{k} \underbrace{\Pr[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]}_{\stackrel{\text{def}}{=} 2^{-e_i}} = 2^{-\sum_{i=1}^{k} e_i}.$$

Let $\beta > 0$ be some parameter and define the set $\mathcal{H}(x) \subseteq [k]$ via $\mathcal{H}(x) := \{i \ : \ e_i \geq (\alpha - \beta)b\}$. Intuitively, these are the indices $i$ where the $i$'th block takes on a low probability value $x_i$ conditioned on previous blocks, and therefore the indices where we should expect to find entropy. Note that $x_1, \ldots, x_i$ fully determines whether $i \in \mathcal{H}(x)$. By an averaging argument, it must be the case that $|\mathcal{H}(x)| \geq \beta \cdot k$.

Now, let us consider the random variable $\mathcal{H}(X)$. We claim that for every $i \in [k]$ and every $\gamma > 0$, at least *one* of the following two conditions must hold:

(I) Either $\Pr[i \in \mathcal{H}(X) \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq \gamma$,
(II) or $\mathbf{H}_\infty(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{H}(X)) \geq (\alpha - \beta) \cdot b - \log(\gamma)$.

Intuitively, the above says that if we condition on any choice of the first $i-1$ blocks then either (I) it is unlikely for the $i$'th block to take on any value whose a-priori probability is too small, or (II) the $i$'th block has high entropy even if we condition on it taking some such value. In particular, if (I) does not hold,

then

$$\begin{aligned}
&\max_z \Pr[X_i = z | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{H}(X)] \\
&= \max_z \frac{\Pr[X_i = z \wedge i \in \mathcal{H}(X) | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]}{\Pr[i \in \mathcal{H}(x) | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]} \\
&\leq \max_{z \in \mathcal{Z}_i(x)} \frac{\Pr[X_i = z | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]}{\gamma} \\
&\quad \text{where } \mathcal{Z}_i(x) = \{z \ : \ \Pr[X_i = z | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq 2^{-(\alpha-\beta)\cdot b}\} \\
&\leq \frac{2^{-(\alpha-\beta)\cdot b}}{\gamma}
\end{aligned}$$

The second inequality follows by noting that $i \in \mathcal{H}(X) \Leftrightarrow X_i \in \mathcal{Z}_i(x)$ whenever $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$.

Now define $\mathcal{A}(x) = \{i \ : \ \Pr[i \in \mathcal{H}(X) \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq \gamma\}$, which corresponds to the indices $i$ for which case (I) holds. Let $\mathcal{I}(x) = \mathcal{H}(x) \setminus \mathcal{A}(x)$, which corresponds to the set of indices for which case (II) holds *and* they are in $\mathcal{H}(x)$. Then, for any $i \in \mathcal{I}(x)$:

$$\begin{aligned}
&\mathbf{H}_\infty(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{I}(X)) \\
&\quad = \mathbf{H}_\infty(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, i \in \mathcal{H}(X)) \geq (\alpha - \beta)b - \log(1/\gamma),
\end{aligned}$$

where the second first equality follows from the fact that $x_1, \ldots, x_{i-1}$ fully determine whether $i \in \mathcal{A}(x)$, and the second inequality follows from the definition of condition (II). This proves part 2.(b) of the lemma with $\alpha_2 = (\alpha - \beta - \log(1/\gamma)/b)$.

We first prove the lemma for the first parameter choice (i). Define $\mathsf{BAD} = \{x \ : \ |\mathcal{I}(x)| < \beta \cdot k\}$. Clearly this ensures that part 2.(a) of the lemma is satisfied with $\alpha_1 = \beta$. Therefore, we are left to prove part 1 of the lemma. Note that $\mathcal{I}(X) = \mathcal{H}(X) \setminus \mathcal{A}(X)$ and $|\mathcal{H}(x)| \geq \beta k$. Intuitively, the only way we can have $X \in \mathsf{BAD}$ is if there is some index $i$ where the $i$'th block $X_i$ was unlikely to take on any low-probability value conditioned on the prior blocks, but it did so anyway, and this is unlikely to occur. Formally

$$\begin{aligned}
Pr[X \in \mathsf{BAD}] &= \Pr[|\mathcal{I}(X)| < \beta k] \\
&\leq \Pr[|\mathcal{H}(X) \cap \mathcal{A}(X)| \geq 1]] \\
&\leq \sum_i \Pr[i \in H(X) \cap \mathcal{A}(X)] \\
&\leq \sum_i \max_{\{(x_1, \ldots, x_{i-1}) \ : i \in \mathcal{A}(x)\}} \Pr[i \in H(X) \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \\
&\leq k \cdot \gamma.
\end{aligned}$$

The second line follows from the fact that $\mathcal{I}(X) = \mathcal{H}(X) \setminus \mathcal{A}(X)$ and $|\mathcal{H}(x)| \geq \beta k$ fopr all $x$. The second to last inequality follows by noting that $(x_1, \ldots, x_{i-1})$ fully determine whether $i \in \mathcal{A}(x)$, and the last inequality follows by noting that the definition of $\mathcal{A}(x)$ guarantees that for any $x_1, \ldots, x_{i-1}$ for which $i \in \mathcal{A}(x)$ we

have $\Pr[i \in \mathcal{H}(X) \ : \ X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq \gamma$. Therefore this proves part 1 of the lemma with $\varepsilon = k \cdot \gamma$.

□

## 5 MAC with Long Tags

In this section, we build a streaming MAC in the streaming BSM where the size of tags grow with the memory bound of the adversary. More precisely, we prove the following theorem:

**Theorem 2.** *For all integers $m, \lambda$, there exists a streaming MAC with memory $n = O(\lambda + \mathrm{polylog}(m, \lambda)))$ (Definition 2) which can authenticate messages of length up to $2^\lambda$, and which is $(2^{-\lambda}, \mathcal{A})$-$\mathsf{uf\text{-}cmva}$-secure (Definition 3), where $\mathcal{A}$ is the set of streaming adversaries with memory $m$ that make a total number of at most $Q = 2^\lambda$ oracle queries in the unforgeability experiment (Definition 5). Furthermore the (streamed) tags are of size $O(m + \lambda \log \lambda)$.*

### 5.1 Construction

*Construction.* Let $|\mu|$ be a parameter. Let $n, m$ be integers such that $n \geq \Omega(\lambda + \mathrm{polylog}(m, \lambda, |\mu|))$. Let $\mathsf{Ext} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^\ell$ be a $(n, m, 2^{-\Omega(\lambda)})$-BSM extractor (Theorem 1), where $\ell = O(\lambda + \log(k + |\mu|))$; and suppose $m \geq \ell$, and $k \geq m/2 + 2\lambda$ while $k = O(m + \lambda \log \lambda)$.

