Doubly-Affine Extractors, and their Applications

Yevgeniy Dodis* Kevin Yeo†

Abstract

In this work we challenge the common misconception that information-theoretic (IT) privacy is too impractical to be used in the real-world: we propose to build simple and *reusable* IT-encryption solutions whose only efficiency penalty (compared to computationally-secure schemes) comes from a large secret key size, which is often a rather minor inconvenience, as storage is cheap. In particular, our solutions are *stateless* and *locally computable at the optimal rate*, meaning that honest parties do not maintain state and read only (optimally) small portions of their large keys with every use.

Moreover, we also propose a novel architecture for outsourcing the storage of these long keys to a network of semi-trusted servers, trading the need to store large secrets with the assumption that it is hard to simultaneously compromise too many publicly accessible ad-hoc servers. Our architecture supports *everlasting privacy* and *post-application security* of the derived one-time keys, resolving two major limitations of a related model for outsourcing key storage, called bounded storage model.

Both of these results come from nearly optimal constructions of so called *doubly-affine extractors*: locally-computable, seeded extractors $\mathbf{Ext}(X, S)$ which are linear functions of X (for any fixed seed S), and protect against bounded affine leakage on X. This holds unconditionally, even if (a) affine leakage may *adaptively depend* on the extracted key $R = \mathbf{Ext}(X, S)$; and (b) the seed S is only *computationally* secure. Neither of properties are possible with general-leakage extractors.

1 Introduction

Information-theoretic (IT) security is very attractive as it enables provably secure schemes that resist advances in computational power, novel cryptanalysis, or the possibility of quantum computers. This is especially important for privacy applications, where huge amounts of encrypted communication are being stored and recorded, with the danger that all these communications could be decrypted years later. Unfortunately, the famous impossibility result of Shannon [29, 10] states that IT-secure schemes come at a price: the secret should be at least as large as the message. The traditional interpretation of this negative result is that one must settle for much weaker computational security, so as to make the problem of key distribution feasible.

1.1 Reusable IT-Encryption

As we observe, just because the secret key must be large does not make the system automatically impractical. In fact, since local storage is often cheap, a (necessarily) large secret key X might be a very reasonable price to pay for *unconditional security*, provided this is the *only* efficiency penalty when compared to computationally-secure schemes. For the purposes of this work, we will interpret this latter requirement by demanding that our solutions are simultaneously *stateless* and *locally-computable*.

Stateless. Our first requirement demands that parties keep no state beyond storing the original key X. This is crucial when there are many parties that cannot easily remain synchronized together. For example, this could be the case when the secret key is shared by $k \gg 2$ parties and each party cannot view all transmissions (possibly intentionally). It also removes out-of-sync errors as a common potential source of insecurity that cause multiples parties to accidentally reuse the same portion of the key. In particular,

*New York University, dodis@cs.nyu.edu, Yevgeniy Dodis was partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1815546, 2055578.

†Google and Columbia University, kwlyeo@google.com

statelessness rules out trivial solutions where parties slowly utilize consecutive (fresh) portions shared key X (e.g., as one-time pads) until X "runs out" that is infeasible with many parties.

Locally Computable. Our second requirement of local compatibility ensures that each concrete application of the scheme (both as sender and receiver) only accesses the long secret key X in very few locations. We call this number of locations p the *probe complexity*, and the ratio $\alpha = \frac{m}{p} \in [0; 1]$ the *locality* of a given solution. Requiring $p \ll |X|$ rules out elegant (stateless) solutions using so called $\mathcal{L}$-wise independent hash functions, because all conventional $\mathcal{L}$-wise independent hash functions read the entire long key X for every evaluation.

Rate of IT-encryption. We can now define our first motivating question more formally. As the "price" for being stateless and locally computable, we introduce a "waste" parameter $\beta > 0$, indicating that the total length of all the messages we wish to (statelessly) encrypt is bounded by $(1-\beta)|X|$. Given this "waste" $\beta > 0$ and message length m, our goal is to minimize the probe complexity p (and maximize locality α).

It is not hard to see (this follows from the more general observation of [31]) that $p \geq m/\beta$ (i.e., $\alpha \leq \beta$), which is indeed independent of the key length $|X|$. Intuitively, since in any stateless scheme up to $(1-\beta)$-fraction of X might have been already used to encrypt prior messages, one needs to sample on average $1/\beta$ bits of X to get an "unused" bit. We ask if this exact bound is tight, and optimal locality $\alpha \approx \beta$ can be achieved, perhaps up to an *sub-linear additive* loss (that we denote by $o(m)$ while omitting the security parameter λ from notation)?

Open Problem 1. *Design practical, stateless and reusable IT-secure encryption schemes of m-bit messages with n-bit key and probe complexity $p = m/\beta + o(m)$, where $(1-\beta)n$ is the upper bound on the total length of the encrypted messages.*

As one of our main results, we present the first affirmative solution to this open question, achieving $p = m/\beta + O(\lambda(m + \lambda))) = m/\beta + o(m)$, whenever message length $m = \omega(\lambda)$. Our scheme is quite practical. We present the exact expression with no hidden constants in Theorems 9 and 11, and demonstrate concrete improvements over prior state-of-the-art solutions [5, 4] in Section 6.

1.2 Delegating Storage

While solving Open Problem 1 means that long secrets is the *only* "price" for unconditional security (when compared to computationally-secure schemes), we would like to do even better, and delegate the storage of these large keys (to a cloud provider as an example). Since this clearly overcomes the Shannon's impossibility result, we require strong trust assumptions with the storage server.

To achieve this ambitious goal, our encryption scheme for message M will compute a ciphertext $C = (S, R \oplus M)$, where $R = \mathbf{Ext}(X, S)$, $\mathbf{Ext}$ is a carefully chosen locally computable extractor [31] and S is a fresh random seed chosen by the sender. At a high level, we would like the server to store X instead of the users, and only the honest sender/receiver(s) be able to retrieve the correct one-time pads $R = \mathbf{Ext}(X, S)$ from the server.

Basic Architecture. As the first attempt, imagine some *virtual server* $\mathbf{T}$ will choose the long random string X, for the altruistic purpose of helping users utilize "reusable IT-security". In our final architecture, the virtual server will be emulated by a network of semi-trusted servers under (still strong, but) more plausible trust and communication assumptions. But, for now, we will think of $\mathbf{T}$ as not only stateless and locally computable, but also *fully trusted, incorruptible, and having private channels to any user who contacts it*.[1]

Since $\mathbf{T}$ might not even know its user base, we assume that $\mathbf{T}$ is truly public, and does not perform any explicit authentication. Hence, anybody, *including the attacker* $\mathbf{A}$, can send a seed S to $\mathbf{T}$, and get back the value $R = \mathbf{Ext}(X, S)$. However, we assume that the total length of one-time pads obtained by the attacker

[1] These strong assumptions will be substantially weakened, once we replace the virtual server by several "real" servers. This is similar in spirit to IT secret sharing [28] and MPC literature [8], which assume private channels to uncorrupted servers. However, we will do even better, as there are no "consistency requirements" for generating randomness. For example, our servers will be ad hoc, and do not communicate (or possibly even know!) about each other. See Section 5.2.

is bounded by $(1 - \beta)|X$ that may be achieved by ensuring that servers stop responding after a certain number of requests.

Sharing the seed. In the setting of Section 1.1, the seed S used to derive $R = \mathbf{Ext}(X, S)$ was sent in the clear, as part of the ciphertext $C = (S, R \oplus M)$. This was fine since access to X was given only to the honest uses, and not the attacker $\mathbf{A}$. Now, however, S cannot be sent in the clear, as then $\mathbf{A}$ can directly query $\mathbf{T}$ on S, just as the honest recipient would.

Now, we need some mechanism how only the authorized parties learn S. Moreover, they need to do this repeatedly for every new message M. This looks like a chicken-and-egg problem, as transmitting fresh seeds while protecting their privacy *unconditionally* is as hard as solving the reusable IT-encryption. To resolve this dilemma, we would like for this idea to work even if the seed S is shared using some *computationally-secure* mechanism. As natural examples, parties can (a) run fresh Diffie-Hellman key agreement to generate S; or (b) share a short symmetric key k once, and have the sender computationally encrypt seed S using k; or (c) in the public-key setting, computationally encrypt S using the receiver's public key. In all of these scenarios, we want to claim that M is remains private *forever*, as long as the privacy of S is not broken *during the lifetime of server* $\mathbf{T}$. We call this notion of privacy *everlasting privacy*, following the terminology of [2].

Allowing adversarial seeds. In the setting of Section 1.1, the attacker could only observe prior extractor outputs $\mathbf{Ext}(X, S_i)$ on *honestly chosen*, random seeds S_i. In contrast, the current setting enables the adversary $\mathbf{A}$ to learn outputs $\mathbf{Ext}(X, S_i)$ on *adversarial* seeds S_i. Moreover, the seed S_i could be chosen effectively depending on the "challenge one-time pad" $R = \mathbf{Ext}(X, S)$. With respect to the security game, if $\mathbf{A}$ observes "challenge encryption" $P = R \oplus M$ and knows the message $M \in \{V_0, V_1\}$. Therefore, $\mathbf{A}$ can deduce $R \in \{P \oplus V_0, P \oplus V_1\}$ without knowing the "challenge seed" S. Hence we want our architecture to be *post-application secure* [12]. That is, it should be safe to let the attacker interact with the servers, even after the honest parties use the challenge encryption P (either large portions of P or all of P may be leaked to $\mathbf{A}$).

To summarize the preceding discussion, to soundly realize our basic architecture for key delegation, the chosen extractor $\mathbf{Ext}(X, S)$ must be (a) everlastingly private with computationally-secure seeds; and (b) post-application secure.

Open Problem 2. *Design IT-secure, locally computable extractor* $\mathbf{Ext}$ *for the sound implementation of the basic delegation architecture. In particular,* $\mathbf{Ext}$ *should be post-application secure and support computationally-secure seeds.*

In this work we design the first architecture which supports these guarantees. Moreover, we show how to distribute the virtual server among several servers, only some of which can be trusted. Our solution is (a) *ad-hoc*, meaning servers do not need to know about or coordinate with each other, and (b) almost fully *stateless*, meaning the servers need to maintain minimal-to-no state except to ensure that adversaries view a bounded amount of leakage.

1.3 Locally Computable Extractors to the Rescue?

Our first hope is that standard locally computable extractors (LCEs), as originally formalized in the context of Bounded Storage Model (BSM) encryption [31], would be precisely what we need to solve Open Problems 1 and 2. Such an extractor $\mathbf{Ext}(X, S)$ is guaranteed to work on a uniform, n-bit key X, even despite the attacker obtaining up $(1 - \beta)n$ leakage bits $L = f(X)$, for any function f of attacker's choice. In particular, by modeling previous extractor outputs $\mathbf{Ext}(X, S_i)$ as leakage on X, the resulting scheme appears to be suffice for our purposes of building reusable IT-encryption. Unfortunately, general LCEs do not work for either of our questions, both quantitatively and qualitatively.

Suboptimal Rate. While the best upper bound [31] on the locality a of LCEs is the same $a \leq \beta$ as we have in our simpler setting, we currently do not have schemes matching this bound. Thus, this approach will not help us resolve Open Problem 1. The best known scheme of [5] achieves $a_0(\beta) \approx -\log_2(1 - h_2^{-1}(\beta))$,

where $h_2(z) = -z \log_2(z) - (1 - z) \log_2(1 - z)$ is the binary entropy function, and $h^{-1}(\beta)$ takes the smaller of two possible inverses. It is easy to see that $a_0(\beta) \ll \beta$ for all β even slightly bounded away from 0 and 1. For example, $a_0(0.5) = 0.168 \ll 0.5$ and $a_0(0.1) = 0.019 \quad 0.1$. In fact, [5] proved the optimality of their particular proof technique based on the so called "subkey prediction lemma" [1, 5, 4] (although it is conceivable a better non-asymptotic LCE bound will be found with a different technique; e.g., those from [24, 31]).

Computationally Secure Seeds. Just like in our setting from Section 1.2, supporting computationally secure seeds would be a huge win for the BSM setting. Surprisingly, several works [19, 14] showed that the BSM (and LCE) is too general to handle computationally-secure seeds. First, everlasting privacy in the BSM may not be reduced in a black-box manner to any computational assumption [19]. Second, there are explicit examples of computationally secure mechanisms to generate S which would break BSM security for *any* LCE. For example, if the attacker knows encryption Z of S under some fully homomorphic encryption (FHE), this still leaves S computationally secure. Yet, the attacker can *efficiently* evaluate $\mathbf{Ext}(X, .)$ inside the ciphertext as its compact leakage function $L = f(X, Z)$, learning FHE of the one-time pad $R = \mathbf{Ext}(X, S)$. When the attacker later becomes unbounded, it can break the FHE, and learn R.

Fortunately, this attack on BSM does not translate to our setting in the delegated storage setting. The servers will refuse to homomorphically evaluate the given ciphertext, as this does not correspond to evaluating $\mathbf{Ext}(X, S_i)$ on some seed S_i. However, it shows that we need a more refined approach to *provably* achieve everlasting privacy.

Post-Application Security. In the traditional BSM setting, the leakage $L = f(X)$ may only depend on the random source X. For post-application security, however, we allow the leakage function to also depend on R; that is, $L = f(X, R)$. Unfortunately, general LCEs cannot be post-application secure, at least for the interesting setting when $|S| < |R|$.[2] To see this, consider a boolean leakage function $f(X, R)$ which is 1 if and only there exists some seed S which yields $R = \mathbf{Ext}(X, S)$. When $|S| < |R|$, such $f(X, R)$ will always be true with "real" R, but almost never true with random R (see Appendix B).

Once again, this attack does not translate to our setting, as such leakage does not correspond to evaluating $\mathbf{Ext}(X, S_i)$ on some seed S_i. However, it shows that we need a more refined approach to *provably* achieve post-application-security.

1.4 Doubly-Affine Extractors

To overcome the limitations of existing LCEs, we notice that, for both of our application scenarios, the leakage of X consists of values $\mathbf{Ext}(X, S_i)$ for various seeds S_i. Hence, we can try to design efficient extractors which are *only secure against leakage of their own outputs*. Moving forward, we will denote LCEs with this property as simply extractors. With this approach, we will resolve both Open Problems 1 and 2.

Linearity. We observe that most existing LCEs [2, 14, 22, 31] are *linear (affine) functions* of X (for any fixed S). For our settings, an affine extractor only needs to be secure against what is called *affine* leakage functions resulting from previous extractor outputs. Such extractors are called *affine extractors* [15], and certainly appear easier than "general leakage" extractors. However, until now affine extractors have only been considered in the seedless setting. As a result, such seedless affine extractors can be neither locally computable, nor linear.

In this work, we initiate the study of *seeded* affine extractors which are both locally computable and linear functions of the source X. For conciseness, we will call such (seeded, locally computable) extractors *doubly-affine*, where "affine" now refers to both the leakage and the extractor itself.

Our Model and its Advantages. The formal security game for doubly-affine extractors is given in Figure 1. The attack is split in two stage. In this first state, the attacker $\mathbf{A}_1$ is given challenge output R (either $\mathbf{Ext}(X, S)$ or uniform), and can make up to $\pounds = (1 \_ \beta)\mathsf{n}$ adaptive affine leakage queries. Since these queries are adversarial and our extractor is linear, they can model extractor outputs $\mathbf{Ext}(X, S^*)$ on adversarial seeds S^*. Thus, *post-application security is built into the definition.*

[2] Setting $|S| \geq |R|$ is uninteresting for BSM, as parties can then use S instead of R.

In the second stage, the attacker $\mathbf{A}_2$ is given the seed S, but cannot make any more leakage queries. This models everlasting privacy, although not necessarily with respect to computationally secure seeds (yet). To model the latter concern, we augment the basic definition in Figure 1 to a seemingly more advanced setting in Figure 2, where some abstract seed-generating procedure $\Sigma(S^t)$ outputs the extractor seed S and the side-information Z. This side information Z is given to $\mathbf{A}_1$ to help making its affine queries, and the entire seed S^t used by the computationally secure seed generator is given to the second-stage attacker $\mathbf{A}_2$. We require that the game in Figure 2 is secure against any computationally bounded $\mathbf{A}_1$ and *unbounded* $\mathbf{A}_2$, as long as S remains pseudorandom to $\mathbf{A}_1$ given Z.

