Speak Much, Remember Little: Cryptography in the Bounded Storage Model, Revisited

Yevgeniy Dodis* NYU
Willy Quach† Northeastern
Daniel Wichs‡ Northeastern and NTT Research

September 22, 2021

Abstract

The goal of the *bounded storage model (BSM)* is to construct unconditionally secure cryptographic protocols, by only restricting the storage capacity of the adversary, but otherwise giving it unbounded computational power. Here, we consider a *streaming* variant of the BSM, where honest parties can stream huge amounts of data to each other so as to overwhelm the adversary's storage, even while their own storage capacity is significantly smaller than that of the adversary. Prior works showed several impressive results in this model, including *key agreement* and *oblivious transfer*, but only as long as adversary's storage $m = O(n^2)$ is at most quadratically larger than the honest user storage n. Moreover, the work of Dziembowski and Maurer (DM) also gave a seemingly matching lower bound, showing that key agreement in the BSM is impossible when $m > n^2$.

In this work, we observe that the DM lower bound only applies to a significantly more restricted version of the BSM, and does not apply to the streaming variant. Surprisingly, we show that it is possible to construct key agreement and oblivious transfer protocols in the streaming BSM, where the adversary's storage can be significantly larger, and even exponential $m = 2^{O(n)}$. The only price of accommodating larger values of m is that the round and communication complexities of our protocols grow accordingly, and we provide lower bounds to show that an increase in rounds and communication is necessary.

As an added benefit of our work, we also show that our oblivious transfer (OT) protocol in the BSM satisfies a simulation-based notion of security. In contrast, even for the restricted case of $m = O(n^2)$, prior solutions only satisfied a weaker indistinguishability based definition. As an application of our OT protocol, we get general multiparty computation (MPC) in the BSM that allows for up to exponentially large gaps between m and n, while also achieving simulation-based security.

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

†E-mail:quach.w@northeastern.edu. Part of this work was completed during an internship at NTT Research.

‡E-mail:wichs@ccs.neu.edu. Research supported by NSF grant CNS-1750795, CNS-2055510 and the Alfred P. Sloan Research Fellowship.

Contents

1 Introduction

It is well known that Alice and Bob cannot agree on a shared secret by communicating over public (authentic) channel, when the eavesdropper Eve has unbounded computational resources. Thus, traditional cryptography assumes that Eve is "resource bounded", and most commonly, bounds her run time. Many key agreement schemes have been constructed in this setting, starting with the seminal work of Diffie and Hellman [DH76], under various computational hardness assumptions. Of course, the dream of cryptography is to construct unconditionally secure protocols, without relying on any unproven assumptions, but unfortunately, this is currently beyond our reach, as it easily implies $P \neq NP$.

In contrast, the *Bounded Storage Model* (BSM), introduced in the pioneering work of Maurer [Mau92], *only* assumes that Eve has bounded space rather than time. A long series of works [CM97, CCM98, Din01, HCR02, DR02, ADR02, DM02, Lu02, Vad04, DHRS07, Raz16, KRT17, Raz17, GRT18, GZ19] showed that it is possible to construct many kinds of unconditionally secure cryptographic schemes in this model, including key agreement and oblivious transfer over a public channel, provided that Eve's storage is not too large.

It turns out that there are several related-but-different variants of the BSM. In this work, we focus on a natural variant, which we refer to as the "streaming BSM". We first discuss this model, which will be the default throughout the paper. We will compare the streaming BSM model to other variants from the literature further below.

"Streaming" BSM. In this model, parties can generate and send huge amounts of data to each other, but only have limited local memory. The model is parameterized by two parameters: the honest parties' space capacity n, and the attacker's space capacity m, where $m \gg n$. We assume parties operate in the streaming model: they generate/receive communication one bit at a time, while only maintaining a small local memory throughout. The total communication can be huge, say $k \gg m \gg n$, and can occur over multiple back-and-forth rounds.

For example, Alice can stream a huge random string X of length k to Bob by sampling it one bit at a time; both Alice and Bob can store some small subset of n physical locations of X, or they can store the parity of X computed in a streaming manner, but neither of them can remember all of X. The attacker Eve is also streaming, just like Alice and Bob, but has much larger memory capacity $m \gg n$. We call the resulting model the (n, m)-BSM, and it will be the default throughout the paper; sometimes, we will explicitly refer to it as the "streaming BSM" to disambiguate from other variants.

Prior Results. As with computational cryptography, in the BSM we can consider a symmetric-key setting, where honest parties can share a short secret key that can be used to encrypt arbitrarily many messages over time, or a public-key setting, where no shared key is available. In both cases the parties can freely communicate over a public channel, and the goal is to achieve unconditional, information-theoretic (IT) security, without making any additional computational assumptions.

In the symmetric-key setting, a series of beautiful papers [Mau92, DR02, ADR02, DM02, Lu02, Vad04, Raz16, KRT17, Raz17, GRT18] showed that it is possible to achieve arbitrarily large gaps between the space of the attacker and that of the honest parties, up to exponential: $m = 2^{O(n)}$. (Of course, the price of allowing large values of m is that the ciphertext size has to grow proportionally, to ensure that we eventually overwhelm the adversary's storage capacity to overcome the Shannon lower bound. Therefore, if we want to limit ourselves to schemes with polynomial ciphertext size, then m is limited to some arbitrarily large polynomial.)

Amazingly, it is even possible to construct unconditionally secure public-key schemes in the BSM, and prior works [CM97, CCM98, Din01, HCR02, DHRS07, GZ19] constructed BSM schemes for key agreement (KA) and oblivious transfer (OT), which is then complete for all multi-party computation (MPC) [Kil88, IPS08]. However, all of the prior works in the public-key setting allowed at most a quadratic gap between the adversarial and the honest users storage: $m = O(n^2)$. In fact, the work of Dziembowski and Maurer [DM04] seemed to suggest that this limitation is inherent, by showing there is no KA protocol in the BSM when $m > n^2$. So it may have appeared that the question of designing public-key cryptographic primitives in the

BSM had been settled.

Our Question and Main Result. However, as we observe in this work, and discuss in Section 1.1, the lower bound of [DM04] was only shown in a *restricted* version of the BSM model, and does not apply to the more general "streaming" BSM. Most significantly, the authors critically assumed that there is *at most one* "long" communication round in the key agreement protocol, where the length k of the streamed message overwhelms the storage capacity m of the attacker. While this restriction was satisfied by many prior work in the BSM (see Section 1.1), this opens the possibility that it might be possible to break the quadratic barrier of [DM04] when parties use the full streaming power of the BSM, including the ability to stream *several* "long" messages to each other. This is the main question of this work:

Main Question: *Do there exist unconditionally secure key agreement (KA) and oblivious trasnfer (OT) protocols in the streaming (n, m)-BSM, when m is allowed to be much larger than n^2?*

We answer this question in the *affirmative*, and show that we can allow arbitrarily large gaps between m and n, up to exponential $m = 2^{O(n)}$. Surprisingly, this shows that unlike time-bounded public-key cryptography, — where we must rely on *additional* computational assumptions, — space-bounded public-key cryptography can be proven *unconditionally*, while supporting *arbitrary gaps* between the powers of honest parties and the attacker. The price of allowing large values of m is that the round and communication complexities of the protocols grow correspondingly and we also provide a lower bound to show that this is inherent. In particular, this means that if we want limit ourselves to protocols with polynomial (round/communication) efficiency, then m is limited to be some arbitrarily large polynomial.

Before describing our results in detail, we start by describing the different variants of the BSM, to understand the gap that we crucially exploit between the model used in the lower bound of [DM04] and the model for our upper bounds.

1.1 Modeling Gap: Breaking the Quadratic Barrier

Many of the prior works in the BSM, including the original work of [Mau92] and the lower bound of Dziembowski and Maurer [DM04], considered a more restricted model, that we refer to as the "traditional BSM" to disambiguate from the "streaming BSM". In particular, they consider a variant where a single long random string X is broadcast by a third party, and the honest users can store a small subset of n physical locations of X (chosen non-adaptively). The adversary, can store arbitrary information about X, as long as the amount of information is bounded by m bits. After this occurs, the adversary's storage becomes unbounded, and the honest parties can run some additional protocol, whose overall space and communication complexity is bounded by n. Protocols in the traditional BSM readily translate into the streaming BSM, by having one of the users stream X as the first message of the protocol.[1]

Compared to the streaming BSM, the traditional BSM can be seen as imposing additional restrictions on the honest parties Alice and Bob, and giving more power to the space-bounded attacker Eve, as follows:

(a) *Restricting Number of "Long" Rounds.* We make a distinction between "long" rounds, in which one of the parties streams a long message consisting of more than m bits of data, versus "short" rounds, consisting of fewer than m bits of data. Note that Eve can store the entire message in a short round. The traditional BSM allows only a single "long" round — the very first round of the protocol.

(b) *Uniformly Random "Long Rounds".* The traditional BSM requires that a "long" round should simply stream a uniformly random string X. When true, such X is called a *randomizer string* [Mau92], and can also come externally (e.g., from nature) rather than being sampled by the parties.

[1] This holds generically in the case of KA. In the case of OT, where the participants can be malicious, it may not be generically safe to allow one of the parties to chose X instead of having it sampled by a trusted third party. However, it was safe to do so for all the protocols in the literature.

(c) *Local Computability for Alice/Bob.* In the streaming BSM, when a party Alice streams a long string X to an honest party Bob, then Bob is allowed to arbitrarily process all of X in a streaming manner, as long as not using more than n bits of space. The traditional BSM demands a stricter property of n-*Local Computatibility* (LC) [Vad04]: The honest parties can only access at most n (a-prori non-adaptively chosen) *physical* locations of each string X sent during a "long" round. [2]

(d) *Unlimited Short-term Memory for Eve.* In the streaming BSM, the adversary Eve is streaming and only has m bits of memory throughout the execution of the protocol. In the traditional BSM, we only require that Eve stores at most m bits immediately after observing each "long" round, but we allow her to use unlimited short-term memory to process the round, and do not restrict her memory during "short" rounds.

Clearly, enforcing any of the restrictions (a)-(d) makes any upper bound stronger, and hence all protocols in the traditional BSM model also apply to the streaming BSM. Indeed, most previous constructions in the traditional BSM satisfied all of these additional properties. For example, the symmetric-key results of [ADR02, DM02, Lu02, Vad04] satisfied all of (a)-(d), as did the public-key results for key agreement and oblivious transfer of [CM97, CCM98, DHRS07]. However, there were exceptions, pointing to the fact that these restrictions were not all seen as crucial. For example, the work of [Din01] required two "long" rounds, and therefore did not satisfy (a). Moreover, if one wanted to use OT as a sub-protocol in general MPC, then this would require running many sequential copies, meaning that even if the OT protocol satisfied (a), the resulting MPC would not.

More recently, the ground-breaking work of Raz et al. [Raz16, KRT17, Raz17, GRT18] (presented in terms of time-space tradeoffs for learning parity), constructed elegant symmetric-key encryption schemes in the streaming BSM that crucially *do not* satisfy (b)-(d); see Section 1.4. The work of [GZ19], then lifted the techniques of Raz et al. [Raz16, KRT17, Raz17, GRT18] to build key agreement, oblivious transfer and bit commitment protocols in the streaming BSM, without satisfying (b)-(d). Nevertheless, the protocols of [GZ19] have some advantages over prior works in the traditional BSM, such as smaller number of communication rounds, and perfect correctness.

Overall, looking at the literature, it appears that many works implicitly viewed the streaming BSM as the real conceptual goal, but ended up satisfying additional properties (a)-(d) that they incorporated into their formal model. This view seems to be shared by the more recent works of [Raz16, KRT17, Raz17, GRT18, GZ19] that did not satisfy the additional properties, but still continued to refer to their model as the BSM, without carefully distinguishing between the variants. We continue in this vein, and view the streaming BSM as the main notion to strive for, while achieving the additional restrictions (a)-(d) can be seen as a nice bonus, but is not essential.

Moving to the lower bound of Dziembowski and Maurer [DM04], it turns out it critically used restriction (a), namely that there is only a single long round having large communication. Hence, to overcome the quadratic barrier imposed by [DM04], our protocols must use multiple long rounds.

Interestingly, we will be able to do so while still satisfying the additional restrictions (b)-(d). In particular, our protocols contains many long rounds, each of which involves generating a long uniformly random string X, while the honest parties store some small set of at most n physical locations of X. The adversary is only restricted to storing at most m bits of information about each X sent in a long round, but gets unlimited memory otherwise (i.e. during the short rounds and for computing the functions that compresses each X into m bits). However, we will mostly view these additional features as secondary, and focus most of our discussion on the fully unrestricted streaming BSM. If follow-up works manage to get further improvements by also dropping the restrictions (b)-(d), much like the works of [Raz16, KRT17, Raz17, GRT18, GZ19], this would be "fair game" and satisfy the main goal from our point of view.

To sum up, even though many prior works already departed from the traditional BSM and considered the streaming BSM as the main model, when it comes to public-key schemes, all prior works in the BSM were stuck at the quadratic gap between honest and adversarial storage. On the other hand, the quadratic lower bound of [DM04] does not extend to the streaming BSM, which opens the door for our results.

[2] For example, if local computability is demanded, parties cannot compute the parity of all the bits of X.

1.2 Our Results

As our main positive results, we design protocols for key agreement (KA), oblivious transfer (OT) and general multiparty computation (MPC) in the (n, m)-BSM, supporting up to an exponential gap between the honest user and adversary storage: $m = 2^{O(n)}$. This *qualitatively matches the positive results in the space-bounded symmetric-key setting*, albeit in (substantially) more rounds. In fact, we also show that large number of long rounds (and also overall large communication complexity) is essential when $m \gg n^2$, by non-trivially extending the lower bound of [DM04] to general BSM protocols. Details follow.

Key Agreement in BSM. Recall, the goal of a KA protocol is for Alice and Bob agree on a ℓ-bit key while talking over an authenticated-but-public channel. In Section 5, we show the following result in the (n, m)-BSM:

Theorem 1.1 (informal)**.** *For any m, λ, there exists some $n_{min} = O(\log m + \lambda)$ such that for all $n \geq n_{min}$ there is an unconditionally secure key agreement protocol in the (n, m)-BSM that outputs an $\Omega(n)$-bit key and achieves security $2^{-\Omega(\lambda)}$. Furthermore:*

- *The number of rounds is $\widetilde{O}(\lceil m/n^2 \rceil \cdot \lambda)$.*
- *The communication complexity is $\widetilde{O}(m\lceil m/n^2 \rceil \cdot \lambda)$.*

Note that, although the adversary's storage bound m can even be exponentially larger than n, this comes at the cost of increasing the number of rounds and bits of communication. If we want the overall protocol to be polynomially efficient, then we must restrict m to be some arbitrarily large polynomial.

Oblivious Transfer and Beyond. As our second main result, we build an OT-protocol in (n, m)-BSM, achieving nearly the same parameters as our KA protocol from Theorem 1.1. Recall, in an OT protocol, sender Alice has two ℓ-bit messages $(\mathsf{msg}_0, \mathsf{msg}_1)$, and receiver Bob has a single choice bit $c \in \{0, 1\}$. At the end of the protocol, Alice should learn nothing, while Bob should learn msg_c, and get no information about msg_{1-c}. When ported to (n, m)-BSM, (1) honest Alice and Bob should use space at most n, (2) the privacy of choice bit c should hold even against malicious Alice with storage m, and (3) the privacy of m_{1-c} should hold even against malicious Bob with storage m.

In our work we will achieve receiver privacy guarantee (2) even against *unbounded* space sender, so we only rely on the BSM for sender privacy (3). Moreover, our protocol satisfies *simulation-based* security, with an efficient simulator. This means that our simulator only uses the attacker as a black-box and is efficient relative to the corresponding attacker. In contrast, prior OT works in the BSM [CCM98, Din01, DHRS07, GZ19] all satisfied a weaker indistinguishability-based variant of sender-privacy, which roughly corresponds to inefficient simulation. The problem of having an efficient simulator was explicitly stated as an interesting and challenging open problem in [DHRS07]. Our result, formally proven in Section 6, is summarized below:

Theorem 1.2 (informal)**.** *For any m, λ, there exists some $n_{min} = O(\log m + \lambda)$ such that for all $n \geq n_{min}$ there is an unconditionally secure OT protocol with efficient simulator in the (n, m)-BSM with message size $\Omega(n)$ and security (and correctness) errors $2^{-\Omega(\lambda)}$. Furthermore:*

- *The number of rounds is $\widetilde{O}(\lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda))$.*
- *The communication complexity is $\widetilde{O}(m \cdot \lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda))$.*
- *Receiver security holds even against a malicious sender with unbounded space.*

To generalize our result to general MPC, recall that OT is information-theoretically complete for general MPC [Kil88, IPS08]. In Section 6.4, we observe that this result also extends to the (n, m)-BSM, provided we allow the honest parties' storage n, round complexity R, and communication complexity C to also polynomially-depend on the circuit size of the corresponding MPC functionality. Note that these parameters are completely independent of the adversary's storage bound m, which can still be arbitrarily larger.

Moreover, efficient simulation of our OT protocol is *critical* to achieve efficient simulation of the resulting MPC. For example, if we apply our MPC to the special case of the zero-knowledge (ZK) functionality, we get the first ZK protocol in (n, m)-BSM with an *efficient simulator* and arbitrary gap between m and n. In contrast, if we only had indistinguishability-based OT, we would get ZK with an inefficient simulator (which is equivalent to witness indistinguishability), which is insufficient/uninteresting in many situations when the witness is unique. Indeed the prior works of [SPY92, AF94] constructed (non-interactive) witness indistinguishable proofs in the BSM, and explicitly left zero-knowledge as an open problem.

Round and Communication Lower Bound. As we already mentioned, circumventing the lower bound [DM04] requires more than one long round. Also, any protocol in the (n, m)-BSM clearly requires more than m bits of communication. However, our protocols in Theorems 1.1 and 1.2 are noticeably less efficient: they use $\Omega(m/n^2)$ rounds and $\Omega(m^2/n^2)$ communication. This begs the question of whether large round and communication complexities of our protocols are inherent. In particular, when $m \gg n^2$, should the number of rounds R grow with m and should the communication C be super-linear in m?

Unfortunately, we show that the answer is affirmative (see Theorem 7.4). Specifically, we show that $R \geq \Omega((m/n^2)^{1/2})$ and $C \geq \Omega(m \cdot (m/n^2)^{1/2})$. While leaving a non-trivial gap with our upper bounds $R = \widetilde{O}(m/n^2)$ and $C = \widetilde{O}(m^2/n^2)$ when $m \gg n^2$, it still shows that the number of rounds grows with m, and the communication must be super-linear in m. It is an interesting open question to close this quantitative gap between our lower and upper bounds.

Our basic lower bound above only holds for BSM protocols where the attacker Eve is allowed unlimited short-term memory, and is only subject to keeping an m-bit state in between rounds (i.e., condition (d)). However, we also non-trivially extend our lower bound to show that it can even handle fully streaming adversaries that are restricted to m-bits of memory throughout the protocol execution, at the cost of a weaker quantitative bound: $R \geq \Omega((m/n^2)^{1/3})$, $C \geq \Omega(m \cdot (m/n^2)^{1/3})$. It is also an interesting open question to close the quantitative gap between this bound and the previous one.

1.3 Our Techniques

Bit-Entropy Lemma. As a crucial tool in our KA and OT constructions, we rely on a new technical lemma for min-entropy (Lemma 3.1). On a high level, the lemma says that if a long string $X \in \{0,1\}^k$ has high min-entropy (e.g., because it was chosen uniformly at random and the adversary could only remember $m \ll k$ bits of information about it), then many individual bits $X[i]$ of X must have non-trivial min-entropy. Specifically, if $\mathbf{H}_\infty(X) \geq \delta \cdot k$, we show that $\sum_{i \in [k]} \mathbf{H}_\infty(X[i]) \geq \rho \cdot k$, where we (optimally) relate ρ to δ. For example, when $\delta = \Omega(1)$, then $\rho = \Omega(1)$. The technical lemma relates to conceptually similar lemmas in [NZ96, Vad04, BKR16], showing that random subsets of bits in X have a high entropy rates, but to our knowledge, the single-bit version was not shown previously.

Key Agreement Protocol. The high-level idea for our KA protocol from Section 5.2 is surprisingly simple. The protocol consists of many rounds i, where Alice streams a $(k = 2m)$-bit random string X to Bob and remembers a single random location in the string $X[a]$. Similarly, as Bob receives the string X, he remembers a single random location $X[b]$. At the end of each round, Alice and Bob exchange their choice of locations a, b with each other; if $a = b$, they set $X[a] = X[b]$ as their shared key and terminate, else they erase all of their memory so far and go to the next round. Their storage only consists of a single index and is therefore $n = O(\log m)$. The probability of Alice and Bob agreeing in any round is $1/(2m)$ and therefore after $O(m)$ rounds they are likely to terminate. In the round i^* where they agree, the attacker can only remember m out of $2m$ bits of arbitrary information about the string X that was sent, and the choice of what information to remember is made before seeing Alice's and Bob's locations a, b. Therefore, the agreed upon location $X[a] = X[b]$ in that round has some constant amount of entropy from Eve's point of view.