Let $c \geq 3$ be a constant. Let $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ be a one-time, information-theoretic deterministic MAC with message space $\{0,1\}^{k+|\mu|}$ that can be evaluated in a streaming manner using memory $O(\lambda + \log(k + |\mu|))$ and tag size at most $\ell$, which is $\frac{(k+|\mu|)Q_V}{2^{c\lambda}}$-secure against adversaries that make at most one tagging query and $Q_V$ verification queries (which exists by Claim 1).

We define the following algorithms:

- $\overline{\mathsf{KeyGen}}(1^\lambda)$: Sample a seed $\mathsf{seed} \leftarrow \{0,1\}^k$ for $\mathsf{Ext}$ and a key $\mathsf{sk}$ for the one-time MAC. Output:
$$\overline{\mathsf{sk}} = (\mathsf{sk}, \mathsf{seed}).$$
- $\overline{\mathsf{MAC}}(\mathsf{sk}, \mu)$: On input $\mu \leftarrow \{0,1\}^{|\mu|}$ and $\overline{\mathsf{sk}} = (\mathsf{sk}, \mathsf{seed})$, sample $x \leftarrow \{0,1\}^k$ and output:
$$\overline{\sigma} = (x, \quad \mathsf{Ext}(x, \mathsf{seed}) \oplus \mathsf{MAC}(\mathsf{sk}, (x \| \mu)) ).$$
We consider here that $x$ is generated and output in a streaming manner, while $\mathsf{Ext}(x; \mathsf{seed})$ and $\mathsf{MAC}(\mathsf{sk}, (x\|\mu))$ are computed in a streaming manner.
- $\overline{\mathsf{Verify}}(\mu, \mathsf{sk}, \sigma)$: on a streamed input $(\mu, \overline{\sigma} = (x, \psi))$, compute
$$\sigma := \psi \oplus \mathsf{Ext}(x; \mathsf{seed}),$$
and output $\mathsf{Verify}(\mathsf{sk}, \sigma)$.

**Claim 2 (Correctness).** *Suppose* $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *is correct. Then* $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{MAC}}, \overline{\mathsf{Verify}})$ *is correct.*

**Claim 3 (Efficiency).** $\overline{\mathsf{KeyGen}}$ *can be computed (in place) with memory* $O(n)$*, and* $\mathsf{MAC}$ *and* $\mathsf{Verify}$ *can be computed in a streaming manner using memory* $O(n)$*.*

*Proof.* Computing $\mathsf{KeyGen}$, as well as the size of $\overline{\mathsf{sk}}$ follow directly by Theorem 1 and Claim 1, which gives a (one-time, information-theoretic) MAC with tag size $O(\lambda + \log(k + |\mu|))$ where keys can be sampled efficiently. Instantiating $\mathsf{Ext}$ with $k = O(m + \lambda \log \lambda)$ and $n \geq \Omega(\lambda + \log(k + |\mu|)) = \Omega(\lambda + \mathrm{polylog}(m, \lambda, |\mu|))$ gives the claim.

As for computing $\mathsf{MAC}$ and $\mathsf{Verify}$ in a streaming manner, given $\mathsf{seed}$ and $\overline{\mathsf{sk}}$, one can sample and output $x$ in a streaming manner. Then one can compute both $\mathsf{Ext}(x, \mathsf{seed})$ and $\mathsf{MAC}(\mathsf{sk}, (x, \mu))$ given $x$ as a streaming input, using memory $O(n)$. □

**Theorem 3 (Security).** *Let* $\mathcal{A}$ *be the set of streaming adversaries running in space* $m/2$ *and that make at most* $Q$ *oracle queries in the* $\mathsf{uf\text{-}cmva}$ *experiment (Definition 5). Let* $\varepsilon \leq Q_T Q_V 2^{m/2-k} + (Q_T + Q_V)2^{-\Omega(\lambda)} + Q_V \cdot (k + |\mu|)/2^{c\lambda}$*. Then* $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{MAC}}, \overline{\mathsf{Verify}})$ *is* $(\varepsilon, \mathcal{A})$*-*$\mathsf{uf\text{-}cmva}$*-secure.*

We refer to the full version for a proof.

Setting $\ell = O(\lambda + \log(k + |\mu|))$, $m = 2^{(c-1)\lambda}$ where $c \geq 3$, and $k \geq m/2 + 2\lambda$ while $k = O(m + \lambda \log \lambda)$, and $|\mu| = 2^\lambda$ yields Theorem 2.

*Remark 2 (Security against Adversaries with Temporary Unbounded Memory).* The construction above is in fact secure against adversaries that have temporary unbounded memory. Namely, on input a streamed tag $(x, \psi)$, they are allowed to use temporary unbounded memory to process $x$, as long as they compress their state down to some size $m$ memory before $\psi$ is received. BSM-extractor security still applies, as $x$ has sufficient min-entropy, and verification queries can then be handled using a similarly strong reduction. In particular the construction is secure even given non-uniform advice of size at most $m$.

## 6 MAC with Short Tags

In this section, we build a streaming MAC in the streaming BSM where the size of tags does *not* grow with the memory bound of the adversary. In particular, honest parties can store entire tags. More precisely, we prove the following theorem:

**Theorem 4 (MAC with Short Tags).** *For all* $m, \lambda$ *there exists a streaming MAC with memory* $n = O(\lambda\sqrt{m} + \lambda^2)$ *(Definition 2) which can authenticate messages of length up to* $2^{O(n)}$*, which is* $(2^{-\lambda}, \mathcal{A})$*-*$\mathsf{uf\text{-}cmva}$*-secure (Definition 3), where* $\mathcal{A}$ *is the set of streaming adversaries with memory* $m$ *that make at most* $Q = 2^\lambda$ *oracle queries in the unforgeability experiment (Definition 5) and that do not query the verification oracle on any input provided by the tagging oracle. Furthermore the resulting MAC has tags of size* $O(n)$*.*

Our main tool is the following result from [25].

