Somewhere Randomness Extraction and Security against Bounded-Storage Mass Surveillance

Jiaxin Guan
Princeton University
jiaxin@guan.io

Daniel Wichs
Northeastern University & NTT Research
danwichs@gmail.com

Mark Zhandry
NTT Research
mzhandry@gmail.com

Abstract

Consider a state-level adversary who observes and stores large amounts of encrypted data from all users on the Internet, but does not have the capacity to store it all. Later, it may target certain "persons of interest" in order to obtain their decryption keys. We would like to guarantee that, if the adversary's storage capacity is only (say) 1% of the total encrypted data size, then even if it can later obtain the decryption keys of arbitrary users, it can only learn something about the contents of (roughly) 1% of the ciphertexts, while the rest will maintain full security. This can be seen as an extension of *incompressible cryptography* (Dziembowski CRYPTO '06, Guan, Wichs and Zhandry EUROCRYPT '22) to the *multi-user* setting. We provide solutions in both the symmetric key and public key setting with various trade-offs in terms of computational assumptions and efficiency.

As the core technical tool, we study an information-theoretic problem which we refer to as "somewhere randomness extraction". Suppose $X_1, \ldots, X_t$ are correlated random variables whose total joint min-entropy rate is α, but we know nothing else about their individual entropies. We choose t random and independent seeds $S_1, \ldots, S_t$ and attempt to individually extract some small amount of randomness $Y_i = \mathsf{Ext}(X_i; S_i)$ from each X_i. We'd like to say that roughly an α-fraction of the extracted outputs Y_i should be indistinguishable from uniform even given all the remaining extracted outputs and all the seeds. We show that this indeed holds for specific extractors based on Hadamard and Reed-Muller codes.

1 Introduction

Bounded-Storage Mass Surveillance. We consider a scenario where a powerful (e.g., state-level) adversary wants to perform mass surveillance of the population. Even if the population uses encryption to secure all communication, the adversary can collect large amounts of encrypted data from the users (e.g., by monitoring encrypted traffic on the Internet). The data is encrypted and hence the adversary does not learn anything about its contents when it is collected. However, the adversary may store this data for the future. Later, it may identify various "persons of interest" and perform expensive targeted attacks to get their secret keys (e.g., by remote hacking or by physically compromising their devices). We will assume the adversary is capable of eventually getting any secret key of any user of its choosing. Can we still achieve any meaningful notion of security against such mass-surveillance?

One option is to rely on cryptosystems having *forward secrecy* [Gün90], which exactly addresses the problem of maintaining security even if the secret key is later compromised. Unfortunately, forward-secure encryption schemes inherently require either multi-round interaction between the sender and receiver or for the receiver to perform key updates, both of which can be impractical or impossible in many natural scenarios. Without these, it may seem that no reasonable security is possible – if the adversary collects all the ciphertexts and later can get any secret key, clearly it can also get any plaintext!

In this work, we restrict the adversary to have *bounded storage*, which is much smaller than the total of size of all the encrypted data it can observe. This is a reasonable assumption since the total communication of an entire population is likely huge.[1] As a running example throughout the introduction, we will assume that the adversary's storage capacity is 1% of the total encrypted data size. We allow the adversary to observe all the encrypted data simultaneously and then compress it in some arbitrary way to fit within its storage budget. Later, the adversary can get any secret key of any user of its choosing, and eventually it may even get all the keys of all the users. What kind of security guarantees can we provide in this setting?

Clearly, the adversary can simply store 1% of the ciphertexts and discard the remaining 99%, which will allow it to later compromise the security of 1% of the users by getting their secret keys. While one may pessimistically see this as a significant privacy violation already, we optimistically regard this as a potentially reasonable privacy outcome that's vastly preferable to the adversary being able to compromise all the users. For example, if the adversary later chooses a random user and wants to learn something about their data, it will only be able to do so with 1% probability, even if it can get their secret key. But can we argue that this is the best that the adversary can do? In particular, we'd like to say that, no mater what compression strategy the adversary employs, it will be unable to learn anything about the contents of 99% of the ciphertexts, even if it later gets all the secret keys. Unfortunately, this is not generically true. For example, the adversary could store the first 1% of the bits of every ciphertext. If the encryption scheme is (e.g.,) the one-time pad, then an adversary who later learns the secret keys would later be able to learn the first 1% of every encrypted message of every user, which may provide a pretty

[1] Global annual Internet traffic has long surpassed 1 zettabyte (10^{21} bytes) [BJ], while *total* world-wide datacenter storage is only a couple zettabytes in 2022 [Dep].

good idea of the overall message contents. In fact, it can get even worse than this. If the encryption scheme is fully homomorphic, the adversary can individually compress each ciphertext into a small evaluated ciphertext encrypting some arbitrary predicate of the data (e.g., was the message insulting of the supreme leader), and therefore learn the outcome of this predicate about the encrypted data of every user. Even worse, if the encryption scheme is multi-key fully homomorphic, the adversary can derive a compressed ciphertext that encrypts the output of a joint computation over all the data of all the users, as long as the output is sufficiently small. Thus, in general, an adversary whose storage capacity is only 1%, may still be able to learn some partial information about the encrypted messages of a 100% of the users. The question is then, whether or not it is indeed possible to guarantee only 1% of users are compromised, and if so to actually design such a scheme.

Connection to Incompressible Cryptography. Encryption schemes that offer protection against bounded-storage mass surveillance can be seen as a generalization of *incompressible encryption* [Dzi06, GWZ22, BDD22] to the setting of multiple ciphertexts. To clarify the distinction, we refer to the earlier notion of incompressible encryption as *individually incompressible* and our new notion as *multi-incompressible*.

In an *individually incompressible encryption* scheme, we can make the size of a ciphertext flexibly large, and potentially huge (e.g., many gigabytes). An adversary observes a single ciphertext, but cannot store it in its entirety and can instead only store some compressed version of it. Security dictates that even if the adversary later gets the user's secret key, it cannot learn anything about the encrypted message. The work of [Dzi06] gave a construction of one-time symmetric-key encryption with information-theoretic security in this setting, and the work of [GWZ22] showed how to achieve public-key encryption in this setting, under the minimal assumption that standard public-key encryption exists. The works of [GWZ22, BDD22] also constructed such public-key encryption schemes having rate 1, meaning that the size of the message can be almost as large as the ciphertext size, and the latter work even showed how to do so under specific but standard public-key assumptions.