The simple template above only outputs a 1-bit shared key, only guarantees that it has some low but non-trivial entropy from the point of view of the attacker (but does not guarantee that it is uniformly random), has a constant correctness error and and requires $O(m)$ rounds. However, it is easy to address

these deficiencies. First, Alice/Bob can store $O(n)$ random locations (not just 1), which means they improve their odds of agreement in a given round from $1/m$ to roughly $O(n^2/m)$, to get round complexity $O(m/n^2)$ and communication complexity $O(m^2/n^2)$, respectively. Second, we can amplify security (and correctness) to ensure that the agreed upon key is $2^{-\lambda}$-statistically close to uniform, while simultaneously making the key longer (say, λ bits), by repeating the above $O(\lambda)$ times, and applying a randomness extractor to the $O(\lambda)$ agreed upon bit locations. Finally, once the symmetric-key is $O(\lambda)$ bits long, we amplify it to be $\Omega(n)$ bits, by adding an additional round, and using any of the optimal symmetric-key BSM protocols (e.g., [Vad04]).

We also notice that, while our protocol takes many rounds (which we show to be inherent and therefore does not satisfy restriction (a), it does satisfy the additional restrictions (b)-(d): each long string is truly random, Alice and Bob are "locally computatable", and security holds even if Eve has an unrestricted amount of short-term local memory, as long as she can only remember at most m bits of information after seeing each string X.

Oblivious Transfer Protocol. In an OT protocol, sender Alice has two messages $(\mathsf{msg}_0, \mathsf{msg}_1)$, and receiver Bob has a single choice bit $c \in \{0,1\}$. At the end of the protocol, Alice should learn nothing, while Bob learns msg_c, and gets no information about msg_{1-c}.

Our oblivious transfer crucially relies on a tool called *interactive hashing* [NOVY93, DHRS07]. This tool was also used to construct OT in the BSM by prior works [CCM98, Din01, DHRS07] achieving a quadratic gap between the honest and adversarial storage. However, our protocol uses it in a substantially different way. In an interactive hashing protocol, a sender Bob has a random input $b \in [k]$, and at the end of the protocol, Alice can narrow down Bob's input to one of two possible choices b_0, b_1 such that $b \in \{b_0, b_1\}$, but Alice does not learn which of them it is; both options are equally likely. On the other hand, Bob cannot simultaneously control both of the values b_0, b_1 that Alice ends up with, and in particular he cannot cause both of them to land in some sparse subset $B \subseteq [k]$. Such interactive hashing protocols can be performed with 4 rounds of interactions and polylog(k) time/space. The security properties hold information-theoretically, even if the parties have unbounded computation and memory.

We now describe a simplified version of our OT protocol, which roughly corresponds to the case where honest users have $n = O(\log m)$ storage. We first rely on a component sub-protocol, which one can think of as an (imperfect) form of Rabin OT [Rab81]: Alice outputs some bit r, and Bob either also outputs r or $\perp$, but Alice does not learn which of these occurred. We set the length of "long rounds" to $k = O(m \log(m))$:

- Alice and Bob choose random indices $a, b \leftarrow [k]$ respectively. Alice samples a random string $X \leftarrow \{0,1\}^k$ and sends it to Bob. Alice stores $X[a]$ and Bob stores $X[b]$.
- Alice and Bob run interactive hashing where Bob uses his index b. Alice learns that it is one of b_0, b_1.
- Alice checks if $a \in \{b_0, b_1\}$, and if not, then the parties go back to the beginning and try again. Else Alice sends a to Bob and outputs $r = X[a]$. Bob checks if $a = b$ and if so he outputs $r = X[b]$ else he outputs $\perp$.

The interactive hashing security ensures that even if Alice is malicious, she does not learn whether Bob outputs $\perp$ or r. On the other hand, even if Bob is malicious and has storage m, there is only a small $O(k/\log k)$ set of bad indices $B \subseteq [k]$ that he "knows" (have very small entropy given his state). The interactive hashing ensures that it's unlikely that both b_0, b_1 are in B, and Alice selects one of them at random (the one that matches her a). Therefore, in the execution where Alice accepts, with probability $\approx 1/2$, Alice's index satisfies $a \notin B$ and therefore Bob does not know $r = X[a]$.

To go from the above sub-protocol to full OT, we employ a variant of the trick of [Cré88] to go from Rabin OT to the more standard 1-out-of-2 OT. The parties run the above sub-protocol for $t = 3\lambda$ iterations, where Alice outputs bits $(r_1, \ldots, r_t) \in \{0,1\}^t$ and Bob outputs $(r'_1, \ldots, r'_t) \in \{0,1,\perp\}^t$ such that $r'_i \in \{r_i, \perp\}$ and roughly $1/2$ of them are $\perp$, but Alice does not know which. Bob selects two disjoint subsets $I_0, I_1 \subseteq [t]$ of size λ each at random, subject to I_c only containing values i for which $r'_i \neq \perp$. Alice applies an extractor on the values r_{I_0}, r_{I_1} and uses the outputs to one-time-pad her messages $\mathsf{msg}_0, \mathsf{msg}_1$. This allows Bob to recover msg_c. It's easy to see that the sets I_0, I_1 look identically distributed to Alice and so she does not

learn Bob's choice bit c. On the other hand, since Bob only knows roughly $\frac{t}{2} = \frac{3\lambda}{2}$ of the values r_i, at least one of r_{I_0}, r_{I_1} must contain roughly $\frac{\lambda}{2}$ values that Bob does not know, and hence the corresponding extracted string will blind the message.

Note that in our scheme, security against an adversarial Bob (receiver) relies on him having bounded storage m, but security against an adversarial Alice (sender) does not impose any restrictions on her storage. The overall protocol requires $\widetilde{O}(m \cdot \lambda)$ rounds to terminate and $\widetilde{O}(m^2\lambda)$ communication. Our full protocol generalizes the above to settings where honest users have larger storage n to get $\widetilde{O}(\lceil m/n^2 \rceil \cdot \lambda)$ rounds and $\widetilde{O}(m \cdot \lceil m/n^2 \rceil \cdot \lambda)$ communication. This requires additional technical ideas to perform interactive hashing on sets of indices rather than just a single index; see Section 6.

One issue with the above idea, and indeed all prior constructions of OT in the BSM [CCM98, Din01, DHRS07, GZ19], is that it only satisfies a weak form of indistinguishability-based security, which is equivalent to security with an inefficient simulator. In particular, to simulate an adversarial Bob, we need to figure out his choice bit c, which requires figuring out which locations $X[a]$ he "knows" and which he does not. This can be done inefficiently (and non-black-box) by looking at Bob's state after processing X and figuring out the conditional entropy of each bit of X given the state; but there seems to be no hope to make this process efficient. We show how to overcome this via an efficient rewinding-based simulation strategy. The simulator forks off many copies of the interactive hashing protocol and figures out which indices show up as one of Alice's outputs with *high frequency*. We show that this serves as a good proxy for the indices that Bob knows – since he only knows $X[a]$ for very few locations a, he has to "play" such locations with high frequency if he wants to have a good chance of Alice selecting them. Therefore, by using the efficiently computable set of high-frequency indices as a proxy for the inefficiently computable set of indices that Bob knows, we can efficiently extract Bob's choice bit c.

Lower Bound. Let us first recall the main intuition of the DM lower bound [DM04]. Let $m > n$ be the storage size of the adversary, and suppose the first message of the protocol is some large message M, potentially of size $|M| \gg m$ much larger than the adversary's storage. In the real protocol, the honest parties Alice and Bob respectively compute states s_A and s_B after processing M. DM shows that there exists some compact information s_E^* of size m, which (1) is publicly-computable given M, and (2) *decorrelates* the states s_A and s_B of Alice and Bob in the following sense: conditioned on s_E^*, the users' states s_A and s_B only share a low amount of mutual information, bounded by n^2/m. Therefore, if $m = O(n^2)$ is sufficiently large, the information shared between Alice and Bob conditioned on the adversary's view becomes too small (much less than 1 bit) for them to agree on a shared random key.

One obstacle towards extending DM to the interactive setting is that, even if the mutual information created in each round is very small $O(n^2/m)$, with sufficiently many rounds it can add up. Indeed, this is exactly what our upper bound exploits, and why one can allow large gaps between m and n with a large numbers of rounds! For our lower bound on rounds and communication, we want to show that this is essentially the best that one can do. There are two main obstacles. Firstly, the DM approach only works if Alice and Bob do not share any mutual information in the first place. This is true at the beginning of the protocol, which results in a candidate adversarial strategy for the first round of the protocol. But it is not clear whether it extends to any intermediate round within the protocol execution, where Alice and Bob managed to already get some, albeit small, amount of mutual information.[3] Moreover, a naive attempt would be to have Eve compute an appropriate s_E^* for every round, but then Eve would need to store all of these values throughout the duration of the protocol, thus blowing up her storage.

Instead, we approach the DM core idea from a different angle, by thinking of it as a *round reduction* step that allows us to convert an R round protocol into an $R-1$ round protocol, with only a small loss in correctness and security. In particular, instead of having Alice send the long message M to Bob in the first round, we remove the first round entirely, and have Bob do the following: (1) sample M as Alice would, (2)

[3]Indeed, it is not true in general that their mutual information can only increase by a small amount in each round; once Alice and Bob share even a small amount of mutual information (e.g., they share a short extractor seed, perhaps even only with small probability), they may be able to leverage it to derive much more mutual information in just one additional round (e.g., send a long message and extract).

(inefficiently) sample s_E^* given M as the adversary in DM, and (3) sample his state s_B conditioned on s_E^*; (4) use it to compute the next message M', and (5) send (s_E^*, M') as the new message to Alice. Alice then: (6) samples s_A conditioned on s_E^*; and (7) processes M' using s_A, as she would have done originally. Note that Alice and Bob are now inefficient, with unlimited short-term memory to process each round, but only keep short n-bit states between rounds, similar to feature (d) of Eve.[4]

We claim that the round reduction step preserves correctness and security up to some small loss. This holds because the original states s_A, s_B had small mutual information conditioned on s_E^*, which implies that they are statistically close to independent. Therefore, the new way of sampling s_A, s_B truly independently conditioned on s_E^* only introduces a small statistical error. On the other hand, any attack Eve can perform on the new protocol by observing both of the values (s_E^*, M') sent by Bob at the same time, she could have also performed originally by computing s_E^* from Alice's original message M, storing s_E^* locally in her m-bit state (here we crucially rely on it being small), and then performing the same computation on the values (s_E^*, M'), once Bob sends M'.

By performing the round-reduction steps iteratively, we eventually get a 0-round key agreement protocol, which leads to a contradiction. However, each time we perform the round-reduction step we incur some statistical error $\sqrt{n^2/m}$. The square-root comes from using Pinsker's inequality to convert from mutual information to statistical distance. Therefore, we only end up with a secure protocol at the end, if the original protocol has $R = O(\sqrt{m/n^2})$ rounds, which gives our lower bound on rounds $R \geq \Omega(\sqrt{m/n^2})$. Note that this also gives a lower bound on communication C since $C \geq R$. However, we can get a stronger lower bound of $C \geq \Omega(m \cdot \sqrt{m/n^2})$ by showing how to remove "small" rounds (i.e., having communication smaller than m) for free, without any loss in correctness/security. We refer to Section 7.2 for more details.

As mentioned previously, this lower bound only rules out protocols secure against strong attackers Eve who have access to unbounded short-term memory to process each round, while storing m bits between rounds. We further adapt the techniques above to handle *fully streaming* adversaries, that are restricted to m bits of memory throughout the protocol. The main observation is that the only step in the round reduction procedure that requires Eve to have unbounded short-term memory is sampling s_E^* given M. We first observe that this step can be performed in a streaming manner using small local memory, as long as Alice and Bob are streaming algorithms with small local memory. However, even if the latter was the case in the initial protocol, once we start removing rounds, we required Alice and Bob to have large local memory to run Eve's attack. This turns into a recursive analysis, where the memory that Alice and Bob need to run the protocol after removing R rounds, depends on the memory Eve needs to attack on the protocol after removing $R-1$ rounds, which depends on the memory Alice and Bob need to run the protocol after removing $R-1$ rounds etc. By carefully analyzing this recursion, we show that Eve's short-term memory can be bounded to only be a factor of R larger than the previous bound we had on her long-term memory, which yields our new new quantitatively weaker bounds of $R \geq \Omega((m/n^2)^{1/3})$ and $C \geq \Omega(m \cdot (m/n^2)^{1/3})$ for the fully streaming model. We refer to Section 7.4 for more details.

1.4 Related Work

We already extensively mentioned the prior work on the symmetric-key BSM [Mau92, ADR02, DM02, Lu02, Vad04, Raz16, KRT17, Raz17, GRT18] and the public-key BSM models [CM97, CCM98, Din01, HCR02, DHRS07, GZ19]. In particular, the work of [Raz16, KRT17, Raz17, GRT18] constructed "reusable" n-bit-key symmetric-key encryption schemes, capable of encrypting exponentially many b-bit messages, where an individual ciphertext is "only" $O(mb/n)$ bits long.[5] When $b \ll n$, this is a huge saving compared to the prior symmetric-key schemes in the BSM, where each individual ciphertext had size greater than m, irrespective of message length. Interestingly, these works did not satisfy restrictions (b)-(d), critically using full features of the streaming BSM.

In the context of proof systems, [SPY92, AF94] constructed non-interactive witness indistinguishable proofs secure against memory-bounded streaming verifiers, allowing arbitrary gap between the values n and

[4] Note that allowing Alice and Bob to be stronger makes the resulting lower bound stronger as well.

[5] This is optimal, as otherwise Eve is capable of storing more than n/b ciphertexts in its memory, allowing the parties to encrypt more than $b \cdot n/b = n$ bits of information using an n-bit key, contradicting Shannon lower bound.

m. In contrast, the proofs systems constructed in this work are full zero-knowledge, with efficient simulation, but use many rounds of interaction. In a related vein, a very recent work of [GZ21] considered the notion of "disappearing cryptography" in the (streaming) BSM. Here, a component of the scheme (e.g., a ciphertext, signature, proof or program) is streamed bit by bit. The space-bounded receiver can get the functionality of the system once, after which the object "disappears" for subsequent use.

The work of [MST09] designed novel "timestamping" schemes in the (traditional) BSM. Here space-bounded sender and receiver have access to a long randomizer string X: the sender will timestamp a given document D at time t, and the receiver will prepare to verify D (which is yet unknown). The sender can then prove the timestamping of D to the receiver at a much later time, and the receiver is guaranteed that the sender is unable to timestamp a "very different" (i.e., high-entropy) document D'.

Finally, we mention the seminal works of [Nis90, NZ96] in the context of designing pseudorandom generators fooling space-bounded distinguishers. Unlike the BSM setting, the the memory n of the generator must be necessarily higher than the memory m of the the distinguisher, and the works of [Nis90, NZ96] come very close to this bound, unconditionally. In a similar vein, the work of [KRVZ11] constructs deterministic randomness extractors for space-bounded sources of randomness.

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 $i \in s$.

Statistical Distance. Let X, Y be random variables with supports S_X, S_Y, respectively. We define their *statistical difference* as

$$\mathbf{SD}(X,Y) = \frac{1}{2} \sum_{u \in S_X \cup S_Y} \left| \Pr[X = u] - \Pr[Y = u] \right|.$$

We write $X \approx_\varepsilon Y$ to denote $\mathbf{SD}(X,Y) \leq \varepsilon$.

Predictability and Entropy. The *predictability* of a random variable X is $\mathbf{Pred}(X) \stackrel{\mathsf{def}}{=} \max_x \Pr[X = x]$. The *min-entropy* of a random variable X is $\mathbf{H}_\infty(X) = -\log(\mathbf{Pred}(X))$. Following Dodis et al. [DORS08], we define the conditional predictability of X given Y as $\mathbf{Pred}(X|Y) \stackrel{\mathsf{def}}{=} \mathbb{E}_{y \leftarrow Y}\left[\mathbf{Pred}(X|Y=y)\right]$ and the (average) conditional min-entropy of X given Y as: $\mathbf{H}_\infty(X|Y) = -\log\left(\mathbf{Pred}(x|Y)\right)$. Note that $\mathbf{Pred}(X|Y)$ is the success probability of the optimal strategy for guessing X given Y.

Lemma 2.1 ([DORS08])**.** *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 2.2 ([DORS08])**.** *For any random variables X, Y,for every $\varepsilon > 0$ we have*

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

Lemma 2.3. *If X and Y are independent conditioned on Z then $\mathbf{H}_\infty(X|Y) \geq \mathbf{H}_\infty(X|Y,Z) \geq \mathbf{H}_\infty(X|Z)$.*

Lemma 2.4. *If X and Y are independent conditioned on Z then $\mathbf{H}_\infty(X,Y|Z) \geq \mathbf{H}_\infty(X|Z) + \mathbf{H}_\infty(Y|Z)$.*

Shannon Entropy. The Shannon entropy of a random variable X is $\mathbf{H}(X) \stackrel{\mathsf{def}}{=} \mathbf{E}_{x \leftarrow X}[-\log(\Pr[X = x])]$. The conditional Shannon entropy of X given Y is $\mathbf{H}(X|Y) \stackrel{\mathsf{def}}{=} \mathbf{E}_{y \leftarrow Y}\, \mathbf{H}(X|Y=y) = \mathbf{E}_{(x,y) \leftarrow (X,Y)}[-\log(\Pr[X = x|Y = y])]$.

For $0 \leq p \leq 1$ we define the binary entropy function $\mathbf{h}(p) \stackrel{\mathsf{def}}{=} \mathbf{H}(B_p)$, where B_p is a Bernoulli variable that outputs 1 with probability p and 0 with probability $1 - p$.

Lemma 2.5. *For any random variables X, Y, we have:* $\mathbf{H}_\infty(X|Y) \leq \mathbf{H}(X|Y)$.

Extractors. We review the notion of randomness extractors and known parameters.

Definition 2.6 ((Strong, Average-Case) Seeded Extractor [NZ96])**.** *We say that an efficient function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ *is an* (α, ε)-extractor *if for all random variables* (X, Z) *such that* X *is supported over* $\{0,1\}^n$ *and* $\mathbf{H}_\infty(X|Z) \geq \alpha$ *we have* $\mathbf{SD}((Z, S, \mathsf{Ext}(X; S)) , (Z, S, U_\ell)) \leq \varepsilon$ *where* S, U_ℓ *are uniformly random and independent bit-strings of length* d, ℓ *respectively.*

Theorem 2.7 ([ILL89])**.** *There exist an* (α, ε)*-extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ *as long as* $\alpha \geq \ell + 2\log(1/\varepsilon)$*. Furthermore, such an extractor can be computed in* $O(n)$ *time and space.*

Definition 2.8 (BSM Extractor [Vad04])**.** *We say that an efficient function* $\mathsf{BSMExt} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^\ell$ *is an* (n, m, ε)-BSM extractor *if:*

- *Given* $\mathsf{seed} \in \{0,1\}^d$ *initially stored in memory, it is possible to compute* $\mathsf{BSMExt}(x; \mathsf{seed})$ *given streaming access to* $x \in \{0,1\}^k$ *using at most* n *bits of total memory. Moreover, it can be done while only accessing at most* n *locations (chosen non-adaptively) in the string* x*.*
- BSMExt *is an* (α, ε)*-extractor (Definition 2.6) for* $\alpha = k - m$*.*

Note that a BSM Extractor gives a simple one-round protocol (n, m)-BSM protocol where Alice and Bob start with a uniformly random shared key key_0 of some small size d and derive a new shared key $\mathsf{key}_1 \in \{0,1\}^\ell$ of a larger size $\ell > d$. Alice just streams a random $x \in \{0,1\}^k$ to Bob and both parties compute $\mathsf{key}_1 = \mathsf{BSMExt}(x; \mathsf{key}_0)$. Security holds since the adversary can only store m-bits of information about x so it has $\alpha \geq k - m$ bits of entropy conditioned on the adversary's view, and key_0 acts as a random seed which is a-priori unknown to the adversary. Therefore $\mathsf{key}_1 = \mathsf{BSMExt}(x; \mathsf{key}_0)$ is ε-close to uniform given the adversary's view of the protocol.

Theorem 2.9 ([Vad04])**.** *For any* $m \geq \ell, \lambda$*, there is a* (n, m, ε)-BSM extractor $\mathsf{BSMExt} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^\ell$ *with* $n = O(\ell + \lambda + \log m)$*,* $\varepsilon = 2^{-\Omega(\lambda)}$*,* $k = O(m + \lambda \log(\lambda))$*,* $d = O(\log m + \lambda)$*.*

KL Divergence and Mutual Information. We recall the notions of KL divergence and (conditional) mutual information.