**Theorem 5 ( [25]).** *Let $k \geq 1$ be an integer. There exist constants $C, \alpha > 0$ such that the following holds. Let* Adv *be an algorithm which takes as input a stream $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_Q, b_Q)$, where $\mathbf{a}_i \in \{0,1\}^k$, $\mathbf{s} \leftarrow \{0,1\}^k$ and $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle \in \mathbb{Z}_2$ and $Q \leq 2^{\alpha k}$, and outputs a vector $\mathbf{s}' \in \mathbb{Z}_2^k$. Suppose that* Adv *uses at most $Ck^2$ memory. Then $\Pr[\mathbf{s}' = \mathbf{s}] \leq O(2^{-\alpha k})$.*

*Futhermore the same conclusion holds if* Adv *is given access to unbounded temporary memory between receiving elements $(\mathbf{a}, b)$, as long as its state is compressed down to $Ck^2$ bits whenever an element $(\mathbf{a}, b)$ is received. We refer to such adversaries as* adversaries with memory $Ck^2$ and unbounded temporary memory *(Remark 1).*

We build our scheme as follows. We first give a construction of a MAC with small tags, with small message space and no verification queries which additionally satisfies message-hiding (Definition 4). We then show that any such MAC can be compiled to handle larger messages. The next step is to show that previously issued tagging queries can be efficiently recognized using a low amount of memory, independent of the number of queries made. Last, we show how to ensure security even given verification queries. We refer to the full version for more details.

## 7 Public-Key Signatures

In this section, we show how to build signature schemes in the streaming BSM. More precisely, we prove the following:

**Theorem 6.** *For all $m, \lambda$, there exists a streaming signature scheme with memory $n = \widetilde{O}(\lambda^3 + \sqrt{m\lambda})$ (Definition 7) which can authenticate messages of length up to $2^\lambda$, and which is $(2^{-\Omega(\lambda)}, \mathcal{A})$-*uf-cmva*-secure (Definition 8) where $\mathcal{A}$ is the set of streaming adversaries with memory $m$ that make a total number of at most $Q = 2^\lambda$ oracle queries in the unforgeability experiment (Definition 9). Furthermore, (streamed) signatures have size $\widetilde{O}(m + \lambda^{3/2})$.*

To do so, we first present a *set key-agreeement protocol* in Section 7.1, using our block entropy lemma Lemma 4 as our main technical tool. Then we show how to upgrade such a protocol to a signature scheme in Section 7.2. Last, we show that the quadratic gap between the adversary's memory bound and the honest users is optimal (up to poly($\lambda$) factors), in Section 7.3 (Theorem 8).

### 7.1 Set Key-Agreement Protocol

Given any parameters $m, \lambda$ and $\ell$, we define additional parameters: $b = 8\lambda(\ell+2)$, $k = \max(\lceil 4m/b \rceil + 4, 64\lambda)$, $q = \lceil 2\sqrt{k\lambda} \rceil$, which guarantees $k \geq 2m/b + 2q$. Let $\mathsf{Ext} : \{0,1\}^b \times \{0,1\}^d \to \{0,1\}^\ell$ be a $(\ell + 2\lambda, 2^{-\lambda})$-seeded extractor with some seed-length $d$, that can be computed using $O(b)$ space ( [18]). Consider the following "set key agreement" protocol between Alice and Bob, with one round of communication from Alice to Bob.

– $\overline{\mathsf{KeyGen}}$: Alice stream a value $\mathsf{vk} = (x = (x_1, \ldots, x_k), \mathcal{S}_A, (\mathsf{seed}_1, \ldots, \mathsf{seed}_q))$ generated as follows.
  - She chooses a uniformly random subset $\mathcal{S}_A \subseteq [k]$ of size $|\mathcal{S}_A| = q$ and stores it in memory as an ordered tuple $\mathcal{S}_A = (i_1^A, \ldots, i_q^A)$.
  - She streams a uniformly random value $x = (x_1, \ldots, x_k) \leftarrow (\{0,1\}^b)^k$ and additionally stores the values $x_{\mathcal{S}_A} = (x_{i_1^A}, \ldots, x_{i_q^A})$ in memory.
  - She streams her set $\mathcal{S}_A$.
  - She chooses $q$ random extractor seeds $\mathsf{seed}_j \leftarrow \{0,1\}^d$ and computes $\mathsf{sk}_j = \mathsf{Ext}(x[i_j^A]; \mathsf{seed}_j)$ for $j \in [q]$. She sends $(\mathsf{seed}_1, \ldots, \mathsf{seed}_q)$ and stores $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_q)$ in memory.

  Her final outputs consists of the secret key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_q) \in (\{0,1\}^\ell)^q$ stored in memory.
– $\overline{\mathsf{KeyReceive}}$: Bob processes the stream $\mathsf{vk} = (x = (x_1, \ldots, x_k), \mathcal{S}_A, (\mathsf{seed}_1, \ldots, \mathsf{seed}_q))$ as follows:
  - He chooses a uniformly random subset $\mathcal{S}_B \subseteq [k]$ of size $|\mathcal{S}_B| = q$ and stores it in memory.
  - As he receives the stream $x$, he additionally stores $x_{\mathcal{S}_B}$ in memory.
  - When he receives $\mathcal{S}_A = (i_1^A, \ldots, i_q^A)$ he stores it in memory.
  - When he receives $(\mathsf{seed}_1, \ldots, \mathsf{seed}_q)$ he computes $\mathcal{T} = \{j \in [q] \ : \ i_j^A \in \mathcal{S}_B\}$. For each $j \in \mathcal{T}$ he computes $\mathsf{sk}_j = \mathsf{Ext}(x_{i_j^A}; \mathsf{seed}_j)$. He stores the value $(\mathcal{T}, \mathsf{sk}_\mathcal{T} = (\mathsf{sk}_j)_{j \in \mathcal{T}})$ in memory.

  Bob's final outputs consists of $\mathsf{vd} = (\mathcal{T}, \mathsf{sk}_\mathcal{T} = (\mathsf{sk}_j)_{j \in \mathcal{T}})$ stored in memory.