For example, to model Diffie-Hellman key exchange, we set $S^t = (a, b)$, $Z = (g^a, g^b)$ and $S = \mathbf{Prg}(g^{ab})$, for some pseudorandom generator **Prg**. The fact that we give the values a and b to $\mathbf{A}_2$ now accurately models the fact that an unbounded attacker can break discrete log of g^a and g^b eventually, and thus learn more information than simply recovering the extractor seed $S = \mathbf{Prg}(g^{ab})$.

Fortunately, unlike the setting of general LCEs, we show that *any* doubly-affine extractor satisfying a the simpler definition in Figure 1 will also automatically satisfy the definition in Figure 2. Thus, our basic definition in Figure 1 also covers *everlasting security against computationally secure seeds*.

Parameters and Efficiency. Last, but not the least, restricting to linear leakage allows to solve our nearly optimal probe complexity $\mathsf{p} \approx \mathsf{m}/\beta$, settling Open Problem 1 in the affirmative. As we show in Section 6, our construction is also concretely efficient, making it attractive for real-world applications where IT-security matters.

As an additional advantage, it gracefully extends to a more refined model of local computability, where in addition to the probe complexity p, we also wish to minimize the number of non-contiguous memory c blocks one needs to read the required p bits. We call this parameter c *cache complexity*, as it roughly corresponds to the number of cache misses to read c non-contiguous regions of memory. Unlike probe complexity, cache complexity does not have to grow with the number of extracted bits m, and can be as small as $\mathsf{c} = O(\lambda)$, where λ is the security parameter. Indeed, our main construction generally gives $\mathsf{p} = \frac{\mathsf{m}}{\beta} \cdot (1 + \frac{O(\lambda)}{\mathsf{c}})$.

To summarize, doubly-affine extractors provide all the properties we need to simultaneously resolve Open Problems 1 and 2.

1.5 Our Constructions and Techniques

Our doubly-affine extractor follow the sample-then-extract approach introduced by Vadhan [31] for LCE. The first sampling step selects a subset I of p bits of X, denoted by $Y = X_{|I}$, and the second step applies a non-local extractor to Y to produce the final output R. In our work, we improve the parameters for both steps, when the leakage is restricted to be affine.

Sampler Improvement. The two samplers we analyze were already considered by prior work on LCEs [23, 24, 31, 5], but our work presents new, improved analyses for the case of affine leakage. As our key insight, we prove (see Theorem 2) that the optimal affine leakage strategy against *any* sampler is to select some *physical* $(1 - \beta)\mathsf{n}$ bits of X. In other words, the best adversarial strategy is to simply try and guess as many locations of the sampled bits as possible.

This result is surprising for two reasons. First, the same equivalence is false for general-leakage samplers: [5] shows that even simple (but highly non-linear) leakage functions may greatly outperform physical-bit leakage. Second, the equivalence is false for the overall setting of doubly-affine extractors. Ignoring locality, for example, parity of all n bits of X is trivially secure against bounded leakage of up to $(\mathsf{n} - 1)$ bits, but is trivially insecure against a single-bit of affine leakage.

Once we reduce to physical-bit leakage, a simple Chernoff bound easily implies that the number of "non-leaked" physical bits in a "random-enough" p-bit sample of X is highly concentrated around its expected value $\beta\mathsf{p}$ — a conclusion which would seem highly non-obvious without our equivalence.

Extractor Improvement. Once we know that the sample Y has entropy of approximately $\beta\mathsf{p}$ in the adversary's view, we can apply any non-local linear $\mathbf{Ext}^t$ to extract $\mathsf{m} \approx \beta\mathsf{p}$ bits from Y. For *concrete security*, especially for small values of m, we still want to optimize the *entropy loss* $(\beta\mathsf{p} - \mathsf{m})$. Using extractors for

general leakage, this entropy loss is known to be at least 2λ [27], and this bound is easily achieved by many linear extractors (e.g., [20]).

Once again, we observe that our non-local extractor only has to withstand affine leakage. In particular, we show that the optimal entropy loss for (non-local) doubly-affine extractors is only λ, *saving a factor of two over general-leakage extractors.* We present a general construction of such non-local, doubly-affine extractors from rank-preserving matrices (see [11]), that may be instantiated from a variety of concrete matrices such as Toeplitz matrices.

Seed Length. In our analysis, we did not optimize the length s of S. Existentially, we show that all our improvements are possible (unconditionally) with $\mathsf{s} = O(\lambda)$, while our concrete constructions use larger seed length $\mathsf{s} = O(\mathsf{m} + \lambda \log(\mathsf{n}))$. Such a seed-length is quite acceptable for most applications, as this only increases the ciphertext length (or communication with the server $\mathbf{T}$) by a constant factor. Moreover, to match computationally-secure encryption schemes with optimal ciphertext length $\mathsf{m} + O(\lambda)$, we use the fact that doubly-affine extractors are *everlastingly private* with computationally-secure seeds. Hence, we can use any stream cipher (e.g., SALSA20 or CHACHA) to expand a λ-bit seed S^t into the required longer seed $S = \mathbf{Prg}(S^t)$, *while maintaining IT-security.*

For example, our reusable IT-encryption in Section 1.1 can have ciphertext $(S^t, \mathbf{Ext}(X, \mathbf{Prg}(S^t)) \oplus M)$ of optimal length $(\mathsf{m} + \lambda)$, matching that of computationally secure schemes! Similarly, the communication complexity when interacting with our virtual server $\mathbf{T}$ from Section 1.2 can be made optimal: λ "upstream" bits S^t from the users, and m "downstream" bits $R = \mathbf{Ext}(X, \mathbf{Prg}(S^t))$ from the server. We stress that we need an extremely weak kind of computational-security security for the $\mathbf{Prg}$: just ability to fool a concrete, and easily computable statistical test (which we know a random expanding function satisfies w.h.p.). Thus, it seems *extremely plausible* that SALSA20 or CHACHA satisfy this combinatorial property *unconditionally.* Nevertheless, it is a good theoretical question to improve the seed length s to the optimal value $O(\lambda)$.

1.6 Applications

Replicated Setting. We generalize the setting of Section 1.1, where the entire long secret key X is replicated among several trusted parties. We already saw that doubly-affine extractors immediately give locally computable, CPA-secure encryption $C = (S, \mathbf{Ext}(X, S) \oplus M)$ *with optimal locality* in this setting. In fact, the same scheme is trivially CCA1-secure, since doubly-affine extractors support leakage of extractor outputs on adversarial seeds S. To get CCA2 security, and even achieve the strongest notion of authenticated encryption (AE) [6], the parties can additionally share a short key for *computationally-secure* MAC, and use this fixed key to authenticate the ciphertext $C = (S, P)$ above. In this variant, the authenticity is computational, but the privacy is everlasting, as long as the MAC is not broken while the post-challenge decryption oracle is used. Moreover, all these schemes are still everlastingly private with computationally-secure seeds S, allowing to achieve optimal ciphertext length $(\mathsf{m} + O(\lambda))$.

Distributed Setting. We generalize the setting of Section 1.2 where parties delegate the storage of X to several servers. We already saw that doubly-affine extractors are enough in the single server case, as they provide the required post-application security and support computationally secure seeds. For example, the server $\mathbf{T}$ can help the parties to achieve all the efficiency benefits of (symmetric-key) authenticated encryption with associated data (AEAD) plus *everlasting privacy.* All the parties need to do is use a computationally-secure AEAD (using a short shared key) to encrypt the seed S,[3] instead of the message M. Similar techniques also work in the public-key setting, where S can be appropriately encrypted using the receiver's public-key. While relying on a single trusted server $\mathbf{T}$ may be challenging due to privacy of the channel, we can, instead, consider distributed setting with multiple servers $t \geq 2$ where we use the standard assumption in the information-theoretic literature that channels between the user and at least g servers are secure and the remaining $t - g$ channels may be compromised. This is a common assumption that has been used in many prior seminal works including information-theoretic secret sharing [28], multi-party computation [8] and secure message transmission [13].

[3]Technically, S should be encrypted with associated data $P = R \oplus M$.

Specifically, we extend our architecture to the setting of $t \geq 2$ servers, who jointly emulate the virtual server $\mathbf{T}$. In more detail, each of the t servers will independently generate and store a subset of the random source X. For the distributed setting, we only assume that $g \leq t$ servers are honest with a private channel to users. Moreover, the servers do not need to coordinate, or even know each other's existence: each simply picks a random string, and provides access to its random string to users.

We also consider two cases where the $(t - g)$ corrupted servers are either honest-but-curious or malicious. Our honest-but-curious solution works for any $g \geq 1$, and achieves multiplicative overhead roughly t/g for the user, as compared to the single-server case. In particular, each server accesses and returns a *sub-linear* number of bits $p^t \approx m/(\beta g)$. For the malicious setting, we necessarily assume that $g > 2t/3$, and use simple error-correcting techniques to achieve overhead roughly $t/(3g - 2t)$, with each server returning $p^t \approx m/(\beta(3g - 2t))$ bits.

2 Definitions

In this section, we formally define doubly-affine extractors that output affine functions of the random source while tolerating affine leakage. We start by presenting the affine oracle that provides linear access to a truly random string. Afterwards, we define doubly-affine extractors in both the information theoretic and computationally secure settings.

2.1 Affine Leakage Model

In the affine leakage model, there is a uniformly random string $X \in \{0,1\}^n$. Throughout our work, X is referred to as the *source* or *random source*. The string X is accessed through an affine oracle that receives a n-bit string $Q \in \{0,1\}^n$ and returns the dot product of $\mathbf{LIN}_X(Q) = Q \cdot X$. In other words, one query to the affine oracle enables retrieving the XOR of a subset of bits of X.

For convenience, multiple queries to the affine oracle may be represented using a single matrix. In particular, q queries may be represented using a $q \times n$ bit-matrix $\mathbf{Q} \in \{0,1\}^{q \times n}$ such that $\mathbf{LIN}_X(\mathbf{Q}) = \mathbf{Q}X$ where the affine oracle returns the multiplication of $\mathbf{Q}$ and X resulting q bits.

Definition 1. *For any $X \in \{0,1\}^n$, the affine oracle $\mathbf{LIN}_X$ receives a $q \times n$ binary matrix $\mathbf{Q} \in \{0,1\}^{q \times n}$ for any $q \geq 1$. Then, $\mathbf{LIN}_X(\mathbf{Q}) = \mathbf{Q}X$.*

2.2 Information-Theoretic Doubly-Affine Extractors

The main focus of our work is to construct efficient extractors in the affine leakage model. The goal of a doubly-affine extractor is to utilize a short random seed along with access to the oracle $\mathbf{LIN}_X$ to derive a random string that may be used at higher level applications. For security, the extractor's output should remain random even if an adversary uses the oracle $\mathbf{LIN}_X$ to learn large (but not all) of the underlying random string X.

In more detail, extractors are defined as algorithms that receive a random s-bit seed and output a m-bit random string where $m > s$ (that is, the output random string is larger than the input seed). Extractors are able to perform queries to the oracle $\mathbf{LIN}_X$ to access the uniformly random string X. The output of extractors should remain random to an adversary that has utilized the oracle $\mathbf{LIN}_X$ to learn at most £ bits about the underlying random string X.

In this paper, we restrict our attention to extractors that only perform non-adaptive queries to $\mathbf{LIN}_X$. In other words, the extractor must pick a single query matrix $\mathbf{Q}$, send it to the $\mathbf{LIN}_X$ and use the response to generate the output. To our knowledge, all prior works also exclusively studied extractors that non-adaptively accessed the underlying random string X. Non-adaptivity may be beneficial in settings where sending queries to $\mathbf{LIN}_X$ may be expensive.

For convenience, we will make the assumption that the output of doubly-affine extractors will simply be the response from the single query to the oracle $\mathbf{LIN}_X$. We show that this limitation is not important as our constructions will be essentially optimal. With this restriction, the output of doubly-affine extractors will

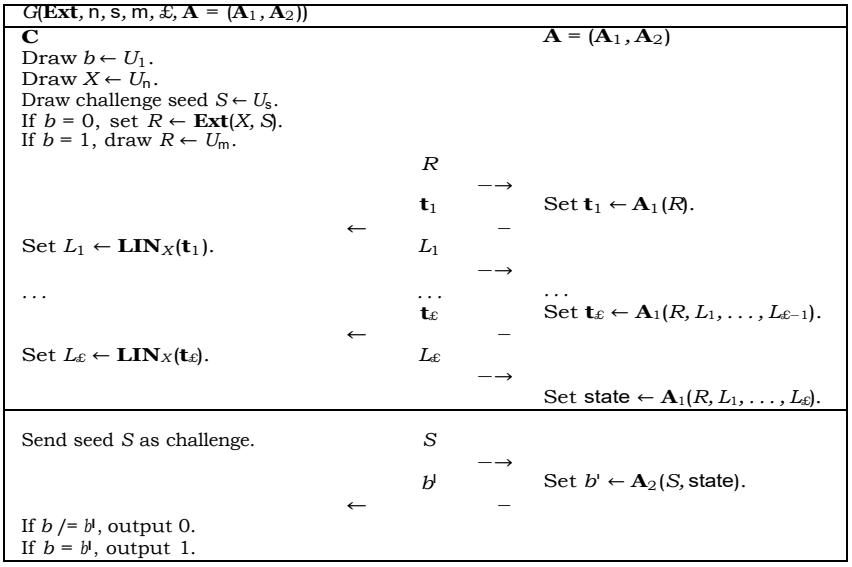


Figure 1: Game $G(\mathbf{Ext}, \mathsf{n}, \mathsf{s}, \mathsf{m}, \pounds, \mathbf{A})$.

also be affine. This property will be integral in settings when the adversary's leakage consists of previous extractor outputs.

We define extractors as $\mathbf{Ext} : \{0,1\}^{\mathsf{n}} \times \{0,1\}^{\mathsf{s}} \to \{0,1\}^{\mathsf{m}}$ where the first argument is the n-bit random string, the second argument is s-bit random seed and the output are the m extracted bits. As we consider non-adaptive extractors, we note that all extractors $\mathbf{Ext}$ are uniquely defined by a query algorithm $\mathbf{Pick}_{\mathbf{Ext}} : \{0,1\}^{\mathsf{s}} \to \{0,1\}^{\mathsf{m} \times \mathsf{n}}$ that outputs a $\mathsf{m} \times \mathsf{n}$ binary matrix that will be query to the oracle $\mathbf{LIN}_X$. Since $\mathbf{Ext}$ returns the output from the oracle, we note that $\mathbf{Ext}(X, S) = \mathbf{LIN}_X(\mathbf{Pick}(S))$ for any n-bit random string X and s-bit random seed. We will use $\mathbf{Ext}$ and $\mathbf{Pick}_{\mathbf{Ext}}$ interchangeably in our paper.

The security of doubly-affine extractors are presented in Figure 1. The adversary $\mathbf{A}$ is given a challenge of either a m-bit unifomly random string or the extractor output. Using the oracle $\mathbf{LIN}_X$, $\mathbf{A}$ may perform $\pounds$ adaptive queries to learn at most $\pounds$ linear functions of X with knowledge of the challenge. Afterwards, $\mathbf{A}$ is given the input seed and must guess the origin of the challenge. We say $\mathbf{A}$ has ε advantage if $\mathbf{A}$ has $1/2 + \varepsilon$ probability of guessing correctly. For ease of presentation, we split $\mathbf{A}$ into stateful adversaries $\mathbf{A}_1$ and $\mathbf{A}_2$ that are responsible for generate oracle queries and computing the final guess respectively. We note that both adversaries are computationally unbounded.

We stress that $\mathbf{A}$ is given the challenge prior to performing any queries to the oracle $\mathbf{LIN}_X$. As a result, we require that doubly-affine extractors provide security against *post-application leakage*. This is a notion that is not achievable in other models with general leakage (we present a counterexample in Appendix B). By restricting to linear leakage, we enable a significant improvement in security.

Definition 2. *A deterministic algorithm* $\mathbf{Ext} : \{0,1\}^{\mathsf{n}} \times \{0,1\}^{\mathsf{s}} \to \{0,1\}^{\mathsf{m}}$ *is* $(\pounds, \varepsilon)$*-secure if for any adversary* $\mathbf{A}$, $\Pr[G(\mathbf{Ext}, \mathsf{n}, \mathsf{s}, \mathsf{m}, \pounds, \mathbf{A}) = 1] \leq \frac{1}{2} + \varepsilon$.