Definition 2.10 (KL Divergence)**.** *For two discrete probability distributions* P*,* Q *with the same support* $\mathrm{Supp}(P)$*, we define the KL divergence as*

$$D_{KL}(P\|Q) = \sum_{x \in \mathrm{Supp}(P)} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

Definition 2.11 ((Conditional) Mutual Information)**.** *If* (X, Y, Z) *denotes a triple of (potentially correlated) random variables with joint distribution* $P_{(X,Y,Z)}$*, we define the mutual information of* X *and* Y *conditioned on* Z *as:*

$$\mathbf{I}(X; Y|Z) = \mathop{\mathbf{E}}_{Z} D_{KL}(P_{(X,Y)|Z} \| P_{X|Z} \times P_{Y|Z}),$$

where $P_{(X,Y)|Z}$ *denotes the conditional distribution of* (X, Y) *given* Z*, and* $P_{X|Z}$ *and* $P_{Y|Z}$ *respectively denote the marginal distributions of* X *and* Y *given* Z*.*

Lemma 2.12 (Pinsker's Inequality)**.** *Let* X*,* Y *be random variables with distributions* P_X, P_Y *with the same support* $\mathrm{Supp}(X)$*. We have:*

$$\mathbf{SD}(X, Y) \leq \sqrt{\frac{1}{2} D_{KL}(P_X \| P_Y)}.$$

We will use the following lemma:

Lemma 2.13. *Let n, k be integers. Let $A \subseteq [k]$ be a uniformly random set such that $|A| = n$. Then for any fixed $B \subseteq [k]$ with $|B| = n$, we have:*

$$\Pr_A [A \cap B \neq \emptyset] \geq \Omega\left(\min\left(\frac{n^2}{k}, 1\right)\right).$$

Proof. Suppose $k > n^2$. The probability over A that $A \cap B = \emptyset$ is at most $\left(\frac{k-n}{k}\right)^n = 1 - O(n^2/k)$. Therefore the probability that $A \cap B \neq \emptyset$ is at least $\Omega(n^2/k)$. □

3 Bit-Entropy Lemma

We prove a new lemma showing that if X has sufficiently high min-entropy, then many individual bits $X[i]$ have sufficiently high min-entropy as well.

For $q \in [0, 1]$, we define $\mathbf{h}_+^{-1}(q)$ to be the unique value p such that $.5 \leq p \leq 1$ and $\mathbf{h}(p) = q$, where $\mathbf{h}$ is the binary entropy function.

Lemma 3.1. *Assume X, Y are random variables, where X is distributed over $\{0,1\}^k$. Let $X[i]$ denote the i'th bit of X. If $\mathbf{H}_\infty(X|Y) \geq \delta k$ the following 3 statements hold:*

1. $\sum_i \mathbf{Pred}(X[i] \mid Y) \leq \mathbf{h}_+^{-1}(\delta)k$.
2. $\sum_i \mathbf{H}_\infty(X[i] \mid Y) \geq -\log(\mathbf{h}_+^{-1}(\delta))k$.
3. *If I is uniformly random over $[k]$ and independent of X, Y then* $\mathbf{H}_\infty(X[I] \mid Y, I) \geq -\log(\mathbf{h}_+^{-1}(\delta))$.

Proof. We have:

$$\delta k \leq \mathbf{H}_\infty(X|Y) \leq \mathbf{H}(X|Y) = \sum_{i \in [k]} \mathbf{H}(X[i] \mid X[1], ..., X[i-1], Y) \leq \sum_{i \in [k]} \mathbf{H}(X[i] \mid Y).$$

Therefore

$$\delta \leq \mathop{\mathbf{E}}_{i \leftarrow [k], Y \leftarrow y} \mathbf{H}(X[i] \mid Y = y).$$

Since $\mathbf{h}_+^{-1}$ is a decreasing and concave function, this means:

$$\begin{aligned}
\mathbf{h}_+^{-1}(\delta) &\geq \mathbf{h}_+^{-1}\left(\mathop{\mathbf{E}}_{i \leftarrow \{0,1\}^k, y \leftarrow Y} \mathbf{H}(X[i] \mid Y = y)\right) \\
&\geq \mathop{\mathbf{E}}_{i \leftarrow [k], y \leftarrow Y} \mathbf{h}_+^{-1}(\mathbf{H}(X[i] \mid Y = y)) \\
&\geq \mathop{\mathbf{E}}_{i \leftarrow [k], y \leftarrow Y} (\max_{b \in \{0,1\}} \Pr[X[i] = b \mid Y = y]) \\
&\geq \mathop{\mathbf{E}}_{i \leftarrow [k]} \mathbf{Pred}(X[i]|Y).
\end{aligned}$$

This proves the first part of the theorem. Also the third part of the theorem follows since $\mathbf{H}_\infty(X[I] \mid Y, I) = -\log(\mathbf{E}_{i \leftarrow I} \mathbf{Pred}(X[I] \mid Y, I = i)) = -\log(\mathbf{E}_{i \leftarrow [k]} \mathbf{Pred}(X[i] \mid Y))$. The second part follows since $(-\log)$ is a decreasing and convex function so

$$\begin{aligned}
-\log(\mathbf{h}_+^{-1}(\delta)) &\leq -\log\left(\mathop{\mathbf{E}}_{i \leftarrow [k]} \mathbf{Pred}(X[i] \mid Y)\right) \\
&\leq \mathop{\mathbf{E}}_{i \leftarrow [k]} -\log(\mathbf{Pred}(X[i] \mid Y)) \\
&\leq \mathop{\mathbf{E}}_{i \leftarrow [k]} \mathbf{H}_\infty(X[i] \mid Y)
\end{aligned}$$

□

Remark 3.2. To the best of our knowledge, the "bit-prediction" lemma above is new, as it talks about individual bit prediction; as opposed to "subkey-predcition" lemma studied in prior BSM literature [NZ96, Vad04, BKR16], which talked about simultaneously predicting a large subset of bits. We also remark that our parameters are tight, as can be seen by taking X to be the uniform distribution over a hamming ball of radius pk, where $p = 1 - \mathbf{h}_+^{-1}(\delta) \leq 1/2$. The volume of this ball is roughly $2^{\mathbf{h}(p)k} = 2^{\delta k}$, so $\mathbf{H}_\infty(X) = \delta k$. Yet, each bit of X can be predicted with probability at least $1 - p = \mathbf{h}_+^{-1}(\delta)$.

Lemma 3.3. *For any $0 < \varepsilon \leq 1$ there is a $\delta = \Omega(\varepsilon^2)$ such that $-\log(\mathbf{h}_+^{-1}(1-\delta)) = (1-\varepsilon)$.*

Proof. Given ε we can solve:

$$\begin{aligned} & -\log(\mathbf{h}_+^{-1}(1-\delta)) = 1-\varepsilon \\ \Rightarrow \quad & \mathbf{h}_+^{-1}(1-\delta) = 2^{\varepsilon}/2 \\ \Rightarrow \quad & 1-\delta = \mathbf{h}(2^{\varepsilon}/2) = \mathbf{h}(1/2 + \Theta(\varepsilon)) = 1 - \Theta(\varepsilon^2). \end{aligned}$$

where we rely on the bound $2^{\varepsilon} = (1 + \Theta(\varepsilon))$ and $\mathbf{h}(1/2 + \Theta(\varepsilon)) = 1 - \Theta(\varepsilon^2)$ (e.g. [Cal09, Theorem 2.2]). □

4 Bounded Storage Model

A (n, m)-*bounded storage model (BSM)* protocol, is parametrized by a bound n on the memory of the honest parties, and a bound $m > n$ on the memory of the adversary. Communication between parties occurs in rounds where one party sends data to another party. Honest parties send and receive data in a *streaming* manner, by generating/reading the stream one bit at a time, while only using n bits of memory overall. The adversary, is also a streaming algorithm with m bits of memory.

For all our constructions, we will satisfy additional properties, corresponding to properties (b)-(d) discussed in the introduction. The protocol consists of two types of rounds: "short rounds" are of size is $< n$, and can be fully generated, sent, and processed by the honest parties using only n bits bits of memory, without needing to be streamed one bit at a time, while "long rounds" are of size $> m$.[6] Our protocols satisfy the following additional properties:

- *Uniformly Random "Long Rounds".* Each long round consists of a uniformly random string x generated by some party A and sent to party B.
- *Local Computability for Honest Parties.* In each long round, the honest parties only read a small set of $< n$ locations of x and use these to update their state, while using only n bits of memory in total. Furthermore, the set of locations accessed is chosen non-adaptively at the beginning of the round, before seeing any bits of x.
- *Unlimited Short-term Memory for Adversary.* The adversary can generate/read the entire long round of communication at once, and can use unlimited amounts of short-term memory during this process, but can only store a compressed m-bit state immediately after the end of each long round. There are no restrictions on the adversary's memory during/after short rounds.

5 Key Agreement

5.1 Definition

A key agreement protocol in the (n, m)-BSM with security ε is a protocol between two honest users Alice and Bob with memory bound n. At the end of the protocol Alice and Bob outputs values $\mathsf{key}_A, \mathsf{key}_B \in \{0,1\}^\ell$

[6] We will allow ourselves to split up the protocol into rounds arbitrarily, and may have two (or more) adjacent rounds where the same party A talks to party B.

respectively. For correctness, we require that when the protocol is executed honestly then $\Pr[\mathsf{key}_A = \mathsf{key}_B] = 1$. For security, we consider a passive BSM adversary Eve with memory bound m. Let view_{Eve} denote Eve's final state at the end of the protocol execution. We require that

$$(\mathsf{view}_{Eve}, \mathsf{key}_A) \approx_\varepsilon (\mathsf{view}_{Eve}, \mathsf{key}^*)$$

where $\mathsf{key}^* \leftarrow \{0,1\}^\ell$ is chosen uniformly at random and independently of the protocol execution.

5.2 Construction

Theorem 5.1. *For any $m \geq \ell, \lambda$ there is some $n_{min} = O(\lambda + \ell + \log m)$ such that for all $n > n_{min}$ there is a key agreement protocol in the (n, m)-BSM that outputs an ℓ-bit key and has security $\varepsilon = 2^{-\Omega(\lambda)}$. The round complexity of the protocol is $O(\lceil (m/n^2) \rceil \cdot \lambda \cdot \mathrm{polylog}(m))$ and the communication complexity $O(m \cdot \lceil (m/n^2) \rceil \cdot \lambda \cdot \mathrm{polylog}(m + \lambda))$.*

Proof. We first give the key agreement protocol between Alice and Bob, then discuss security, and then show how to set parameters to get the claimed efficiency.

Construction. Given m, λ, ℓ we define additional parameters as follows.

- Let $k = 2m$.
- Let $d_1 = O(\lambda + \log m)$ and $n_1 = O(\lambda + \log m + \ell)$ and $k' = O(m + \lambda \log(\lambda))$ be some values such that there is a $(n_1, m, \varepsilon = 2^{-\Omega(\lambda)})$-*BSM extractor* $\mathsf{BSMExt} : \{0,1\}^{k'} \times \{0,1\}^{d_1} \to \{0,1\}^\ell$ per Theorem 2.9.
- Let $t = O(\lambda + \log m)$ and $d_0 = O(t), n_0 = O(t)$ be some value such that there is a $(t/10, \varepsilon = 2^{-\Omega(\lambda)})$-extractor $\mathsf{Ext} : \{0,1\}^t \times \{0,1\}^{d_0} \to \{0,1\}^{d_1}$ that can be computed using n_0 space per Theorem 2.7.
- Define $n_{min} = \max(n_0, n_1, 2t + \lceil \log k \rceil + 1) = O(\lambda + \log m + \ell)$.
- For any $n \geq n_{min}$, define $\tilde{n} = \lfloor (n - t)/(\lceil \log k \rceil + 1) \rfloor = \Omega(n / \log m)$.

The protocol works as follows.

1. Set $i := 0$. Repeat the following until $i = t$:
   (a) Alice and Bob select uniformly random subsets $s_A, s_B \subseteq [k]$ of size $|s_A| = |s_B| = \tilde{n}$ respectively.
   Alice streams a uniformly random string $x \leftarrow \{0,1\}^k$ to Bob.
   Alice stores $x[s_A]$ while Bob stores $x[s_B]$.
   (b) Bob sends s_B to Alice.
   (c) If $s_A \cap s_B \neq \emptyset$ then Alice selects a random index $j \leftarrow s_A \cap s_B$ and sends j to Bob.
   Both Alice and Bob set $r_i = x[j]$ and increment $i := i + 1$.
   Else if $s_A \cap s_B = \emptyset$ then Alice simply sends $j = \perp$ to Bob.
2. Alice and Bob set $r := (r_1, \ldots, r_t)$.
   Alice sends a random $\mathsf{seed}_0 \leftarrow \{0,1\}^{d_0}$ to Bob and both of them compute $\mathsf{seed}_1 = \mathsf{Ext}(r; \mathsf{seed}_0)$.
3. Alice streams a uniformly random string $x \leftarrow \{0,1\}^{k'}$ to Bob and both parties compute $\mathsf{key} = \mathsf{BSMExt}(x; \mathsf{seed}_1)$.

We refer to each execution of the sub-protocol in steps 1.(a) - 1.(c) as an *epoch*. We say that an epoch is *successful* if Alice does not send $j = \perp$ in step 1.(c). Note that step 1.(a) and step 3 are "long rounds" and all other steps are "short rounds".

Honest Party Memory. Note that Alice and Bob only need $\widetilde{n}(\lceil \log k \rceil + 1)$ memory to execute each epoch, and need to store up to t bits from previous epochs for a total of $\widetilde{n}(\lceil \log k \rceil + 1) + t \leq n$ memory to execute part 1 of the protocol. They need $n_0 \leq n$ memory to execute part 2 and $n_1 \leq n$ memory to execute part 3. Therefore, in total, the protocol can be executed using at most n memory as required.

Security. For $i \in [k]$, let us define random variables corresponding to the values in the i'th successful epoch:

- X^i: the value of $x \in \{0,1\}^k$ sent by Alice during that epoch,
- Y^i: the state of the adversary immediately after processing x but before Bob sends s_B,
- J^i: the index $j \in [k]$ sent by Alice,
- $\widehat{Y}^i$: the state of the adversary at the end of the epoch after Alice sends j,
- $R^i = X^i[J^i]$ the value r_i stored by Alice and Bob.

Firstly, we see that:

$$\mathbf{H}_\infty(X^i | \widehat{Y}^{i-1}, Y^i) \geq \mathbf{H}_\infty(X^i | \widehat{Y}^{i-1}) - m \geq \mathbf{H}_\infty(X^i) - m \geq k - m = k/2$$

where the first inequality follows by Lemma 2.1 and the fact that $Y^i \in \{0,1\}^m$, the second inequality follows since $X^i, \widehat{Y}^{i-1}$ are independent, and the third since X^i is uniformly random over $\{0,1\}^k$. Furthermore J^i is uniformly random over $[k]$ and independent of $\widehat{Y}^{i-1}, X^i, Y^i$; it only depends on the sets s_A, s_B selected by Alice and Bob during the epoch and, by the symmetry of the protocol, every index j is equally likely to be selected. Therefore, by Lemma 3.1, we have

$$\mathbf{H}_\infty(R^i = X^i[J^i] | \widehat{Y}^{i-1}, Y^i, J^i) \geq \rho,$$

where $\rho = -\log(\mathbf{h}_+^{-1}(1/2) \geq .1$.

The variable $\widehat{Y}^t$ denotes the state of the adversary at the end of the final epoch. We have:

$$\begin{aligned}
\mathbf{H}_\infty(R^1, \ldots, R^t | \widehat{Y}^t) &\geq \mathbf{H}_\infty(R^1, \ldots, R^t | (Y^1, J^1), \ldots, (Y^t, J^t)) && (1)\\
&\geq \sum_{i \in [t]} \mathbf{H}_\infty(R^i | (Y^1, J^1), \ldots, (Y^t, J^t)) && (2)\\
&\geq \sum_{i \in [t]} \mathbf{H}_\infty(R^i | \bar{Y}^{i-1}, Y^i, J^i) && (3)\\
&\geq t \cdot \rho \geq .1\ t, && (4)
\end{aligned}$$

where:

- (1) follows from Lemma 2.3 since $(R^1, \ldots, R^t)$ and $\widehat{Y}^t$ are independent conditioned on $((Y^1, J^1), \ldots, (Y^t, J^t))$.
- (2) follows from Lemma 2.4 since the random variables R^i are independent conditioned on $((Y^1, J^1), \ldots, (Y^t, J^t))$.
- (3) follows from Lemma 2.3 since R^i and $((Y^1, J^1), \ldots, (Y^t, J^t))$ are independent conditioned on $(\bar{Y}^{i-1}, Y^i, J^i)$.

Therefore, by extractor security, we have $\mathsf{seed}_1 = \mathsf{Ext}(r = (r_1, \ldots, r_t); \mathsf{seed}_0)$ is $2^{-\Omega(\lambda)}$ statistically close to uniform, given the adversary's state at the end of part 2 of the protocol execution (this state is fully determined by the values $\widehat{Y}^t, \mathsf{seed}_0$). This allows us to switch to a hybrid world where we choose seed_1 uniformly at random and independently of the view of the adversary after part 2 of the protocol, and this hybrid is $2^{-\Omega(\lambda)}$ indistinugishable from real.

Now, we can rely on the security of BSMExt to argue that $\mathsf{key} = \mathsf{BSMExt}(x; \mathsf{seed}_1)$ is $2^{-\Omega(\lambda)}$ statistically close to uniform even given the adversary's view of the entire protocol. Note that, in the hybrid world, seed_1 is uniformly random and independent of the adversary's view, and the adversary only gets m bits of leakage on the value x, which is chosen uniformly at random. This proves security as needed.

Round and Communication Complexity. To analyze the round and communication efficiency, notice that each epoch is successful if $s_A \cap s_B \neq \emptyset$, which occurs with probability $p = \Omega(\min(\widetilde{n}^2/k, 1)) = \Omega(\min(\widetilde{n}^2/m, 1))$ by Lemma 2.13, and the probabilities are independent for each epoch. Therefore, in $2t/p$ epochs, we expect $2t$ successes, and by the Chernoff bound, the probability of having fewer than t successes is at most $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$. Therefore, with all but $2^{-\Omega(\lambda)}$ probability, in part 1 of the protocol, the number of rounds is

$$r = O(t/p) = O(t/\min(\widetilde{n}^2/m, 1)) = O(\lceil (m/n^2) \rceil \cdot \lambda \cdot \text{polylog}(m),$$

and the communication complexity is then $O(m \cdot r)$. Parts 2 and 3 of the protocol together add only 2 rounds and $d_0 + k' = O(m + \lambda \log(\lambda))$ bits of of communication, which gives the bounds of the theorem.

Note that we can modify the protocol to get worst-case round and communication complexities with the above bounds by having Alice and Bob always stop after at most $2t/p$ epochs, and if they did not get t successful ones, they simply output an all 0 key. Since this bad event only occurs with $2^{-\Omega(\lambda)}$ probability, security is preserved.

□

6 Oblivious Transfer and Multiparty Computation

6.1 Definition of Oblivious Transfer

We define oblivious transfer (OT) in the BSM via a real/ideal framework. In the ideal model the sender (Alice) gives two messages $(\mathsf{msg}_0, \mathsf{msg}_1) \in (\{0,1\}^\ell)^2$ to an ideal functionality $\mathcal{F}_{OT}$ and the receiver (Bob) gives a bit $c \in \{0,1\}$. The ideal functionality $\mathcal{F}_{OT}$ gives msg_c to the receiver and gives nothing to the sender.

A protocol Π realizes $\mathcal{F}_{OT}$ in the (n,m)-BSM with security ε if:

- Π can be executed by honest parties with n-bit memory.
- There exists an efficient black-box simulator $\mathsf{Sim}^{\mathcal{A}}$ that runs in time $\text{poly}(n, m, \lambda = \log(1/\varepsilon))$ with black-box (rewinding) access to the adversary $\mathcal{A}$, such that for any (inefficient) BSM-adversary $\mathcal{A}$ with m-bit state corrupting either the sender or the receiver and for any choice of inputs $\mathcal{Z} = (\mathsf{msg}_0, \mathsf{msg}_1, c)$ from the environment, we have

  $$\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}} \approx_\varepsilon \mathsf{IDEAL}_{\mathsf{Sim}^{\mathcal{A}},\mathcal{F}_{OT},\mathcal{Z}}$$

  where we define the distributions:

  $\mathsf{REAL}_{\mathcal{A},\Pi,\mathcal{Z}}$: denotes the real execution of Π with the adversary $\mathcal{A}$ taking on the role of either the sender or the receiver while the honest party uses the input specified by $\mathcal{Z}$; the output of the distribution consists of the output of $\mathcal{A}$ together with the inputs/outputs of the honest party.