In our new notion of *multi-incompressible encryption*, we also have the flexibility to make the ciphertext size arbitrarily large. But now the adversary observes a large number of ciphertexts from many users and compresses them down to something that's roughly an α-fraction of the size of all the original ciphertexts, for some α. In particular, the adversary's storage may be much larger than a single ciphertext. Later the adversary gets all the secret keys, and we want to say that the adversary can only learn something about a (roughly) α-fraction of the messages, but cannot learn anything about the rest.

Our new notion of multi-incompressibility implies individual incompressibility. In particular, in the case of a single ciphertext, unless the adversary stores essentially all of it (i.e., $\alpha \approx 1$), it cannot learn anything about the encrypted message (= 100% of the messages). But our notion is significantly more general. For example, individual incompressibility does not even offer any guarantees if an adversary can take even 2 ciphertexts and compress them down to the size of 1, while multi-incompressibility ensures that one of the messages stays secure.

Formalizing multi-incompressibility is tricky: the natural indistinguishability-based approach would be to insist that the encryptions of two lists of messages are indistin-

guishable. But unlike individually incompressible encryption, in our setting the adversary can always learn *something*, namely the messages contained in ciphertexts it chose to store. We therefore need a fine-grained notion which captures that some messages to be learned, but other messages remain completely hidden. We give details on our solution below.

Extracting randomness against correlated sources. Before getting to our results, we discuss randomness extraction, which is a central tool in all existing constructions of incompressible encryption. A randomness extractor Ext takes as input a source of imperfect randomness X and uses it to distill out some (nearly) uniformly random string Y. Here, we consider seeded extractors, which use a public uniformly random seed S as a catalyst to extract $Y = \mathsf{Ext}(X; S)$, such that Y should be (nearly) uniform even conditioned on the seed S.

While randomness extraction is very well studied, it is most often in the *single-use* case, where a single string $Y = \mathsf{Ext}(X; S)$ is extracted from a single source X having sufficient entropy. Here we ask: what if many strings $Y_i = \mathsf{Ext}(X_i; S_i)$ are extracted from multiple sources X_i respectively (using independent random seeds S_i), but where the sources X_i may be arbitrarily correlated? What guarantees can be made? We consider the case where we only know that the total joint entropy of all the sources is high, but we know nothing else about their individual entropies; indeed some of the sources may have no entropy at all! In this case, clearly not all of the extracted values Y_i can be uniform, and some may even be entirely deterministic. One may nevertheless hope that *some* of the extracted values remain uniform, where the fraction of uniform values roughly correlates to combined total entropy rate of all the sources. To the best of our knowledge, randomness extraction in this setting has not been studied before.

1.1 Our Results.

Formalizing Multi-user Incompressible Encryption. We first provide definitions for multi-user incompressible encryption. We depart from the indistinguishability-based definitions of the prior work on incompressible cryptography [Dzi06, GWZ22, BDD22], and instead give a *simulation*-based definition. Essentially, the definition says that anything that an adversary can learn by taking many ciphertexts of different users, compressing them down sufficiently, and later getting all the secret keys, can be simulated by a simulator that can only ask to see some small fraction of the plaintexts but does not learn anything about the remaining ones. In the single-instance case, this definition implies indistinguishability-based security, but appears stronger. Nevertheless, existing constructions and proofs are readily adapted to satisfy simulation security. The distinction becomes more important in the multi-user setting, however, where simulation security allows us to naturally define what it means for some messages to be revealed and some to remain hidden.

Somewhere Randomness Extractors. As our main technical tool, we explore a new kind of extractor that we call a somewhere randomness extractor, which aims to solve the extraction problem outlined above. Syntactically, this is a standard extractor $Y = \mathsf{Ext}(X; S)$ that takes as input a source X and a seed S and outputs some short

randomness Y. However, we now imagine that the extractor is applied separately to t correlated sources X_i, with each invocation using an independent seed S_i, to derive extracted values $Y_i = \mathsf{Ext}(X_i; S_i)$. The only guarantee on the sources is that the total joint min-entropy of $X = (X_1, \ldots, X_t)$ is sufficiently high. Any individual source X_i, however, may actually be deterministic (have 0 entropy), in which case the corresponding extracted value Y_i is of course not random. However, provided the total min-enropy rate of X is high, it is guaranteed that *many* of the t extracted values are statistically-close uniform. Ideally, if the joint min-entropy rate of X is α, we would hope that roughly αt of the extracted values are uniform.

Formalizing the above requires some care. For example, it may be the case that X is chosen by selecting a random index $i^* \leftarrow [t]$, setting X_{i^*} to be all 0's, and choosing the remaining block X_j for $j \neq i^*$ uniformly at random. In that case X has a very high entropy rate, but for any fixed index i, the min-entropy of X_i is small (at most $\log t$ since with polynomial probability $1/t$ the value of X_i is all 0's), and not enough to extract even 1 bit with negligible bias. Therefore, we cannot argue that $Y_i = \mathsf{Ext}(X_i; S_i)$ is close to uniform for any particular index i! Instead, we allow the set of indices i, for which Y_i is close to uniform, itself be a random variable correlated with X. (See Definition 3.1.)