**Lemma 5.** *The procedures* $\overline{\mathsf{KeyGen}}$ *and* $\overline{\mathsf{KeyReceive}}$ *can be computed by streaming algorithms with memory* $\widetilde{O}(bq) = \widetilde{O}(\lambda^2 \ell + \lambda\sqrt{\ell \cdot \lambda})$*. Furthermore* $\mathsf{sk}$ *and* $\mathsf{vd}$ *have size* $\widetilde{O}(q \cdot \ell) = \widetilde{O}(\sqrt{m\ell} + \lambda\ell)$*.*

**Lemma 6 (Set Key-Agreement Lemma).** *Let Eve be a streaming attacker with $m$ bits of memory, who observes the stream* $\mathsf{vk}$ *and outputs* $\mathsf{view}_{Eve}$ *in the above protocol. Let* $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_q)$ *be Alice's output and let* $\mathsf{vd} = (\mathcal{T}, \mathsf{sk}_\mathcal{T})$ *be Bob's output in the protocol. Then there is some index* $t \in \mathcal{T} \cup \{\perp\}$ *defined as a random variable depending on the entire protocol execution and the view of Eve, and some* $\varepsilon = 2^{-\Omega(\lambda)}$ *such that:*

$$\begin{aligned}(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j \ : \ j \in [q] \setminus \{t\}), \mathsf{sk}_t)\\ \approx_\varepsilon (\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j \ : \ j \in [q] \setminus \{t\}), u)\end{aligned}$$

*where* $u \leftarrow \{0,1\}^\ell$ *is random and independent of all other values, and we define* $\mathsf{sk}_t := \perp$ *if* $t = \perp$*.*

*Proof.* At the end of the protocol execution, we select the index $t$ as follows:

1. If $|\mathcal{S}_A \cap \mathcal{S}_B| < \lambda$ then set $t = \perp$. (We refer to this event as $\perp_0$.)
2. Choose a uniformly random $\mathcal{V} \subseteq \mathcal{S}_A \cap \mathcal{S}_B$ such that $|\mathcal{V}| = \lambda$. Let $\mathcal{W} = \mathcal{S}_A \setminus \mathcal{V}$.
3. Let $\mathsf{view}_{Eve}^0$ denote the view of Eve immediately after processing the $x$ component of the stream but before receiving $\mathcal{S}_A$.

4. Let $\mathsf{aux} = (\mathsf{view}^0_{Eve}, \mathcal{W}, x_{\mathcal{W}})$, and let $\mathsf{AUX}$ be a random variable corresponding to $\mathsf{aux}$. Let $X = (U_{kb}|\mathsf{AUX} = \mathsf{aux})$ be the random variable corresponding to choosing $x \leftarrow (\{0,1\}^b)^k$ from the uniform distribution, conditioned on $\mathsf{AUX} = \mathsf{aux}$. If, for the given value of $\mathsf{aux}$, we have $\mathbf{H}_\infty(X) < kb/2$ then set $t = \perp$. (We refer to this event as $\perp_1$.)
5. Let the sets $\mathsf{BAD} \subseteq (\{0,1\}^b)^k$ and $\mathcal{I}(x) \subseteq [k]$ be the sets from Lemma 4 defined with respect to the distribution $X$ with $\alpha = 1/2$, $\alpha_1 = \alpha_2 = 1/8$. Let $x$ be the value sent during the protocol execution.
   If $\mathcal{I}(x) \cap \mathcal{V} = \emptyset$ then set $t = \perp$. (We refer to this event as $\perp_2$.)
6. Let $i^*$ be the smallest value in $\mathcal{I}(x) \cap \mathcal{V} \subseteq \mathcal{S}_A = \{i^A_1, \ldots, i^A_q\}$ and let $t^*$ be the value such $i^A_{t^*} = i^*$. Set $t = t^*$.

We show a sequence of hybrid distributions that are statistically close.

*Hybrid 0.* This is the distribution on the left-hand side of the Lemma

$$(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j \ : \ j \in [q] \setminus \{t\}), \mathsf{sk}_t)$$

*Hybrid 1.* We now change how we select the sets $\mathcal{S}_A, \mathcal{S}_B, \mathcal{V}, \mathcal{W}$.

In hybrid 0, $\mathcal{S}_A, \mathcal{S}_B \subseteq [k]$ are chosen uniformly at random with $|\mathcal{S}_A| = |\mathcal{S}_B| = q$. Then, if $|\mathcal{S}_A \cap \mathcal{S}_B| > \lambda$, we choose $\mathcal{V} \subseteq \mathcal{S}_A \cap \mathcal{S}_B$ of size $|\mathcal{V}| = \lambda$ uniformly at random and define $\mathcal{W} = \mathcal{S}_A \setminus \mathcal{V}$.

In hybrid 1, we instead choose $\mathcal{W} \subseteq [k]$ of size $|\mathcal{W}| = q - \lambda$ uniformly at random. Then we select $\mathcal{V} \subseteq [k] \setminus \mathcal{W}$ of size $|\mathcal{V}| = \lambda$ uniformly at random. We define $\mathcal{S}_A = \mathcal{V} \cup \mathcal{W}$. Then we choose $\mathcal{S}_B \subseteq [k]$ of size $|\mathcal{S}_B| = q$ uniformly at randomly subject to $\mathcal{V} \subseteq [k]$.

Note that the above change ensures that, in hybrid 1, $\mathcal{V} \subseteq \mathcal{S}_A \cap \mathcal{S}_B$ with $|\mathcal{V}| = \lambda$, and therefore the event $\perp_0$ never occurs.

Hybrids 1 is distributed identically to hybrid 0 if we condition on $\perp_0$ not occurring in hybrid 0. Therefore the statistical distance between the hybrids is bounded by the probability that $\perp_0$ occurs, meaning that $|\mathcal{S}_A \cap \mathcal{S}_B| < \lambda$, in hybrid 0. This is bounded by $\varepsilon_1 \leq 2^{-\lambda/4} = 2^{-\Omega(\lambda)}$ by Lemma 3 as $q \geq 2\sqrt{k\lambda}$.

*Hybrid 2.* In hybrid 1, if the event $\perp_1$ occurs (i.e., $\mathsf{aux}$ is such hat $\mathbf{H}_\infty(X) < kb/2$) then we set $t = \perp$ and $\mathsf{sk}_t = \perp$. In hybrid 2, if $\perp_1$ occurs then we set $t = \perp$ and choose $\mathsf{sk}_t \leftarrow \{0,1\}^\ell$ uniformly at random.