  $\mathsf{IDEAL}_{\mathsf{Sim}^{\mathcal{A}},\mathcal{F}_{OT},\mathcal{Z}}$: denotes the ideal execution of $\mathcal{F}_{OT}$ with an ideal-adversary $\mathsf{Sim}^{\mathcal{A}}$ taking on the same role as $\mathcal{A}$, while the honest party uses the input specified by $\mathcal{Z}$; the output of the distribution consists of the output of $\mathsf{Sim}^{\mathcal{A}}$ together with the inputs/outputs of the honest party.

We further say that the protocol is secure against an *unbounded-memory* sender (resp. receiver) if we can drop the requirement on the storage of $\mathcal{A}$ when it corrupts the sender (resp. receiver).

We say that the protocol is only secure with *inefficient simulation*, if we drop the requirement on the efficiency of the simulator. Our default notion will be *efficient simulation*.

On Efficient Simulation. Note that we require efficient simulation even though the adversary may be computationally unbounded. This may seem strange at first, but is natural and is analogous to (e.g.,) requiring an efficient simulator for statistical Zero Knowledge proofs [GMR89] or for information-theoretically secure MPC protocols. In particular, the definition is agnostic to whether or not the adversary is efficient, but ensures that the adversary cannot learn anything in the real world that it could not also learn with only polynomially *more* computational power in the ideal world. The need for an efficient simulator is crucial when leveraging OT to construct other more complex functionalities, as we will do in Section 6.4. For example, we can use our OT in the BSM to construct zero-knowledge (ZK) proofs in the BSM. If the OT simulator were inefficient, the resulting ZK proof would only be inefficiently simulatable (equivalently, would only be witness indistinguishable), which is completely meaningless in many scenarios where the witness is unique; the prover may as well just send the witness in the clear.

On the other hand, our simulator does not have bounded storage and can use more memory than the adversary. This naturally corresponds to the idea that having some a-priori (polynomial) bound on storage is only assumed to be a limitation in the real world, and is a useful imitation in helping us build secure protocols, but is not a fundamental restriction that we need to also preserve for the ideal-world adversary interacting with the ideal functionality.

6.2 Interactive Hashing

Basic Interactive Hashing. In an interactive hashing protocol a sender Bob has an input $u \in [k]$. The goal of the protocol is for Alice to narrow down Bob's input to one of two possible choices u_0, u_1 such that $u = u_b$ for one of $b = 0$ or $b = 1$, but Alice does not learn which. In particular, even if Alice acts maliciously, when Bob chooses his input $u \leftarrow [k]$ at random, then both choices of b appear equally likely from Alice's point of view. On the other hand, although Bob can choose an arbitrary input u and ensures $u = u_b$ for some $b \in \{0, 1\}$, he cannot control the "other" value u_{1-b} too much. In particular, even if Bob is malicious during the protocol, for any sufficiently sparse subset $B \subseteq [k]$, it is highly unlikely that *both* of u_0, u_1 are contained in B.

Definition 6.1. *An* interactive hashing protocol *is a protocol between a public-coin randomized Alice (receiver) and a deterministic Bob (sender). Alice has no input and Bob has some input $u \in [k]$. At the end of the protocol, we denote the transcript (h, v), consisting of all the random messages h sent by Alice and all the responses v sent by Bob. We can think of h as defining a hash function that maps Bob's input u to his set of responses $v = h(u)$. The protocol has the following properties:*

- *2-to-1 Hash: Every possible choice of Alice's messages results in a hash function h which is 2-to-1, meaning that for every v in the image of h has exactly two pre-images: $|h^{-1}(v)| = 2$.*
- *(α, β)-Security: For any set $B \subseteq [k]$ of size $|B| \leq \beta \cdot k$, if Alice follows the protocol honestly and Bob acts arbitrarily resulting in some transcript (h, v) such that $\{u_0, u_1\} = h^{-1}(v)$ then $\Pr[\{u_0, u_1\} \subseteq B] \leq \alpha$.*

Note: the 2-to-1 hash property ensures that, if Bob chooses $u \leftarrow [k]$ uniformly at random and acts honestly during the protocol, then even if Alice acts maliciously resulting in some transcript (h, v) at the end of the protocol, if we define $\{u_0, u_1\} = h^{-1}(v)$ such that $u = u_b$, Alice cannot distinguish between b and $1 - b$.

Theorem 6.2 ([NOVY93, DHRS07])**.** *There is an 4-round interactive hashing protocol with (α, β)-security for any $\beta < 1$ with $\alpha = O(\beta \log k)$. Furthermore, the execution of the protocol and the computation of h^{-1} can be done in* polylog$k$ *time and space.*

Definition of Set Interactive Hashing. Here, we extend the notion of interactive hashing to the case where the sender Bob has an entire set of inputs $s_B \subseteq [k]$. Alice has her own set of inputs $s_A \subseteq [k]$. The goal of the protocol is to ensure that when there is a value in the intersection $s_A \cap s_B$ then there is a good chance that Alice will accept and output some value $u \in S_A$, in which case it then holds with probability 1/2, that $u \in S_B$ and Bob accepts and outputs it, while with probability 1/2 Bob rejects. Alice should not learn which

of these two cases occur, even if she acts maliciously. On the other hand, even if Bob is malicious, he cannot have too much control over the value that Alice outputs: for any sufficiently sparse set $B \subseteq [k]$, he cannot ensures that the value u that Alice outputs (conditioned on her accepting) is in the set B with probability much higher than $1/2$.

Definition 6.3. *In a set interactive hashing protocol, Alice and Bob have sets $s_A, s_B \subseteq [k]$ of size $|s_A| = |s_B| = n$. At the end of the protocol, Alice either rejects by sending a special $\perp$ message to Bob, or she accepts and sends some $u \in S_A$ to Bob. If Alice sends $\perp$, then Bob always rejects and outputs $\perp$. Otherwise, Bob can either accept, in which case he outputs the same u as Alice and it must hold that $u \in S_B$, or he rejects. The protocol has (α, β)-security if it satisfies the following properties:*

- *Correctness: If Alice and Bob both execute the protocol honestly using random subsets $s_A, s_B \subseteq [k]$ of size $|s_A| = |s_B| = n$ then:*

$$\Pr[\textit{Alice accepts}] \geq \Omega(\min(n^2/k, 1)) \quad , \quad \Pr[\textit{Bob accepts} \mid \textit{Alice accepts}] = \frac{1}{2}.$$

  Furthermore whenever Alice accepts with some value u, then it must be the case that $u \in S_A$ and if Bob also accepts then it must be the case that $u \in S_A \cap S_B$.

- *Security for Honest Bob: If Bob follows the protocol honestly using a random subset $s_B \subseteq [k]$ of size $|s_B| = n$ and Alice follows the protocol arbitrarily, then, even condition on any arbitrary protocol transcript in which Alice accepts (i.e., does not send $\perp$ to Bob as the last message) we have:*

$$\Pr[\textit{Bob accepts}] = \frac{1}{2}.$$

- *(α, β)-Security for honest Alice: Let $B \subseteq [k]$ be a set of size $|B| \leq \beta \cdot k$. If Alice follows the protocol honestly using a random subset $s_A \subseteq [k]$ of size $|s_A| = n$ and Bob follows the protocol arbitrarily, then*

$$\Pr[\textit{Alice outputs } u \in B \mid \textit{Alice accepts}] \leq \frac{1}{2} + \alpha.$$

Construction of Set Interactive Hashing. The most natural idea for set interactive hashing is to just run n parallel copies of basic interactive hashing to every element of Bob's set $S_B = \{u_1, \ldots, u_n\}$, using the same hash function h for all of them. Alice then learns the outputs $v_1, \ldots, v_n$ and computes $2n$ pre-images $\{u_i^0, u_i^1\} \leftarrow h^{-1}(v_i)$. She checks if any of these $2n$ values lie in s_A: if not she rejects, else she selects the corresponding u_i^b. Bob accepts if $u_i^b = u_i$.

Unfortunately, there is a subtle issue that prevents us from proving the security of this scheme when Alice is honest and Bob is malicious. The issue is that the v_i values may have some repetitions (even if Bob is honest), since it may be the case that $h(u_i) = h(u_j)$ so that $v_i = v_j$. And since Alice accepts if some u_b^i appears in S_A, and then selects a random such value u_b^i, she is more likely to accept if there are more distinct values, and more likely to select an index i that does not repeat vs one that repeats. In other words, both Alice's probability of accepting and the index i that she chooses conditioned on accepting, now depend on the outcome of the basic interactive hashing execution itself, which prevents us from being able to rely on the security of a single instance of the protocol chosen a-priori.

To fix this issue, we notice that each of the values v_i can appear at most twice in the list $v_1, \ldots, v_n$ since the u_i's are distinct, so there are at least $n/2$ distinct values. We have Alice randomly sub-select a list $I \subseteq [n]$ distinct values $v_i \ : \ i \in I$ and the continue the execution as before with the narrowed list. Moreover, the way Alice performs the sub-selection ensures that the outcome of sub-selecting I and choosing a random $i \leftarrow I$ is distributed identically to $i \leftarrow [n]$. This is done by taking every pair $v_i = v_j$ that appears twice in the list and selecting exactly one of $i \neq j$ at random, and then selecting a random subset of exactly $1/2$ of the indices i such that v_i only appears once in the list. It's easy to see that each index $i \in [n]$ is included in I with probability exactly $1/2$. This solves the above issue since the probability of Alice accepting is now the same for every execution of interactive hashing and, conditioned on her accepting, the index i that she chooses is uniformly random over $[n]$. The full construction and analysis are described below.

We consider the following protocol for set interactive hashing using basic interactive hashing. Alice and Bob have sets $s_A, s_B \subseteq [k]$ of size $|s_A| = |s_B| = n$. Let us denote $s_B = \{u_1, \ldots, u_n\}$. Assume n is even.

- Alice and Bob run n parallel copies of a basic interactive hashing protocol where Alice uses the same random coins for all copies, and Bob uses the input u_i in copy i. The resulting transcript consists of a hash function h and an ordered list of n hash outputs $v_1, \ldots, v_n$ with $v_i = h(u_i)$.

  Since the h is 2-to-1, each v_i should appear at most 2 times in the list, and if this is not the case then Alice rejects and sends $\perp$. Let I_{once} be the set of $i \in [n]$ such that v_i only appears once in the list and I_{twice} be the set of $i \in [n]$ such that v_i appears twice.

  Alice sub-selects a set of indices $I \subseteq [n]$ of size $|I| = n/2$ as follows:

  - Select a random subset of exactly $\frac{1}{2}$ of the indices from I_{once}.
  - For each $i \neq j$ in I_{twice} such that $v_i = v_j$, select exactly one of $\{i, j\}$.

- If there exist some values $i \in I, u \in h^{-1}(v_i)$ such that $u \in s_A$ then Alice sends a random such pair (i, u) to Bob and outputs u. Else she sends $\perp$ and rejects.[7]

- If Bob receives $\perp$ from Alice, then he also rejects and outputs $\perp$. Else if he receives the pair (i, u) from Alice then he does the following:

  - if $u = u_i \in S_B$ is the input Bob used in the i'th execution of the interactive hashing protocol, then he outputs u,
  - else he outputs $\perp$.

Lemma 6.4. *Assuming the basic interactive hashing protocol has (α, β)-security, the above construction yields a set interactive hashing protocol with (α, β)-security. In particular, there is an 5-round set interactive hashing protocol with (α, β)-security for any $\beta < 1$ with $\alpha = O(\beta \log k)$. Furthermore, the execution of the protocol can be done in $n \cdot \text{polylog}k$ time and space.*

Proof. Firstly, for correctness, Alice accepts if $s_A \cap U \neq \emptyset$, where $U = h^{-1}(\{v_i \ : \ i \in I\}) \subseteq [k]$ is some set of size n and s_A is a uniformly random and independent set of size n. Therefore $\Pr[\text{Alice accepts}] \geq \Omega(\min(n^2/k, 1))$ by Lemma 2.13.

Secondly, assume Bob follows the protocol honestly using a random subset $s_B = \{u_1, \ldots, u_n\} \subseteq [k]$ of size $|s_B| = n$ and Alice follows the protocol arbitrarily. The protocol transcript consists of the interactive hashing transcript $(h, v_1, \ldots, v_n)$ with $v_j = h(u_j)$ and Alice's choice of (i, u) with $u \in h^{-1}(v_i)$. Condition on any worst-case choice of these values. Bob accepts if $u_i = u$ and outputs $\perp$ otherwise. Since h is a 2-to-1 function, conditioned on the above values, Bob's input u_i is uniformly random over $\{u_i^0, u_i^1\} = h^{-1}(v_i)$ and therefore the probability that $u_i = u$ is exactly $1/2$. Note that, as a special case, this also implies that if Alice and Bob are both honest then $\Pr[\text{Bob accepts} \mid \text{Alice accepts}] = \frac{1}{2}$.

Thirdly, let $B \subseteq [k]$ be a set of size $|B| \leq \beta \cdot k$, and assume Alice follows the protocol honestly using a random subset $s_A \subseteq [k]$ of size $|s_A| = n$ and Bob follows the protocol arbitrarily. Notice that Alice accepts if $s_A \cap U \neq \emptyset$, where $U = h^{-1}(\{v_i \ : \ i \in I\}) \subseteq [k]$ is some set of size n and s_A is a uniformly random and independent set of size n. Therefore the probability of Alice accepting (over the randomness of s_A) is the same no matter what U is and is hence independent of what happens in the executions of the basic interactive hashing protocol. Furthermore, conditioned on Alice accepting, the pair (i, u) that Alice chooses is identical to choosing uniformly random $i \leftarrow [n], u \leftarrow h^{-1}(v_i)$. This is because every $i \in [n]$ has the same probability $(1/2)$ of being included in I, and every pair (i, u) with $i \in I, u \in h^{-1}(v_i)$ then has the same probability of being selected by Alice, since she selects such a pair by selecting a random such u that belongs to her random set s_A and all the u values contained in the n pairs are distinct. Therefore the probability that Alice outputs $u \in B$ conditioned on Alice accepting is the same as the probability that if we choose a

[7] We can assume without loss of generality that even a malicious Alice always sends $u \in h^{-1}(v_i)$ since Bob can check this condition, and we can re-interpret Alice's message as $\perp$ if it does not hold.

uniformly random $i \leftarrow [n], u \in h^{-1}(v_i)$, then $u \in B$. Since we are selecting a uniformly random execution i of the basic interactive hashing protocol, we can rely on the (α, β)-security of basic interactive hashing, to argue that the probability that $\{u_i^0, u_i^1\} = h^{-1}(v_i) \subseteq B$ is at most α and therefore the probability that $u \leftarrow h^{-1}(v_i)$ satisfies $u \in B$ is at most $\frac{1}{2} + \alpha$. □

6.3 OT Construction

Theorem 6.5. *For any $m \geq \ell, \lambda$ there is some $n_{min} = \Omega(\log m + \ell + \lambda)$ such that for all $n \geq n_{min}$ there is an oblivious transfer protocol in the (n, m)-BSM with ℓ-bit messages and security $\varepsilon = 2^{-\Omega(\lambda)}$. The protocol is secure with efficient simulation, and it achieves security against an unbounded-memory sender. The round complexity is $O(\lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda, \log m))$ and the communication complexity $O(m \cdot \lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda, \log(m)))$.*

Proof. We first describe the OT protocol between sender Alice and receiver Bob, then discuss security, and then show how to set parameters to get the claimed efficiency.

Construction. Given m, λ, ℓ we define additional parameters as follows:

- Let $d_1 = O(\lambda + \log m)$ and $n_1 = O(\lambda + \log m + \ell)$ and $k' = O(m + \ell + \lambda \log(\lambda))$ be some values such that there is a $(n_1, m + \ell, \varepsilon = 2^{-\Omega(\lambda)})$-*BSM extractor* $\mathsf{BSMExt} : \{0,1\}^{k'} \times \{0,1\}^{d_1} \to \{0,1\}^{\ell}$ per Theorem 2.9.
- Let $t = O(\lambda + \log m)$ and $d_0 = O(t), n_0 = O(t)$ be some value such that there is a $(t/40, \varepsilon = 2^{-\Omega(\lambda)})$-extractor $\mathsf{Ext} : \{0,1\}^t \times \{0,1\}^{d_0} \to \{0,1\}^{d_1}$ that can be computed using n_0 space per Theorem 2.7.
- Set $k = (m + \lambda) \log^3(m + \lambda)$.
- Set $\alpha = \frac{1}{20}$ and let $\beta = 1/O(\log k)$ be such that (α, β)-security for interactive set hashing holds.
- Set $\delta = \Omega(\beta^2) = 1/O(\log^2 k))$ be such that $-\log(\mathbf{h}_+^{-1}(1 - \delta)) = (1 - \beta/2)$ by Lemma 3.3. This ensures that $\delta k \geq \Omega((m + \lambda) \log(m + \lambda))$.
- Let $g(k) = \mathrm{poly} \log k$ be the parameter from Lemma 6.4, such that an execution of the set interactive hashing protocol with parameters n, k can be done in $n \cdot g(k)$ time and space. Assume $g(k) \geq \lceil \log k + 1 \rceil$.
- Define $n_{min} = \max(2n_0, 2n_1, 3t + g(k)) = O(\lambda + \log m + \ell)$.
- For any $n \geq n_{min}$, define $\tilde{n} = \lfloor (n - 2t)/g(k) \rfloor = \Omega(n/\mathrm{polylog}(m + \lambda))$.
- Let $p = \Omega(\min(\tilde{n}^2/k, 1))$ be the correctness probability of Alice accepting during an honest execution of the set interactive hashing protocol with parameters $\tilde{n}, k$,per Definition 6.3. Set $R_{max} = 2t/p = O(t \cdot \lceil k/\tilde{n}^2 \rceil) = O(\lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda, \log m)$.

The protocol works as follows.

Bob has a choice bit $c \in \{0,1\}$ and Alice has two messages $\mathsf{msg}_0, \mathsf{msg}_1 \in \{0,1\}^{\ell}$.

1. Alice and Bob initiate vectors $r_A \in \{0,1\}^t, r_B \in \{0,1,\perp\}^t$ respectively. They set $i := 0$. Repeat the following until $i = t$:
   (a) Alice and Bob select uniformly random subsets $s_A, s_B \subseteq [k]$ of size $|s_A| = |s_B| = \tilde{n}$ respectively. Alice streams a uniformly random string $x \leftarrow \{0,1\}^k$ to Bob. Alice stores $x[s_A]$, and Bob stores $x[s_B]$.
   (b) Alice and Bob perform set interactive hashing, with Bob's input being s_B.
      - If Alice rejects, then both parties move to the next iteration.

- Else, if Alice accepts with some value $u \in s_A$, then she sets $r_A[i] = x[u]$.
  - If Bob also accepts then it must be the case that $u \in s_B$ and he sets $r_B[i] = x[u]$.
  - Else, Bob sets $r_B[i] = \perp$.

  Both parties increment $i := i + 1$.

If the number of iterations reaches R_{max} before $i = t$, the parties abort.

2. Bob sets $I := \{i \in [t] : r_B[i] \neq \perp\}$. If $|I| < \frac{2 \cdot t}{5}$ then Bob aborts. Else he chooses two sets I_0, I_1 of size $|I_0| = |I_1| = \frac{2 \cdot t}{5}$ by sub-selecting $I_c \subseteq I$ and $I_{1-c} \subseteq [t] \setminus I_c$ uniformly at random.
   Bob sends I_0, I_1 to Alice.

3. Alice checks that $|I_0| = |I_1| = \frac{2 \cdot t}{5}$ and $I_0 \cap I_1 = \emptyset$ and aborts otherwise.
   She chooses an extractor seed $\mathsf{seed} \leftarrow \{0,1\}^{d_0}$ and sends seed to Bob.
   Alice computes $\mathsf{seed}_0 = \mathsf{Ext}(r_A[I_0]; \mathsf{seed})$, $\mathsf{seed}_1 = \mathsf{Ext}(r_A[I_1]; \mathsf{seed})$.
   Bob computes $\mathsf{seed}_c = \mathsf{Ext}(r_B[I_c]; \mathsf{seed})$.

4. Alice streams a uniformly random string $x \leftarrow \{0,1\}^{k'}$ to Bob.
   Alice computes $\mathsf{key}_0 = \mathsf{BSMExt}(x; \mathsf{seed}_0), \mathsf{key}_1 = \mathsf{BSMExt}(x; \mathsf{seed}_1)$.
   Bob computes $\mathsf{key}_c = \mathsf{BSMExt}(x; \mathsf{seed}_c)$.

5. Alice sends to Bob:
$$\mathsf{ct}_0 = \mathsf{key}_0 \oplus \mathsf{msg}_0, \quad \mathsf{ct}_1 = \mathsf{key}_1 \oplus \mathsf{msg}_1$$
   and Bob outputs $\mathsf{msg} = \mathsf{ct}_c \oplus \mathsf{key}_c$.