We show constructions of somewhere randomness extractors nearing the optimal number of uniform extracted values. In particular, we show that if the joint min-entropy rate of $X = (X_1, \ldots, X_t)$ is α then there exists some random variable I_X denoting a subset of $\approx \alpha \cdot t$ indices in $[t]$ such that nobody can distinguish between seeing all the extracted values $Y_i = \mathsf{Ext}(X_i; S_i)$ versus replacing all the Y_i for $i \in I_X$ by uniform, even given all the seeds S_i. (See Corollary 3.4.) Our constructions are based on Hadamard codes (long seed) and Reed-Muller codes (short seed). While the constructions themselves are standard, our analysis is novel, leveraging the list-decodability of the codes, plus a property we identify called *hinting*. Hinting roughly means that the values of $\{\mathsf{Ext}(x; S_i)\}_i$ on some particular exponentially large set of pairwise independent seeds S_i can be compressed into a single small hint, of size much smaller than x. This hinting property is a crucial feature in the *local* list-decoding algorithms for these codes, but appears not to have been separately formalized/utilized as a design goal for an extractor.[2]

Applications. We then show that somewhere randomness extraction can be used essentially as a drop-in replacement for standard randomness extractors in prior constructions of individual incompressible encryption, lifting them to multi-incompressible encryption. As concrete applications, we obtain multi-incompressible encryption in a variety of settings:

- A symmetric key scheme with information-theoretic security, by replacing the extractor in [Dzi06].

[2]The work of [AOR+20] studied a notion of extractors for "Somewhere Honest Entropic Look Ahead" (SHELA) sources. The notions are largely different and unrelated. In particular: (i) in our work X is an arbitrary source of sufficient entropy while [AOR+20] places additional restrictions, (ii) we use a seeded extractor while [AOR+20] wants a deterministic extractor, (iii) we apply the seeded extractor separately on each X_i while [AOR+20] applies it jointly on the entire X, (iv) we guarantee that a large fraction of extracted outputs is uniform even if the adversary sees the rest, while in [AOR+20] the adversary cannot see the rest.

- A "rate-1" symmetric key scheme, meaning the ciphertext is only slightly larger than the message. Here, we assume either decisional composite residuosity (DCR) or learning with errors (LWE), matching [BDD22][3].
- A public key scheme, assuming any ordinary public key encryption scheme, matching [GWZ22].
- A rate-1 public key scheme, under the same assumptions as [BDD22][4]. The scheme has large public keys.
- A rate-1 public key scheme that additionally has succinct public keys, assuming general functional encryption, matching [GWZ22].

In all cases, we guarantee that if the adversary's storage is an α fraction of the total size of all the ciphertexts, then it can only learn something about a $\beta \approx \alpha$ fraction of the encrypted messages. We can make $\beta = \alpha - 1/p(\lambda)$ for any polynomial p in the security parameter λ, by choosing a sufficiently large ciphertext size.

Multiple ciphertexts per user. Prior work, in addition to only considering a single user, also only considers a single ciphertext per user. Perhaps surprisingly, security does not compose, and indeed for any fixed secret key size, we explain that simulation security for unbounded messages is impossible.

We therefore develop schemes for achieving a bounded number of ciphertexts per user. We show how to modify each of the constructions above to achieve multi-ciphertext security under the same assumptions.

The Random Oracle Model. We show how to construct symmetric key multi-user incompressible encryption with an unbounded number of ciphertexts per user and also essentially optimal secret key and ciphertext sizes, from random oracles. This shows that public key tools are potentially not inherent to rate-1 symmetric incompressible encryption.

1.2 Our Techniques: Somewhere Randomness Extraction

We discuss how to construct a somewhere randomness extractor Ext. Recall, we want to show that, if the joint min-entropy rate of $X = (X_1, \ldots, X_t)$ is α then there exists some random variable I_X denoting a subset of $\approx \alpha \cdot t$ indices in $[t]$ such that the distribution $(S_i, Y_i = \mathsf{Ext}(X_i; S_i))_{i\in[t]}$ is statistically indistinguishable from $(S_i, Z_i)_{i\in[t]}$ where Z_i is uniformly random for $i \in I_X$ and $Z_i = Y_i$ otherwise.

A failed approach. A natural approach would be to try to show that every standard seeded extractor is also a "somewhere randomness extractor". As a first step, we would show that there is some random variable I_X denoting a large subset of $[t]$ such that the values X_i for $i \in I_X$ have large min-entropy conditioned on $i \in I_X$. Indeed, such results are

[3]One subtlety is that, for all of our rate-1 constructions, we need a PRG secure against *non-uniform* adversaries, whereas the prior work could have used a PRG against uniform adversaries.

[4][BDD22] explores CCA security, but in this work for simplicity we focus only on CPA security.

known; see for example the "block-entropy lemma" of [DQW22] (also [DKZ18, DFR+07]). In fact, one can even show a slightly stronger statement that the random variables X_i for $i \in I_X$ have high min-entropy even conditioned on all past blocks $X_1, \ldots, X_{i-1}$. However, it cannot be true that X_i has high min-entropy conditioned on *all* other blocks past and future (for example, think of X being uniform subject to $\bigoplus_{i=1}^{t} X_i = 0$). Unfortunately, this prevents us for using the block-entropy lemma to analyze somewhere extraction, where the adversary sees some extracted outputs from all the blocks.[5] It remains as a fascinating open problem whether every standard seeded extractor is also a somewhere randomness extractor or if there is some counterexample.[6]

Our approach. We are able to show that particular seeded extractors Ext based on Hadamard or Reed-Muller codes are good somewhere randomness extractors. For concreteness, let us consider the Hadamard extractor $\mathsf{Ext}(x; s) = \langle x, s \rangle$. Our analysis is inspired by the techniques used to prove the Goldreich-Levin theorem [GL89], even though our goal is completely different. Our proof proceeds in 3 steps:

Step 1: Switch quantifiers. We need to show that there *exists* some random variable I_X such that *every* statistical distinguisher fails to distinguish between the two distributions $(S_i, Y_i)_{i \in [t]}$ and $(S_i, Z_i)_{i \in [t]}$. We can use von Neumann's minimax theorem to switch the quantifiers: it suffices to show that for every (randomized) statistical distinguisher D there is some random variable I_X such that D fails to distinguish the above distributions.

Step 2: Define I_X. For any fixed $x = (x_1, \ldots, x_t)$ we define the set I_x to consist of indices $i \in [t]$ such that D fails to distinguish between the hybrid distributions $(\{S_j\}_{j \in [t]}, Z_1, \ldots, Z_{i-1}, Y_i, \ldots, Y_t)$ versus $(\{S_j\}_{j \in [t]}, Z_1, \ldots, Z_i, Y_{i+1}, \ldots, Y_t)$, where in both distributions we condition on $X = x$. We then define the random variable I_X that chooses the correct set I_x according to X. It is easy to show via a simple hybrid argument that with this definition of I_X it is indeed true that D fails to distinguish $(S_i, Y_i)_{i \in [t]}$ and $(S_i, Z_i)_{i \in [t]}$.