We move onto the efficiency of extractors. We define the *probe complexity* of an extractor as the number of bits of X accessed by the oracle in a single extractor execution. We define the *cache complexity* of an extractor as the number of disjoint regions of X accessed by the oracle in a single extractor execution. Probe complexity measures the total running time of oracle in a single extractor execution while cache complexity measures the number of cache misses incurred in a single extractor execution. Note that probe complexity may be measured as the number of non-zero columns in the query matrix produced by $\mathbf{Pick}_{\mathbf{Ext}}$. Cache complexity corresponds to the number of consecutive groups of non-zero columns found in the query matrix produced by $\mathbf{Pick}_{\mathbf{Ext}}$.

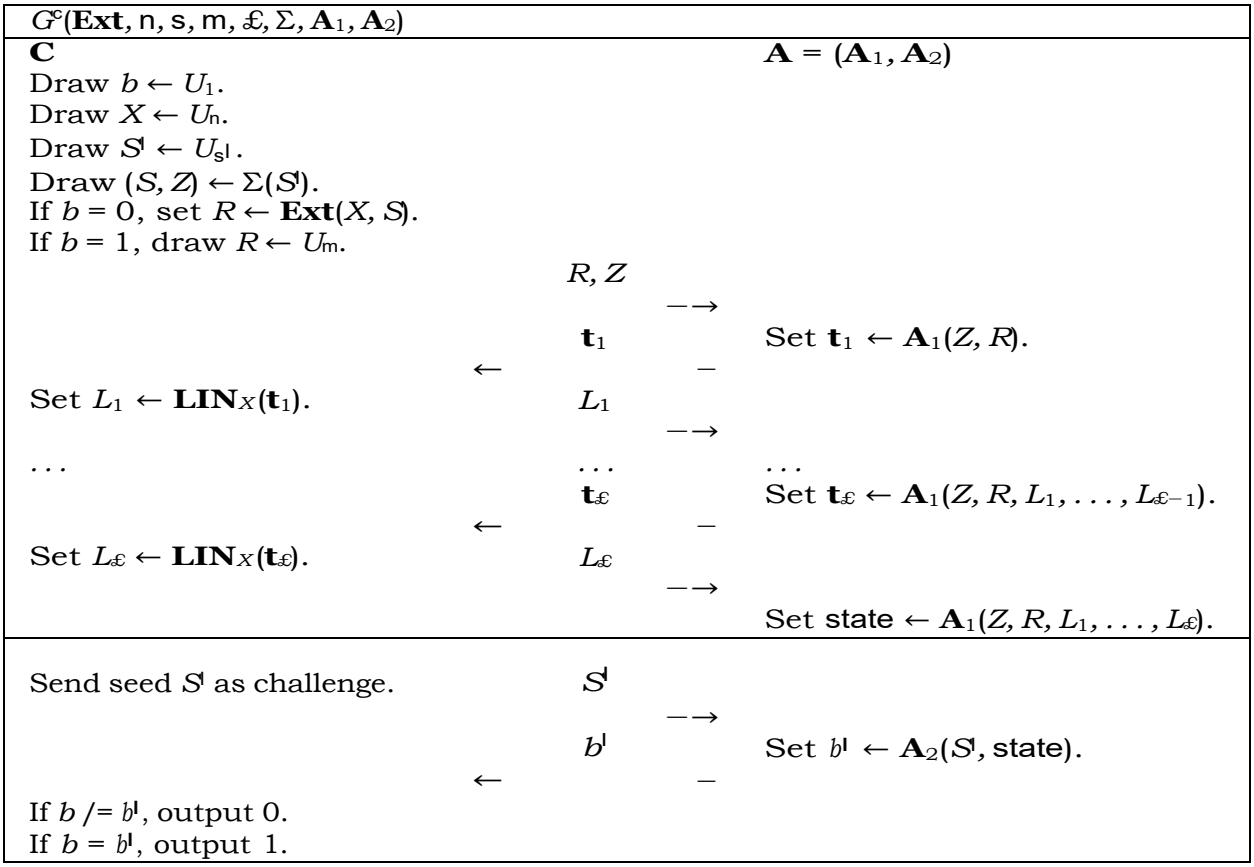


Figure 2: Game $G^{c}(\mathbf{Ext}, n, s, m, \ell, \Sigma, \mathbf{A}_1, \mathbf{A}_2)$.

Definition 3. $\mathbf{Ext}$ *is* (p, c)*-local if for every seed* S, $\mathbf{Pick}_{\mathbf{Ext}}(S)$ *has at most* p *non-zero columns and at most* c *consecutive non-zero column groups.*

2.3 Computational Doubly-Affine Extractors

As another advantage of doubly-affine extractors, we show that they may be built even when using seeds that are only computationally-secure. In more detail, suppose the extractor's input seed is computationally-secure with respect to leakage seen by the adversary. Is it possible for the extractor's output to remain information-theoretically random with help from the affine oracle? This is impossible for computationally unbounded adversaries with access to the oracle as the adversary may compute the seed and query the oracle to obtain the extractor's output. Instead, we want computational extractors to produce outputs that are secure against adversaries that are computationally-bounded only when the oracle is available and may become computationally-unbounded afterwards. The ability to handle computationally-secure seeds is a benefit of the affine leakage model that is impossible in general leakage models (see [14, 19]).

The security game for computational extractors is shown in Figure 2. As a major result, we will prove that the security games in Figure 1 and Figure 2 are equivalent (see Section 4). That is, every information-theoretic extractor is also a computational extractor. We consider *hybrid adversaries* $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ where $\mathbf{A}_1$ is a stateful PPT adversary and $\mathbf{A}_2$ is a computationally unbounded adversary. The game uses a computationally-secure protocol Σ that produces a computational seed S as well as leakage Z using a (typically) shorter random seed S'. $\mathbf{A}_1$ is given both the extractor's output and the leakage Z of Σ. The role of $\mathbf{A}_1$ is to adaptively query the oracle $\mathbf{LIN}_X$ to learn ℓ bits about X. $\mathbf{A}_2$ will use the knowledge gained

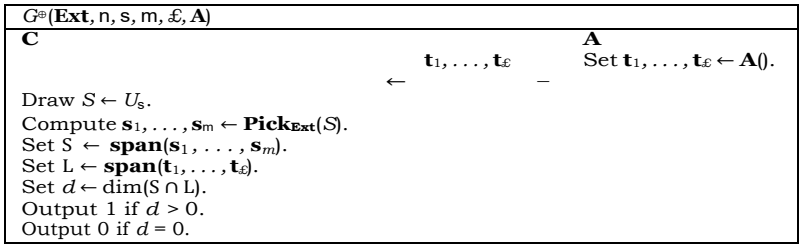


Figure 3: Linear Span Game $G^{\oplus}(\mathbf{Ext}, \mathsf{n}, \mathsf{s}, \mathsf{m}, \ell, \mathbf{A})$.

by $\mathbf{A}_1$ as well as the original seed S^t to distinguish between challenges of either uniformly random strings or extractor outputs.

Definition 4. *A deterministic algorithm* $\mathbf{Ext} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ *is* $(\ell, \Sigma, \varepsilon)$*-computationally-secure if for any hybrid adversary* $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ *such that* $\mathbf{A}_1$ *is PPT,* $\Pr[G^c(\mathbf{Ext}, \mathsf{n}, \mathsf{s}, \mathsf{m}, \ell, \Sigma, \mathbf{A}) = 1] \leq \frac{1}{2} + \varepsilon$.

3 Information-Theoretic Doubly-Affine Extractors

In this section, we present our constructions for information-theoretic doubly-affine extractors. We start by reducing the security game of doubly-affine extractors to linear algebraic concepts. Afterwards, we show that constructing doubly-affine extractors requires two simpler primitives: *samplers* and *non-local doubly-affine extractors* (or non-local extractor, for short). By presenting efficient samplers and non-local doubly-affine extractors, we obtain our final efficient extractor. We also present various lower bounds for the studied primitives.

3.1 Optimal Doubly-Affine Extractor Adversary

One of the main results in our paper is the ability to reduce the complex security game of doubly-affine extractors to a simpler game. Consider the following adversarial approach to compromise doubly-affine extractors according to the security game in Figure 1. Suppose the adversary has chosen ℓ queries to $\mathbf{LIN}_X$. Next, the adversary receives the seed S and computes the query matrix $\mathbf{Pick}_{\mathbf{Ext}}(S)$. Next, the adversary checks whether extractor's output bits is a linear combination of any of the ℓ leakage bits. This is equivalent to checking whether the span of the extractor's queries intersects the span of the adversary's queries. In the case of an intersection, the adversary can check whether the output bit matches the linear combination. For real challenges, this is always true. For random challenges, this is only true with probability $1/2$. Therefore, the adversary has significant advantage as long as the intersection of query spans is non-empty.

We show that the above adversary is essentially optimal up to choosing the oracle queries. Formally, we prove this by showing the security game in Figure 1 is equivalent to the same simpler game in Figure 3. The game in Figure 3 severely limits the adversary by forcing the adversary to follow the above adversarial approach. The adversary must ignore both ignore the extractor output and non-adaptively query the oracle. Additionally, the adversary loses the ability to post-process the oracle results. Instead, the challenger determines the winner of the game by checking whether the intersection of the adversarial query subspace and the extractor query subspace is non-empty. We show the security games in Figures 1 and 3 are identical. As a caveat, we note that the adversary could also check whether the extractor's outputs bit are linearly independent. If any output bit is a linear combination of the other output bits, then the adversary will already win the game. For real challenges, the linearly dependent output bit must match the linear combination of the other output bits. For random challenges, this only happens $1/2$ of the time. Therefore, we will assume that the extractor outputs bits will be linearly independent. In other words, the extractor's oracle queries will always be linearly independent without loss of generality.

Theorem 1. *Suppose that* **Ext** *is* (ℓ, ε)*-secure with respect to security game* $G^{\oplus}$*. That is, for any adversary* $\mathbf{A}$, $\Pr[G^{\oplus}(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}) = 1] \leq \varepsilon$*. Then,* **Ext** *is* (ℓ, ε)*-secure with respect to* G *according to Definition 2.*

Proof. We use the H-coefficient technique to upper bound the advantage of any adversary. We denote G_0 to be the real experiment where $b = 0$ and G_1 to be the ideal experiment in Figure 1. Let $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ be the leakage vectors chosen by the adversary $\mathbf{A}$ while $\mathbf{s}_1, \ldots, \mathbf{s}_m$ are the output m row vectors outputted by $\mathbf{Pick}_{\mathbf{Ext}}(S)$. We denote $L := \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ and $S := \mathbf{span}(\mathbf{s}_1, \ldots, \mathbf{s}_m)$. We will define the set of all good transcripts T defined such that $\dim(L \cap S) = 0$.

Fix any transcript in $t \in$. We will consider a step-by-step analysis of G_0 and G_1 and compute the probabilities of the next message conditioned on the previous messages being fixed. Note, the output of $\mathbf{A}$ depends only its input and its internal randomness. Therefore, the output distribution of $\mathbf{A}$ will always be the same as long as the input is the same. So, we only focus on the messages sent by the challenger. Let us start with the seed S, which is drawn uniformly at random in both G_0 and G_1. Note that $\mathbf{s}_1, \ldots, \mathbf{s}_m$ are all linearly independent by our assmption on the extractor. Therefore, as X is uniformly random, each of $R_i = \mathbf{LIN}_X(\mathbf{s}_i) = \mathbf{s}_i \cdot X$ are uniformly random.

By a similar assumption on the adversary, we note that all $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ will be linearly independent. As we are assuming that $\dim(L \cap S) = 0$, we know that $\mathbf{t}_i$ is linearly independent from all of $\mathbf{s}_1, \ldots, \mathbf{s}_m$ and $\mathbf{t}_1, \ldots, \mathbf{t}_{i-1}$. As X is uniformly random, we know that $\mathbf{LIN}_X(\mathbf{t}_i) = \mathbf{t}_i \cdot X$ is a uniformly random bit. So, $\Pr[T(G_0) = t] = \Pr[T(G_1) = t]$ for all $t \in T$ and their ratio is always 1. By the H-coefficient technique (see Theorem 13 in Appendix A), the advantage of $\mathbf{A}$ is upper bounded by $\Pr[T(G_1) \notin T]$.

Therefore, we only need to bound $\Pr[T(G_1) \notin T]$. In particular, we can reinterpret this as a new game where the adversary $\mathbf{A}$ is attempting to maximize the probability that $\dim(L \cap S) > 0$ in G_1. We will show that this game corresponds to the game in Figure 3 and $\Pr[T(G_1) \in T] \leq \max_{\mathbf{A}^t} \Pr[G^{\oplus}(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}^t) = 1]$ for any adversary $\mathbf{A}^t$, which would complete our proof.

To prove this, we will convert any adversary $\mathbf{A}$ for Figure 1 into an adversary $\mathbf{A}^t$ for Figure 3. Note in G_1, that each of $R, L_1, \ldots, L_\ell$ and S are generated uniformly at random. We construct $\mathbf{A}^t$ who generates $R, L_1, \ldots, L_\ell$ on their own without interacting with the challenger and feeds them to $\mathbf{A}$ to get the vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$. Note, the view of $\mathbf{A}$ is identical to G_1. As a result, $\Pr[T(G_1) \in T] = \Pr[G^{\oplus}(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}^{\oplus}) = 1]$ as required. □

With this theorem, we already see that the main challenge for extractors is to ensure output bits are not a linear combination of the adversarial oracle queries.

3.2 Sample-then-Extract Paradigm

To construct doubly-affine extractors, we use the sample-then-extract paradigm that was introduced by Vadhan [31]. This paradigm constructs extractors in two steps. First, a subset of the random string X is sampled such that the sample contains a large amount of entropy conditioned on the adversary's leakage. Next, a non-local doubly-affine extractor (that we will also denote as a non-local extractor) is executed on the sampled subset. The non-local extractor is expected to utilize the entirety of the sampled subset to produce as many random bits as possible (since no locality is required, they are denoted as non-local).

We now explain at a high level why the sample-then-extract algorithm results in an extractor. All the sampled bits will not be random after the adversary views leakage bits. If the adversary sees ℓ bits of the n-bit random string X, then we expect only $(1 - \ell/n)$-fraction of the sampled bits to be random. The role of the non-local extractor is to condense the mixture of random and non-random sampled bits into a smaller string of truly random bits. We will both define and construct samplers and non-local extractors in the upcoming sections.

3.3 Samplers

The notion of *samplers* has been well studied in the past (see [7, 9, 24, 32, 31, 17] as some examples). Prior works studied samplers with respect to general functionalities and/or general leakage. In our work, we

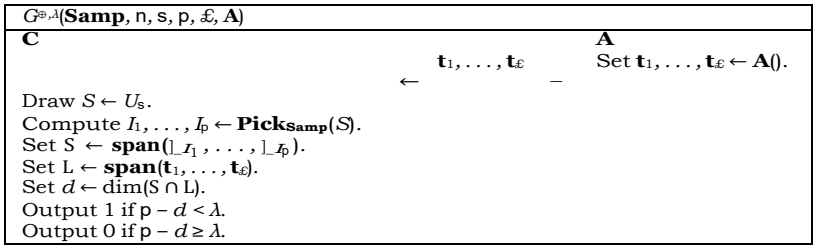


Figure 4: Relaxed Linear Span Game $G^{\oplus,\lambda}$(**Samp**, n, s, p, £, **A**).

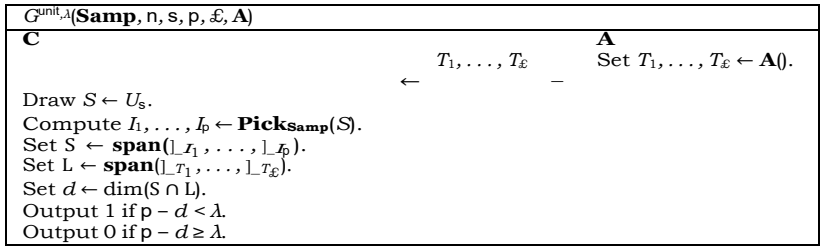


Figure 5: Relaxed Unit Span Game $G^{\text{unit},\lambda}$(**Samp**, n, s, p, £, **A**).

define samplers in a narrower manner within the affine leakage model that will be easily composable in the sample-then-extract paradigm.

Samplers are deterministic algorithms $\mathbf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^p$ with inputs of a n-bit random string X and a s-bit seed S that outputs p sampled bits of X. The goal is to sample as many random bits as possible in the view of the adversary with leakage bits. As discussed in the previous section, it is unlikely that all sampled bits will be secure. If an adversary has £ random leakage bits of X, then only $(1 - £/n)p$ sampled bits will be secure in expectation.