Hybrids 1 and 2 are distributed identically as long as $\perp_1$ does not occur, and so the statisticaly distance between them is bounded by the probability that $\mathbf{H}_\infty(X) < kb/2$. Note that $X = (U_{kb}|\mathsf{AUX} = \mathsf{aux})$. Furthermore, we have:

$$\mathbf{H}_\infty(U_{kb}|\mathsf{AUX}) \geq \mathbf{H}_\infty(U_{kb}|\mathcal{W}) - m - (q-\lambda)\cdot b \geq kb - m - (q-\lambda)\cdot b \geq kb/2 + \lambda,$$

where the first inequality follows from Lemma 2 since $\mathsf{aux} = (\mathsf{view}^0_{Eve}, \mathcal{W}, x_{\mathcal{W}})$ where $|\mathsf{view}^0_{Eve}| = m$ and $|x_{\mathcal{W}}| = (q-\lambda)\cdot b$, and the second inequality follows since $\mathcal{W}$ is chosen independently of $x \leftarrow U_{kb}$ and hence does not reduce entropy. Therefore, by Lemma 1:

$$\Pr_{\mathsf{aux} \leftarrow \mathsf{AUX}}[\underbrace{\mathbf{H}_\infty(U_{kb}|\mathsf{AUX} = \mathsf{aux})}_{\mathbf{H}_\infty(X)} < kb/2] \leq 2^{-\lambda},$$

and so the statistical distance between the hybrids is $\varepsilon_2 \leq 2^{-\lambda}$.

*Hybrid 3.* In hybrid 3, if $\bot_1$ does not occur and $x \in \mathsf{BAD}$, then we define $t = \bot$ and choose $\mathsf{sk}_t \leftarrow \{0,1\}^\ell$ uniformly at random. We refer to this event as $\bot_{1.5}$.

Hybrids 1 and 2 are distributed identically unless: $\bot_1$ does not occur and $x \in \mathsf{BAD}$. But, if we fix any $\mathsf{aux}$ for which $\bot_1$ does not occur, then by Lemma 4 we can bound the probability that $x \in \mathsf{BAD}$ by $k \cdot 2^{-b/4} = 2^{-\Omega(\lambda)}$. Therefore, the statistical distance between the hybrids is $\varepsilon_3 \leq 2^{-\Omega(\lambda)}$.

*Hybrid 4.* In hybrid 3, if $\bot_1$ and $\bot_{1.5}$ don't occur but $|\mathcal{I}(x) \cap \mathcal{V}| = \emptyset$ then we set $t = \bot$ and $\mathsf{sk}_t = \bot$. In hybrid 4, if $\bot_1$ and $\bot_{1.5}$ don't occur but $|\mathcal{I}(x) \cap \mathcal{V}| = \emptyset$, then we set $t = \bot$ and choose $\mathsf{sk}_t \leftarrow \{0,1\}^\ell$ uniformly at random.

Hybrids 3 and 4 are distributed identically unless: $\bot_1$ and $\bot_{1.5}$ don't occur but $|\mathcal{I}(x) \cap \mathcal{V}| = \emptyset$. Let us fix any $\mathsf{aux}, x$ such that $\bot_1, \bot_{1.5}$ do not occur. This also fixes $\mathcal{I}(x)$. Furthermore, by Lemma 4, we have $|\mathcal{I}(x)| \geq k/8$. Moreover, $I(x) \subseteq [k] \setminus \mathcal{W}$ since, for $i \in \mathcal{W}$, the values $X_i$ are completely fixed by $\mathsf{aux}$ and hence have entropy 0. On the other hand $\mathcal{V}$ is uniformly random over $[k] \setminus \mathcal{W}$ with $|\mathcal{V}| = \lambda$, and independent of $\mathcal{I}(x)$. Therefore the probability that $\mathcal{V} \cap \mathcal{I}(x) = \emptyset$ is bounded by $\varepsilon_4 \leq (7/8)^\lambda = 2^{-\Omega(\lambda)}$, which also bounds the statistical distance between these hybrids.

*Hybrid 5.* We undo the change introduced in hybrid 3. That is, we no longer check if $x \in \mathsf{BAD}$ and take any special action if it is.

Hybrids 4 and 5 are statistically close for the same reason hybrids 2 and 3 are statistically close, with distance $\varepsilon_5 \leq 2^{-\Omega(\lambda)}$.

*Hybrid 6.* In hybrid 6, we now always choose $\mathsf{sk}_t \leftarrow \{0,1\}^\ell$ uniformly random.

We argue that hybrids 5 and 6 are statistically close by the security of the extractor. Let us fix and condition on any choice of values

$$(\mathsf{aux} = (\mathsf{view}^0_{Eve}, \mathcal{W}, x_{\mathcal{W}}), \mathcal{V}, \mathcal{S}_B, i^*, x_1, \ldots, x_{i^*-1}, (\mathsf{seed}_j)_{j \in [q] \setminus t^*})$$

chosen during the experiment, subject to $\bot_1, \bot_2$ not occurring. We define $i^*$ as being the smallest value in $\mathcal{V} \cap \mathcal{I}(x)$ and $t^*$ is defined as the value such $i^A_{t^*} = i^*$, which is fixed once $i^*$ and $\mathcal{S}_A = \mathcal{V} \cup \mathcal{W} = \{i^A_1, \ldots, i^A_q\}$ are fixed. The above values also fix $\mathsf{sk}_{\mathcal{W}} := (\mathsf{sk}_j = \mathsf{Ext}(x_{i^A_j}; \mathsf{seed}_j)_{i^A_j \in \mathcal{W}}$.

If either $\bot_1$ occurs or $\bot_2$ occurs then the hybrids are identical, and therefore it suffices to only show that they are statistically close for any fixed choice of the values as above for which $\bot_1, \bot_2$ do not occur.

Note that, by Lemma 4, fixing $x_1, \ldots, x_{i^*-1}$ also fixes whether $i \in \mathcal{I}(x)$ for all $i \leq i^*$ and therefore, once we fix all of the above, conditioning on $i^*$ being the smallest value in $\mathcal{V} \cap \mathcal{I}(x)$ is equivalent to just conditioning on $i^* \in \mathcal{I}(x)$. Therefore, the distribution of $X_{i^*}$ (i.e., the $i^*$ block of $x$) conditioned on the above is equivalent to $(X_{i^*} | X_1 = x_1, \ldots, X_{i^*-1} = x_{i^*_1}, i^* \in \mathcal{I}(x))$, and By Lemma 4, it has min-entropy $\mathbf{H}_\infty(X_{i^*}) \geq b/8$.[7]

[7] For the rest, of the argument we will condition on all the fixed values implicitly and will not write this conditioning explicitly.