Step 3: Argue that I_X is large. We still need to show that I_X is a large set, containing $\approx \alpha \cdot t$ indices. To do so, we show that if I_X were small (with non negligible probability) then we could "guess" X with sufficiently high probability that would contradict X having high min-entropy. In particular, we provide a guessing strategy such that for any x for which I_x is small, our strategy has a sufficiently high chance of guessing x. First, we guess the small set $I_x \subseteq [t]$ as well as all of the blocks x_i for $i \in I_x$ uniformly at random. For the rest of the blocks $i \notin I_x$, we come up with a guessing strategy that does significantly better than guessing randomly. We rely on the fact that distinguishing implies predicting, to convert the distinguisher D into a predictor P such that for all $i \notin I_x$ we have: $P(S_i, \{S_j, \mathsf{Ext}(x_j; S_j)\}_{j \in [t] \setminus \{i\}}) = \mathsf{Ext}(x_i; S_i)$ with probability significantly better than $1/2$. Now we would like to use the fact that the Hadamard code $(\mathsf{Ext}(x; s) = \langle x, s \rangle)_s$ is list-decodable to argue that we can use such predictor P to derive a small list of

[5] This strategy would allow us to only prove a very weak version of somewhere extraction when the number of blocks t is sufficiently small so that many blocks will have more than t bits of entropy. In this case we can afford to lose the t extracted output bits from the entropy of each block. However, in our setting, we think of the number of blocks t as huge, much larger than the size of each individual block.

[6] We were initially convinced that the general result does hold and invested much effort trying to prove it via some variant of the above approach without success. We also mentioned the problem to several experts in the field who had a similar initial reaction, but were not able to come up with a proof.

possibilities for x. Unfortunately, there is a problem with this argument. To call the predictor, we need to feed it with some auxiliary values $\mathsf{aux}_i = \{S_j, \mathsf{Ext}(x_j; S_j)\}_{j\in[t]\setminus\{i\}}$, which requires knowing t bits about x. We could hope to guess a good choice of aux_i, but there may be a different good choice for each $i \in [t]$, and therefore we'd need to guess a fresh t bits of information about x just to recover each block x_i, which is worse than the trivial approach of guessing x_i directly when $|x_i| < t$! Instead, we use a trick inspired by the proof of the Goldreich-Levin theorem. For each block $j \in [t]$, we guess the values of $\langle x_j, S_j^{(k)}\rangle$ for a very small "base set" of Q random seeds $S_j^{(1)}, \ldots, S_j^{(Q)}$. We can then expand this small "base set" of guesses into an exponentially larger "expanded set" of $2^Q - 1$ guesses for the values $\langle x_j, \sum_{k\in K} S_j^{(k)}\rangle = \sum_{k\in K}\langle x_j, S_j^{(k)}\rangle$ for all sets $K \subseteq [Q]$ with $K \neq \emptyset$. The expanded set of guesses is correct if the base set is correct and the expanded sets of seeds $(\sum_{k\in K} S_j^{(k)})_K$ are pairwise independent for different sets K. Therefore, each such set K gives us some corresponding $\mathsf{aux}_i^{(K)}$. We can now apply Chebyshev's bound to argue that if for each i we take the majority value for $P(S_i, \mathsf{aux}_i^{(K)})$ across all $2^Q - 1$ sets K, it is likely equal to the true majority value of $P(S_i, \{s_j, \mathsf{Ext}(x_j; s_j)\}_{j\in[t]\setminus\{i\}})$ which is equal to $\mathsf{Ext}(x_i; S_i)$ with probability significantly better than $1/2$. Notice that we got our saving by only guessing Qt bits about $x = (x_1, \ldots, x_t)$ for some small value Q (roughly $\log(1/\varepsilon)$ if we want indistinguishability ε) and were able to use these guesses to recover all the blocks x_i for $i \notin I_x$.

Generalizing. We generalize the above analysis for the Hadamard extractor to any extractor that is list-decodable and has a "hinting" property as discussed above. In particular, this also allows us to use a Reed-Muller based extractor construction with a much smaller seed and longer output length.

1.3 Our Techniques: Multi-Incompressible Encryption

We then move to considering incompressible encryption in the multi-user setting.

Definition. We propose a simulation-based security definition for multi-incompressible encryption. Roughly, the simulator first needs to simulate all the ciphertexts for all the instances *without* seeing any of the message queries, corresponding to the fact that at this point the adversary can't learn anything about any of the messages. To model the adversary then learning the secret keys, we add a second phase where the simulator can query for a *subset* of the messages, and then must simulate *all* the private keys. We require that no *space-bounded* distinguisher can distinguish between the receiving real encryptions/real private keys vs receiving simulated encryptions/keys. The number of messages the simulator can query will be related to the storage bound of the distinguisher.

Upgrading to multi-incompressible encryption using somewhere randomness extraction. All prior standard-model constructions of individual incompressible encryption [Dzi06, GWZ22, BDD22] utilize a randomness extractor. For example, Dziembowski [Dzi06] gives the following simple construction of a symmetric key incompressible encryption scheme:

- The secret key k is parsed as (s, k') where s is a seed for a randomness extractor, and k' is another random key.

- To encrypt a message m, choose a large random string R, and output $c = (R, d = \mathsf{Ext}(R; s) \oplus k' \oplus m)$.

The intuition for (individual) incompressible security is that an adversary that cannot store essentially all of c can in particular not store all of R, meaning R has min-entropy conditioned on the adversary's state. The extraction guarantee then shows that $\mathsf{Ext}(R; s)$ can be replaced with a random string, thus masking the message m.