For any **Samp**, we denote $\mathbf{Pick}_{\mathbf{Samp}}$ as the queries sent to the oracle (similar to extractors). The output of **Samp** will also be the response from the oracle. In other words, $\mathbf{Samp}(X, S) = \mathbf{LIN}_X(\mathbf{Pick}_{\mathbf{Samp}}(S))$. As **Samp** samples bits, each column of $\mathbf{Pick}_{\mathbf{Samp}}(S)$ will be a unit vector. We will use **Samp** and $\mathbf{Pick}_{\mathbf{Samp}}$ interchangeably in the remainder of the paper.

We relax the security game of doubly-affine extractors (Figure 3) to obtain a security game for samplers in affine leakage model presented in Figure 4. Recall that the extractor adversary should not compromise any output bits. We modify the definition for samplers so at least λ sampled bits are random in the adversary's view. We chose to immediately define samplers with respect to the optimal adversary for convenience. One could re-define sampler security using a natural game by relaxing the security of doubly-affine extractors in Figure 1.

Definition 5. *A deterministic algorithm* $\mathbf{Samp} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^p$ *is* $(£, \varepsilon, \lambda)$*-secure if for any adversary* **A**, $\Pr[G^{\oplus,\lambda}(\mathbf{Ext}, n, s, m, £, \mathbf{A}) = 1] \leq \varepsilon$.

Specific to samplers, we immediately show that the adversary may immediately be weakened without loss of generality. Consider a simple adversary for samplers that also samples £ bits from X. If the adversary samples $\lambda + 1$ bits identical to the sampler, the adversary will distinguish the real-or-random challenge with high advantage. We prove that this adversary is optimal. In other words, sampler adversaries do not gain advantage by learning linear combinations of X as opposed to sampling single bits of X. To formalize this idea, we modify the sampler game in Figure 4 such that the adversary may only sample physical bits of X. The new game may be found in Figure 5.

Theorem 2. *Suppose that* **Samp** *is* $(£, \varepsilon, \lambda)$*-secure with respect to security game* $G^{\text{unit},\lambda}$*. That is, for any adversary* **A**, $\Pr[G^{\text{unit},\lambda}(\mathbf{Samp}, n, s, m, £, \mathbf{A}) = 1] \leq \varepsilon$*. Then,* **Samp** *is* $(£, \varepsilon, \lambda)$*-secure with respect to* $G^{\oplus,\lambda}$ *according to Definition 5.*

Proof. Take any adversary $\mathbf{A}^{\oplus}$ and we will construct an adversary $\mathbf{A}^{\text{unit}}$ with no worst advantage. Consider any output of $\mathbf{A}^{\oplus}$ as $\mathbf{T}$ with the ℓ row vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$. We will show that there exists a strategy that replaces $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ with unit vectors without affecting the adversary's advantage. First, perform Gaussian elimination on $\mathbf{T}$ without the ability to swap columns. Without loss of generality, we get row vectors $\mathbf{t}'_1, \ldots, \mathbf{t}'_\ell$ such that each $\mathbf{t}'_i$ will have a leading 1 entry in column j_i and $1 \le j_1 < \ldots < j_\ell \le n$. Furthermore, we know that $\mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ and $\mathbf{span}(\mathbf{t}'_1, \ldots, \mathbf{t}'_\ell)$ are identical. We will choose our unit vectors as $T_1 := \mathsf{R}_{j_1}, \ldots, T_\ell := \mathsf{R}_{j_\ell}$.

Consider the matrix output by $\mathbf{Pick}_{\mathbf{Samp}}(S)$ with the row unit vectors $I_1, \ldots, I_p$ where S is drawn uniformly at random. Suppose the dimension of $\mathbf{span}(\mathbf{t}'_1, \ldots, \mathbf{t}'_\ell) \cap \mathbf{span}(\mathsf{R}_{I_1}, \ldots, \mathsf{R}_{I_p})$ is d. Denote the basis of this intersection by $\mathbf{v}_1, \ldots, \mathbf{v}_d$. Note each $\mathbf{v}_k$ is a linear combination of $\mathbf{t}'_1, \ldots, \mathbf{t}'_\ell$. That is, $\mathbf{v}_k = b_1\mathbf{t}'_1 + \ldots + b_\ell\mathbf{t}'_\ell$. If $b_i = 1$, then we know that the j_i-th entry of $\mathbf{v}_k$ must be 1. Then, for all $b_i = 1$, this implies that R_{j_i} was chosen as a row unit vector by $\mathbf{Pick}_{\mathbf{Samp}}(S)$. Consider the new vector $\mathbf{v}'_k = b_1T_1 + \ldots + b_\ell T_\ell$. The set of indices of $\mathbf{v}'_k$ that are 1 are exactly $\{j_i \mid b_i = 1\}$. As a result, $\mathbf{v}'_k$ also belongs to $\mathbf{span}(\mathsf{R}_{I_1}, \ldots, \mathsf{R}_{I_p})$. Note, the set $\{\mathbf{v}'_1, \ldots, \mathbf{v}'_d\}$ is linearly independent since $\{\mathbf{v}_1, \ldots, \mathbf{v}_s\}$ is linearly independent. Therefore, $\mathbf{span}(\mathbf{v}'_1, \ldots, \mathbf{v}'_d)$ has dimension d and is contained in the intersection of $\mathbf{span}(\mathsf{R}_{T_1}, \ldots, \mathsf{R}_{T_\ell})$ and $\mathbf{span}(\mathsf{R}_{I_1}, \ldots, \mathsf{R}_{I_p})$. This mean the dimension d' of the intersection achieved by the new adversary $\mathbf{A}^{\text{unit}}$ is at least d achieved by $\mathbf{A}^{\oplus}$, completing the proof. □

The above security reductions significantly simplify analyzing sampler security. Before moving on, we highlight that the majority of our efficiency gains are achieved from our improved samplers. In particular, the insight that optimal sampler adversaries in the affine leakage model model are limited is the key reason as to why our doubly-affine extractors are more efficient than previous constructions in the general leakage model.

(p, c)-Local Sampler Construction. We start by presenting an efficient (p, c)-local sampler. The idea is to view the n-bit source X as a two-dimensional matrix with c rows and n/c columns and the seed $S = (S_1, \ldots, S_c)$ as c integers from the set $[n/c]$ that are represented using $\log(n/c)$ bits. The sampler will sample p/c bits from each row. In the i-th row, the sampler chooses the consecutive p/c bits starting from the S_i-th column. **Samp** is (p, c)-local as a total of p bits are sampled that may be arranged into c consecutive groups (one group per row). Our construction is similar to ones presented in [14], but is was never analyzed or defined as a stand-alone sampler.

Theorem 3. *For any $0 \le \ell < n$, $c > 0$, $\varepsilon > 0$ and $\lambda > 0$, there exists a (p, c)-local sampler that is $(\ell, \varepsilon, \lambda)$-secure with seed length $s = c \log(n/c)$ and probe complexity*

$$p = \max\left\{c, \left\lceil \frac{\lambda}{1 - \frac{\ell}{n} - \frac{\ln(1/\varepsilon)}{c}} \right\rceil\right\}.$$

Proof. Fix any output of indices $T_1, \ldots, T_\ell$ from the adversary $\mathbf{A}$. Suppose that $I_1, \ldots, I_p$ are the indices of bits sampled by **Samp**. Then,

$$\dim(\mathbf{span}(\mathsf{R}_{T_1}, \ldots, \mathsf{R}_{T_\ell}) \cap \mathbf{span}(\mathsf{R}_{I_1}, \ldots, \mathsf{R}_{I_p})) = |\{T_1, \ldots, T_\ell\} \cap \{I_1, \ldots, I_p\}|.$$

Consider any index I_j that appears in row i. There are exactly p/c different choices of S_i such that the **Samp** will sample the bit at index I_j. Overall, there are n/c choices of S_i. Therefore, I_j is sampled by **Samp** with probability exactly $(p/c)/(n/c) = p/n$.

Suppose that ℓ_i indices appear in row i of $\mathbf{M}$, for all $i \in [c]$. Note that $\ell = \ell_0 + \ldots + \ell_{c-1}$. Denote X_i to be the number of bits sampled by **Samp** that do not overlap with leaked bits. We know that $\mathsf{E}[X_i] = p/c - (\ell_i p)/n$ by linearity of expectation. If we set $\mu = X_0 + \ldots + X_{c-1}$, then

$$\mu = p - \frac{\ell p}{n} = p \cdot \left(1 - \frac{\ell}{n}\right).$$

As exactly p/c bits are sampled in each row, $X_i \leq p/c$. We now apply Chernoff bounds (see [16]) to get that for any constant $0 < \delta < 1$,

$$\Pr[X \leq (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu^2}{c(p/c)^2}} \leq e^{-\delta^2 c\left(1-\frac{\ell}{n}\right)^2}.$$

Therefore, we know that there exists at least $\lambda = p(1-\delta)(1-\ell/n)$ sampled bits that are not leaked to the adversary except with probability $e^{-\delta^2 c\left(1-\frac{\ell}{n}\right)^2}$.

We know that $\varepsilon \leq e^{-\delta^2 c\left(1-\frac{\ell}{n}\right)^2}$ so $\delta \leq \sqrt{\ln(1/\varepsilon)/(c(1-\ell/n)^2)}$ whenever $c > \ln(1/\varepsilon)/(1-\ell/n)^2$. Plugging back into $\lambda = p(1-\delta)(1-\ell/n)$, we get that $\lambda = p(\ell/n + \sqrt{\ln(1/\varepsilon)/c})$. We can set $\lambda = p - m$. After re-arranging, we get the desired theorem. □

We note that the efficiency of our sampler is almost optimal. Note that an adversary may always pick ℓ random bits of X as leakage. If p bits are sampled, it is expected that only $(1-\ell/n)p$ sampled bits are secure. Our construction secure samples $\lambda = (1-\ell/n-\sqrt{\ln(1/\varepsilon)/c})p$ bits implying that at most $(\sqrt{\ln(1/\varepsilon)/c})$-fraction of bits are lost beyond the expectation.

(p, p)-Local Sampler Construction. For the setting where cache complexity is irrelevant, we present a (p, p)-local sampler that ends up being more concretely efficient. Our construction simply picks a subset of p bits from X uniformly at random using a seed S that encodes a random subset using $\log\binom{n}{p}$ bits.

Theorem 4. *For any $0 \leq \ell < n$ and $\lambda > 0$, there exists a (p, p)-local sampler that is $(\ell, \varepsilon, \lambda)$-secure with seed length $\log\binom{n}{p}$ and probe complexity p satisfying*

$$\varepsilon \leq (p-\lambda)\left(\frac{n-\ell}{p-\lambda}\right)\left(\frac{1}{2}\right)^{\lambda+1}$$

In Appendix C, we prove the above theorem. We also analytically show that our (p, p)-local sampler is concretely more efficient than our (p, c)-local sampler whenever $p = c$.

3.4 Non-Local Doubly-Affine Extractors

In the sample-then-extract paradigm, the goal of non-local extractors is to take sampled bits containing both random and non-random bits and produce a truly random string. We abstract the setting to the original doubly-affine extractor security game by assuming that the non-random bits are leakage viewed by the adversary. In other words, we assume that the random source X has n random bits except that ℓ bits are not random as they have been viewed by the adversary.

The simplest way to build non-local extractors is to use affine universal hash functions and the leftover hash lemma [20]. As the input string X has $n - \ell$ random bits, the leftover hash lemma states that there exists a non-local extractor that outputs $n - \ell - 2\log(1/\varepsilon)$ random bits except with probability ε.

In our work, we improve upon this result by constructing non-local extractors that produces $n - \ell - \log(1/\varepsilon)$ random bits. This reduces the lost entropy by a factor of two. The ability to build improved non-local extractors is another advantage of doubly-affine extractors.

To do this, we generically reduce the construction of non-local extractors to a special family of matrices with certain properties. Consider m matrices $A_1, \ldots, A_m$ with n rows and s columns. For any non-empty subset $\emptyset \neq I \subseteq \{1, \ldots, m\}$, the matrix $A_I = \sum_{i \in I} A_i$ has rank n. We present an instantiation using field multiplication.

Special Matrix Family from Toeplitz Matrices. We instantiate the matrix family using Toeplitz matrices to achieve $s = n + m - 1$. Note, as $m \leq n$, the seed length is at most $s \leq 2n$. Toeplitz matrices are the matrices where all diagonals are equal. For any Toeplitz matrix T, it is always the case that $T_{i,j} = T_{i+1,j+1}$. In particular, we will use Toeplitz matrices of dimension $n \times (n + m - 1)$. We define the set $T_1, \ldots, T_m$ in the following way. T_i consists of the first $i - 1$ columns only of 0's followed by the $n \times n$ identity matrix occupying the next n columns. All remaining columns will also consist of only 0's. As an example,

T_1 consists of the identity matrix occupying the first n columns followed by all 0's. Using this construction, we get that for any seed $S \in \{0,1\}^{n+m-1}$, $T_i S = (S_i, S_{i+1}, \ldots, S_{i+n-1})$. We note that the computational cost of this instantiation is $O(n \log n)$ using FFT (see [25] for more details). In Appendix D, we present an instantiation from field multiplication with smaller seed lengths, but higher computational costs. We use the Toeplitz matrix instantiation for practical considerations.

Theorem 5. *For any subset* $\emptyset = I \subseteq [1, \ldots, m]$, $T_I = \bigsqcup_{i \in I} T_i$ *has rank* n.

Proof. To show this, we prove there will always exist n linearly independent column vectors in T_I for any $I = \emptyset$. We show that all n unit vectors always appear as column vectors. Consider all column vectors where all non-zero entries are contributed by exactly one of T_i where $i \in I$. There are exactly n of these vectors each with exactly a single 1 entry. Furthermore, they all correspond to unique unit vectors implying that the rank of T_I is n. □

Non-Local Extractor Construction. Our non-local extractor is built using the m matrix family $A_1, \ldots, A_m$ described above. The non-local extractor receives a s-bit seed S and a n-bit input string X with m random bits. Then, the output of the non-local extractor is $X \cdot (A_1 S), \ldots, X \cdot (A_m S)$. In other words, the i-th output bit is the dot product of X and the matrix-vector multiplication of the A_i and the seed S.[4] We note that our construction is similar to the one [11], but our security analysis is different, as we extracting more bits in our setting.

Theorem 6. *For any* $0 \leq \ell < n$ *and* $\varepsilon > 0$, *there exists a non-local extractor that is* (ℓ, ε)*-secure that outputs* $m = n - \ell - \log(1/\varepsilon)$ *bits with seed length* n.

Proof. We use the game $G^{\oplus}$ to evaluate the security of **NLExt**. Fix any leakage vectors $\mathbf{T} := (\mathbf{t}_1, \ldots, \mathbf{t}_\ell)$. $\mathbf{Pick}_{\mathbf{NLExt}}(S)$ will output the vectors $A_1 S, \ldots, A_m S$. We need to bound the probability that the intersection of $L := \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ and $S := \mathbf{span}(A_1 S, \ldots, A_m S)$ will contain a non-zero vector. To do this, we pick any non-zero vector $\mathbf{v} \in S$ and compute the probability that $\mathbf{v}$ occurs in the intersection. Note that L contains at most 2^ℓ vectors. We show that $\mathbf{v}$ is a vector chosen uniformly at random from a set of 2^n. As $\mathbf{v}$ is non-zero, we rewrite

$$\mathbf{v} = \bigsqcup_{i \in I} A_i S = \left(\bigsqcup_{i \in I} A_i\right) S = A_I S$$

where $I = \emptyset$. As a result, A_I has rank at least n and $\mathbf{v}$ is a random vector from a set of 2^n. Therefore, $\Pr[\mathbf{v} \in L] \leq 2^{\ell - n}$. To complete the proof, we apply a Union bound over all at most $2^m - 1$ non-zero vectors in S. Therefore,

$$\Pr[G^{\oplus}(\mathbf{NLExt}, n, s, m, \ell, \mathbf{A}) = 1 \mid \mathbf{A}() = \mathbf{T}] \leq (2^m - 1)\frac{2^\ell}{2^n} \leq 2^{m+\ell-n}$$

for any fixed leakage vectors $\mathbf{T}$. As a result, we get that

$$\begin{aligned}
&\Pr[G^{\oplus}(\mathbf{NLExt}, n, s, m, \ell, \mathbf{A}) = 1] \\
&\quad \leq \bigsqcup_{\mathbf{T}} \Pr[\mathbf{A}() = \mathbf{T}] \cdot \Pr[G^{\oplus}(\mathbf{NLExt}, n, s, m, \ell, \mathbf{A}) = 1 \mid \mathbf{A}() = \mathbf{T}] \\
&\quad \leq 2^{m+\ell-n} \cdot \bigsqcup_{\mathbf{T}} \Pr[\mathbf{A}() = \mathbf{T}] \leq 2^{m+\ell-n}.
\end{aligned}$$

We derive $\varepsilon \geq 2^{m+\ell-n}$ by choosing $m \leq n - \ell - \log(1/\varepsilon)$. □

We also present a lower bound on the number of extractable bits by non-local extractors that is tight up to an additive constant factor.