We refer to each execution of the sub-protocol in steps (a), (b) as an *epoch*. We say that an epoch is successful if Alice *accepts* during the set interactive hashing protocol, and the parties increment i. Note that step 1.(a) and step 4 are "long rounds", and all other steps are "short rounds".

Honest Party Memory. Note that Alice and Bob only need $\widetilde{n}g(k)$ memory to execute each epoch, and need to store up to $2t$ bits from previous epochs for a total of $\widetilde{n}g(k) + t \leq n$ memory to execute part 1 of the protocol. They need $2n_0 \leq n$ memory to execute part 2 and $2n_1 \leq n$ memory to execute part 3. Therefore, in total, the protocol can be executed using at most n memory as required.

Efficiency. The number of rounds is $O(R_{max}) = O(\lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda, \log m)$. Each of the at most R_{max} epochs requires communication $k + \mathrm{polylog}(k) = O(k) = O((m + \lambda)\mathrm{polylog}(m + \lambda)$. Furthermore, step 2 requires $t = O(\lambda + \log m)$ bits of communication, step 3 required $d_0 = O(\lambda + \log m)$ bits of communication, and step 4 requires $k' = O(m + \lambda \log(\lambda))$ bits and step 5 require $\ell = O(m)$ bits, for a total communication of $O(m \cdot \lceil m/n^2 \rceil \cdot \mathrm{poly}(\lambda, \log m)$.

Correctness. It's easy to see that correctness holds unless the parties abort. Firstly, we argue that the probability of aborting due to the number of epochs exceeding R_{max} is $2^{-\Omega(\lambda)}$. By the correctness of set interactive hashing, each epoch is successful with probability $p = \Omega(\widetilde{n}^2/k)$. Therefore, in $R_{max} = 2t/p$ epochs, we expect $2t$ successes, and by Chernoff, the probability of having fewer than t successes is at most $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$.

Second, we argue that the probability of Bob aborting due to $|I| < \frac{2 \cdot t}{5}$ is $2^{-\Omega(\lambda)}$. For each successful epoch i, the probability that $i \in I$ is exactly the probability that Bob accepts in the set interactive hashing protocol, conditioned on Alice accepting, which is $1/2$. Therefore the expected size of I is $t/2$ and, by Chernoff, the probability that $|I| < \frac{2 \cdot t}{5}$ is $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$.

Security for Honest Bob (Receiver). First, we consider the setting where Bob is honest and Alice is adversarial. We show how to efficiently simulate Alice's view, even if Alice has unlimited computational power and memory. The simulator uses Alice's code as a black-box and runs in polynomial time. It simply runs the protocol with Alice by taking on the role of Bob. It follows Bob's specification during all the epochs, but remembers the entire message x sent by Alice. Therefore, no matter what u Alice picks in the i'th successful epoch, the simulator will set $r_B[i] = x[u]$. If there are t successful epochs and Alice does not abort, then the simulator picks completely random sets $I_0 \subseteq [t]$ and $I_1 \subseteq [t] \setminus I_0$ and sends them to Alice. It receives seed from Alice and computes $\mathsf{seed}_b = \mathsf{Ext}(r_B[I_b]; \mathsf{seed})$ for $b \in \{0,1\}$. It receives x from Alice in step 4 and computes $\mathsf{key}_b \mathsf{BSMExt}(x, \mathsf{seed}_b)$ for $b \in \{0,1\}$. Finally, it receives $\mathsf{ct}_0, \mathsf{ct}_1$ and computes $\mathsf{msg}_b = \mathsf{ct}_b \oplus \mathsf{key}_b$ for $b \in \{0,1\}$, and gives $(\mathsf{msg}_0, \mathsf{msg}_1)$ to the ideal functionality on behalf of Alice.

We show that the real execution and the simulation are statistically indistinguihsable. We define the following hybrids.

- H_0: Real execution.
- H_1: In each epoch, Bob remembers the entire x. If the epoch is successful and Alice accepts with some index u, then Bob flips a coin and with probability $1/2$ he sets $r_B[i] = x[u]$ and with probability $1/2$ he sets $r_B[i] = \perp$.
- H_2: This is the simulation.

We have $H_0 \equiv H_1$ by the security of set interactive hashing (for honest Bob), which guarantees that conditioned on any worst-case Alice's view of the protocol, the probability of Bob accepting is exactly $1/2$.

We claim $H_1 \approx H_2$. The only differences between H_1, H_2 are the following. In H_1, the set $I := \{i \in [t] : r_B[i] \neq \perp\}$ is chosen by including the index $i \in [t]$ with independent probability $1/2$. Bob aborts if $|I| \leq \frac{2 \cdot t}{5}$ and else he chooses two sets: $I_c \subseteq I$ and $I_{1-c} \subseteq [t] \setminus I_c$ of size $\frac{2 \cdot t}{5}$ each. On the other hand, in H_2 Bob chooses uniformly random sets $I_c \subseteq [t]$ and $I_{1-c} \subseteq [t] \setminus I_c$ of size $\frac{2 \cdot t}{5}$ each. Notice that in H_1, conditioned on Bob not aborting, the set $I_c \subseteq [t]$ is a uniformly random set of size $\frac{2 \cdot t}{5}$. Therefore, the only difference between the hybrids is the probability that Bob aborts in H_1. By the Chernoff bound, this probability is $2^{-\Omega(t)} \leq 2^{-\Omega(\lambda)}$. This concludes the proof of security for honest Bob.

Security for Honest Alice (Sender): Inefficient Simulation. We now consider the setting where the sender Alice is honest and the receiver Bob is adversarial. We first show that we can inefficiently simulate Bob's view of the protocol, as long as Bob is a BSM adversary whose storage is at most m bits. We then modify the proof further to get efficient simulation via a more complex argument.

We begin by describing the inefficient simulator and then proceed to prove indistinguishability.

Inefficient Simulator: *The simulator runs a copy of Bob and begins executing the protocol acting as the honest Alice. It initiates a set $I^* = \emptyset$. In each epoch i, let y^i_{pre} be Bob's state prior to seeing Alice's message x^i and let y^i_{post} be the state immediately after during the execution. Let the random variables $X^i, Y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}}$ correspond to these values. If*

$$\mathbf{H}_\infty(X^i | Y^i_{\mathsf{pre}} = y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}} = y^i_{\mathsf{post}}) \leq (1 - \delta)k$$

*then abort (denoted: **abort 1**).*
Define the set of indices that Bob "knows" (i.e., have low entropy from his point of view) via:

$$\mathcal{B}^i = \left\{ j \ : \ \mathbf{H}_\infty(X^i[j] \mid Y^i_{\mathsf{pre}} = y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}} = y^i_{\mathsf{post}}) \leq \frac{1}{2} \right\} \subseteq [k].$$

If the i'th epoch is the $\hat{i}$'th successful one and Alice accepts with some value u^i such that $u^i \in \mathcal{B}^i$ then the simulator adds $\hat{i}$ to I^* .
*If at the end of all the epochs $|I^*| \geq \frac{6 \cdot t}{10}$ then the simulator aborts (denoted: **abort 2**).*
When Bob sends I_0, I_1, there must be at least one value of c such that $|I_{1-c} \setminus I^*| \geq \frac{t}{10}$. The simulator picks any such c, sends it to the ideal functionality and gets msg_c. It simulates Alice's communication in step 3,4,5 by choosing seed, x and ct_c just like an honest Alice, but replacing ct_{1-c} by a random independent value.

There are three main differences between the simulation and the honest execution: the presence of the abort conditions abort 1 and abort 2, and the fact that we replace ct_{1-c} by uniform. We do a sequence of hybrids argument to show that these are indistinguishable:

- H_0: Real Execution
- H_1: Add abort condition 1.
- H_2: Add abort condition 2.
- H_3: Choose seed_{1-c} uniformly at random.
- H_4: Simulation: choose ct_{1-c} uniformly at random.

Claim 6.6. $\mathbf{SD}(H_0, H_1) \leq R_{max} \cdot 2^{-\Omega((m+\lambda)\log(m+\lambda))} = 2^{-\Omega(\lambda)}$.

We can bound the statistical distance by the probability of abort 1 occurring in H_1. For each epoch i, if we set $v = (k-m) - (1-\delta)k \geq \Omega((m+\lambda)\log(m+\lambda))$, then with all but 2^{-v} probability over the choice of $y^i_{\mathsf{pre}}, y^i_{\mathsf{post}}$, we have:

$$\begin{aligned}
\mathbf{H}_\infty(X^i | Y^i_{\mathsf{pre}} = y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}} = y^i_{\mathsf{post}}) &\geq \mathbf{H}_\infty(X^i | Y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}}) - v \\
&\geq \mathbf{H}_\infty(X^i | Y^i_{\mathsf{pre}}) - m - v \\
&\geq \mathbf{H}_\infty(X^i) - m - v \\
&\geq k - m - v \\
&\geq (1-\delta)k
\end{aligned}$$

Where the first inequality follows from Lemma 2.2, the second inequality follows by Lemma 2.1 and the fact that $Y^i_{\mathsf{post}} \in \{0,1\}^m$, the third inequality follows from the fact that X^i is independent of Y^i_{pre} and the last follows since X^i is uniform over $\{0,1\}^k$. The claim follows by taking a union bound over all $i \in [R_{max}]$.

Claim 6.7. $\mathbf{SD}(H_1, H_2) \leq 2^{-\Omega(\lambda)}$.

We can bound the statistical distance by the probability of abort 2 occurring in H_2. In each epoch i, let us condition on values values $y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}}$ for which abort 1 does not occur. Then

$$\mathbf{H}_\infty(X^i | Y^i_{\mathsf{pre}} = y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}} = y^i_{\mathsf{post}}) \geq k - m - \lambda \geq (1-\delta)k.$$

Then, by Lemma 3.1, we have

$$\sum_{j \in [k]} \underbrace{\mathbf{H}_\infty(X^i[j] \mid Y^i_{\mathsf{pre}} = y^i_{\mathsf{pre}}, Y^i_{\mathsf{post}} = y^i_{\mathsf{post}})}_{e^i_j} \geq -\log(\mathbf{h}_+^{-1}(\delta))k \geq (1-\beta/2)k.$$

By an averaging argument, this implies that $|\mathcal{B}^i = \{j \ : \ e^i_j < 1/2\}| \leq \beta k$. We now rely on the security of the set interactive hashing, which guarantees that conditioned on the ith epoch being successful, the probability of Alice accepting with some $u^i \in \mathcal{B}^i$ is at most $1/2 + \alpha$ for $\alpha \leq \frac{1}{20}$. Since this holds for each successful epoch, even if we condition on all the previous epochs, we can rely on the Chernoff bound to argue that $|I^*| < (\frac{1}{2} + \frac{1}{10}) \cdot t$ except with $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$ probability.

Claim 6.8. $\mathbf{SD}(H_2, H_3) \leq 2^{-\Omega(\lambda)}$.

Let us condition on the any worst-case sequence of states $y^i_{\mathsf{pre}}, y^i_{\mathsf{post}}$ of Bob during the execution of the protocol, and all the other messages exchanged during parts 1 and 2 of the protocol *except for* all the values X^i, subject to abort 1 and abort 2 not occurring. This also fixed Bob's view through the end of part 2 of the protocol. The remainder of the analysis holds for any such fixed choice. Note that once we fix $y^i_{\mathsf{pre}}, y^i_{\mathsf{post}}$, then the X^i random variables are completely independent of everything else we conditioned on, and they are

also independent of each other. In addition, this fixes the indices of the epochs i that are successful. Let us consider the random variables $R_A[\hat{i}] = X^i[u^i]$ for the values $r_A[\hat{i}]$ kept by Alice in the $\hat{i}$'th successful epoch, which occurs in iteration i. By definition, if $\hat{i} \notin I^*$ then $u^i \notin \mathcal{B}^i$ and therefore $\mathbf{H}_\infty(R_A[\hat{i}] = X^i[u^i]) > \frac{1}{2}$. Furthermore, the variables $R_A[\hat{i}]$ are independent of each other. By definition, for the bit c extracted by the simulator, we have $|I_{1-c} \setminus I^*| \geq \frac{t}{10}$ and hence, conditioned on all the values we fixed, we have:

$$\mathbf{H}_\infty(R_A[I_{1-c}] \mid R_A[I_c]) = \mathbf{H}_\infty(R_A[I_{1-c}]) \geq \mathbf{H}_\infty(R_A[I_{1-c} \setminus I^*]) \geq \sum_{\hat{i} \in I_{1-c} \setminus I^*} \mathbf{H}_\infty(R_A[\hat{i}]) \geq \frac{t}{20}.$$

In part 3 of the execution, Alice chooses a random seed and sets $\mathsf{seed}_c = \mathsf{Ext}(R_A[I_c]; \mathsf{seed}), \mathsf{seed}_{1-c} = \mathsf{Ext}(R_A[I_{1-c}]; \mathsf{seed})$. By the security of Ext, we have that seed_{1-c} is $2^{-\Omega(\lambda)}$ close to uniform even given $\mathsf{seed}, \mathsf{seed}_c$. Note that parts 4,5 of the protocol only depend on $\mathsf{seed}_0, \mathsf{seed}_1$ and are otherwise independent of R_A. Therefore, the hybrids are $2^{-\Omega(\lambda)}$ close.

Claim 6.9. $\mathbf{SD}(H_3, H_4) \leq 2^{-\Omega(\lambda)}$.

This follows by the security of the BSM Extractor. In particular, even given Bob's m-bit state immediately after seeing the string x sent in part 4 of the protocol and the ℓ-bit value $\mathsf{key}_c = \mathsf{BSMExt}(x; \mathsf{seed}_c)$, we have that $\mathsf{key}_{1-c} = \mathsf{BSMExt}(x; \mathsf{seed}_{1-c})$, is $2^{-\Omega(\lambda)}$ close to uniform. Once we replace key_{1-c} by uniform, we get hybrid 4.

Security for Honest Alice (Sender): Efficient Simulation. We now show how to modify the preceding argument to make the simulator efficient, using Bob's code as a black box. Previously, the simulator was non-black-box and inefficient since it needed to know Bob's states y^i_{pre} and y^i_{post} and to compute the set $\mathcal{B}^i$, corresponding to the indices $j \in [k]$ that Bob "knows" (i.e., $x^i[j]$ has low entropy from Bob's point of view), which is not an efficient operation. To make the simulator black-box/efficient, we replace the set $\mathcal{B}^i$ with a different set $\mathcal{H}^i$, corresponding to the "high-frequency" indices $j \in [k]$ that Alice is likely to output in interactive hashing portion of the ith epoch, conditioned on the epoch being successful. We can efficiently compute $\mathcal{H}^i$ by forking many executions of the interactive hashing sub-protocol and rewinding to estimate these frequencies. The main observation is that the set $\mathcal{B}^i$ is sufficiently small, so that if Bob wants to ensure that *some* index from $\mathcal{B}^i$ is chosen with good probability, he has to "play" such indices with high frequency in the interactive hashing protocol. More specifically, since the probability of the interactive hashing outputting *any specific* "low frequency" index $j \notin \mathcal{H}^i$ is small, we can rely on the union bound over the small set $\mathcal{B}^i$, to show that the probability of outputting *some* such $j \in \mathcal{B}^i$ such that $j \notin \mathcal{H}^i$ is still small. Therefore $\mathcal{H}^i$ is an efficiently commutable small set that can serve as a good proxy for $\mathcal{B}^i$ in the previous argument. We now proceed with the formal proof.

Efficient Simulator: *Define parameters $\mu := \log^{1.5}(k)/k$, $\rho_{min} = \frac{1}{40R_{max}}$, $\gamma = (\rho_{min}\mu)/10$. The simulator runs Bob and begins executing the protocol acting as the honest Alice. It initiates a set $I^*_{\mathcal{H}} = \emptyset$. In each epoch i, prior to executing the interactive hashing protocol in step (b) with Bob, the simulator computes a set $\mathcal{H}^i$ via a rewinding strategy that forks many independent copies of the execution for the interactive hashing protocol and estimates the following parameters:*

- *$\tilde{\rho}^i$ is the estimated probability that the epoch is successful and Alice accepts in the interactive hashing protocol, where the estimate has precision γ, meaning that the true probability ρ^i is equal to $\tilde{\rho}^i \pm \gamma$ except with a sampling error of $2^{-\Omega(\lambda+\log(R_{max}))}$ over the coins of the estimator.*[8] *If $\tilde{\rho}^i < \rho_{min}$ then set $\mathcal{H}^i := \emptyset$ and exit.*
- *For each $j \in [k]$: $\tilde{g}^i_j$ be the estimated probability that Alice accepts* and *outputs the index $u^i = j$ in epoch i, with precision γ, meaning that the true probability g^i_j is equal to $\tilde{g}^i \pm \gamma$ except with error $2^{-\Omega(\lambda+\log(R_{max})+\log k)}$ over the coins of sampling.*[8]

[8] This can be done efficiently via the Chernoff bound.

- *For each $j \in [k]$: Set $\tilde{f}^i_j = \tilde{g}^i_j/\tilde{\rho}^i$ to be the estimated conditional probability that Alice outputs the index $u^i = j$ in epoch i conditioned on epoch i being successful. Then $\tilde{f}^i_j$ is an estimate for the true probability $f^i_j = g^i/\rho^i$. Assuming that the estimates for $\tilde{g}^i_j, \tilde{\rho}^i$ are within their target precision and $\tilde{\rho}^i \geq \rho_{min}$, then the estimate for $\tilde{f}^i_j$ is with precision $\mu/2$ since:*

$$f^i_j = \frac{g^i_j}{\rho^i} = \frac{\tilde{g}^i_j \pm \gamma}{\tilde{\rho}_i \pm \gamma} = \frac{\tilde{g}^i_j}{\tilde{\rho}_i} \pm \mu/2.$$

- *Set $\mathcal{H}^i := \{j \ : \ \tilde{f}^i_j \geq \mu\}$.*

If the i'th epoch is the $\hat{i}$'th successful one and Alice accepts with some value u^i such that $u^i \in \mathcal{H}^i$ then the simulator adds $\hat{i}$ to $I^*_{\mathcal{H}}$.
If at the end of all the epochs $|I^*_{\mathcal{H}}| \geq \frac{6 \cdot t}{10}$ then the simulator aborts (denoted: abort 2).
When Bob sends I_0, I_1, there must be at least one value of c such that $|I_{1-c} \setminus I^*_{\mathcal{H}}| \geq \frac{t}{10}$. The simulator picks any such c, sends it to the ideal functionality and gets msg_c. It simulates Alice's communication in step 3,4,5 by choosing seed, x and ct_c just like an honest Alice, but replacing ct_{1-c} by a random independent value.

Note that the simulator above is essentially the same as the inefficient simulator with the differences that there is no abort 1 condition (since it cannot be checked efficiently) and the simulator uses $\mathcal{H}^i$ in place of $\mathcal{B}^i$ to define the set that we previously denoted by I^* and now denote by $I^*_{\mathcal{H}}$ to disambiguate. For the analysis, it will be convenient to still define the sets $\mathcal{B}^i$ and $I^*_{\mathcal{B}}$ corresponding to the way I^* was defined previously for the inefficient simulation. Moreover, it will be useful to define the same abort 1 condition as in the inefficient simulation (even though our efficient simulator does not check it). Moreover we define two additional abort conditions: abort 0 says that the execution aborts if any of the estimates $\tilde{\rho}_i, \tilde{g}^i_j$ are not within the specified precision of the true probabilities, and abort 3 says that the execution aborts if $|I^*_{\mathcal{B}} \setminus I^*_{\mathcal{H}}| \geq t/20$.

There are two main differences between the simulation and the honest execution: the presence of the abort condition 2, and the fact that we replace ct_{1-c} by uniform. We do a sequence of hybrids argument to show that these are indistinguishable:

- H_0: Real Execution
- $H_{1/2}$: Add abort condition 0
- H_1: Add abort condition 1.
- H_2: Add abort condition 2.
- H_3: Add abort condition 3.
- H_4: Replace seed_{1-c} by uniformly random.
- H_5: Replace key_{1-c} by uniformly random.
- H_6: Simulation: Remove abort conditions 0,1,3.

Claim 6.10. $\mathbf{SD}(H_0, H_{1/2}) \leq 2^{-\Omega(\lambda)}$.

This follows from our bound on the sampling errors of the estimates $\tilde{\rho}^i$ and g^i_j and a union bound over $i \in R_{max}, j \in [k]$.

Claim 6.11. $\mathbf{SD}(H_{1/2}, H_1) \leq 2^{-\Omega(\lambda)}$.

This follows by the same argument as in the proof of inefficient simulation (Claim 6.6).

Claim 6.12. $\mathbf{SD}(H_1, H_2) \leq 2^{-\Omega(\lambda)}$.