We demonstrate that our somewhere randomness extractors can be used as a drop-in replacement for ordinary random extractors in all prior constructions of individual incompressible encryption, upgrading them to multi-incompressible encryption. In the case of [Dzi06], this is almost an immediate consequence of our somewhere random extractor definition. Our simulator works by first choosing random s for each user, and sets the ciphertexts of each user to be random strings. Then it obtains from the somewhere randomness extractor guarantee the set of indices i where Y_i is close to uniform. For these indices, it sets k' to be a uniform random string. This correctly simulates the secret keys for these i.

For i where Y_i is not uniform, the simulator then queries for messages for these i. It programs k' as $k' = d \oplus \mathsf{Ext}(R; s) \oplus m$; decryption under such k' will correctly yield m. Thus, we correctly simulate the view of the adversary, demonstrating multi-incompressible security.

Remark 1.1. *The set of indicies where Y_i is uniform will in general not be efficiently computable, and somewhere randomness extraction only implies that the set of indices exist. Since our simulator must know these indices, our simulator is therefore inefficient. In general, an inefficient simulator seems inherent in the standard model, since the adversary's state could be scrambled in a way that hides which ciphertexts it is storing.*

We proceed to show that various constructions from [GWZ22, BDD22] are also secure in the multi-user setting, when plugging in somewhere randomness extractors. In all cases, the proof is essentially identical to the original single-user counterpart, except that the crucial step involving extraction is replaced with the somewhere randomness extraction guarantee. We thus obtain a variety of parameter size/security assumption trade-offs, essentially matching what is known for the single-user setting.

One small issue that comes up is that, once we have invoked the somewhere randomness extractor, the simulation is inefficient. This presents a problem in some of the security proofs, specifically in the "rate-1" setting where messages can be almost as large as ciphertexts. In the existing proofs in this setting, there is a computational hybrid step that comes *after* applying the extractor. Naively, this hybrid step would seem to be invalid since the reduction now has to be inefficient. We show, however, that the reduction can be made efficient as long as it is *non-uniform*, essentially having the choice of indices (and maybe some other quantities) provided as non-uniform advice. As long as the underlying primitive for these post-extraction hybrids has non-uniform security, the security proof follows.

Multiple ciphertexts per user. We also consider the setting where there may be multiple ciphertexts per user, which has not been considered previously.

It is not hard to see that having an *unbounded* number of ciphertexts per user is impossible in the standard model. This is because the simulator has to simulate everything but the secret key without knowing the message. Then, for the ciphertexts stored by the adversary, the simulator queries for the underlying messages and must generate the secret key so that those ciphertexts decrypt to the given messages. By incompressiblity, this means the secret key length must be at least as large as the number of messages.

We instead consider the case of bounded ciphertexts per user. For a stateful encryption scheme, it is trivial to upgrade a scheme supporting one ciphertext per user into one supporting many: simply have the secret key be a list of one-time secret keys. In the symmetric key setting, this can be made stateless by utilizing k-wise independent hash functions.

In the public key setting, achieving a stateless construction requires more work, and we do not believe there is a simple generic construction. We show instead how to modify all the existing constructions to achieve multiple ciphertexts per user. Along the way, we show an interesting combinatorial approach to generically lifting non-committing encryption to the many-time setting without sacrificing ciphertext rate.

Random Oracle Model. In Section 7, we finally turn to constructions in the random oracle model. We give a construction of symmetric key incompressible encryption with optimal key and ciphertext length, achieving security for an unbounded number of users and unbounded number of ciphertexts per user. As explained above, this is only possible because our simulator utilizes the random oracle: the incompressibility argument no longer applies since the simulator can covertly set the messages by programming random oracle queries. The construction is essentially a 2-round unbalanced Feistel network.

We also show that standard hybrid encryption lifts essentially any random oracle-based symmetric key incompressible encryption to a public key scheme, assuming only general public key encryption. This significantly generalizes a construction of [BDD22]. Note, however, that as observed by [BDD22], the security of the scheme in the standard model is problematic: they show that if the PKE scheme is instantiated with fully homomorphic encryption, then there is a simple efficient attack that completely violates incompressible security. This gives a very natural random oracle uninstantiability result. In particular, all prior random oracle uninstantiabilities require a contrived instantiation of some building block[7], whereas this uninstantiability only requires instantiating hybrid encryption with fully homomorphic encryption.

Remark 1.2. *Note that the underlying symmetric key scheme in [BDD22] uses ideal ciphers instead of random oracles. Thus, their uninstantiability is only for the ideal cipher model. [BDD22] claims the counterexample applies to random oracles, since random oracles and ideal ciphers are supposedly equivalent [HKT11]. However, this is incorrect, as the equivalence only holds in the "single stage" setting [RSS11]. Importantly, incompressible encryption is not a single stage game, owing to the space bound on the adversary's storage between receiving the ciphertexts and receiving the secret keys. In the more general multi-stage setting encompassing incompressible encryption, the equivalence*

[7]For example, even the "natural" uninstnatiability of Fiat-Shamir [GK03] requires a contrived proof system.

of ideal ciphers and random oracles is open. Our generalized construction fixes this issue by directly designing our symmetric key scheme from random oracles.

2 Preliminaries

Notation-wise, for $n \in \mathbb{N}$, we let $[n]$ denote the ordered set $\{1, 2, \ldots, n\}$. We use capital bold letters to denote a matrix $\mathbf{M}$. Lowercase bold letters denote vectors $\mathbf{v}$. Let $\mathbf{M}_{i,j}$ denote the element on the i-th row, and j-th column of $\mathbf{M}$, and $\mathbf{v}_i$ denote the i-th element of $\mathbf{v}$.