Let $\mathsf{SK}_{\mathcal{V}^-} := (\mathsf{SK}_j = \mathsf{Ext}(X_{i_j^A}; \mathsf{seed}_j)_{i_A^j \in \mathcal{V} \setminus \{i^*\}}$. Then

$$\mathbf{H}_\infty(X_{i^*} | \mathsf{SK}_{\mathcal{V}}^-) \geq \mathbf{H}_\infty(X_{i^*}) - (\lambda - 1) \cdot \ell \geq b/8 - (\lambda - 1) \cdot \ell \geq \ell + 2\lambda.$$

Let $\mathsf{SEED}_{t^*}$ to be a random variable for $\mathsf{seed}_{t^*}$. Then

$$(\mathsf{SK}_{t^*} = \mathsf{Ext}(X_{i^*}; \mathsf{SEED}_{t^*}), \mathsf{SEED}_{t^*}, \mathsf{SK}_{\mathcal{V}^-}) \approx_{\varepsilon_6} (U_\ell, \mathsf{SEED}_{t^*}, \mathsf{SK}_{\mathcal{V}^-})$$

where $\varepsilon_6 \leq 2^{-\lambda}$, by the security of the extractor.

Now observe that, conditioned on the fixed values, the outputs of hybrid 5 and 6 are completely defined given the additional values $\mathsf{SEED}_{t^*}, \mathsf{SK}_{\mathcal{V}^-}$ and either $\mathsf{SK}_{t^*} = \mathsf{Ext}(X_{i^*}; \mathsf{SEED}_{t^*})$ in hybrid 5 or $U_\ell$ in hybrid 6. In particular, everything else in the hybrid is defines as follows:

- $\mathcal{T}, t$ are completely determined by the fixed values $\mathcal{S}_A = \mathcal{V} \cup \mathcal{W} = \{i_1^A, \ldots, i_q^A\}$, $\mathcal{S}_B, i^*$.
- $\mathsf{view}_{Eve}$ is defined in terms of $\mathsf{view}_{Eve}^0, \mathcal{S}_A, \mathcal{S}_B, (\mathsf{seed}_j)_{j \in [q] \setminus t^*}, \mathsf{SEED}_{t^*}$.
- $\mathsf{sk}_{-t}$ consists of $\mathsf{SK}_{\mathcal{V}^-}$ and $\mathsf{sk}_{\mathcal{W}} := (\mathsf{sk}_j = \mathsf{Ext}(x_{i_j^A}, \mathsf{seed}_j)_{i_j^A \in \mathcal{W}}$, where the latter only depends on the fixed values $\mathcal{W}, x_{\mathcal{W}}, (\mathsf{seed}_j)_{j \in [q] \setminus t^*}$.
- And the last component in the hybrid is $\mathsf{SK}_{t^*}$ in hybrid 5 and $U_\ell$ in hybrid 6.

Therefore, the statistical distance between the hybrids is bounded by $\varepsilon_6 \leq 2^{-\lambda}$.

*Hybrid 7.* Undo the change from Hybrid 1 and select $\mathcal{S}_A, \mathcal{S}_B, \mathcal{V}, \mathcal{W}$ as in Hybrid 0.

Hybrids 6 and 7 are statistically close for the same reason hybrids 0 and 1 are statistically close, with distance $\varepsilon_7 \leq 2^{-\Omega(\lambda)}$.

This hybrid is equivalent to the right-hand side distribution of the Lemma.

$$(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j \ : \ j \in [q] \setminus \{t\}), u)$$

where $u \leftarrow \{0,1\}^\ell$ is random and independent of all other values.

Combining the above hybrids, the Lemma holds with $\varepsilon \leq \sum_{i=1}^{7} \varepsilon_i = 2^{-\Omega(\lambda)}$. □

### 7.2 From Set Key Agreement to Signatures

*Construction.* Let $\ell, m, |\mu|$ be parameters. Let $\mathcal{A} = \mathcal{A}(O(m), Q)$ be any class of *non-uniform* streaming algorithms such that there exists a constant $C'$ such that algorithms in $\mathcal{A}$ run with memory at most $C'm$ making at most $Q$ oracle calls (Definition 5). Let $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ be a streaming MAC with memory $\ell$ and message space $\{0,1\}^{|\mu|}$ satisfying $(\varepsilon, \mathcal{A})$-$\mathsf{uf\text{-}cmva}$-security (Definition 3) for some $\varepsilon > 0$, such that signing keys (of size at most $\ell$) are uniformly random.

Let $b$, $k$, $q$, be the parameters instantiated in Section 7.1, and let $\mathsf{Ext} : \{0,1\}^b \times \{0,1\}^d \to \{0,1\}^\ell$ be a $(\ell + 2\lambda, 2^{-\lambda})$-seeded extractor.

We define a streaming signature scheme $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{KeyReceive}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Verify}})$ as follows.

– $\overline{\mathsf{KeyGen}}(1^\lambda)$: stream $\mathsf{vk}$ and store $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_q)$, as defined in Section 7.1.
– $\overline{\mathsf{KeyReceive}}(1^\lambda, \mathsf{vk})$: on input a streamed verification key $\mathsf{vk}$, store $\mathsf{vd} = (\mathcal{T}, \mathsf{sk}_\mathcal{T} = (\mathsf{sk}_j)_{j\in\mathcal{T}})$ as specified in Section 7.1.
– $\overline{\mathsf{Sign}}(\mathsf{sk}, \mu)$: on input a (potentially streamed) message $\mu$, compute and output (potentially in a streaming manner):

$$\sigma = \{\sigma_j = \mathsf{MAC}(\mathsf{sk}_j, \mu)\}_{j\in[q]}.$$

– $\overline{\mathsf{Verify}}(\mathsf{vd}, \mu, \sigma)$: on input $\mathsf{vd} = (\mathcal{T}, \{\mathsf{sk}_j\}_{j\in\mathcal{T}})$, a (potentially streamed) message $\mu$, and a (potentially streamed) signature $\sigma = \{\sigma_j\}_{j\in\mathcal{T}}$, output

$$\bigwedge_{i\in\mathcal{T}} \mathsf{Verify}(\mathsf{sk}_j, \mu, \sigma_j),$$

with the convention that it outputs 1 if $\mathcal{T} = \emptyset$.