We bound the probability of abort 2 occurring in H_2. Assume abort 0 does not occur. The either $\mathcal{H}^i = \emptyset$ or $\mathcal{H}^i = \{j \ : \ \tilde{f}^i_j \geq \mu\} \subseteq \{j \ : \ f^i_j \geq \mu/2\}$. Since $\sum_j f^i_j = 1$ we have $|\{j \ : \ f^i_j \geq \mu/2\}| \leq 2/\mu \leq k/\log^{1.5}(k)$. Therefore in either case $|\mathcal{H}^i| \leq k/\log^{1.5}(k)$. By the $(\alpha = \frac{1}{20}, k/\log^{1.5}(k))$-security of the set interactive hashing, with the bad set $\mathcal{H}^i$, the probability of $u^i \in \mathcal{H}^i$ conditioned on Alice accepting is at most $1/2 + \alpha$ for $\alpha \leq \frac{1}{20}$. Since this holds for each successful epcoh even conditioned on all previous epochs, we can rely on the Chernoff bound to argue that $|I^*_{\mathcal{H}}| < (\frac{1}{2} + \frac{1}{10}) \cdot t$ except with $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$ probability.

Claim 6.13. $\mathbf{SD}(H_2, H_3) \leq 2^{-\Omega(\lambda)}$.

We bound the probability of abort 3 occurring in $\mathbf{H}_3$. Assume abort 0 does not occur. We can write $|I^*_B \setminus I^*_{\mathcal{H}}| \leq q_1 + q_2$, where we define q_1 to be the number of epochs i in which $\tilde{\rho}_i \leq \rho_{min}$ but epoch i is successful, and we define q_2 to be the number of successful epochs i in which Alice accepts with some value u^i such that $u^i \in \mathcal{B}^i \setminus \mathcal{H}^i$ and $\tilde{\rho}_i > \rho_{min}$. Firstly, we can bound $q_1 \leq t/40$ since when abort 0 does not occur then $\tilde{\rho}_i \leq \rho_{min}$ implies that the probability of epoch i succeeding is $\rho_i \leq 2\rho_{min} = \frac{1}{20R_{max}}$. Therefore the probability that there exist $t/40$ epochs where this occurs is $\leq \binom{R_{max}}{t/40}(\frac{1}{20R_{max}})^{t/40} \leq (1/t)^{\Omega(t)} \leq 2^{-\Omega(\lambda)}$. Secondly, we can bound $q_2 \leq t/40$ since for any epoch i such that $\tilde{\rho}_i > \rho_{min}$ and any $j \notin \mathcal{H}^i$ we have

$$\Pr[u^i = j \mid \text{epoch } i \text{ successful}] = f^i_j \leq \tilde{f}^i_j + \mu/2 < (1.5)\mu.$$

Furthermore, by the same calculation as done in the proof of inefficient simulation, we have $|\mathcal{B}^i| \leq \beta k \leq k/\log^2 k$. Therefore, we can rely on the union bound:

$$\begin{aligned}
\Pr[u^i \in \mathcal{B}^i \setminus \mathcal{H}^i \mid \text{epoch } i \text{ successful}] &\leq \sum_{j \in \mathcal{B}^i \setminus \mathcal{H}^i} \Pr[u^i = j \mid \text{epoch } i \text{ successful}] \\
&\leq \sum_{j \in \mathcal{B}^i \setminus \mathcal{H}^i} (1.5\mu) \\
&\leq |\mathcal{B}^i| \cdot (1.5) \cdot \mu = O(1/\log^{.5}(k) = o(1).
\end{aligned}$$

Therefore within the t successful epochs, we expect $o(t)$ of them to satisfy $u^i \in \mathcal{B}^i \setminus \mathcal{H}^i$ and, by Chernoff, the probability that more than $t/40$ of them satisfy this is $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$.

Claim 6.14. $\mathbf{SD}(H_3, H_4) \leq 2^{-\Omega(\lambda)}$.

To argue that hybrids H_3 and H_4 are $2^{-\Omega(\lambda)}$ statistically close, we employ a similar argument as in Claim 6.8 for inefficient simulation. In particular, by definition, for the bit c extracted by the simulator, we have $|I_{1-c} \setminus I^*_{\mathcal{H}}| \geq \frac{t}{10}$. Assuming abort 3 does not occur, we have $|I^*_B \setminus I^*_{\mathcal{H}}| \leq \frac{t}{20}$. Therefore $|I_{1-c} \setminus I^*_{\mathcal{B}}| \geq \frac{t}{20}$. The rest of the argument is essentially the same as before. Using the same notation as previously.

Let us condition on the any worst-case sequence of states $y^i_{\mathsf{pre}}, y^i_{\mathsf{post}}$ of Bob during the execution of the protocol, and all the other messages exchanged during parts 1 and 2 of the protocol *except for* all the values X^i, subject to abort 0,1,2,3 not occurring. Let us consider the random variables $R_A[\hat{i}] = X^i[u^i]$ for the values $r_A[\hat{i}]$ kept by Alice in the $\hat{i}$'th successful epoch, which occurs in iteration i. We have:

$$\mathbf{H}_\infty(R_A[I_{1-c}] \mid R_A[I_c]) = \mathbf{H}_\infty(R_A[I_{1-c}]) \geq \mathbf{H}_\infty(R_A[I_{1-c} \setminus I^*_{\mathcal{B}}]) \geq \sum_{\hat{i} \in I_{1-c} \setminus I^*_{\mathcal{B}}} \mathbf{H}_\infty(R_A[\hat{i}]) \geq \frac{t}{40}.$$

Note that we conditioned on the entire view of Bob except for the last message, which is defined as

$$\mathsf{seed}, \mathsf{ct}_c = \mathsf{Ext}(R_A[I_c]; \mathsf{seed}) \oplus \mathsf{msg}_c, \mathsf{ct}_{1-c} = \mathsf{Ext}(R_A[I_{1-c}]; \mathsf{seed}) \oplus \mathsf{msg}_{1-c}.$$

In part 3 of the execution, Alice chooses a random seed and sets $\mathsf{seed}_c = \mathsf{Ext}(R_A[I_c]; \mathsf{seed}), \mathsf{seed}_{1-c} = \mathsf{Ext}(R_A[I_{1-c}]; \mathsf{seed})$. By the security of Ext, we have that seed_{1-c} is $2^{-\Omega(\lambda)}$ close to uniform even given $\mathsf{seed}, \mathsf{seed}_c$. Note that parts 4,5 of the protocol only depend on $\mathsf{seed}_0, \mathsf{seed}_1$ and are otherwise independent of R_A. Therefore, the hybrids are $2^{-\Omega(\lambda)}$ close.

Claim 6.15. $\mathbf{SD}(H_4, H_5) \leq 2^{-\Omega(\lambda)}$.

This follows by the security of the BSM Extractor. The proof is identical to Claim 6.9 for inefficient simulation.

Claim 6.16. $\mathbf{SD}(H_5, H_6) \leq 2^{-\Omega(\lambda)}$.

The argument that removing the abort conditions is statistically close is the same as the argument that adding them is statistically close in Claims 6.13, 6.11, 6.10. □

6.4 Multiparty Computation from OT

It is known that one can use the oblivious transfer (OT) ideal functionality as a black box to achieve general multi-party computation in the OT-hybrid model [Kil88, IPS08]. By plugging in our construction of OT in the BSM, one therefore gets general multiparty computation in the BSM with efficient simulation. The same observation was also explicitly made by [DHRS07], with the only difference that in their case, the simulation was inefficient.

We provide some additional details. Assume we want to perform a multiparty computation of some circuit C with N parties and security parameter λ.

- Honest user storage: If we start with an OT protocol in the (n, m)-BSM and use it to construct MPC, the honest users need to keep in memory all of the intermediate state of the external MPC protocol in the OT-hybrid model. The size of this state is some $\text{poly}(|C|, N, \lambda)$ completely independent of n, m. Therefore the resulting protocol will be in the (n', m)-BSM model with $n' = n + \text{poly}(|C|, N, \lambda)$, which can still be arbitrarily smaller than the adversarial storage m.

- Adversary storage: We note that the MPC protocol only executes copies of the OT protocol sequentially. When the "outer" simulator of the overall MPC needs to simulate each OT execution, it can spawn of a fresh copy of an "inner" OT simulator. Although the outer simulator may need to store some additional state related to the outer MPC execution, this is completely unrelated to the inner OT. Therefore, the OT protocol only needs to achieve security against an OT adversary with the same storage bound m as the overall MPC adversary.

Summarizing we get the following theorem as a corollary of our OT protocol (Theorem 6.5) and the works of [Kil88, IPS08].

Theorem 6.17. *For any m, λ and any N-party ideal functionality $\mathcal{F}$ having circuit size $|\mathcal{F}|$, there is some $n_{min} = O(\log m) + \text{poly}(|\mathcal{F}|, N, \lambda)$ such that for all $n \geq n_{min}$ there is a secure MPC protocol in the (n, m)-BSM with $\varepsilon = 2^{-\Omega(\lambda)}$ security against an adversary that can maliciously corrupt any number of parties. The round complexity is $O(\lceil m/n^2 \rceil \cdot \text{poly}(|\mathcal{F}|, N, \lambda))$ and the communication complexity $O(m \cdot \lceil m/n^2 \rceil \cdot \text{poly}(|\mathcal{F}|, N, \lambda))$.*

7 Lower Bounds on Rounds and Communication

In this section, we prove that achieving large memory gaps between adversaries and honest parties in the bounded storage model inherently requires large round complexity and communication. In Section 7.1, we introduce the specific BSM we use for our lower bound. In Section 7.2, we prove a lower bound on round complexity and communication in this model. Looking ahead, one drawback of this lower bound is that it only rules out protocols secure against somewhat strong, non-streaming adversaries. In Section 7.3, we introduce another variant of the BSM where adversaries are streaming, and prove an associated lower bound in Section 7.4.

7.1 Model for the Lower Bound: the Unbounded Processing Model

As mentioned in the introduction, our lower bound holds in a stronger model than the variant of streaming BSM we use for our positive results Section 4. The main conceptual difference is that both the honest parties are only bound by their storage used between the rounds, but could compute its contents using unbounded temporary memory. We describe that model, and introduce notation in more details below. We develop in more details the relation with previously discussed notions of BSM in Remark 7.1.

A (n, m)-bounded storage model protocol Π in the *unbounded processing model*, is parametrized by a bound n on the storage of honest parties and a bound m on the storage of the adversary. In the case of two parties, Alice and Bob send (potentially large) messages to each other at every round. Every round i consists of one party, say Alice, sending a message to the other, say Bob, as follows: she computes

$$(s_A^{(i)}, M^{(i)}) \leftarrow \mathsf{send}_A^{(i)}(s_A^{(i-1)})$$

and Bob computes

$$s_B^{(i)} \leftarrow \mathsf{receive}_B^{(i)}(s_B^{(i-1)}, M^{(i)}),$$

and vice-versa if Bob sends the message in round i. $s_A^{(i)}$ and $s_B^{(i)}$ denote the local states kept by Alice and Bob respectively after round i, and $M^{(i)}$ denotes the message sent at round i. By convention their starting states are $s_A^{(0)} = s_B^{(0)} = \emptyset$. We require the states s_A and s_B to be of bounded size, namely $|s_A^{(i)}|, |s_B^{(i)}| \leq n$ for all i. There are however no restrictions on the complexity of the functions $\mathsf{send}_A^{(i)}, \mathsf{receive}_B^{(i)}$ in rounds i where Alice sends a message, or $\mathsf{send}_B^{(i)}, \mathsf{receive}_A^{(i)}$ in rounds i where Bob sends a message. We'll assume for convenience of notation that parties speak turn by turn, namely Alice sends messages in odd rounds and Bob sends messages in even rounds, or vice-versa.

Adversaries Adv in this model are similarly modeled as functions $\mathsf{Adv}^{(i)} : (s_E^{(i-1)}, M^{(i)})^{(i)} \mapsto s_E^{(i)}$, where there are no restrictions on the complexity of $\mathsf{Adv}^{(i)}$, up to the state $s_E^{(i)}$ having size at most m for all i.

We will respectively denote by C and R some upper bounds on the total communication and the number of rounds of Π, which hold over all possible executions of Π. In the case of key agreement, this is without loss of generality up to a constant loss in either security (having both parties abort and output 0) or correctness (having both parties abort and output a random value), using Markov's inequality.

We will furthermore suppose that the length of the message sent in any fixed round i is fixed by the protocol, and in particular does not depend on its internal randomness. We discuss how to relax this requirement later in Remark 7.6.

We now define key agreement (with 1-bit output) in this model. A key agreement protocol Π in the (n, m)-BSM is a protocol with two parties, Alice and Bob, which results in a single-bit final state $s_A^{(R)}, s_B^{(R)} \in \{0, 1\}$. We require the following properties:

- δ-correctness: We have
$$\Pr[s_A^{(R)} = s_B^{(R)}] \geq 1/2 + \delta.$$
for some constant $\delta \leq 1/2$.
- (m, ε)-Sscurity: No adversary Adv with memory m (with the specifications above) can guess Alice's output $s_A^{(R)}$ at the end of the protocol:
$$\forall \mathsf{Adv}, \Pr[s_E^{(R)} = s_A^{(R)}] \leq 1/2 + \varepsilon,$$
for constant $\varepsilon \leq 1/2$.
- We furthermore require $\delta - \varepsilon = \Theta(1)$.

The last requirement enforces that adversaries have strictly smaller probability of guessing the output of the honest parties than the other honest party.

Remark 7.1 (Comparison with previously discussed models)**.** As mentioned before, this defines a more expressive model than the one in Section 4, as honest users for the definition above are stronger than in Section 4. The main differences are (1) there are no restrictions on the computational power of the *honest users* to compute their states kept between the rounds of the protocol, who can in particular use arbitrary large temporary memory, (2) they are neither bound to send uniformly random "long" messages, nor restricted to have local access to it. In the terminology we used in the introduction, the lower bound holds for honest users without restrictions (b), (c), but with the same capability (d) as Eve. All these capabilities make the resulting lower bound stronger.

However, we only consider strong adversaries with unlimited short-term memory (restriction (d) in the introduction). This does make our lower bound weaker than ideal, and leaves open the possibility of a tighter lower bound for more restricted classes of "streaming" adversaries. Looking ahead, in Sections 7.3 and 7.4, we adapt this model and the subsequent lower bound to restrict adversaries to be streaming, albeit at the cost of slightly worse quantitative bounds.

To sum up, in this new model, the honest users have the same capabilities as the adversary, up to a smaller storage between rounds.

In terms of key agreement, we relax correctness and security to only be constants, as long as honest users have some non-trivial advantage in agreeing on the output bit compared to an adversary; this again makes our lower bound stronger.

7.2 Lower Bound in the Unbounded Processing Model

Before proving our lower bounds, we state some useful lemmas.

Lemma 7.2 ([DM04])**.** *Let P be a random process that outputs a tuple of (potentially correlated) values (Z, Y). Let q be some integer. Let P' denote the process of sampling $(Z_0, Y) \leftarrow P$, computing $X = f(Y)$ for some randomized function f, and independently sampling for $j \in [q]$:*

$$Z_j \leftarrow P_{Z|Y}$$

where $P_{Z|Y}$ denotes the marginal distribution of Z induced by P, conditioned on Y. Then there exists some index $i \in [q]$ such that:

$$\mathbf{I}(X; Z_0 | Z_1, \cdots, Z_i) \leq \frac{\mathbf{H}(X)}{q+1}. \tag{5}$$

Furthermore, one can (inefficiently) compute such an index i.

Proof. The proof directly follows from the proof of Theorem 1 of [DM04]. We restate it here for completeness. It suffices to argue that $Z_0, \cdots, Z_q$ are symmetric with respect to X, namely:

$$P'_{X, Z_{i_1}, \ldots, Z_{i_w}}(x, z_1, \ldots, z_w) = P'_{X, Z_{i'_1}, \ldots, Z_{i'_w}}(x, z_1, \ldots, z_w)$$

for all $w \leq q$, and all sets of distinct indices $(i_1, \ldots, i_w)$ and $(i'_1, \ldots, i'_w)$. Indeed, a chain rule for conditional mutual information gives:

$$\sum_{i=0}^{n} \mathbf{I}(X; Z_i | Z_{i-1}, \ldots, Z_0) = \mathbf{I}(X; Z_0, \ldots Z_q) \leq \mathbf{H}(X),$$

so that there exists some $i \in [q]$ (computable inefficiently) such that:

$$\mathbf{I}(X; Z_i | Z_{i-1}, \ldots, Z_0) \leq \frac{\mathbf{H}(X)}{q+1}$$

and the conclusion follows as $\mathbf{I}(X; Z_0 | Z_1, \ldots, Z_i) = \mathbf{I}(X; Z_i | Z_{i-1}, \ldots, Z_0)$ by symmetry.

To prove that $Z_0, \cdots, Z_q$ are symmetric with respect to X, we observe that they are symmetric with respect to Y, as sampled independently conditioned on Y. They are therefore symmetric with respect to $X = f(Y)$. □

Lemma 7.3. *Consider the process P', the random variables X, Z and some index $i \in [q]$ defined in Lemma 7.2.*

Consider the random variable $Z' \leftarrow P'_{Z_0|Z_1,\cdots,Z_i}$ obtained by sampling Z_0 conditionally on $Z_1, \cdots, Z_i$. Then:

$$\mathbf{SD}\left((X, Z, \{Z_j\}_{j\le i}),\ (X, Z', \{Z_j\}_{j\le i})\right) \le \sqrt{\frac{\mathbf{H}(X)}{2(q+1)}},$$

Proof. We start with Eq. (5) of Lemma 7.2:

$$\mathbf{I}(X; Z_0|Z_1, \cdots, Z_i) \le \frac{\mathbf{H}(X)}{q+1}. \tag{5}$$

By definition of mutual information, the left-hand-side is:

$$\mathbf{I}(X; Z_0|Z_1, \cdots, Z_i) = D_{KL}(P'_{(X,Z_0)|\{Z_j\}_{j\le i}} \| P'_{X|\{Z_j\}_{j\le i}} \times P'_{Z_0|\{Z_j\}_{j\le i}}) \tag{6}$$

Conditioned on $\{Z_j\}_{j\le i}$, the conditional distribution of (X, Z_0) is $P'_{(X,Z_0)|\{Z_j\}_{j\le i}}$ (by definition), and the conditional distribution of (X, Z') is $P'_{X|\{Z_j\}_{j\le i}} \times P'_{Z_0|\{Z_j\}_{j\le i}}$. This is because the marginal distribution of X conditioned on $\{Z_j\}_{j\le i}$ is $P'_{X|\{Z_j\}_{j\le i}}$, and Z' is independently sampled from $P'_{Z_0|\{Z_j\}_{j\le i}}$.

We therefore have:

$$\begin{aligned}
\mathbf{SD}\left(X, Z, \{Z_j\}_{j\le i}),\ (X, Z', \{Z_j\}_{j\le i})\right) &\le \mathop{\mathbf{E}}_{\{Z_j\}} \mathbf{SD}\left(((X, Z)|\{Z_j\}_{j\le i}),\ (X, Z')|\{Z_j\}_{j\le i})\right) \\
&\le \mathop{\mathbf{E}}_{\{Z_j\}} \sqrt{\frac{1}{2} \cdot D_{KL}\left(P'_{(X,Z_0)|\{Z_j\}_{j\le i}} \| P'_{X|\{Z_j\}_{j\le i}} \times P'_{Z_0|\{Z_j\}_{j\le i}}\right)} \\
&\le \sqrt{\frac{1}{2} \cdot \mathop{\mathbf{E}}_{\{Z_j\}} D_{KL}\left(P'_{(X,Z_0)|\{Z_j\}_{j\le i}} \| P'_{X|\{Z_j\}_{j\le i}} \times P'_{Z_0|\{Z_j\}_{j\le i}}\right)} \\
&= \sqrt{\frac{1}{2} \cdot \mathbf{I}(X; Z_0|Z_1, \cdots, Z_i)} \\
&\le \sqrt{\frac{\mathbf{H}(X)}{2(q+1)}},
\end{aligned}$$

where the second inequality follows by Pinsker's inequality (Lemma 2.12), the third by concavity of the square root function, the equality by Eq. (6), and the last inequality by Eq. (5). □

We are now ready to prove our main theorem of this section.

Theorem 7.4. *Let Π be a key agreement protocol in the unbounded processing model (Section 7.1), with honest storage n, satisfying δ-correctness and (m, ε) security, where $\delta - \varepsilon = \Omega(1)$. Suppose furthermore that for any execution of Π, the total communication between Alice and Bob is at most C and consists of at most R rounds. Then $C \ge \Omega\left(\frac{m^{3/2}}{n}\right)$, and $R \ge \Omega\left(\frac{\sqrt{m}}{n}\right)$.*

Proof. We define the following transformation, which takes a protocol Π in the unbounded processing model, with total communication (upper-bounded by) C, and number of rounds (upper bounded by) R, and outputs a protocol $\overline{\Pi}$ with number of rounds (upper bounded by) $R - 1$, such that correctness and security of Π respectively imply correctness and security of $\overline{\Pi}$ with a small loss.