Lemma 2.1 (Johnson Bound)**.** *Let $\mathcal{C} \subseteq \Sigma^n$ with $|\Sigma| = q$ be any q-ary error-correcting code with relative distance $p_0 = 1 - (1+\rho)\frac{1}{q}$ for $\rho > 0$, meaning that for any two distinct values $x, y \in \mathcal{C}$, the Hamming distance between x, y is at least $p_0 \cdot n$. Then for any $\delta > \sqrt{\rho(q-1)}$ there exists some $L \leq \frac{(q-1)^2}{\delta^2 - \rho(q-1)}$ such that the code is $(p_1 = (1-(1+\delta)\frac{1}{q}), L)$-list decodable, meaning that for any $y \in \Sigma_q^n$ there exist at most L codewords $x \in \mathcal{C}$ that are within Hamming distance $p_1 n$ of y.*

Lemma 2.2 (Distinguishing implies Predicting)**.** *For any randomized function $D : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ there exists some randomized function $P : \{0,1\}^n \to \{0,1\}^m$ such that for any jointly distributed random variables (A, B) over $\{0,1\}^n \times \{0,1\}^m$:*

$$\textit{if } \Pr[D(A,B)=1] - \Pr[D(A,U_m)=1] \geq \varepsilon \textit{ then } \Pr[P(A)=B] \geq \frac{1}{2^m}(1+\varepsilon).$$

Proof. Define $P(a)$ as follows. Sample a random $b_0 \leftarrow \{0,1\}^m$, if $D(a, b_0) = 1$ output b_0 else sample a fresh $b_1 \leftarrow \{0,1\}^m$ and output b_1.

Define $p = \Pr[D(A, U_m) = 1]$. Let B_0, B_1 be independent random variables that are uniform over $\{0,1\}^m$ corresponding to the bits b_0, b_1 . Then we have

$$\begin{aligned}
\Pr[P(A)=B] &= \Pr[D(A,B_0)=1 \wedge B_0 = B] + \Pr[D(A,B_0)=0 \wedge B_1 = B] \\
&= \Pr[B_0 = B]\Pr[D(A,B)=1] + \Pr[D(A,B_0)=0]\Pr[B_1 = B] \\
&= \frac{1}{2^m}(\varepsilon + p) + (1-p)\frac{1}{2^m} = \frac{1}{2^m}(1+\varepsilon).
\end{aligned}$$

□

Min-Entropy Extractor. Recall the definition for average min-entropy:

Definition 2.1 (Average Min-Entropy)**.** *For two jointly distributed random variables (X, Y), the average min-entropy of X conditioned on Y is defined as*

$$H_\infty(X|Y) = -\log \mathbf{E}_{y \xleftarrow{\$} Y}[\max_x \Pr[X = x | Y = y]].$$

Lemma 2.3 ([DRS04])**.** *For random variables X, Y where Y is supported over a set of size T, we have*

$$H_\infty(X|Y) \geq H_\infty(X,Y) - \log T \geq H_\infty(X) - \log T.$$

Definition 2.2 (Extractor [Nis90])**.** *A function* $\mathsf{Extract} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k, ϵ) *strong average min-entropy extractor if, for all jointly distributed random variables* (X, Y) *where* X *takes values in* $\{0,1\}^n$ *and* $H_\infty(X|Y) \geq k$*, we have that* $(U_d, \mathsf{Extract}(X; U_d), Y)$ *is* ϵ*-close to* (s, U_m, Y)*, where* U_d *and* U_m *are uniformly random strings of length* d *and* m *respectively.*

Remark 2.1. *Any strong randomness extractor is also a strong* average *min-entropy extractor, with a constant loss in* ϵ*.*

2.1 Incompressible Encryption

Incompressible PKE [GWZ22]. First, recall the definition of incompressible public key encryption (PKE) by Guan, Wichs, and Zhandry [GWZ22]. The syntax of an incompressible PKE scheme is analogous to that of a classical PKE scheme, except that Gen takes an additional security parameter S, which is the space bound of the adversary('s long term storage). The "rate" of the scheme is defined as the ratio of the message length to the ciphertext length. Note that the rate is always between 0 and 1, with 1 being the ideal rate, meaning that the ciphertext does not add any overhead to the message length.

The security is defined through the following experiment $\mathsf{Dist}^{\mathsf{IncomPKE}}_{\mathcal{A},\Pi}(\lambda)$ [GWZ22]:

1. The adversary $\mathcal{A}_1$ takes 1^λ, and outputs a space bound 1^S.
2. Run $\mathsf{Gen}(1^\lambda, 1^S)$ to obtain keys $(\mathsf{pk}, \mathsf{sk})$.
3. Sample a uniform bit $b \in \{0,1\}$.
4. The adversary is given the public key pk and submits an auxiliary input aux.
5. The adversary submits the challenge query consisting of two messages m_0 and m_1, and receives $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.
6. $\mathcal{A}_1$ produces a state st of size at most S.
7. The adversary $\mathcal{A}_2$ is given the tuple $(\mathsf{pk}, \mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs a guess b' for b. If $b' = b$, we say that the adversary succeeds and the output of the experiment is 1. Otherwise, the experiment outputs 0.

Definition 2.3 (Incompressible PKE Security)**.** *Let* λ *and* S *be security parameters. A public key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to have incompressible PKE security if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*:*

$$\Pr\left[\mathsf{Dist}^{\mathsf{IncomPKE}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Incompressible SKE [Dzi06]. One can also imagine an analogous incompressible *symmetric* key encryption (SKE) scheme. This object has been studied earlier by Dziembowski under the name *forward-secure storage* [Dzi06]. The syntax of an incompressible SKE also follows a standard SKE scheme. The "rate" is also defined the same as the ratio of the message length to the ciphertext length. The security of an incompressible SKE can be analogously defined through the following experiment $\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A},\Pi}(\lambda)$:

1. The adversary $\mathcal{A}_1$ takes 1^λ, and outputs a space bound 1^S.
2. Run $\mathsf{Gen}(1^\lambda, 1^S)$ to obtain the key k.
3. Sample a uniform bit $b \in \{0,1\}$.
4. The adversary submits an auxiliary input aux.
5. The adversary submits the challenge query consisting of two messages m_0 and m_1, and receives $\mathsf{ct} \leftarrow \mathsf{Enc}(k, m_b)$.
6. $\mathcal{A}_1$ produces a state st of size at most S.
7. The adversary $\mathcal{A}_2$ is given the tuple $(k, \mathsf{aux}, \mathsf{st})$ and outputs a guess b' for b. If $b' = b$, we say that the adversary succeeds and the output of the experiment is 1. Otherwise, the experiment outputs 0.