**Claim 4 (Correctness).** *Suppose* $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *is correct. Then* $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{MAC}}, \overline{\mathsf{Verify}})$ *is correct.*

**Claim 5 (Efficiency).** *Suppose* $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *be a streaming MAC with memory* $\ell$ *(Definition 2). Then* $\overline{\mathsf{KeyGen}}$ *and* $\overline{\mathsf{KeyReceive}}$ *can be computed by streaming algorithms with memory* $\widetilde{O}(\lambda^2\ell + \lambda\sqrt{\ell\cdot\lambda})$*, and* $\mathsf{sk}$ *and* $\mathsf{vd}$ *have size* $\widetilde{O}(\sqrt{m\ell} + \lambda\ell)$*.* $\overline{\mathsf{Sign}}$ *and* $\overline{\mathsf{Verify}}$ *can be computed by streaming algorithms with memory* $O(\sqrt{m\ell} + \lambda\ell)$*.*

**Theorem 7 (Security).** *Let* $\mathcal{A}_{MAC} = \mathcal{A}(O(m), Q)$ *be any class of* non-uniform[8] *streaming algorithms such that there exists a constant* $C'$ *such that algorithms in* $\mathcal{A}_{MAC}$ *run with memory at most* $C'm$ *making at most* $Q$ *oracle calls (Definition 5), and* $\mathcal{A}_{Sig}$ *be its signature counterpart (Definition 9). Suppose* $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *is* $(\varepsilon, \mathcal{A}_{MAC})$-$\mathsf{uf\text{-}cmva}$*-secure (Definition 3). Then* $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{KeyReceive}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Verify}})$ *is* $(2\varepsilon, \mathcal{A}_{Sig})$-$\mathsf{uf\text{-}cmva}$*-secure (Definition 8).*

*Proof.* Let $\mathsf{Adv} \in \mathcal{A}_{MAC}$ be a streaming adversary with memory $m$ for the $\mathsf{uf\text{-}cmva}$ experiment for $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{MAC}}, \overline{\mathsf{Verify}})$. Consider the following hybrid experiments.

*Hybrid* $H_0$. This corresponds to the $\mathsf{uf\text{-}cmva}$ experiment $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$.

*Hybrid* $H_1$. We change how $\overline{\mathsf{KeyReceive}}$ in Step 1 of $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$ is computed. $\mathsf{vd}$ is now computed as $(\mathcal{T}, \mathsf{sk}_{-t}, u)$ where $u \leftarrow \{0,1\}^\ell$. In other words, the secret key $\mathsf{sk}_t$ is replaced by uniformly random, where $t$ is given by Lemma 6.

**Lemma 7.** *Suppose* $b, k, q$ *are instantiated as in Section 7.1. Then the advantage of* $\mathsf{Adv}$ *decreases by at most* $\varepsilon$ *between* $H_0$ *and* $H_1$*:*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_1}(1^\lambda, \mathsf{Adv}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_0}(1^\lambda, \mathsf{Adv}) = 1\right]\right| \le \varepsilon,$$

*where, for* $b \in \{0,1\}$, $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_b}$ *denotes the* $\mathsf{uf\text{-}cmva}$ *experiment in Hybrid* $H_b$*.*

[8] Looking ahead, this will not affect our final result, because our base MAC is secure against non-uniform adversaries.

*Proof.* Denote by $\mathsf{view}_{Eve}$ the state of $\mathsf{Adv}$ after Step 1 of $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$. Let $\mathsf{sk} = (\mathsf{sk}_j)_{j\in[q]}$ and let $\mathsf{vd} = (\mathcal{T}, (\mathsf{sk}_j)_\mathcal{T})$ be the respective outputs of $\overline{\mathsf{KeyGen}}$ and $\overline{\mathsf{KeyReceive}}$ in Step 1.

The lemma follows as the output of $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$ can be computed using $(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j)_{j\neq t}, \mathsf{sk}_t)$ alone (with the convention that $\mathsf{Verify}(\perp, \mu, \sigma) = 1$ for all $\mu, \sigma$.). Namely, run $\mathsf{Adv}$ starting from Step 2 with state $\mathsf{view}_{Eve}$. Signing queries are answered using the knowledge of $\mathsf{sk}_{-t}$, $\mathsf{sk}_t$ and $t$. Verification queries as well as the final forgery are answered using the knowledge of $\mathsf{sk}_{\mathcal{T}\setminus\{t\}}$, $\mathcal{T}$, $\mathsf{sk}_t$ and $t$.

By Lemma 6:

$$(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j)_{j\neq t}, \mathsf{sk}_t) \approx_\varepsilon (\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j)_{j\neq t}, u).$$

Now, computing the output of $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}(1^\lambda, \mathsf{Adv})$ using $(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j)_{j\neq t}, \mathsf{sk}_t)$ corresponds to $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_0}(1^\lambda, \mathsf{Adv})$, and using $(\mathsf{view}_{Eve}, \mathcal{T}, t, \mathsf{sk}_{-t} = (\mathsf{sk}_j)_{j\neq t}, u)$ corresponds to $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_1}(1^\lambda, \mathsf{Adv})$, and the lemma follows. □

**Lemma 8.** *Suppose* $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *is* $(\varepsilon, \mathcal{A})$-$\mathsf{uf\text{-}cmva}$-*secure. Then*

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_1}(1^\lambda, \mathsf{Adv}) = 1] \leq \varepsilon.$$

*Proof.* Let $\mathsf{Adv}$ be an adversary for $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{KeyReceive}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Verify}})$ with memory $m$ having advantage $\varepsilon'$ in $\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_1}$. We build a (non-uniform) adversary against the $\mathsf{uf\text{-}cmva}$-security of $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ with memory $m + O(q\ell)$ and advantage $\varepsilon'$ as follows.

Our reduction computes $\overline{\mathsf{KeyGen}}$, streams $\mathsf{vk}$ to $\mathsf{Adv}$ and stores a secret key $(\mathsf{sk}_i)_{i\in[q]}$. It computes $\mathsf{vd} = (\mathcal{T}, \mathsf{sk}_\mathcal{T})$. It receives the (inefficiently computable) index $t$ defined in Lemma 6 as non-uniform advice: note that $t$ only depends on the execution of the set key agreement and the resulting view of the adversary, and is therefore independent of the unforgeability experiment for $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$.