On a high level, $\overline{\Pi}$ compresses the first two rounds of Π into a single round. Without loss of generality, suppose that in Π, Alice sends the first message $M^{(1)}$. In $\overline{\Pi}$, Bob will instead send the first message. Namely, $(\mathsf{send}_A^{(1)}, \mathsf{receive}_B^{(1)}, \mathsf{send}_B^{(2)}, \mathsf{receive}_A^{(2)})$ are compressed into two algorithms $(\overline{\mathsf{send}}_B^{(1)}, \overline{\mathsf{receive}}_A^{(1)})$.

The subsequent rounds are identical to Π, that is, Alice and Bob compute the subsequent messages and update their states as in Π. Formally, for $1 \le k \le \lfloor (R-1)/2 \rfloor$

$$\overline{\mathsf{send}}_A^{(2k)}(s_A) = \mathsf{send}_A^{(2k+1)}(s_A), \quad \overline{\mathsf{receive}}_B(2k)(s_B, M) = \mathsf{receive}_B^{(2k+1)}(s_B, M),$$
$$\overline{\mathsf{send}}_B^{(2k+1)}(s_B) = \mathsf{send}_B^{(2k+2)}(s_B), \quad \overline{\mathsf{receive}}_A(2k+1)(s_A, M) = \mathsf{receive}_A^{(2k+2)}(s_B, M).$$

Therefore, we now focus on defining $\overline{\mathsf{send}}_B^{(1)}$ and $\overline{\mathsf{receive}}_A^{(1)}$, or, equivalently, $(\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, \overline{M}^{(1)})$, that is the states of Alice and Bob after one round of $\overline{\Pi}$, along with the first message sent.

Our transformation acts differently according to the length of the first message $M^{(1)}$ of Π, which we recall is assumed of fixed length.

Case 1: $|M^{(1)}| \le m$**.** If the first message $M^{(1)}$ of the original protocol Π is of size $|M^{(1)}| \le m$, the transformation works as follows. Bob computes the two messages $M^{(1)}, M^{(2)}$ of the first two rounds of Π himself, and in particular computes himself Alice's first round message $M^{(1)}$ in Π. He sends both messages $(M^{(1)}, M^{(2)})$ in the first round of the new protocol $\overline{\Pi}$. Given this new message, Alice (inefficiently) samples a state compatible with the original first message $M^{(1)}$, and then processes $M^{(2)}$ as in the original protocol Π.

Formally, we define $\overline{\mathsf{send}}_B^{(1)}$, $\overline{\mathsf{receive}}_A^{(1)}$ as follows:

- $\overline{\mathsf{send}}_B^{(1)}$: Compute $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$. Compute $s_B^{(1)} \leftarrow \mathsf{receive}_B^{(1)}(M^{(1)})$. Compute $(s_B^{(2)}, M^{(2)}) \leftarrow \mathsf{send}_B^{(2)}(s_B^{(1)})$. Output:
$$\overline{s_B}^{(1)} = s_B^{(2)}, \quad \overline{M}^{(1)} = (M^{(1)}, M^{(2)}).$$
- $\overline{\mathsf{receive}}_A^{(1)}(\overline{M}^{(1)} = (M^{(1)}, M^{(2)}))$: Given $M^{(1)}$, sample $\overline{s_A}^{(0)}$ as
$$\overline{s_A}^{(0)} \leftarrow P_{s_A^{(1)}|M^{(1)}}$$
where $P_{s_A^{(1)}|M^{(1)}}$ is the marginal distribution of Alice's state $s_A^{(1)}$ after the first round of Π, induced by $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$ conditioned on $M^{(1)}$. Note that this is an inefficient process.
Output:
$$\overline{s_A}^{(1)} \leftarrow \mathsf{receive}_A^{(2)}(\overline{s_A}^{(0)}, M^{(2)}).$$

We first argue that $\overline{\Pi}$ is δ-correct. The conditional sampling makes the two following distributions identically distributed:

$$\Big((s_A^{(1)}, s_B^{(1)}, M^{(1)}); \quad (s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}, \quad s_B^{(1)} \leftarrow \mathsf{receive}_B^{(1)}(M^{(1)})\Big)$$
$$\Big((\overline{s_A}^{(0)}, s_B^{(1)}, M^{(1)}); \quad (s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}, \quad \overline{s_A}^{(0)} \leftarrow P_{s_A^{(1)}|M^{(1)}}, \quad s_B^{(1)} \leftarrow \mathsf{receive}_B^{(1)}(M^{(1)})\Big)$$

Therefore the distributions remain identical after computing $\mathsf{send}_B^{(2)}$, $\mathsf{receive}^{(2)}$, that is

$$(s_A^{(2)}, s_B^{(2)}, M^{(2)}) \equiv (\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, M^{(2)}).$$

In particular, the user states $(\overline{s_A}^{(1)}, \overline{s_B}^{(1)})$ of $\overline{\Pi}$ after round one are identically distributed as the user states $(s_A^{(2)}, s_B^{(2)})$ of Π after round two. δ-correctness of $\overline{\Pi}$ then follows by δ-correctness of Π.

Next, we show that (m, ε)-security of Π implies (m, ε)-security for $\overline{\Pi}$. Let $\overline{\mathsf{Adv}}$ be an unbounded processing model adversary with storage m and advantage ε against $\overline{\Pi}$. We build an adversary Adv against Π as follows. Define:

$$s_E^{(1)} = \mathsf{Adv}^{(1)}(M^{(1)}) = M^{(1)},$$

and $\mathsf{Adv}^{(i)} = \overline{\mathsf{Adv}}^{(i-1)}$ for all $i \ge 2$. In particular, $\mathsf{Adv}^{(2)}$ is given as input $(s_E^{(1)} = M^{(1)}, M^{(2)})$, and can therefore run $\overline{\mathsf{Adv}}^{(1)}(\overline{M}^{(1)})$ with input distributed identically as $\overline{M}^{(1)} = (M^{(1)}, M^{(2)})$ by construction of $\overline{\Pi}$. In particular, Adv has the same advantage ε as $\overline{\mathsf{Adv}}$. Moreover, Adv is an adversary in the unbounded processing model with storage m as $|M_1| \le m$ and $\overline{\mathsf{Adv}}$ has storage at most m.

Case 2: $|M^{(1)}| > m$**.** If the first message $M^{(1)}$ of Π is of size $|M^{(1)}| > m$, the transformation works as follows. Bob computes the first message $M^{(1)}$ of Alice in Π, and computes the second message $M^{(2)}$ of Π. Instead of sending $M^{(1)}$ directly, Bob (inefficiently) samples many independent copies of Alice's state after the first round of Π conditioned on $M^{(1)}$, that we denote $\{\overline{s_{A,j}}\}$. He instead sends $\overline{M}^{(1)} = (\{\overline{s_{A,j}}\}, M^{(2)})$ as the first round message of $\overline{\Pi}$. Given this new first message, Alice (inefficiently) samples a state $\overline{s_A}^{(0)}$ (corresponding to a state of Π after the first round) conditioned on $\{\overline{s_{A,j}}\}$, and processes $M^{(2)}$ given this state $\overline{s_A}^{(0)}$.

Formally, let (Z_0, Y) be correlated random variables corresponding to computing $(Z_0, Y) \coloneqq (s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$. For $j \in [q]$, define Z_j as the random variable corresponding to sampling

$$Z_j \coloneqq \overline{s_{A,j}} \leftarrow P_{s_A^{(1)}|M^{(1)}}$$

where $P_{s_A^{(1)}|M^{(1)}}$ is the marginal distribution of Alice's state $s_A^{(1)}$ induced by $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$ conditioned on $M^{(1)}$. Note that this is an inefficient process. Let $f = \mathsf{receive}_B^{(1)}$, and $X = s_B^{(1)} \leftarrow f(Y)$.

By Lemma 7.2, there exists an (inefficiently computable) index $i \in [q]$ such that

$$\mathbf{I}(X; Z_0 | Z_1, \cdots, Z_i) \leq \frac{\mathbf{H}(X)}{q+1}. \tag{5}$$

We define $\overline{\mathsf{send}}_B^{(1)}$, $\overline{\mathsf{receive}}_A^{(1)}$ as follows:

- $\overline{\mathsf{send}}_B^{(1)}$: Compute $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$. Compute an index i such that Eq. (5) holds.
  For $j \in [i]$, given $M^{(1)}$, sample $\overline{s_{A,j}}$ as

  $$\overline{s_{A,j}} \leftarrow Q_{s_A^{(1)}|M^{(1)}}$$

  where $Q_{s_A^{(1)}|M^{(1)}}$ is the marginal distribution of $s_A^{(1)}$ induced by $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$, conditioned on $M^{(1)}$. Note that both computing i and sampling from $Q_{s_A^{(1)}|M^{(1)}}$ are inefficient.

  Compute $s_B^{(1)} \leftarrow \mathsf{receive}_B^{(1)}(M^{(1)})$, $(s_B^{(2)}, M^{(2)}) \leftarrow \mathsf{send}_B^{(2)}(s_B^{(1)})$, and output:

  $$\overline{s_B}^{(1)} = s_B^{(2)}, \quad \overline{M}^{(1)} = (\{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)}).$$

- $\overline{\mathsf{receive}}_A^{(1)}(\overline{M}^{(1)})$: On input $\overline{M}^{(1)} = (\{\overline{s_{A,j}}\}_{j\in[q]}, M^{(2)})$, sample $\overline{s_A}^{(0)}$ as

  $$\overline{s_A}^{(0)} \leftarrow P_{s_A^{(1)}|\{\overline{s_{A,j}}\}_{j\in[i]}}$$

  where $Q_{s_A^{(1)}|\{\overline{s_{A,j}}\}_{j\in[i]}}$ is the marginal distribution of $s_A^{(1)}$ conditioned on $\{\overline{s_{A,j}}\}_{j\in[i]}$ induced by the following process: sample $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$, sample for $j \in [i]$, $\overline{s_{A,j}} \leftarrow P_{s_A^{(1)}|M^{(1)}}$, and output $(s_A^{(1)}, \{\overline{s_{A,j}}\}_{j\in[q]})$. Note that such a sampling is inefficient.
  Output:

  $$\overline{s_A}^{(1)} \leftarrow \mathsf{receive}_A^{(2)}(\overline{s_A}^{(0)}, M^{(2)})$$

Next, we argue that the distribution of users states after round two of Π is statistically close to the distribution of states after round one of $\overline{\Pi}$.

Claim 7.5. *We have*

$$\mathbf{SD}\left((s_A^{(2)}, s_B^{(2)}, \{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)}), \quad (\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, \{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)})\right) \leq \sqrt{\frac{n}{2(q+1)}}.$$

Proof. Let $X = s_B^{(1)}$, and $Z_0 = s_A^{(1)}$. For $j \in [q]$, let $Z_j = \overline{s_{A,j}}$. By Lemma 7.3, we have:

$$\mathbf{SD}\left((s_B^{(1)}, s_A^{(1)}, \{Z_j\}_{j\le i}),\ (s_B^{(1)}, \overline{s_A}^{(0)}, \{Z_j\}_{j\le i})\right) \le \sqrt{\frac{n}{2(q+1)}}.$$

noting that X has length at most n, so that $\mathbf{H}(X) \le n$.

We observe that $(s_A^{(2)}, s_B^{(2)}, M^{(2)}), (\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, M^{(2)})$ are the respective output of some randomized function g on input $(s_A^{(1)}, s_B^{(1)})$ and $(\overline{s_A}^{(0)}, s_B^{(1)})$, where g computes the second round of Π. Namely, $g(s_A, s_B)$ computes $(s'_B, M') \leftarrow \mathsf{send}_B^{(2)}(s_B)$, $s'_A \leftarrow \mathsf{receive}_A^{(2)}(s_A, M')$, and outputs (s'_A, s'_B, M'). We therefore obtain:

$$\mathbf{SD}\left((s_A^{(2)}, s_B^{(2)}, \{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)}),\quad (\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, \{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)})\right) \le \sqrt{\frac{n}{2(q+1)}}.$$

□

We now argue that $\overline{\Pi}$ is correct and secure as long as Π is.

Suppose Π satisfies δ-correctness. Because the respective outputs of Alice and Bob $s_A^{(R)}$, $s_B^{(R)}$ in Π are randomized functions of $(s_A^{(2)}, s_B^{(2)})$, Claim 7.5 implies that

$$\Pr[\overline{s_A}^{(R)} = \overline{s_B}^{(R)}] \ge 1/2 + \delta - \sqrt{\frac{n}{2(q+1)}}, \tag{7}$$

that is, $\overline{\Pi}$ satisfies $\overline{\delta}$-correctness with $\overline{\delta} \ge \delta - \sqrt{\frac{n}{2(q+1)}}$.

Next, suppose $q \le \lfloor \frac{m}{n} \rfloor$. We show that (m, ε)-security of Π implies $(m, \overline{\varepsilon})$-security for $\overline{\Pi}$, with

$$\overline{\varepsilon} \le \varepsilon + \sqrt{\frac{n}{2(q+1)}}. \tag{8}$$

Let $\overline{\mathsf{Adv}}$ be an unbounded processing model adversary against $\overline{\Pi}$ with storage m and advantage $\overline{\varepsilon}$. Define:

- $\mathsf{Adv}^{(1)}(M^{(1)})$: Compute $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$. Compute as in $\overline{\mathsf{send}}_B^{(1)}$ an index i such that Eq. (5) holds.

  For $j \in [i]$, given $M^{(1)}$, sample $\overline{s_{A,j}}$ as

  $$\overline{s_{A,j}} \leftarrow Q_{s_A^{(1)}|M^{(1)}}$$

  where $Q_{s_A^{(1)}|M^{(1)}}$ is the marginal distribution of $s_A^{(1)}$ induced by $(s_A^{(1)}, M^{(1)}) \leftarrow \mathsf{send}_A^{(1)}$ conditioned on $M^{(1)}$, and output

  $$s_E^{(1)} = \{\overline{s_{A,j}}\}_{j\in[i]}.$$

  Note that both computing i and sampling $\overline{s_{A,j}}$ are inefficient.

- $\mathsf{Adv}^{(i)} = \overline{\mathsf{Adv}}^{(i-1)}$ for all $i \ge 2$.

In particular, $\mathsf{Adv}^{(2)}$ is given as input $(s_E^{(1)} = \{\overline{s_{A,j}}\}_{j\in[i]}, M^{(2)})$, and can therefore run $\overline{\mathsf{Adv}}^{(1)}(\overline{M}^{(1)})$ as it can compute $\overline{M}^{(1)} = (\{\overline{s_{A,j}}\}_{j\in[q]}, M^{(2)})$. Now, $M^{(2)}$ and all the further inputs to $\mathsf{Adv}^{(i)}$ and $\overline{\mathsf{Adv}}^{(i-1)}$ are randomized functions of $(s_A^{(2)}, s_B^{(2)}, \overline{s_{A,j}}_{j\in[i]}, M^{(2)})$ (in the case of Adv) or $(\overline{s_A}^{(1)}, \overline{s_B}^{(1)}, \overline{s_{A,j}}_{j\in[i]}, M^{(2)})$ (in the case of $\overline{\mathsf{Adv}}$). In particular, by Claim 7.5, the statistical distance of the outputs of Adv and $\overline{\mathsf{Adv}}$ is at most $\sqrt{\frac{n}{2(q+1)}}$, so that the advantage of Adv is at least $\varepsilon \ge \overline{\varepsilon} - \sqrt{\frac{n}{2(q+1)}}$.

Moreover, Adv is an adversary in the unbounded processing model with storage $\max(m, nq)$ as $\overline{\mathsf{Adv}}$ is m-bounded and $|\overline{s_{A,j}}| \le n$ for $j \in [i]$. Therefore, if $q \le \lfloor \frac{m}{n} \rfloor$, this induces an adversary Adv with storage m.

Applying the transformation. Set $q = \lfloor \frac{m}{2n} \rfloor$. We iteratively apply this transformation up to R times to the base protocol Π, until obtaining a 0-round protocol $\widetilde{\Pi}$, namely, where Alice and Bob do not communicate, which satisfies $\widetilde{\delta}$-correctness and $(m, \widetilde{\varepsilon})$-security. As, for a 0-round protocol, an adversary can always run Bob's algorithm to guess Alice's bit, this implies $\widetilde{\varepsilon} \geq \widetilde{\delta}$.

Denoting $t \leq R$ the number of times Case 2 occurred in the (up to) R transformations, namely the number of rounds of Π with message sent of length greater than m, the degradation for correctness (Eq. (7)) and security (Eq. (8)) of $\overline{\Pi}$ gives

$$\varepsilon + t \cdot \sqrt{\frac{n}{2(q+1)}} \geq \widetilde{\varepsilon} \geq \widetilde{\delta} \geq \delta - t \cdot \sqrt{\frac{n}{2(q+1)}},$$

so that

$$2t \cdot \sqrt{\frac{n}{2(q+1)}} \geq \delta - \varepsilon,$$

where $\delta - \varepsilon > 0$ is a constant by assumption on Π. As t is upper-bounded by the maximum number of rounds R of Π, and recalling that $q = \lfloor \frac{m}{2n} \rfloor$, this gives:

$$R \geq \Omega\left(\frac{\sqrt{m}}{n}\right).$$

Furthermore, every application of Case 2 decreases the total communication of the resulting $\overline{\Pi}$ by at least $m - nq \geq m/2$. As Π has total communication at most C, the number of applications Case 2 is therefore at most $t \leq \frac{C}{m/2}$, so that:

$$C \geq \Omega\left(\frac{m^{3/2}}{n}\right),$$

which concludes the proof.

□

Remark 7.6 (Variable length messages)**.** In our model (Section 7.1), and therefore Theorem 7.4, we assumed the lengths of the messages of protocols to have a fixed length.

First, we observe that our lower bound on the number of rounds applies even if there is no such restriction. Indeed, one can modify any protocol Π as follows: consider the maximal message length sent in Π (across all possible executions), and pad all the messages to have that maximal length. This does not affect the round complexity of the protocol, and therefore the lower bound on the number of rounds still applies.

Second, we note that our lower bound for communication also applies for variable length messages protocols if we restrict ourselves to the streaming model Section 4. Indeed, if the honest parties are streaming, we can assume that they send exactly one bit per round, turn by turn: this at most doubles the communication cost the initial protocol. Then, all messages are then by definition of fixed length, and Theorem 7.4 applies.

7.3 Model for a Lower Bound against Streaming Adversaries

In the unbounded processing model for our lower bounds of Sections 7.1 and 7.2, the only restriction, both for the honest parties and the adversary, is that their maintained state between rounds of communications has bounded size. In particular, they all can process messages from the protocol using potentially *unbounded* temporary memory, so long as they compress it to some limited amount of storage afterwards.

One natural setting left open, however, is the case where the adversary has bounded storage throughout the entire attack and only streaming access to messages sent. This makes the adversary *weaker* than in the model of Section 7.1, and it is not clear whether the subsequent lower bound extends. In this section, along with Section 7.3, we extend the lower bound of Sections 7.1 and 7.2 to such adversaries, albeit at the cost of slightly worse parameters. Another difference is that while Sections 7.1 and 7.2 also rule out protocols with unbounded processing *honest parties*, the model of this section and the subsequent lower bound in Section 7.3 only rule out *streaming* honest parties.

We first describe our model, that we call the *streaming model with CRS* . Honest parties send and receive messages in a *streaming manner*, using some bounded memory n, without any other restriction on the messages sent nor on the receiving algorithm. We will also consider adversaries which are similarly treating messages sent between the parties in a streaming manner using bounded memory $m > n$.

For comparison with Section 4, honest users are still more powerful, as having general streaming access to messages (as opposed to local access), and are not required to send uniformly random messages. In other words, honest parties neither have restriction (b) nor (c), but are still required now to be treating messages in a streaming manner. The adversary, however, is *weakened* to only have streaming access to the messages of the protocol, similar to honest users. Doing so makes the resulting lower bound stronger.

Compared with our previous lower bound (Sections 7.1 and 7.2), both the honest parties and the adversary are weaker, as they are now both streaming, as opposed to having unbounded preprocessing. As a result, the resulting lower bounds are technically incomparable. Still, we believe that restricting ourselves to protocols where honest parties use bounded memory during the whole execution of the protocol is an extremely natural setting for protocols in the bounded storage model.

Optionally, we will consider a streaming model *in the common reference string model*, where a common reference string is available prior to protocol execution. The CRS is used to (independently) derive starting states for the parties of the protocol. We consider these processes (namely, the CRS generation and the user state generation) to be performed by a trusted party, which can potentially run in memory larger than n. Honest parties do not require knowledge of the CRS to execute the remainder protocol, but adversaries do have access to the CRS to mount attacks. For simplicity, we will only consider CRS that directly fit in the adversary's memory.