[4] Some of our constructions will yield universal hash functions. Unfortunately, we are unable to generically prove $\log(1/\varepsilon)$ loss using only the universality property.

Theorem 7. *Non-local extractors extract at most* $n - £ - \log(1/\varepsilon) - O(1)$ *bits.*

Seed Length. While our construction extracts an almost optimal number of bits, it requires a large seed length of n. Our seed length is similar to those obtained using the leftover hash lemma resulting in seed lengths of at least $n - £ + \log(1/\varepsilon)$ $O(1)$ bits [20]. In Appendix D, we existentially show there exists a matrix family that would result in a non-local extractor with seed length $\log(n£/\varepsilon)$ while extracting the same number of bits. We also present a seed length lower bound showing that the existential construction is almost optimal. We leave it as an open question to construct such a matrix family explicitly, but remind that: (a) in our setting it is safe to expand the seed computationally (unlike general extractors [3]); (b) in the random oracle model, one can expand the seed using the random oracle as done in [5, 4]; (c) one can use theoretical extractors of [18] which are linear, have seed $O(\log n + \log(1/\varepsilon))$, but double the entropy loss to $2\log(1/\varepsilon)$.

3.5 Doubly-Affine Extractors

With constructions of both samplers and non-local extractors, we finally construct our local extractors. First, we will formally define the sample-then-extract paradigm and show that it is secure. Afterwards, we plug in our sampler and non-local extractor constructions to obtain our efficient local extractors.

Sample-then-Extract. We formally define the sample-then-extract composition. Suppose we have a sampler **Samp** with seed length $S_{\mathbf{Samp}}$ that samples p bits. Note, the corresponding $\mathbf{Pick}_{\mathbf{Samp}}$ algorithm outputs a $p \times n$ query matrix. Additionally, assume we have a non-local extractor **NLExt** with seed length $S_{\mathbf{NLExt}}$ that extracts from an p-bit input and produces m-bit outputs. The corresponding $\mathbf{Pick}_{\mathbf{NLExt}}$ algorithm outputs a $m \times p$ query matrix. The resulting local extractor **Ext** is defined by its oracle query function

$$\mathbf{Pick}_{\mathbf{Ext}}(S = (S_{\mathbf{Samp}}, S_{\mathbf{NLExt}})) = \mathbf{Pick}_{\mathbf{NLExt}}(S_{\mathbf{NLExt}})\mathbf{Pick}_{\mathbf{Samp}}(S_{\mathbf{Samp}}).$$

In other words, the oracle query sent by **Ext** is the matrix multiplication of the query matrices chosen by **NLExt** and **Samp**.

Theorem 8. *Let* **Samp** *be a* $(£, \varepsilon_1, \lambda)$*-secure sampler with seed length* S_1 *that samples* p *bits from an* n*-bit source and* **NLExt** *be a* $(p - \lambda, \varepsilon_2)$*-secure non-local extractor with seed length* S_2 *that receives a* p*-bit source and outputs* m *bits. Then, there exists an extractor* **Ext** *that that is* $(£, \varepsilon_1 + \varepsilon_2)$*-secure with seed length* $S_1 + S_2$ *that outputs* m *bits. If* **Samp** *is* (p, c)*-local, then* **Ext** *is* (p, c)*-local.*

The proof may be found in Appendix E. Using the above theorem, we present our constructions using our sampler and non-local extractors. The resulting extractors are more efficient than all prior works (see Section 6 for comparisons).

Theorem 9 ((p, c)-Local Extractor)**.** *For any* $0 \leq £ < n$, $m > 0$ *and* $\varepsilon > 0$, *there exists an* (p, c)*-local extractor that is* $(£, \varepsilon)$*-secure with seed length* $s = c\log(n/c) + p$ *with probe complexity*

$$p = \max\left(c, \frac{m + \log(2/\varepsilon)}{1 - \frac{£}{n} - \frac{\ln(2/\varepsilon)}{c}}\right)$$

for any cache complexity $c \geq \ln(2/\varepsilon)(1 - £/n)^{-2}$.

We note that our extractor is almost optimal in terms of randomness efficiency. Once again, we can consider a simple adversary that samples $£$ leakage bits of X randomly. For any extractor with probe complexity p, only $(1 - £/n)$-fraction of the probed bits will be random in expectation. Our extractor is able to produce $m = (1 - £/n - \sqrt{\ln(2/\varepsilon)/c})p - \log(2/\varepsilon)$ bits that is only $(\sqrt{\ln(2/\varepsilon)/c})$-fraction from the expectation along with another $\log(2/\varepsilon)$ additive bit loss.

We also present the following lower bound on cache complexity for extractors whose proof may be found in Appendix E.

Theorem 10. *Any* (p, c)*-local extractor that is* $(\pounds, \varepsilon)$*-secure with one output bit must have cache complexity* $c = \Omega(\log(1/\varepsilon)/\log(np/\pounds))$.

In terms of cache complexity, we note that our extractor requires cache complexity at least $\ln(2/\varepsilon)(1 - \pounds/n)^{-2}$. Our extractor's cache complexity matches the lower bound in the setting that a constant fraction of bits are leaked, $\pounds = \Theta(n)$.

We also present an extractor when cache complexity is ignored ($c = p$). While both constructions are asymptotically identical, our (p, p)-local extractor is more concretely efficient. This construction is also more efficient than all previous schemes (see Section 6 for comparison).

Theorem 11 ((p, p)-Local Extractor)**.** *For any* $0 \leq \pounds < n$, $m > 0$ *and* $\varepsilon > 0$*, there exists an* (p, p)*-local extractor that is* $(\pounds, \varepsilon)$*-secure with seed length* $s = p\log(n/p) + p$ *with probe complexity* p *satisfying*

$$\varepsilon \leq 2(m + \log(2/\varepsilon)) \binom{n - \pounds}{m + \log(2/\varepsilon) - 1} \cdot \frac{1}{\binom{n}{p - m - \log(2/\varepsilon) + 1}}.$$

4 Computational Doubly-Affine Extractors

We move onto constructing extractors that are secure when using computationally-secure seeds. To refresh readers, recall that computational extractors considered security games against hybrid adversaries $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ where $\mathbf{A}_1$ is a PPT adversary and $\mathbf{A}_2$ is computationally unbounded. $\mathbf{A}_1$ queries the oracle for information about the random source X using the real-or-random challenge and seed leakage. After $\mathbf{A}_1$ learns $\pounds$ bit of leakage from the oracle, $\mathbf{A}_2$ is able to use all learned information to distinguish a real-or-random challenge. Note that $\mathbf{A}_1$ cannot be computationally unbounded as, otherwise, $\mathbf{A}_1$ could derive the original seed, query the oracle to compute the extractor's output and compare with the challenge. For the full definition, we refer readers back to Section 2.3. We re-iterate that computational seeds are not possible with general leakage [14, 19].

As a major result of our work, we show that any information-theoretic extractor is already a computational extractor. In other words, our constructions from Section 3.5 are also computational extractors with similar parameters. In particular, we prove that the security game in Figure 1 generically implies security with respect to Figure 2.

To prove this result, we will utilize insights similar to the ones presented in Theorem 1. Recall that Theorem 1 proved that after receiving the extractor seed, the optimal adversary for information-theoretic extractors simply checked whether any extractor output is a linear combination of the leakage bits received from the oracle. Note that this post-processing adversary may be executed in PPT adversary, which is the key reason why doubly-affine extractors may utilize computational seeds. This statement is not true in other general leakage models that prevent their usage of computational seeds. We present the proof of the following result in Appendix F.

Theorem 12. *If* **Ext** *is information-theoretically* $(\pounds, \varepsilon)$*-secure, then* **Ext** *is computationally* $(\pounds, \varepsilon + \varepsilon^t)$*-secure if the computationally-secure seed generator is a PPT algorithm that is secure against PPT adversaries except with probability* ε^t*.*

Therefore, our efficient information-theoretic extractors from Section 3.5 are also computational extractors that may be used with computational seeds.

5 Applications

In this section, we utilize our doubly-affine extractors to present information-theoretic encryption solutions. We critically leverage both restrictions of affine output and leakage in the following way. Let the random source X be the secret key. To encrypt a message M, one first samples a random seed S and produces the ciphertext $(S, \mathbf{Ext}(X, S) \oplus M)$. To decrypt a ciphertext (S, M^t), one can compute the value $M^t \oplus \mathbf{Ext}(X, S) =$

M . In the above protocol, an adversary with access to encryption/decryption oracles end up only learning extractor outputs that are affine. As a result, the adversary obtains only affine leakage about X. So, doubly-affine extractors end up being the perfect primitive for information-theoretic encryption. While one may also use seeded extractors with general output and leakage, we show our doubly-affine extractors result in better probe/cache complexity (see Section 6 for numerical comparisons).

5.1 Replicated Setting

We consider the problem with $k \geq 2$ parties that need reusable information-theoretic encryption to communicate multiple messages over potentially insecure channels. This may also be referred to as the *group communication* problem. For the case of $k = 2$, we note there is a simple stateful solution. Each party consumes the secret key X as one-time pads starting from different ends and stops when meeting in the middle. However, this solution does not scale well for $k > 2$ parties where all parties do not see all encrypted messages. The natural generalization is to split X into k parts for each party. This is sub-optimal in terms of utilizing X when some parties encrypt infrequently and other parties encrypt frequently. By using doubly-affine extractors, we avoid these issues.

IND-CCA1 Encryption. We show that the simple example presented earlier is already an IND-CCA1 secure for secret key X and messages M.

- $\mathbf{Enc}(X, M) = (S, M \oplus \mathbf{Ext}(X, S))$ with uniformly random S.
- $\mathbf{Dec}(X, (S, M^{\mathsf{t}})) = M^{\mathsf{t}} \oplus \mathbf{Ext}(X, S)$.

We prove security in a stronger variant of real-or-random challenges denoted by IND-CCA1\$. The adversary submits a challenge message M . The challenger returns either an encryption to M or a random string that must be distinguished by the adversary. Note, this requires that ciphertexts are indistinguishable from random strings. We formally prove define and prove security in Appendix G.1.

IND-CCA2 Authenticated Encryption. We present the first application of our computational doubly-affine extractors for constructing IT authenticated encryption (AE) that is IND-CCA2\$ secure. We assume the existence of a standard, computationally-secure AE with associated data (AEAD) scheme, $\mathbf{Enc}_{\mathbf{AEAD}}$ and $\mathbf{Dec}_{\mathbf{AEAD}}$. We assume that the secret key is $(X, K_{\mathbf{AEAD}})$ where $K_{\mathbf{AEAD}}$ is an AE secret key.

- $\mathbf{Enc}((X, K_{\mathbf{AEAD}}), M) = (\mathbf{Enc}_{\mathbf{AEAD}}(K_{\mathbf{AEAD}}, S, M^{\mathsf{t}}), M^{\mathsf{t}})$ with $M^{\mathsf{t}} = M \oplus \mathbf{Ext}(X, S)$ and uniformly random S.
- $\mathbf{Dec}((X, K_{\mathbf{AEAD}}), (S^{\mathsf{t}}, M^{\mathsf{t}})) = M^{\mathsf{t}} \oplus \mathbf{Ext}(X, \mathbf{Dec}_{\mathbf{AEAD}}(K_{\mathbf{AEAD}}, S^{\mathsf{t}}, M^{\mathsf{t}}))$ and rejecting whenever $\mathbf{Dec}_{\mathbf{AEAD}}(K_{\mathbf{AEAD}}, S^{\mathsf{t}}, M^{\mathsf{t}})$ fails authenticity verification.

It is straightforward to adapt our scheme to an IT AEAD by appending associated data with M^{t} in the above scheme. Our construction also achieves authenticity against any PPT adversaries from the underlying AEAD scheme.

For security, we adapt our real-or-random challenge from the prior sections where the adversary must distinguish between an encryption of an adversarially chosen message and a random string. For IND-CCA2\$, the adversary is able to make both encryption and decryption queries after the challenge. A computationally unbounded adversary in IND-CCA2\$ may trivially break the scheme. For the challenge ciphertext $(S^{\mathsf{t}}, M^{\mathsf{t}})$, the adversary sends a decryption query for ciphertext $(S^{\mathsf{t}}, M^{\mathsf{tt}})$ where $M^{\mathsf{tt}} \neq M^{\mathsf{t}}$ to receive $M^{\mathsf{tt}} \quad \mathbf{Ext}(X, S)$ where S is the encrypted seed. Therefore, the adversary computes $\mathbf{Ext}(X, S)$ and may break the scheme. Due to this limitation, we consider adversaries that are PPT when encryption/decryption queries are available but computationally unbounded afterwards. The security game and proof are presented in Appendix G.2.

5.2 Distributed Setting

We consider another setting where the random source X is jointly stored by $t \geq 2$ servers. Each server stores a disjoint subset of X. We will use the standard assumption from information-theoretic cryptography literature where the user will have a private channel with only a subset of $g < t$ trusted servers and the remaining $t - g$ channels may be compromised and/or the servers may be compromised and using faulty randomness. This is a common assumption that appears in many prior important works such as information-theoretic secret sharing [28], multi-party computation [8] and secure message transmission [13]. Note users do not know which servers are compromised. For simplicity, we assume each server stores an equal portion of X (our analysis is trivial to extend when this is not the case). If there are $t - g$ bad servers, a $(t - g)/t$ fraction of X will be leaked. Adversaries may also query the servers to learn affine leakage about the random source X.

We modify our extractors for the multi-server setting. Recall that our extractors first sample bits then apply a non-local extractor on sampled bits. With multiple servers, our extractor first executes the sampler portion with each server individually. Afterwards, the non-local extractor will be executed on all sampled bits to obtain the final extractor output. In terms of efficiency, we note that the virtual random source has only $\mathsf{n} := (g/t)|X|$ random bits excluding parts of X stored by the bad servers. Any (p, c)-local extractor will probe $(g/t)\mathsf{p}$ bits from good servers and the remaining sampled bits are assumed to be compromised already. Executing a non-local extractor over all sampled bits can output $(g/t)\mathsf{p} - \log(1/\varepsilon)$ random bits. By considering the multi-server setting, we must increase the probe complexity of our extractors by a multiplicative factor of (t/g). In particular, consider any extractor with probe complexity p in the replicated setting that outputs m bits. To obtain m output bits in the multi-server setting, our extractors must instead probe $(t/g)\mathsf{p}$ bits.

We emphasize that our doubly-affine extractors are easily amenable to the distributed setting. In particular, the distributed servers do not need any communication or coordination. The seeds that are sent to each server may be computationally-secure. To encrypt/decrypt a message M, the communication and computation from each server is sub-linear in the length of M. Our encryption schemes are easily adaptable to settings where servers may go offline and new servers join as well if servers are malicious. Most of these great properties are not be obtainable by other primitives when moving to distributed settings (such as MPC, secret sharing, etc).

As one requirement, servers must limit the leakage that may be obtained by an adversary. To do this, the servers may keep track of the number of unique bits that are sampled by all users. To limit adversaries to $\pounds$ leakage bits, the servers should stop responding after $\pounds/g$ unique queries. As a result, the g servers return at most $\pounds$ leakage bits. Note that keeping track of the unique sampled bits may be done very efficiently with small storage (using HyperLogLog [21] as an example).

Public-Key IND-CCA2 Encryption. With the server setting, there may be a need for public-key encryption as parties no longer share the secret key. We show that we may build public-key IND-CCA2$ encryption schemes using our computational doubly-affine extractors. We re-use our previous IND-CCA2$ definition with the only difference that the adversary obtains the public key before any encryption/decryption queries.

We will use any public-key encryption (PKE) scheme $(\mathbf{Enc_{PK}}, \mathbf{Dec_{PK}})$ with IND-CCA2 security against PPT adversaries with label support [30] where the label is not private but is required for decryption. Suppose that the PKE has key pair $(\mathbf{pk}, \mathbf{sk})$ and $\mathbf{Ext}(X, S)$ is computed using distributed variant of our doubly-affine extractors.