Definition 2.4 (Incompressible SKE Security)**.** *Let λ and S be security parameters. A symmetric key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to have incompressible SKE security if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:*

$$\Pr\left[\mathsf{Dist}^{\mathsf{IncomSKE}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.2 Functional Encryption

The constructions by [GWZ22] use single-key game-based functional encryption as a building block. Let λ be the security parameter. Let $\{\mathcal{C}_\lambda\}$ be a class of circuits with input space $\mathcal{X}_\lambda$ and output space $\mathcal{Y}_\lambda$. A *functional encryption* scheme for the circuit class $\{\mathcal{C}_\lambda\}$ is a tuple of PPT algorithms $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$ takes as input the security parameter λ, and outputs the master public key mpk and the master secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$ takes as input the master secret key msk and a circuit $C \in \{\mathcal{C}_\lambda\}$, and outputs a function key sk_C.
- $\mathsf{Enc}(\mathsf{mpk}, m) \to \mathsf{ct}$ takes as input the public key mpk and a message $m \in \mathcal{X}_\lambda$, and outputs the ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to y$ takes as input a function key sk_C and a ciphertext ct, and outputs a value $y \in \mathcal{Y}_\lambda$.

We can analogously define the "rate" of an FE scheme to be the ratio between the message length to the ciphertext length. We require correctness and security of a functional encryption scheme.

Definition 2.5 (Correctness)**.** *A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be correct if for all $C \in \{\mathcal{C}_\lambda\}$ and $m \in \mathcal{X}_\lambda$:*

$$\Pr\left[y = C(m) : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m) \\ y \leftarrow \mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Consider the following *Semi-Adaptive Security Experiment*, $\mathsf{Dist}^{\mathsf{SemiAdpt}}_{\mathsf{FE},\mathcal{A}}(\lambda)$:

- Run $\mathsf{FE.Setup}(1^\lambda)$ to obtain $(\mathsf{mpk}, \mathsf{msk})$ and sample a random bit $b \leftarrow \{0,1\}$.
- On input 1^λ and mpk, The adversary $\mathcal{A}$ submits the challenge query consisting of two messages m_0 and m_1. It then receives $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, m_b)$.
- The adversary now submits a circuit $C \in \{\mathcal{C}_\lambda\}$ s.t. $C(m_0) = C(m_1)$, and receives $\mathsf{sk}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, C)$.
- The adversary $\mathcal{A}$ outputs a guess b' for b. If $b' = b$, we say that the adversary succeeds and experiment outputs 1. Otherwise, the experiment outputs 0.

Definition 2.6 (Single-Key Semi-Adaptive Security)**.** *For security parameter λ, a functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to have single-key semi-adaptive security if for all PPT adversaries $\mathcal{A}$:*

$$\Pr\left[\mathsf{Dist}^{\mathsf{SemiAdpt}}_{\mathsf{FE},\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

We can also consider *selective* security, where the adversary only receives mpk *after* sending the challenge messages.

3 Somewhere Randomness Extraction

3.1 Defining Somewhere Extraction

Definition 3.1 (Somewhere Randomness Extraction)**.** *A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is $(t, \alpha, \beta, \varepsilon)$*-somewhere extracting *if the following holds. Let $X = (X_1, \ldots, X_t)$ be any random variable consisting of blocks $X_i \in \{0,1\}^n$ such that $H_\infty(X) \geq \alpha \cdot tn$. Then, there exists some random variable I_X jointly distributed with X, such that I_X is supported over sets $\mathcal{I} \subseteq [t]$ of size $|\mathcal{I}| \geq \beta \cdot t$ and:*

$$(S_1, \ldots, S_t, \mathsf{Ext}(X_1; S_1), \ldots, \mathsf{Ext}(X_t; S_t)) \approx_\varepsilon (S_1, \ldots, S_t, Z_1, \ldots, Z_t)$$

where $S_i \in \{0,1\}^d$ are uniformly random and independent seeds, and $Z_i \in \{0,1\}^m$ is sampled uniformly random for $i \in I_X$ while $Z_i = \mathsf{Ext}(X_i; S_i)$ for $i \notin I_X$.

In other words, the above definition says that if we use a "somewhere extracting" extractor with independent seeds to individually extract from t correlated blocks that have a joint entropy-rate of α, then seeing all the extracted outputs is indistinguishable from replacing some carefully chosen β-fraction by uniform.

3.2 Hinting Extractors

Definition 3.2 (Hinting Extractor)**.** *A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (δ, L, h, Q)-hinting extractor if it satisfies the following:*

- *List Decodable: If we think of* $\mathsf{ECC}(x) = (\mathsf{Ext}(x;s))_{s\in\{0,1\}^d}$ *as a* $(2^d, n)_{\Sigma=\{0,1\}^m}$ *error-correcting code over the alphabet* $\Sigma = \{0,1\}^m$*, then the code is* $(p = 1-(1+\delta)2^{-m}, L)$*-list decodable, meaning that for any* $y \in \Sigma^{2^d}$*, the number of codewords that are within Hamming distance* $p \cdot 2^d$ *of* y *is at most* L*.*

- *Pairwise-Independent Hint: There exists some functions* $\mathsf{hint} : \{0,1\}^n \times \{0,1\}^\tau \to \{0,1\}^h$*, along with* rec_0 *and* rec_1 *such that:*
  - *For all* $x \in \{0,1\}^n, r \in \{0,1\}^\tau$*, if we define* $\sigma = \mathsf{hint}(x;r)$*,* $\{s_1,\ldots,s_Q\} = \mathsf{rec}_0(r)$*, and* $\{y_1,\ldots,y_Q\} = \mathsf{rec}_1(\sigma, r)$*, then* $\mathsf{Ext}(x;s_i) = y_i$ *for all* $i \in [Q]$*.*
  - *Over a uniformly random* $r \leftarrow \{0,1\}^\tau$*, the* Q *seeds* $\{s_1,\ldots,s_Q\} = \mathsf{rec}_0(r)$*, are individually uniform over* $\{0,1\}^d$ *and pairwise independent.*

Intuitively, the pairwise-independent hint property says that there is a small (size h) hint about x that allows us to compute $\mathsf{Ext}(x;s_i)$ for a large (size Q) set of pairwise independent seeds s_i. We generally want Q to be exponential in h.