We define key agreement (with one-bit output) in a very similar way as in Section 7.1: we refer to that section for our notion of δ-correctness and (m, ε)-security. We further consider security against *non-uniform attacks* where adversaries obtain some non-uniform advice that can be generated using unbounded memory.

7.4 Lower Bound against Streaming Adversaries

Theorem 7.7. *Let Π be a key agreement protocol in the streaming model (Section 7.3) with honest storage n, satisfying δ-correctness and (m, ε) security against non-uniform attacks, where $\delta - \varepsilon = \Omega(1)$. Suppose furthermore that for any execution of Π, the total communication between Alice and Bob is at most C and consists of at most R rounds. Then $C \geq \Omega\left(m \cdot \left(\frac{m}{n^2}\right)^{1/3}\right)$, and $R \geq \Omega\left(\left(\frac{m}{n^2}\right)^{1/3}\right)$.*

Proof. We define the following transformation, which takes a protocol Π in the streaming model *with CRS*, with number of rounds (upper bounded by) R, and outputs a protocol $\overline{\Pi}$ with at most $R - 1$ rounds, which preserves correctness and security of Π in the streaming model with CRS with some small loss in parameters. Looking ahead, such a transformation will directly extends starting with protocols without any CRS (but still resulting in a protocol in the CRS model).

Let us first fix some notation (similar to the notation of Section 7.2). Let Π be a key agreement protocol in the streaming model with CRS. Up to renaming Alice and Bob, Π has the following structure:

- $\mathsf{crsGen} \to \mathsf{crs}$: generates a CRS for Π. It will be convenient for us to separate crs into two parts, that we call crs_s and crs_M, respectively.
- $\mathsf{stateGen}_A(\mathsf{crs}) \to s_A$: on input crs, generates a state s_A (for Alice),
- $\mathsf{stateGen}_B(\mathsf{crs}) \to s_B$: on input crs, generates a state s_B (for Bob).

Every odd rounds $1 \leq k \leq R$, consists of Alice sending a message to the other to Bob:

$$(s_A^{(k)}, M^{(i)}) \leftarrow \mathsf{send}_A^{(k)}(s_A^{(i-1)})$$

and Bob computes

$$s_B^{(k)} \leftarrow \mathsf{receive}_B^{(k)}(s_B^{(k-1)}, M^{(k)}),$$

and vice-versa for even rounds k, where Bob sends the message.

The new protocol $\overline{\Pi}$ is defined as follows. Let i, q be integers such that $i \leq q$, to be determined later. We will now simply denote by M the first message sent in Π (by Alice to Bob).

We distinguish two cases, depending on whether the first message M sent in Π is larger than m.

Case 1: $|\mathsf{crs}_M| + |M| \leq m/2$**.** This case is extremely similar to the associated case in the proof of Theorem 7.4, the main difference being that the first message M is now moved in the new CRS as opposed to generated by Bob in $\overline{\Pi}$.

In more details, we define $\overline{\Pi}$ as follows:

- $\overline{\mathsf{crsGen}}$: Compute $\mathsf{crs} = (\mathsf{crs}_s, \mathsf{crs}_M) \leftarrow \mathsf{crsGen}$. Compute $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$, and $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$. Set $\overline{\mathsf{crs}}_s = \mathsf{crs}_s$, $\overline{\mathsf{crs}}_M = (\mathsf{crs}_M \| M)$, and output
$$\overline{\mathsf{crs}} = (\overline{\mathsf{crs}}_s, \overline{\mathsf{crs}}_M).$$
- $\overline{\mathsf{stateGen}}_B(\overline{\mathsf{crs}})$: On input $\overline{\mathsf{crs}} = (\mathsf{crs}_s, (\mathsf{crs}_M \| M))$, set $\mathsf{crs} = (\mathsf{crs}_s, \mathsf{crs}_M)$. Generate $s_B \leftarrow \mathsf{stateGen}_B(\mathsf{crs})$, and compute $s'_B \leftarrow \mathsf{receive}_B(s_B, M)$. Output s'_B.
- $\overline{\mathsf{stateGen}}_A(\overline{\mathsf{crs}})$: On input $\overline{\mathsf{crs}} = (\mathsf{crs}_s, (\mathsf{crs}_M \| M))$, set $\mathsf{crs} = (\mathsf{crs}_s, \mathsf{crs}_M)$. Generate $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$, and compute $(s'_A, M') \leftarrow \mathsf{send}_A(s_A)$. If $M = M'$, output s'_A, otherwise repeat.
- For appropriate $k \leq (R-1)/2$, $\overline{\mathsf{send}}_B^{(2k+1)} = \mathsf{send}_B^{(2k+2)}$, $\overline{\mathsf{receive}}_A^{(2k+1)} = \mathsf{receive}_A^{(2k+2)}$, $\overline{\mathsf{send}}_A^{(2k+2)} = \mathsf{send}_A^{(2k+3)}$ $\overline{\mathsf{receive}}_A^{(2k+2)} = \mathsf{receive}_A^{(2k+3)}$.

Case 2: $|\mathsf{crs}_M| + |M| > m/2$**.**

- $\overline{\mathsf{crsGen}} \to \overline{\mathsf{crs}}$: generate $\mathsf{crs} \leftarrow \mathsf{crsGen}$, along with $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$ and $s_{B,j} \leftarrow \mathsf{stateGen}_B(\mathsf{crs})$ for $j \leq i$. Compute $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$ and, for $j \leq i$: $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. Output:
$$\overline{\mathsf{crs}} = \left(\overline{\mathsf{crs}}_s = \left(s'_{B,1}, \ldots, s'_{B,i}\right), \quad \overline{\mathsf{crs}}_M = \emptyset\right).$$
- $\overline{\mathsf{stateGen}}_A(\overline{\mathsf{crs}}) \to \overline{s_A}$: on input $\overline{\mathsf{crs}}$, generate $\mathsf{crs} \leftarrow \mathsf{crsGen}$, along with $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$ and $s_{B,j} \leftarrow \mathsf{stateGen}_B(\mathsf{crs})$ for $j \leq i$. Compute $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$ and, for $j \leq i$: $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. If $\overline{\mathsf{crs}} = \left(s'_{B,1}, \ldots, s'_{B,i}\right)$, output
$$\overline{s_A} = s'_A.$$
- $\overline{\mathsf{stateGen}}_B(o\mathsf{crs}) \to \overline{s_B}$: on input $\overline{\mathsf{crs}}$, generate $\mathsf{crs} \leftarrow \mathsf{crsGen}$, along with $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$ and $s_{B,j} \leftarrow \mathsf{stateGen}_B(\mathsf{crs})$ for $j \leq i+1$. Compute $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$ and, for $j \leq i+1$: $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. If $\overline{\mathsf{crs}} = \left(s'_{B,1}, \ldots, s'_{B,i}\right)$, output
$$\overline{s_B} = s'_{B,i+1}.$$
- For even $2 \leq k \leq R-1$: $\overline{\mathsf{send}}_A^{(k)}(s'_A) = \mathsf{send}_A^{(k+1)}(s'_A)$, $\overline{\mathsf{receive}}_B^{(k)}(s'_B, M) = \mathsf{receive}_B^{(k+1)}(s'_B, M)$,
- For odd $1 \leq k \leq R-1$: $\overline{\mathsf{send}}_B^{(k)}(s'_B) = \mathsf{send}_B^{(k+1)}(s'_B)$, $\overline{\mathsf{receive}}_A^{(k)}(s'_A, M) = \mathsf{receive}_B^{(k+1)}(s'_A, M)$.

We now analyze the cost of the transformation given the two cases. To do so, consider the following algorithms $\mathsf{Samp}, \overline{\mathsf{Samp}}$, which combines all of the algorithms $\mathsf{crsGen}, \mathsf{stateGen}_A, \mathsf{stateGen}_B$.

- Samp: sample $\mathsf{crs} \leftarrow \mathsf{crsGen}$. Sample $s_A \leftarrow \mathsf{stateGen}_A(\mathsf{crs})$, and, for $j \leq i+1$, $s_{B,j} \leftarrow \mathsf{stateGen}_B(\mathsf{crs})$. Output:
$$\left(\mathsf{crs}, s_A, s_{B,1}, \ldots, s_{B,i+1}\right).$$

- $\overline{\mathsf{Samp}}$: sample $\overline{\mathsf{crs}} \leftarrow \overline{\mathsf{crsGen}}$. Sample $\overline{s}_B \leftarrow \overline{\mathsf{stateGen}}_B(\overline{\mathsf{crs}})$, and, for $j \le i+1$, $\overline{s}_{A,j} \leftarrow \overline{\mathsf{stateGen}}_A(\overline{\mathsf{crs}})$. Output:
$$(\overline{\mathsf{crs}}, \overline{s}_B, \overline{s}_{A,1}, \ldots, \overline{s}_{A,i+1}) .$$

The distribution of the states and transcripts in $\overline{\Pi}$ after Case 1 occurs is identically distributed as in Π. We now argue that the transformation in Case 2 only slightly degrades security and correctness.

Claim 7.8. *Suppose* Samp *is computable by a streaming algorithm using memory K_1, and that Π is δ-correct and (K_2, ε)-secure with $K_2 \ge K_1 + in$. Suppose that Case 2 occurs. Then $\overline{\Pi}$ is $\left(\delta - \sqrt{\frac{n}{q+1}}\right)$-correct and $\left(K_2 - in, \varepsilon - \sqrt{\frac{n}{q+1}}\right)$-secure.*

Proof. Let $\overline{\mathsf{Adv}}$ be a (streaming) attack on $\overline{\Pi}$ with memory K_3 and advantage $\varepsilon' = \varepsilon - \sqrt{\frac{n}{q+1}}$. We build a (streaming) attack Adv on Π as follows.

- Adv :
  1. On input crs, Adv calls Samp to generate $(\mathsf{crs}', s_A, s_{B,1}, \ldots, s_{B,i+1})$. While $\mathsf{crs}' \neq \mathsf{crs}$, repeat the above. Store $(s_{B,1}, \ldots, s_{B,i})$.
  2. When receiving the first message M for Π in a streaming manner, call in parallel i copies of $\mathsf{receive}_B$:
  $$\forall j \in [i], \quad s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M),$$
  and set:
  $$\overline{\mathsf{crs}} := (s'_{B,1}, \ldots, s'_{B,i}).$$
  3. Call $\overline{\mathsf{Adv}}$, with CRS $\overline{\mathsf{crs}}$ and with every subsequent messages relayed in a streaming way from Π, namely the jth streamed message of $\overline{\Pi}$ is the $(j+1)$th streamed message of Π, for $j \le R-1$. Relay the output of $\overline{\mathsf{Adv}}$ as the final output.

Let us first analyze the properties of Adv. Let K_3 be the memory cost of $\overline{\mathsf{Adv}}$. Calling Samp in Step 1 costs memory $K_1 + in$. Step 2 costs memory in, namely the cost of running i copies of Bob in parallel. Step 3 costs memory $K_3 + in$. So overall, Adv can be run with memory $\max(K_1, K_3) + in$. Given that $K_2 \ge K_1 + in$, this is a contradiction as long as $K_3 \le K_2 - in$. Furthermore, Adv processes all the messages sent by Π in a streaming manner, as $\overline{\mathsf{Adv}}$ does as well.

Next, we argue that the distribution $(\overline{\mathsf{crs}}, s'_A, s'_B)$ of Alice and Bob in Π after receiving the first message M (where s'_A is defined implicitly), along with $\overline{\mathsf{crs}}$ generated by Adv (Step 2 above), is within statistical distance $\sqrt{\frac{n}{q+1}}$ to $(\overline{\mathsf{crs}}, \overline{s}_A, \overline{s}_B)$ in $\overline{\Pi}$ where $\overline{s}_B$ is sampled conditioned on $\overline{\mathsf{crs}}$, and $\overline{s}_A$ is implicitly sampled conditioned on crs and M. First, note that the distributions $(\overline{\mathsf{crs}}, s'_A)$ and $(\overline{\mathsf{crs}}, \overline{s}_A)$ are identical by definition of $\overline{\Pi}$. Bounding the distance between the two distributions then follows similarly than in Section 7 using Lemma 7.3 and Claim 7.5, which define the index $i \in [q]$, and taking $Y = (\mathsf{crs}, M)$, $X = s'_A$ and $Z_j = s'_{B,j}$ for $j \le i$. This shows that if $\overline{\Pi}$ has correctness $\delta - \sqrt{\frac{n}{q+1}}$, and security $\varepsilon - \sqrt{\frac{n}{q+1}}$, then Π has correctness at least δ and security at least ε.

□

Next, we analyze the memory complexity of computing Samp. We do so by inheriting an efficient way of implementing $\overline{\mathsf{Samp}}$ given an efficient way to implement Samp.

Claim 7.9. *Suppose* Samp *is computable by a streaming algorithm using memory K_1. Then* $\overline{\mathsf{Samp}}$ *is computable by a streaming algorithm using memory*
$$\max\left((2i+2)n + \max(K_1, (i+2)n), \quad m/2 + (i+2)n\right).$$

Proof. Suppose Samp is computable using memory K_1. We provide an alternative way to sample from the distribution output by $\overline{\mathsf{Samp}}$ with low memory, depending on whether Case 1 or Case 2 occurs, as follows:

- Case 1: $|\mathsf{crs}_M| + |M| \leq m/2$. We build $\overline{\mathsf{Samp}}$ as follows:
  1. Compute $(\mathsf{crs}, s_A, (s_{B_1}, \ldots, s_{B,i+1})) \leftarrow \mathsf{Samp}$; parse $\mathsf{crs} = (\mathsf{crs}_s, \mathsf{crs}_M)$.
  2. Compute $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$ (in a streaming manner), and set $\overline{\mathsf{crs}}_s = \mathsf{crs}_s$, $\overline{\mathsf{crs}}_M = (\mathsf{crs}_M \| M)$.
  3. Compute, for $j \leq i+1$, $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$.
  4. Output $(\overline{\mathsf{crs}} = (\overline{\mathsf{crs}}_s, \overline{\mathsf{crs}}_M), s'_A, (s'_{B,1}, \ldots, s'_{B,i+1}))$.
- Case 2: $|\mathsf{crs}_M| + |M| > m/2$. We construct $\overline{\mathsf{Samp}}$ as follows:
  1. Compute $(\mathsf{crs}, s_A, (s_{B_1}, \ldots, s_{B,i+1}) \leftarrow \mathsf{Samp}$. Compute $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$, and, for all $j \in [i]$, $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. Store $\overline{\mathsf{crs}}_s = \left(s'_{B,1}, \ldots, s'_{B,i}\right)$ and set $\overline{\mathsf{crs}}_M = \emptyset$.
  2. For $\ell \in [q+1]$, do the following:
     - Sample fresh $(\mathsf{crs}, s_A, (s_{B_1}, \ldots, s_{B,i+1})) \leftarrow \mathsf{Samp}$. Compute fresh $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$, and, for all $j \in [i]$, $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. If for all $j \in [i], s'_{B,j} = \overline{\mathsf{crs}}_j$, output $\overline{s}_{A,\ell} = s'_A$. Otherwise repeat.
  3. Sample fresh $(\mathsf{crs}, s_A, (s_{B_1}, \ldots, s_{B,i+1})) \leftarrow \mathsf{Samp}$. Compute fresh $(s'_A, M) \leftarrow \mathsf{send}_A(s_A)$, and, for all $i \in [q+1]$, $s'_{B,j} \leftarrow \mathsf{receive}_B(s_{B,j}, M)$. If for all $j \in [i], s'_{B,j} = \overline{\mathsf{crs}}_j$, output $\overline{s}_B = s'_{B,i+1}$. Otherwise repeat.
  4. Output $(\overline{\mathsf{crs}}, (\overline{s}_{A,1}, \ldots, \overline{s}_{A,i+1}), \overline{s}_B)$.

In Case 1, the memory cost of $\overline{\mathsf{Samp}}$ is $\max(K_1, m/2 + (i+2)n)$, where $(i+2)n$ bounds the cost of implementing the send and $\mathsf{receive}$ (in a streaming manner) (Step 2,3) while updating $\overline{\mathsf{crs}}$ by storing M.

In Case 2, Step 1 uses memory $\max(K_1, (i+1)n)$. Every loop of Step 2 uses memory $in + \max(K_1, (i+1)n)$. Every loop of Step 3 uses memory $in + \max(K_1, (i+2)n)$. Noting that all the quantities above already take into account $\overline{\mathsf{crs}}$ output by $\overline{\mathsf{Samp}}$, the output additionally takes $(i+2)n$ memory.

Overall, $\overline{\mathsf{Samp}}$ can be implemented using memory $\max\left((2i+2)n + \max(K_1, (i+2)n), m/2 + (i+2)n\right)$. □

To finish the proof of Theorem 7.7, we apply the transformation repeatedly R times. Defining the intermediate protocols $\overline{\Pi}$ requires knowledge of the special index $i \in [q]$ (where q is still to be determined) given by Lemma 7.3 Claim 7.5 for application of the transformation; our attack receives these indices as non-uniform advice (which can be stored using memory $R \log q$).

Repeatedly applying Claim 7.9 gives that the sampling algorithm Samp associated with the final protocol without interaction can be implemented with memory $K_1 = \max(R(2q+2)n, m/2 + qn)$, using the fact that the initial R-round protocol Π has both an empty CRS and empty starting states for Alice and Bob.

Then, note that any protocol in the CRS without interaction has a streaming attack with memory n and guessing probability δ (where δ is the correctness of the protocol without interaction), which corresponds to either running the Alice or Bob algorithm. Therefore, applying Claim 7.8 repeatedly R times gives that there is a generic attack on Π with memory $\max(R(2q+2)n, m/2 + qn) + Rqn$ and guessing probability $\delta - R\sqrt{\frac{n}{q+1}}$. Therefore, for an adversary with memory m, setting

$$q = \Theta\left(\frac{m}{Rn}\right)$$

gives that the original Π can only be secure if:

$$R\sqrt{\frac{n}{q+1}} = \Omega\left(R^{3/2}\sqrt{\frac{n^2}{m}}\right) \geq \Omega(1),$$

that is

$$R \geq \Omega\left(\left(\frac{m}{n^2}\right)^{1/3}\right).$$

Turning to communication, applying the transformation with Case 1 incurs no overhead in the memory complexity of sample (up to taking a maximum of the cost with $m/2 + (q+2)n$) and no loss in advantage. Observe that the number of applications of Case 2 is bounded by $O(C/m)$: this is because the combined length of crs_M and the entire transcript decreases by at least $m/2$ at every application of the transformation with Case 2, and is left unchanged at every application with Case 1. This yields:

$$C \geq \Omega\left(m \cdot \left(\frac{m}{n^2}\right)^{1/3}\right).$$

□

References

[ADR02] 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.

[AF94] Yonatan Aumann and Uriel Feige. On message proof systems with known space verifiers. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 85–99. Springer, Heidelberg, August 1994.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 373–402. Springer, Heidelberg, August 2016.

[Cal09] Chris Calabro. *The exponential complexity of satisfiability problems*. PhD thesis, University of California, San Diego, USA, 2009.

[CCM98] 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.

[CM97] 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.

[Cré88] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 350–354. Springer, Heidelberg, August 1988.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[DHRS07] 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.

[Din01] 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.

[DM02] 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.

[DM04] 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.

[DORS08] 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.

[DR02] 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.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989.

[GRT18] 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.

[GZ19] 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.

[GZ21] Jiaxin Guan and Mark Zhandry. Disappearing cryptography in the bounded storage model. Cryptology ePrint Archive, Report 2021/406, 2021. https://eprint.iacr.org/2021/406.

[HCR02] 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.

[ILL89] 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.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[KRT17] 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.

[KRVZ11] Jesse Kamp, Anup Rao, Salil Vadhan, and David Zuckerman. Deterministic extractors for small-space sources. *Journal of Computer and System Sciences*, 77(1):191–220, 2011. Celebrating Karp's Kyoto Prize.

[Lu02] 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.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, January 1992.

[MST09] Tal Moran, Ronen Shaltiel, and Amnon Ta-Shma. Non-interactive timestamping in the bounded-storage model. *Journal of Cryptology*, 22(2):189–226, April 2009.

[Nis90] N. Nisan. Pseudorandom generators for space-bounded computations. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 204–212, New York, NY, USA, 1990. Association for Computing Machinery.

[NOVY93] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP can be based on general complexity assumptions (extended abstract). In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 196–214. Springer, Heidelberg, August 1993.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[Rab81] Michael O. Rabin. How to Exchange Secrets with Oblivious Transfer, 1981. Harvard Aiken Computational Laboratory TR-81.

[Raz16] 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.

[Raz17] 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.

[SPY92] Alfredo De Santis, Giuseppe Persiano, and Moti Yung. One-message statistical zero-knowledge proofs and space-bounded verifier. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, page 28–40, Berlin, Heidelberg, 1992. Springer-Verlag.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, January 2004.