- $\mathbf{Enc}(X, \mathbf{pk}, M) = (\mathbf{Enc_{PK}}(\mathbf{pk}, S, M^t), M^t)$ with $M^t = M \oplus \mathbf{Ext}(X, S)$ and uniformly random S.
- $\mathbf{Dec}(X, \mathbf{sk}, (S^t, M^t)) = M^t \oplus \mathbf{Ext}(X, \mathbf{Dec_{PK}}(\mathbf{sk}, S^t, M^t))$.

We formally define and prove security in Appendix H.1.

Computationally-Secure Keys. In the server setting, parties no longer need to meet and share the secret key as they may generate shared randomness through the servers. We show that two parties may utilize

information-theoretic encryption schemes without meeting. First, the two parties use key exchange to agree on a computationally-secure seed S. To encrypt a message M, we may use any of the prior constructions where $\mathbf{Ext}(X, S)$ is computed using the servers.

Malicious Servers. In this last section, we consider when the $t - g$ corrupted servers are malicious. Malicious servers are able to answer in an arbitrary manner including no response, wrong responses and responses that are inconsistent across multiple requests. As an example, malicious servers may ignore sampler seeds and return random bits for each request. Therefore, malicious servers can force our extractor outputs to no longer be deterministic. For two queries with the same seeds, the outputs might be difference that will be problematic for many applications.

In this section, we present a solution to this problem. The main modification is to utilize Reed-Solomon codes to handle both errors (servers returning wrong responses) and erasures (servers not returning any response). Using a Reed-Solomon code that adds z check bits, the decoding algorithms may handle up to a errors and b erasures such that $2a + b \leq \lfloor z/2 \rfloor$. All p sampled bits will be encoded using a Reed-Solomon code. If there are at $t - g$ malicious servers, that means at most $(t - g)/t \cdot \mathsf{p}$ errors and/or erasures may occur in the sampled bits. Therefore, we choose to use a Reed-Solomon code with $z \geq 2(t - g)/t \cdot \mathsf{p}$ check bits. In applications, multiple parties will have to share both the same seeds as well as the z check bits to guarantee the same output.

As the z check bits generated by the Reed-Solomon code must be shared between multiple parties, we consider the z check bits to be public and, thus, available to the adversary. One reason we chose to use Reed-Solomon code is that that check bits are linear in the message and, thus, linear in the sampled bits. So, we may consider the leaked z check bits as general linear leakage obtained by the adversary. This requires several modifications to **DExt**. Note that our extractors require that the sampled bits from the g good servers have $\mathsf{m} + \log(2/\varepsilon)$ bits of residual entropy as the non-linear extractor will lose $\log(2/\varepsilon)$ bits of entropy. For malicious server setting, the z check bits will be publicly released. Therefore, we need the sampled bits from the good servers to have residual entropy of $\mathsf{m} + z + \log(2/\varepsilon)$ bits as we will lose z bits of entropy for the public check bits and another $\log(2/\varepsilon)$ from the non-linear extractor. Additionally, at most $t/3$ of the servers may be malicious.

6 Experimental Evaluation

We now analyze the efficiency of our extractors in several dimensions. In this section, we aim to answer several concrete questions. First, how do our extractors compare to previous works? Secondly, how efficient are our constructions with respect to randomness extraction? Finally, what is the relationship between probe and cache complexity in our constructions?

6.1 Comparison with Previous Works

We compare our extractors with those that appeared in previous works. The majority of previous works such as [14, 31] focused on asymptotic as opposed to concrete efficiency. We focus on two recent works that present concretely efficient (p, p)-local extractors [5] and (p, c)-local extractors [4]. We caveat that these extractors were built for general leakage as opposed to only linear leakage like our extractors. Therefore, we expect our constructions to be more efficient.

Both [5] and [4] assume the existence of a random oracle. The random oracle is utilized for both seed generation and non-local extraction. To make a fair comparison with our doubly-affine extractors, we replace random oracles with our non-local extractor (see Theorem 6) and require that the requisite seed length is provided as input. With these modifications, our constructions have smaller or identical seed lengths. The (p, c)-local extractor of [4] uses seeds of length $\mathsf{p} \log(\mathsf{n}/\mathsf{c})$ that is larger than our extractor's seed length of $\mathsf{c} \log(\mathsf{n}/\mathsf{c})$ (Theorem 9). Our (p, p)-local extractors in Theorem 11 use $\log \binom{\mathsf{n}}{\mathsf{p}}$ seeds that are identical to the ones used in [5]. We present numerical comparisons in Figures 6 and 7 verifying that our extractors are more efficient. For our comparison, we use security parameter $\varepsilon = 2^{-64}$. In all settings, our constructions extract more bits compared to both previous works when using the same probe and/or cache complexity.

p	Ours	[5]
500	309	210
1000	734	484
2000	1638	1031
4000	3399	2127
279	**128**	88
436	**256**	174
742	**512**	342
351	188	**128**
584	381	**256**
1052	775	**512**

p	Ours	[5]
500	82	20
1000	282	104
2000	721	272
4000	1723	608
621	**128**	40
934	**256**	93
1527	**512**	192
1142	343	**128**
1903	678	**256**
3426	1452	**512**

Figure 6: **Numerical examples, comparison with [5].** Both tables consider 100 GB random sources and $\varepsilon = 2^{-64}$. The left table considers 10% leakage and the right table 50% leakage. The leftmost columns denote the probe complexity p. The second and third column denote the number of random bits extracted.

c	**1**		**8**		**32**		**64**		**512**	
	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]
250	53	53	885	524	3738	695	7542	723	60796	748
500	234	189	2334	1130	9532	1471	19129	1572	153488	1575
1000	622	463	5436	2343	21942	3024	43950	3134	352057	3229
5000	3690	2655	31237	12048	128746	15445	257558	15994	2060924	16464
10000	8263	5395	66565	24178	266455	30972	532976	32068	4264226	33008

c	**1**		**8**		**32**		**64**		**512**	
	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]
250	0	0	85	90	538	146	1142	157	9596	168
500	34	0	734	262	3132	374	6329	396	51088	417
1000	222	84	2236	608	9142	831	18350	874	147257	914
5000	1960	757	16137	3371	64746	4482	129558	4697	1036924	4892
10000	4263	1598	34565	6825	138455	9046	276976	9475	2216266	9864

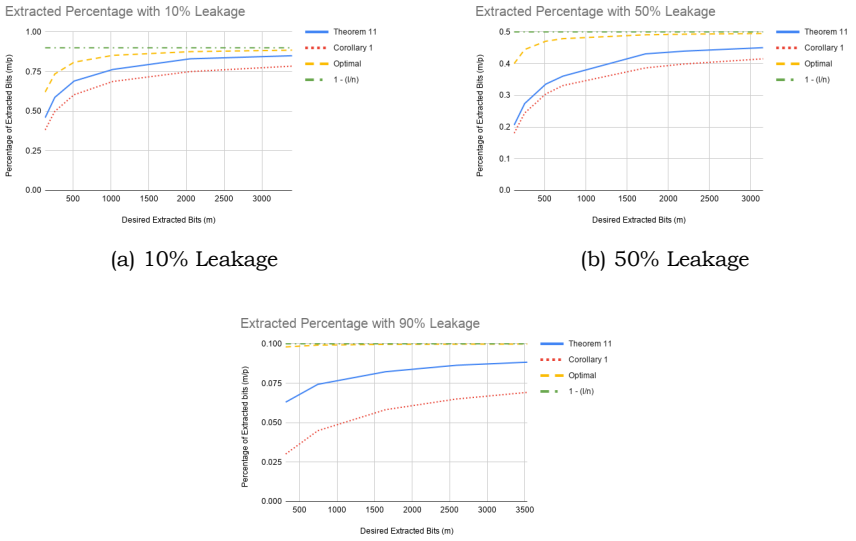
Figure 7: **Numerical examples, comparison with [4].** Both tables consider 100 GB random sources and $\varepsilon = 2^{-64}$. The left table considers 10% leakage and the right table 50% leakage. The leftmost column denotes the cache complexity c. The first row denotes the number of probed bits in each of the c groups. The remaining entries denote the number of random bits extracted.

6.2 Randomness Efficiency

Next, we evaluate the randomness efficiency of our constructions. Consider an extractor with probe complexity p and an adversary with ℓ leakage bits from a n-bit random source, Only a $(1 - \ell/n)$-fraction of the p probed bits will be random in the adversary's view. Our lower bound in Theorem 7 shows that at least $\log(1/\varepsilon)$ additional bits will be lost. Therefore, a theoretically optimal extractor would extract $(1 - \ell/n)p - \log(1/\varepsilon)$ bits. In Figure 8, we show that our extractors are concretely very close to the theoretically optimal extractor. We consider three settings where the adversary learns 10%, 50% and 90% of the random source (that is, $\ell/n \in \{0.1, 0.5, 0.9\}$). As the number of probed bits increase, the number of extracted bits approach the theoretical optimum. We also show that our (p, p)-local extractor (Theorem 11) outperforms our general (p, c)-local extractor (Theorem 9 using Corollary 1).

6.3 Probe And Cache Complexity

Finally, we investigate the cost of small cache complexity in terms of probe complexity. In particular, suppose we fix all parameters except cache and probe complexity. Does smaller cache complexity require larger probe complexity? We show this is the case in Figure 9 where the required probe complexity decreases exponentially as cache complexity increases. This justifies that cache complexity is an important efficiency dimension for extractors.



(a) 10% Leakage

(b) 50% Leakage

(c) 90% Leakage

Figure 8: Percentage of probed bits that are extracted for (p, p)-local extractor from Theorem 11 and Corollary 1 with a 100 GB random source and $\varepsilon = 2^{-64}$.

7 Conclusions

In this paper, we define the notion of *doubly-affine extractors* where both the output and leakage must be affine. Using these two restrictions, we present a series of reductions showing that optimal doubly-affine extractors end up being simple PPT algorithms that check intersection of linear subspaces. Using these insights, we show that doubly-affine extractors may tolerate *post- application leakage* and *computational seeds*, which are impossible in general leakage models. We present extractor constructions that are almost concretely optimal in terms of randomness usage, probe and cache complexity beating prior works. Finally, we show that the doubly-affine restrictions are perfect for the application of information-theoretic encryption. Using our concretely efficient extractors, we obtain state-of-the-art information-theoretic encryption.

References

[1] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO 2009*. 2009.

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

[3] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover hash lemma, revisited. In *Annual Cryptology Conference*, pages 1–20. Springer, 2011.

[4] M. Bellare and W. Dai. Defending against key exfiltration: Efficiency improvements for big-key cryptography via large-alphabet subkey prediction. In *CCS*, 2017.

[5] M. Bellare, D. Kane, and P. Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In *CRYPTO*, 2016.

[6] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, 2000.

[7] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *FOCS'94*.

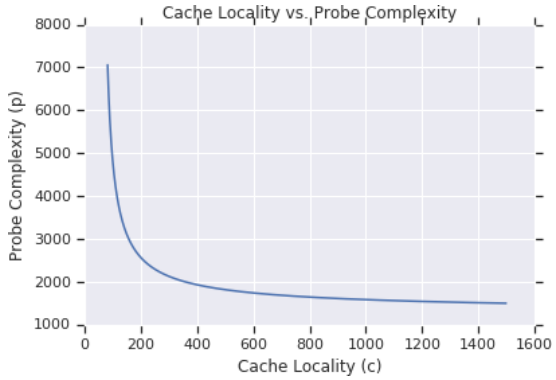


Figure 9: Probe complexity of extractor from Theorem 9 to produce $m = 1024$ bits with varying cache locality requirements and 10% leakage.

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, 1988.

[9] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 1995.

[10] Y. Dodis. Shannon impossibility, revisited. In *ICITS*, 2012.

[11] Y. Dodis, A. Elbaz, R. Oliveira, and R. Raz. Improved randomness extraction from two independent sources. In *APPROX-RANDOM*. 2004.

[12] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IEEE Transactions on Information Theory*, 58(9):6207–6222, 2012.

[13] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM (JACM)*, 40(1):17–47, 1993.

[14] S. Dziembowski and U. Maurer. Tight security proofs for the bounded-storage model. In *STOC*, 2002.

[15] A. Gabizon and R. Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 2008.

[16] M. Goemans. Chernoff bounds, and some applications. 2015.

[17] O. Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. 2011.

[18] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

[19] D. Harnik and M. Naor. On everlasting security in the hybrid bounded storage model. In *International Colloquium on Automata, Languages, and Programming*, pages 192–203. Springer, 2006.

[20] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 1999.

[21] S. Heule, M. Nunkesser, and A. Hall. Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *EDBT*, 2013.

[22] C.-J. Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 2004.

[23] U. M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 1992.

[24] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 1996.

[25] M. Ohsmann. Fast transforms of toeplitz matrices. *Linear algebra and its applications*, 231:181–192, 1995.

[26] J. Patarin. The “coefficients h” technique. In *SAC*, 2008.

[27] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 2000.

[28] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[29] C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 1949.

[30] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 2002.

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

[32] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 1997.

A The H-Coefficient Technique

In this section, we describe the *H-coefficient technique* [26] that will be useful for analyzing the best advantage of an adversary in distinguishing between real and ideal experiments.

We denote the *transcript* as the messages that are communicated between the challenger and adversary in a game. The H-coefficient technique partitions the set of all transcripts into *good* and *bad* sets. For good transcripts, a lower bound on the ratio of the probability of any fixed good transcript appearing in the real experiment compared to the probability of the good transcript appearing in the bad experiment. Bad transcripts are typically chosen where a lower bound on the ratio cannot be proven. Instead, we derive an upper bound on the probability that any bad transcript will ever appear. Furthermore, this bound will be done using the ideal experiment, which typically simplifies the proofs.

Consider a real game G_0 and an ideal game G_1. We denote $T(G_0)$ and $T(G_1)$ to be the distribution of transcripts that appear when simulating G_0 and G_1 respectively. Let τ be the set of all good transcripts. The H-coefficient technique states the following result:

Theorem 13 (H-Coefficient Technique [26])**.** *For two experiments G_0 and G_1, if there exists a set of transcripts τ and $\varepsilon, \delta \geq 0$ satisfying*

1. $\Pr[T(G_0) = t] / \Pr[T(G_1) = t] \geq 1 - \varepsilon$ *for all* $t \in \tau$ *;*
2. $\Pr[T(G_1) \notin \tau] \leq \delta$*;*

then the best distinguishing advantage for any adversary **A** *is at most* $\varepsilon + \delta$*.*

B Impossibility of Security against Post-Application Leakage with General Leakage

Theorem 14. *For models with general leakage, any extractor whose output of* m*-bits is longer than its input seed of* s*-bits will be generically secure against post-application leakage*

Proof. We present a simple counterexample. Consider the binary leakage function $f(X, R)$ where X is the uniformly random string and R are potential extractor outputs. Suppose the leakage function $f(X, R) = 1$ if and only if there exists a seed S such that the extractor given the seed S and the input string X outputs the string R. As the input seed length is shorter than the extractor output bit length, it is impossible for the extractor's output to be a uniformly random string when the adversary has access to the leakage $f(X, R)$. □

C Proof and Analysis of (p, p)-Sampler (Theorem 4)

Proof of Theorem 4. Fix any output of ℓ indices $T_1, \ldots, T_\ell$ from the adversary $\mathbf{A}$ and let $I_1, \ldots, I_p$ be the random p bits sampled by **Samp**. Denote the variable X as the intersection of the sampled and leakage subspaces

$$X := \dim(\mathbf{span}(R_{T_1}, \ldots, R_{T_\ell}) \cap \mathbf{span}(R_{I_1}, \ldots, R_{I_p}) = |\{T_1, \ldots, T_\ell\} \cap \{I_1, \ldots, I_p\}|.$$

We need to analyze $\Pr[X \geq p - m + 1]$, which we decompose in the following way:

$$\Pr[X \geq p - m + 1] = \sum_{z=p-m+1}^{p} \Pr[X = z].$$

To analyze $\Pr[X = z]$, note that there are $\binom{\ell}{z} \cdot \binom{n-\ell}{p-z}$ ways for the adversary learn z bits out of $\binom{n}{p}$ different ways for **Samp** to sample bits. Therefore, we get that

$$\sum_{z=p-m+1}^{p} \Pr[X = z] \leq \sum_{z=p-m+1}^{p} \frac{\binom{\ell}{z} \cdot \binom{n-\ell}{p-z}}{\binom{n}{p}} \leq m \cdot \frac{\binom{\ell}{p-m+1}\binom{n-\ell}{m-1}}{\binom{n}{p}}$$

as $\Pr[X = z]$ is maximized when $z = p - m + 1$. □

Next, we analytically compare our (p, p)-local sampler with the (p, c)-local sampler of Theorem 3 to show that the (p, p)-local sampler is more concretely efficient. To do this, we first set $p = c$ in Theorem 3 to get the following corollary.