To answer signing queries on message $\mu$, the reduction computes for all $j \in [q], j \neq t$: $\sigma_j = \mathsf{Sign}(\mathsf{sk}_j, \mu)$, and makes a signing query $\mu$ to the $\mathsf{MAC}$ oracle (which implicitly uses $\mathsf{MAC.sk} = \mathsf{sk}_t$), thus obtaining $\sigma_t$. It forwards $(\sigma_j)_{j\in[q]}$ to $\mathsf{Adv}$ (using its knowledge of $t$).

To answer verification queries with message $\mu$ and signature $\sigma = (\sigma_j)_{j\in[q]}$, the reduction computes for all $j \in \mathcal{T}, j \neq t$: $b_j \leftarrow \mathsf{Verify}(\mathsf{sk}_j, \mu, \sigma_j)$ and makes a verification query with input $(\mu, \sigma_j)$ to $\mathsf{Verify}$ oracle, obtaining a bit $b_t$. It outputs $\bigwedge_{j\in T} b_j$. The final output of the experiment is computed identically.

Our reduction runs in memory $m+q\ell$, makes the same number of queries $Q$ as $\mathsf{Adv}$, and succeeds if $\mathsf{Adv}$ successfully produces a forgery. Therefore its advantage is at least $\varepsilon'$ which is at most $\varepsilon$ by $\mathsf{uf\text{-}cmva}$-security of $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{KeyReceive}}, \overline{\mathsf{Sign}}, \overline{\mathsf{Verify}})$. □

Overall, the advantage of $\mathsf{Adv}$ in the $\mathsf{uf\text{-}cmva}$ experiment for $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{MAC}}, \overline{\mathsf{Verify}})$ is at most

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}cmva}}_{H_0}(1^\lambda, \mathsf{Adv}) = 1] \leq 2\varepsilon.$$

□

We instantiate $(\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ with our construction in Section 5, which has uniformly random MAC keys, using $|\mu| = 2^\lambda$. Recall that this construction is secure against non-uniform adversaries (and in fact, adversaries with unbounded temporary memory, see Remark 2). Setting $\ell = O(\lambda + \mathrm{polylog}(m, \lambda))$, gives $n = \widetilde{O}(\lambda^3 + \sqrt{m\lambda})$. Combined with the following observation with $\tau = m$, we obtain Theorem 6.

*Remark 3 (Optimizing the Communication Cost).* The signatures for our scheme consist of $[q]$ independent copies of $\mathsf{MAC}(\mathsf{sk}_j, \mu)$, $j \in [q]$, where $\mathsf{MAC}$ is our base MAC. Using our scheme from Section 5, and noting that (1) security only relies on security of a single of the copies, and (2) tags of the form $(x, \psi)$ can be computed given $x$ and $\mathsf{sk}$, we can instead compute our signatures as $(x, (\psi_j)_{j\in[q]})$, namely reusing the $x$ part across the different copies of the base MAC, while preserving correctness and security. This makes our signatures of size $\widetilde{O}(\tau + q\lambda)$, where $\tau$ is the size of the tags from $\mathsf{MAC}$.

*Remark 4 (Weaker Notions of Security).* Our construction constructs a $\mathsf{uf\text{-}cmva}$-secure signature scheme starting from any $\mathsf{uf\text{-}cmva}$-secure MAC. We note that this extends to weaker notions of security, namely starting from a (selectively)-unforgeable MAC with signing (and verification) queries, one obtains a signature scheme satisfying the same notion of security.

### 7.3 Lower Bound for Signatures

We show here that any streaming signature with memory $n$, namely, where all the procedures can be run in a streaming manner with memory $n$, can only be secure against streaming adversaries with memory $m = O(n^2)$.

**Theorem 8.** *Suppose* $(\mathsf{KeyGen}, \mathsf{KeyReceive}, \mathsf{Sign}, \mathsf{Verify})$ *is a streaming signature with memory* $n$ *(Definition 7). Let* $\mathcal{A} = \mathcal{A}(m)$ *be the set of streaming adversaries running with memory* $m$ *(Definition 9). Suppose that* $(\mathsf{KeyGen}, \mathsf{KeyReceive}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(\varepsilon, \mathcal{A})$-$\mathsf{suf\text{-}cma}$*-secure (Definition 8). Then* $m = O(n^2)$.

Theorem 8 holds even for weakly secure schemes, where adversaries are not even allowed to make any oracle queries in the unforgeability experiment. We refer to the full version for a proof, which uses similar ideas to the lower bounds from [11, 9].

## References


1. Y. Aumann, Yan Zong Ding, and M.O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.
2. Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *39th FOCS*, pages 493–502. IEEE Computer Society Press, November 1998.

3. Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 292–306. Springer, Heidelberg, August 1997.
4. Yan Zong Ding. Oblivious transfer in the bounded storage model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 155–170. Springer, Heidelberg, August 2001.
5. Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. *Journal of Cryptology*, 20(2):165–202, April 2007.
6. Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '02, page 1–26, Berlin, Heidelberg, 2002. Springer-Verlag.
7. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Heidelberg, April 2012.
8. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
9. Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. Cryptology ePrint Archive, Report 2021/1270, 2021. https://ia.cr/2021/1270.
10. Stefan Dziembowski and Ueli M. Maurer. Tight security proofs for the bounded-storage model. In *34th ACM STOC*, pages 341–350. ACM Press, May 2002.
11. Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 126–137. Springer, Heidelberg, May 2004.
12. Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 990–1002. ACM Press, June 2018.
13. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.
14. Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 500–524. Springer, Heidelberg, May 2019.
15. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.
16. Dowon Hong, Ku-Young Chang, and Heuisu Ryu. Efficient oblivious transfer in the bounded-storage model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 143–159. Springer, Heidelberg, December 2002.
17. Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 52–66. Springer, Heidelberg, December 2001.
18. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 12–24, New York, NY, USA, 1989. Association for Computing Machinery.

19. Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 293–308. Springer, Heidelberg, August 2005.
20. Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, Heidelberg, May 2011.
21. Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 1067–1080. ACM Press, June 2017.
22. Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 257–271. Springer, Heidelberg, August 2002.
23. Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, January 1992.
24. Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.
25. Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In Irit Dinur, editor, *57th FOCS*, pages 266–275. IEEE Computer Society Press, October 2016.
26. Ran Raz. A time-space lower bound for a large class of learning problems. In Chris Umans, editor, *58th FOCS*, pages 732–742. IEEE Computer Society Press, October 2017.
27. Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, January 2004.