Corollary 1. *For any $0 \leq \ell < n$ and $\lambda > 0$, there exists a (p, p)-local sampler that is $(\ell, \varepsilon, \lambda)$-secure with seed length $p \log(n/p)$ and probe complexity*

$$p \leq \frac{m}{1-\ell/n} + \frac{\ln(1/\varepsilon) + \sqrt{\ln^2(1/\varepsilon) + 4(1-\ell/n)m\ln(1/\varepsilon)}}{2(1-\ell/n)^2} \approx \frac{1}{1-\ell/n} \cdot \max\left(m, \frac{\ln(1/\varepsilon)}{1-\ell/n}\right).$$

Proof. By Theorem 3, we know that $p = \frac{m}{1-(\ell/n)-\sqrt{\ln(1/\varepsilon)/p}}$, which is a quadratic function in p. By solving the quadratic equation, we can determine the optimal value of p. By rearranging, we get that

$$\begin{aligned}
\sqrt{p\ln(1/\varepsilon)} &= p\left(1 - \frac{\ell}{n}\right) - m \\
p\ln(1/\varepsilon) &= p^2\left(1 - \frac{\ell}{n}\right)^2 - p \cdot 2m\left(1 - \frac{\ell}{n}\right) + m^2 \\
0 &= p^2\left(1 - \frac{\ell}{n}\right)^2 - p \cdot \left(2m\left(1 - \frac{\ell}{n}\right) + \ln(1/\varepsilon)\right) + m^2
\end{aligned}$$

Note, the above requires that $p(1-\ell/n) - m \geq 0$. In other words, it must be the case that $p \geq m/(1-\ell/n)$. We will utilize this condition later to pick the correct root when solving the quadratic equation. After solving the quadratic equation, we get

$$p = \frac{2m(1-\ell/n) + \ln(1/\varepsilon) \pm \sqrt{(2m(1-\ell/n) + \ln(1/\varepsilon))^2 - 4(1-\ell/n)^2 m^2}}{2(1-\ell/n)^2}$$

If we choose the negative sign, we note that $p < m/(1-\ell/n)$ which is not allowed. Therefore, we have to pick the positive sign. After re-arranging, we can see that

$$\begin{aligned}
p &= \frac{m}{(1-\ell/n)} + \frac{\ln(1/\varepsilon) + \sqrt{\ln^2(1/\varepsilon) + 4(1-\ell/n)m\ln(1/\varepsilon)}}{2(1-\ell/n)^2} \\
&\leq \frac{m}{1-\ell/n} + \frac{\ln(1/\varepsilon)}{(1-\ell/n)^2} + \sqrt{\frac{m\ln(1/\varepsilon)}{(1-\ell/n)^3}}
\end{aligned}$$

where the last inequality used $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. □

We numerically analyze the two different samplers in Section 6 where we show that Theorem 4 is better than using the above corollary deriving from Theorem 3.

D Non-Local Doubly-Affine Extractor Proofs

We start by presenting two instantiations of the special family of matrices from field multiplication with smaller seed length but larger computational costs compared to our Toeplitz matrix instantiation. Then, we present our lower bound. Finally, we present an existential construction with very short seed length, but leave it as an open question to construct explicitly.

Special Matrix Family from Field Multiplication. We present one instantiation of the matrix set with $\mathsf{s} = \mathsf{n}$. For our construction, we consider the Galois field of size n, $GF[2^{\mathsf{n}}]$ and let $p(x)$ be an irreducible polynomial of degree n that generates $GF[2^{\mathsf{n}}]$. In particular,

$$p(x) = x^{\mathsf{n}} + p_{\mathsf{n}-1}x^{\mathsf{n}-1} + \ldots p_1 x + 1 \mod 2.$$

We represent $p(x)$ using the n-dimensional vector $\vec{p} = (p_{\mathsf{n}-1}, \ldots, p_1, 1)$ so that

$$p(x) = (1, \vec{p}) \cdot (x^{\mathsf{n}}, x^{\mathsf{n}-1}, \ldots, x, 1) \mod 2.$$

We will define the matrix P which represents the operation of multiplying a polynomial by x modulo $p(x)$. So, for any seed $S \in \{0,1\}^{\mathsf{n}}$:

$$PS = (S_2, S_3, \ldots, S_n, 0) \oplus S_1 \vec{p}\,.$$

Note, that the result of PS is linear, so it easy to construct such a matrix P. We define the set of matrices $P_1, \ldots, P_{\mathsf{m}}$ by setting $P_i = P^{i-1}$. This construction also appears in [11] as well as the proof of the following theorem.

Theorem 15. *For any subset* $\emptyset = \mathsf{I} \subseteq [1, \ldots, \mathsf{m}]$, $P_{\mathsf{I}} = \bigoplus_{i \in \mathsf{I}} P_i$ *has rank* n.

Note, the computation based on $P_1, \ldots, P_{\mathsf{m}}$ is very efficient consisting of only cyclic shifts and XOR operations on average half the time. However, this assumes that one already has an irreducible polynomial $p(x)$ of degree n. For large n, generating these irreducible polynomials may be computationally expensive and difficult.

Lower Bound for Non-Local Extractors. We present our lower bound on non-local extractors (Theorem 7).

Proof of Theorem 7. Consider $\mathfrak{L}$ random but linearly independent leakage vectors $\mathbf{t}_1, \ldots, \mathbf{t}_{\mathfrak{L}}$. For any seed S, consider the m-dimensional extracted subspace A_S. Then, we get that over the random choices of $\mathbf{t}_1, \ldots, \mathbf{t}_{\mathfrak{L}}$:

$$\begin{aligned}
&\Pr_{\mathbf{t}_1,\ldots,\mathbf{t}_{\mathfrak{L}}}[\mathbf{span}(A_S) \cap \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_{\mathfrak{L}}) = \{\vec{0}\}] \\
&\quad = (1 - 2^{\mathsf{m}-\mathsf{n}})(1 - 2^{\mathsf{m}+1-\mathsf{n}}) \cdot \cdots \cdot (1 - 2^{\mathsf{m}+\mathfrak{L}-\mathsf{n}}) \\
&\quad = e^{-2^{\mathsf{m}-\mathsf{n}}(2^{\mathsf{m}}+2^{\mathsf{m}+1}+\ldots+2^{\mathsf{m}+\mathfrak{L}})} \\
&\quad = e^{-2^{\mathsf{m}-\mathsf{n}}(2^{\mathfrak{L}+1}-1)} \\
&\quad \leq e^{-2^{\mathsf{m}-\mathsf{n}+\mathfrak{L}-O(1)}}.
\end{aligned}$$

Towards a contradiction, suppose that $\mathsf{m} > \mathsf{n} - \mathfrak{L} - \log(1/\varepsilon) + O(1)$. As a result, we know that $\mathsf{m} - \mathsf{n} + \mathfrak{L} - O(1) > \log(\varepsilon)$. Plugging this into the above formula, we get that for any fixed seed S

$$\Pr_{\mathbf{t}_1,\ldots,\mathbf{t}_{\mathfrak{L}}}[\mathbf{span}(A_S) \cap \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_{\mathfrak{L}}) = \{\vec{0}\}] < e^{-2^{\log(\varepsilon)}} = e^{-\varepsilon} < 1 - \varepsilon$$

for small enough $\varepsilon > 0$. As this is true for every fixed seed S, it is also true over any distribution over seeds as well. In particular,

$$\Pr_{S,\mathbf{t}_1,\ldots,\mathbf{t}_\ell}[\mathbf{span}(A_S) \cap \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell) = \{\vec{0}\}] < 1 - \varepsilon$$

over the random choice of S and $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$. Finally, this implies that

$$\mathsf{E}_{\mathbf{t}_1,\ldots,\mathbf{t}_\ell}[\Pr_S[\mathbf{span}(A_S) \cap \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell) = \{\vec{0}\}]] < 1 - \varepsilon$$

which means there exists bad leakage vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ such that at least one of the m output bits are not random with probability greater than ε providing the desired contradiction. □

Existence of Almost Optimal Extractor. Finally, we show that there exists a non-local extractor with very small entropy loss and seed length. However, we can only show the existence of such an extractor as opposed to a concrete instantiation at this point in time.

Theorem 16. *There exists a non-local extractor* $\mathbf{NLExt} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ *that extracts* $m = n - \ell - \log(1/\varepsilon) - O(1)$ *bits with a seed length of* $s = \log(n\ell/\varepsilon) \leq 2\log(n/\varepsilon)$.

Proof. To do this, we consider $D := 2^s$ random matrices $A_1, \ldots, A_D$ of dimension $n \times m$ representing all possible seeds. We now bound the probability that there exists ℓ leakage vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ that compromise an ε-fraction of the D possible seeds.

$$\begin{aligned}
&\Pr[\exists \mathbf{t}_1, \ldots, \mathbf{t}_\ell \text{ that compromise } \varepsilon D \text{ seeds}] \\
&\leq \sum_{\mathbf{t}_1,\ldots,\mathbf{t}_\ell \subseteq \{0,1\}^n} \Pr[\mathbf{t}_1, \ldots, \mathbf{t}_\ell \text{ compromises } \varepsilon D \text{ seeds}] \\
&\leq \binom{2^n}{\varepsilon D} \left(\frac{1}{2^n}\right)^{\varepsilon D} \Pr[\exists A_i \text{ such that } \mathbf{t}_1, \ldots, \mathbf{t}_\ell \text{ compromises } A_i] \\
&\leq \binom{2^n}{\varepsilon D} \left(\frac{1}{2^n}\right)^{\varepsilon D} \left(2^{\ell-n} + 2^{\ell+1-n} + \ldots + 2^{\ell+m-n}\right)^{\varepsilon D} \\
&\leq 2^{n\ell} \cdot (e/\varepsilon)^{\varepsilon D} \cdot \left(2^{\ell+m-n}\right)^{\varepsilon D} \\
&= 2^{n\ell + \varepsilon D(\log(e/\varepsilon) - n + \ell + m)}.
\end{aligned}$$

We note that the second inequality comes from the fact that the probability a ℓ-dimensional leakage subspace intersects the m-dimensional extracted subspace of A_i is at most $2^{\ell-n} + 2^{\ell+1-n} + \ldots + 2^{\ell+m-n}$.

Note, we require that $\log(e/\varepsilon) < n - \ell - m$ which implies that the number of extracted bits is at most $m < n - \ell - \log(1/\varepsilon) - O(1)$.

Additionally, it must be that $n\ell < \varepsilon D(n - \ell - m - \log(e/\varepsilon))$ So, we know that $D = \Omega(n\ell/\varepsilon)$ implying that the seed length must be $s = \Omega(\log(n\ell/\varepsilon))$ bits. If we consider the natural setting where $\ell = n/c$ for some constant $c > 0$, then $s = \Omega(\log(\ell/\varepsilon))$. As a result, there exists D matrices $A_1, \ldots, A_D$ that specify a non-local extractors with the desired properties. □

E Local Doubly-Affine Extractor Proofs

Sample-then-Extract Proof. We prove the sample-then-extract theorem used by our local extractor constructions.

Proof of Theorem 8. Let's consider an adversary $\mathbf{A}$ for $G^{\oplus}(\mathbf{Ext}, n, s_{\mathbf{Samp}} + s_{\mathbf{NLExt}}, m, \ell, \mathbf{A})$. We will show that $\mathbf{A}$ has advantage at most $\varepsilon_{\mathbf{Samp}} + \varepsilon_{\mathbf{Ext}}$. Consider the security of $\mathbf{Samp}$ and the game $G^{\oplus,\lambda}(\mathbf{Samp}, n, s_{\mathbf{Samp}}, p, \ell, \mathbf{A})$. Suppose that $\mathbf{A}$ chooses $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$ as leakage vectors. For convenience, denote $L := \mathbf{span}(\mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ and $S := \mathbf{span}(R_{I_1}, \ldots, R_{I_p})$ where $I_1, \ldots, I_p$ are the row vectors outputted by $\mathbf{Pick}_{\mathbf{Samp}}(S)$. Except with probability $\varepsilon_{\mathbf{Samp}}$, we know that $\dim(L \cap S) \leq p - \lambda$. Let us suppose that $\dim(L \cap S) \leq p - \lambda$. In this case, we

can reduce the adversary to winning the game $G^{\oplus}(\mathbf{NLExt}, \mathsf{p}, s_{\mathbf{NLExt}}, \mathsf{m}, \lambda, \mathbf{A})$ as the adversary $\mathbf{A}$ may ask at most $\mathsf{p} - \lambda$ leakage vectors about $X_{\mathbf{Samp}(S)}$. By Theorem 6, we know $\mathbf{A}$ has advantage at most $\varepsilon_{\mathbf{NLExt}}$. Therefore, $\mathbf{A}$ has advantage at most $\varepsilon_{\mathbf{Samp}} + \varepsilon_{\mathbf{NLExt}}$ as required. Since **NLExt** only operates over sampled bits, the resulting extractor remains (p, c)-local. □

Local Extractor Lower Bounds. We present a lower bound on cache complexity, c, of any local extractor. In particular, we prove a lower bound single-bit extractors (so they also apply for all extractors).

Proof of Theorem 10. Consider the following adversary that picks $£/\mathsf{p}$ locations in the n-bit source X uniformly at random. For each of $£/\mathsf{p}$ locations, the adversary queries and recovers the $£$ bits following and including the chosen location. Therefore, the adversary queries at most $£$ times. such that the adversary learns $£/\mathsf{p}$ groups of exactly $£$ consecutive bits.

Since **Ext** is (p, c)-local, the single output bit is a deterministic function of at most c groups where each group consists of at most $£$ bits each. Suppose **Ext** queries $1 \le i \le \mathsf{c}$ consecutive groups. The probability that the adversary chose a group starting at the same location as each of these i consecutive groups is $\binom{£/\mathsf{p}}{i} / \binom{\mathsf{n}}{i}$. This probability is minimized by setting $i = \mathsf{c}$. Therefore, there exists an adversary with advantage at least $\varepsilon \ge \binom{£/\mathsf{p}}{\mathsf{c}} / \binom{\mathsf{n}}{\mathsf{c}}$. Note,

$$\frac{\binom{£/\mathsf{p}}{\mathsf{c}}}{\binom{\mathsf{n}}{\mathsf{c}}} = \frac{(£/\mathsf{p}) \cdot (£/\mathsf{p} - 1) \cdot \cdots \cdot (£/\mathsf{p} - \mathsf{c} + 1)}{\mathsf{n} \cdot (\mathsf{n} - 1) \cdot \cdots \cdot (\mathsf{n} - \mathsf{c} + 1)} \ge \left(\frac{£}{\mathsf{pn}} - \frac{\mathsf{c}}{\mathsf{n}} \right)^{\mathsf{c}}.$$

As typically $\mathsf{c} \ll £/\mathsf{p}$, we get $\varepsilon \ge (\frac{£}{\mathsf{p}})^{\mathsf{c}}$ or $\mathsf{c} = \Omega(\log(1/\varepsilon)/\log(\mathsf{pn}/£))$. □

F Computational Doubly-Affine Extractors

Proof of Theorem 12. Towards a contradiction, assume there exists an adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ with advantage strictly greater than $\varepsilon + \varepsilon^t$. We will convert this into a PPT adversary $\mathbf{A}_\Sigma$ for the computational seed generator Σ with advantage strictly greater than ε^t.

Consider the modified version of game G^{c} in Figure 2 where the seed S is uniformly random as opposed to being passed by Σ. Note this game is identical to game G in Figure 1 except that the leakage Z is also given to $\mathbf{A}_1$. Note that we may essentially repeat the proof of Theorem 1 to show that there exists an optimal PPT adversary $\mathbf{A}_2$ with advantage at most ε. The only difference is that $\mathbf{A}_2$ is given the seed S^t to Σ as opposed to the computational seed S itself. However, $\mathbf{A}_2$ may recover S by executing $\Sigma(S^t)$ as Σ is assumed to be PPT. Therefore, if the computational seed S is indistinguishable from a random seed, then $\mathbf{A}$ has advantage at most ε.

Since the assumption is that $\mathbf{A}$ has advantage strictly greater than $\varepsilon + \varepsilon^t$, it must be that $\mathbf{A}$ is able to distinguish between the computational seed and a random string with probability strictly greater than ε^t. With the usage of the optimal PPT $\mathbf{A}_2$ adversary, $\mathbf{A}$ is now a PPT adversary. Therefore, there exists a PPT adversary to break Σ with probability strictly larger than ε^t providing the contradiction. □

G Replicated Setting

In this section, we formally define and prove security of our encryption schemes from Section 5.1. For our definitions of IND-CCA1$ and IND-CCA2$ security, we will suppose that adversaries may perform at most $£_1$ oracle queries prior to the challenge and at most $£_2$ oracle queries after the challenge. We say that a secure is $(£_1, £_2, \varepsilon)$-secure if the protocol is secure against an adversary with such oracle query upper bounds except with ε probability. An encryption scheme is (p, c)-local if it probes at most p bits of the secret key X that may be arranged into at most c consecutive groups.

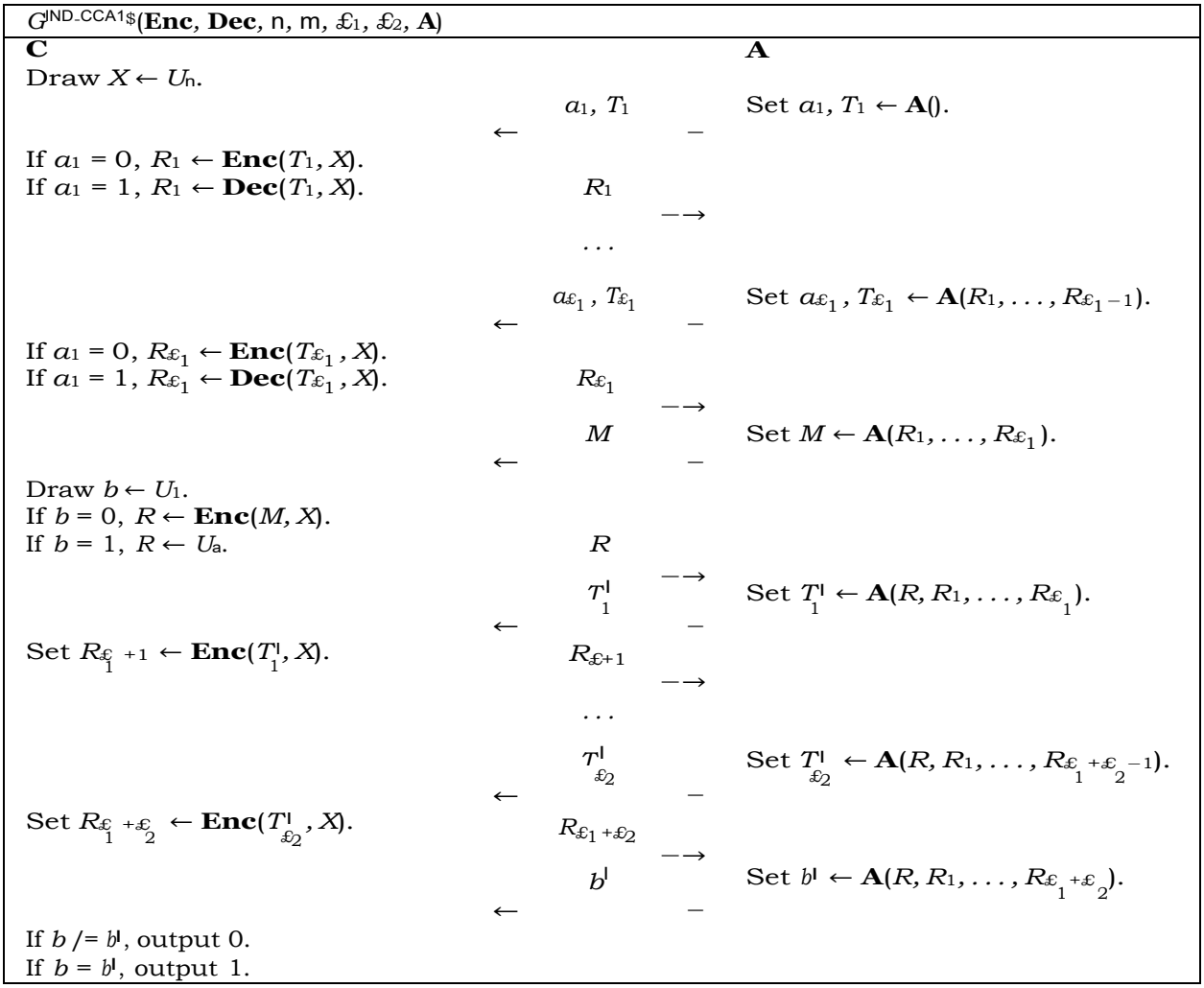


Figure 10: Game $G^{\text{IND-CCA1\$}}(\mathbf{Enc}, \mathbf{Dec}, \mathsf{n}, \mathsf{m}, \pounds_1, \pounds_2, \mathbf{A})$.

G.1 IND-CCA1$ Encryption

The IND-CCA1$ security game is presented in Figure 10. We present the following theorem.

Theorem 17. *Let* **Ext** *be* (p, c)*-local extractor that is* $(\pounds, \varepsilon)$*-secure. For any* $\pounds_1, \pounds_2 \geq 0$ *such that* $\pounds \leq \mathsf{m}(\pounds_1 + \pounds_2)$*, there exists an* (p, c)*-local IND-CCA1$ encryption scheme that is* $(\pounds_1, \pounds_2, \varepsilon)$*-secure.*

Proof. For our above construction, if an adversary sends a decryption oracle query (S_i, C_i), the adversary can retrieve the value $C \oplus \mathbf{Ext}(X, S_i)$. So, the adversary can retrieve $\mathbf{Ext}(X, S_i)$ for the chosen seed S_i. For encryption oracle queries, the adversary sends a message M_i and will receive $(S_i, M \oplus \mathbf{Ext}(X, S_i))$ for a $S_i \leftarrow U_{\mathsf{s}}$. So, the adversary learns $\mathbf{Ext}(X, S_i)$ for a seed chosen uniformly at random. Without loss of generality, we may assume that the adversary always performs $\pounds$ decryption oracle queries before the challenge.

Towards a contradiction, suppose there exists $\mathbf{A}^{\text{IND-CCA1\$}}$ that wins game $G^{\text{IND-CCA1\$}}$ against $(\mathbf{Enc}, \mathbf{Dec})$ with probability larger than ε. We construct $\mathbf{A}$ that wins game G against **Ext** with probability greater than ε. $\mathbf{A}$ executes $\mathbf{A}^{\text{IND-CCA1\$}}$ to get decryption oracle queries (S_i, C_i). $\mathbf{A}$ forwards the m row vectors of S_i to get $\mathbf{Ext}(X, S_i)$ and computes $\mathbf{Dec}((S_i, C_i), X) = C_i \oplus \mathbf{Ext}(X, S_i)$ that is sent back to $\mathbf{A}^{\text{IND-CCA1\$}}$. For the

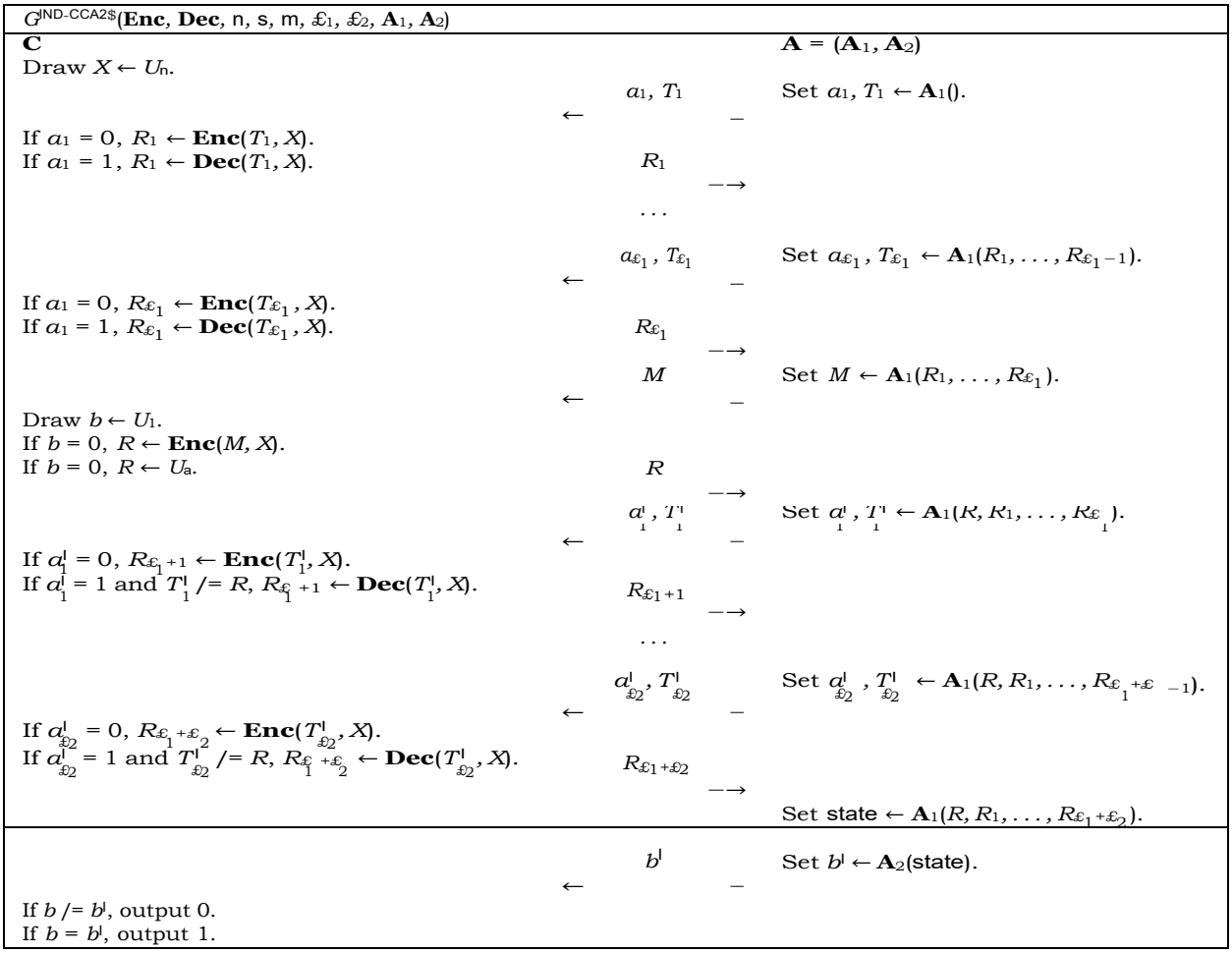


Figure 11: Game $G^{\text{IND-CCA2\$}}(\mathbf{Enc}, \mathbf{Dec}, \mathsf{n}, \mathsf{s}, \mathsf{m}, £_1, £_2, \mathbf{A}_1, \mathbf{A}_2)$.

challenge, $\mathbf{A}$ receives M from $\mathbf{A}^{\text{IND-CCA1\$}}$. Furthermore, $\mathbf{A}$ receives the challenge seed S and the value R that is either uniformly random or $\mathbf{Ext}(X, S)$. $\mathbf{A}$ computes $(S, M \oplus R)$ and sends it to $\mathbf{A}^{\text{IND-CCA1\$}}$. Finally, $\mathbf{A}$ receives b^t from $\mathbf{A}^{\text{IND-CCA1\$}}$ and outputs b^t.

With probability $1/2$, we know that $R = \mathbf{Ext}(X, S)$. Then, $\mathbf{Enc}(M, X) = (S, M \oplus \mathbf{Ext}(X, S))$ with probability $1/2$. With the remaining $1/2$ probability, R is uniformly random, so $(S, M \oplus R)$ is also uniformly random as S is always uniformly random. Therefore, $\mathbf{A}$ wins with probability at least ε. Since $\mathbf{A}^{\text{IND-CCA1\$}}$ sends the same answer, $\mathbf{A}^{\text{IND-CCA1\$}}$ also wins with probability ε. □

G.2 IND-CCA2$ Authenticated Encryption

Our security game of IND-CCA2$ security is found in Figure 11. We prove the following theorem about our AE scheme.

Theorem 18. *Let* $\mathbf{Ext}$ *be a* (p, c)*-local extractor that is* $(£, \varepsilon)$*-secure and* $(\mathbf{Enc}_{\text{AEAD}}, \mathbf{Dec}_{\text{AEAD}})$ *be a computationally-secure AEAD scheme that is both IND-CCA2$ secure and provides ciphertext integrity (CXT-INT) except with probability* ε^t*. For any* $£_1, £_2 \geq 0$ *such that* $£ \leq \mathsf{m}(£_1 + £_2)$*, there exists an* (p, c)*-local IND-CCA2$ encryption scheme that is* $(£_1, £_2, \varepsilon + \varepsilon^t)$*-secure. Furthermore, the scheme provides ciphertext integrity CXT-INT with respect to PPT adversaries.*

Proof. If a PPT adversary can find a (C, P) such that **Dec** properly decrypts implies finding a (C, P) such that $\mathbf{Dec}_{\mathbf{AEAD}}(K_{\mathbf{AEAD}}, C, P)$ decrypts properly. So, computational CXT-INT follows directly from the AEAD scheme.

To show that (**Enc**, **Dec**) is an IND-CCA2$ encryption scheme, we first use the standard observation [6] (which holds with hybrid attackers) that CXT-INT property shown above means that we can ignore all decryption queries (still made when $\mathbf{A}$ is PPT), and effectively show security against an IND-CPA variant of the security game in Figure 11. Next, we replace the second stage unbounded attacker $\mathbf{A}_2$ with the optimal PPT adversary described in Section 3.1. We omit the proof as it follows similarly to Theorem 1 and Theorem 12.

Now that our overall attacker $\mathbf{A}$ is PPT, we present a series of hybrid games to show that the attacker's advantage in distinguishing $b = 0$ from $b = 1$ is at most $\varepsilon + \varepsilon^{t}$. In the resulting game where $b = 0$ denoted by G_0, we know that $R = (\mathbf{Enc}_{\mathbf{AEAD}}(K_{\mathbf{AEAD}}, S, P), P)$ such that $P = M \oplus \mathbf{Ext}(X, S)$. Consider the game G_1 where $R = (C, P)$ and $P = M \oplus \mathbf{Ext}(X, S)$ such that C is a uniformly random string. Note, G_0 and G_1 are indistinguishable with probability at most ε^{t} by the fact that the underlying AEAD scheme is computationally IND-CPA$ secure, and our adversary is PPT. Consider the next hybrid game G_2 where we replace $P = M \oplus \mathbf{Ext}(X, S)$ with P that is chosen uniformly at random. Since **Ext** is an $(\pounds, \varepsilon)$-secure extractor and all encryption oracles queries are equivalent to m oracle queries, games G_1 and G_2 are indistinguishable except with probability at most ε. Finally, note that G_2 is equivalent to our final game where $b = 1$, as $R = (C, P)$ is chosen uniformly at random. □

H Distributed Setting

In this setting, we prove the security of our public-key IND-CCA2$ encryption scheme. Furthermore, we present usage of error-correcting codes to deal with malicious servers. Recall there are t total servers and g good servers (thus, $t - g$ bad servers).

H.1 Public-Key IND-CCA2$ Encryption

The security game for our IND-CCA2$ security notion is found in Figure 11. We prove the security of our public-key scheme from Section 5.2.

Theorem 19. *Let* **Ext** *be a* (p, c)*-local extractor that is* $(\pounds, \varepsilon)$*-secure and* $(\mathbf{Enc}_{\mathbf{PK}}, \mathbf{Dec}_{\mathbf{PK}})$ *be a computationally-secure public-key encryption scheme that is IND-CCA2$ secure with label support except with probability* ε^{t}*. For any* $\pounds_1, \pounds_2 \geq 0$ *such that* $\pounds \leq \mathsf{m}(\pounds_1 + \pounds_2)$*, there exists an* (p, c)*-local IND-CCA2$ encryption scheme that is* $(\pounds_1, \pounds_2, \varepsilon + \varepsilon^{t})$*-secure.*

Proof. The proof follows identically to the proof of Theorem 18 with the modification that the decryption oracle for the underlying public-key scheme must be used instead of the AEAD decryption oracle. □