The list-decoding property, on the other hand, is closely related to the standard definition of strong randomness extractors. Namely, if Ext is a (k,ε)-extractor then it is also $(p = 1-(1+\delta)2^{-m}, 2^k)$-list decodable for $\delta = \varepsilon \cdot 2^m$, and conversely, if it is $(p = 1-(1+\delta)2^{-m}, 2^k)$-list deocdable then it is a $(k+m+\log(1/\delta), \delta)$-extractor (see Proposition 6.25 in [V+12]).

Construction 1: Hadamard. Define $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ via $\mathsf{Ext}(x;s) = \langle x, s\rangle$, where we interpret x, s as elements of $\mathbb{F}_{2^m}^{\hat{n}}$ for $\hat{n} := n/m$ and all the operations are over $\mathbb{F}_{2^m}$. The seed length is $d = n$ bits and the output length is m bits.

Lemma 3.1. *The above* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *is a* (δ, L, h, Q)*-hinting extractor for any* $h, \delta > 0$ *with* $Q \geq 2^{h-m}$ *and* $L \leq 2^{2m}/\delta^2$*.*

Proof. The list-decoding bounds on δ, L come from the Johnson bound (Lemma 2.1) with $q = 2^m$, $\rho = 0$. For pairwise-independent hints, let $\hat{h} = h/m$ and define $\mathsf{hint}(x;R)$ to parse $R \in \mathbb{F}_{2^m}^{\hat{h}\times\hat{n}}$ and output $\sigma = R \cdot x^\top$, which has bit-size h. Let $\mathcal{V} \subseteq \mathbb{F}_{2^m}^{\hat{h}}$ be a set of vectors such that any two distinct vectors $v_1 \neq v_2 \in \mathcal{V}$ are linearly independent. Such a set $\mathcal{V}$ exists of size $Q = (2^m)^{\hat{h}-1} + (2^m)^{\hat{h}-2} + \cdots + 2^m + 1 \geq 2^{h-m}$, e.g., by letting $\mathcal{V}$ be the set of all non-zero vectors whose left-most non-zero entry is a 1. Define $\mathsf{rec}_0(R)$ so that it outputs $\{s_v = v \cdot R\}_{v\in\mathcal{V}}$. Correspondingly, $\mathsf{rec}_1(\sigma, R)$ outputs $\{y_v = \langle v, \sigma\rangle\}_{v\in\mathcal{V}}$. It's easy to see that the seeds s_v are individually uniform and pairwise independent, since for any linearly-independent $v_1 \neq v_2 \in \mathcal{V}$ and the value $s_{v_1} = v_1 R$ and $s_{v_2} = v_2 R$ are random and independent over a random choice of the matrix R. Moreover for all seeds s_v we have

$$\mathsf{Ext}(x, s_v) = \langle s_v, x\rangle = v \cdot R \cdot x^\top = \langle v, \sigma\rangle = y_v.$$

□

Construction 2: Hadamard ∘ Reed-Muller. Define $\mathsf{Ext}(f; s = (s_1, s_2)) = \langle f(s_1), s_2\rangle$, where $f \in \mathbb{F}_{2^w}^{\binom{\ell+g}{g}}$ is interpreted as a ℓ-variate polynomial of total degree g over some field of size $2^w > g$, and $s_1 \in \mathbb{F}_{2^w}^\ell$ is interpreted as an input to the polynomial (this

is Reed-Muler).[8] Then $y = f(s_1)$ and s_2 are interpreted as a values in $\mathbb{F}_{2^m}^{w/m}$ and the inner-product $\langle y, s_2 \rangle$ is computed over $\mathbb{F}_{2^m}$ (this is Hadamard). So overall, in bits, the input length is $n = w \cdot \binom{\ell+g}{g}$, the seed length is $d = w(\ell+1)$ and the output length is m. This code has relative distance $1 - (\frac{1}{2^m} + \frac{g}{2^w}) = 1 - \frac{1}{2^m}(1 + \frac{g}{2^{w-m}})$.

Lemma 3.2. *For any w, ℓ, g, m, δ such that $2^w > g$ and m divides w, if we set $n = w \cdot \binom{\ell+g}{g}$, $d = w(\ell+1)$ then the above* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a (δ, L, h, Q)-hinting extractor with $\delta = \sqrt{g 2^{2m}/2^w}$, $L = \frac{2^{2m}}{\delta^2 - g 2^{2m}/2^w}$, $h = w \cdot (g+1)$, $Q = 2^w$.*

In particular, for any n, m, w such that m divides w, we can set $\ell = g = \log n$ to get an (δ, L, h, Q)-hinting extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $d = O(w \log n)$, $\delta = 2^{m + \log\log n - w/2}$, $h = O(w \log n)$ and $Q = 2^w$.*

Proof. The list-decoding bounds on δ, L come from the Johnson bound (Lemma 2.1) with $q = 2^m$, $\rho = \frac{g}{2^{w-m}}$. On the other hand, for pairwise-independent hints, we can define $\mathsf{hint}(f; r)$ as follows. Parse $r = (r^0, r^1, s_1^1, \ldots, s_1^Q)$ with $r^0, r^1 \in \mathbb{F}_{2^w}^\ell$ and $s_1^i \in \mathbb{F}_{2^m}^{w/m}$. Let $\hat{f}(i) = f(r^0 + i \cdot r^1)$ be a univariate polynomial of degree g and define the hint $\sigma = \hat{f}$ to be the description of this polynomial. Define $\{s_i = (s_0^i, s_1^i)) = \mathsf{rec}_0(r)$ for $i \in \mathbb{F}_{2^w}$ by setting $s_0^i = r^0 + i \cdot r^1$. Define $\{y_i\} = \mathsf{rec}_1(\sigma, r)$ via $y_i = \langle \hat{f}(i), s_1^i \rangle$. It is easy to check correctness and pairwise independence follows from the fact that the values $s_0^i = r^0 + i \cdot r^1$ are pairwise independent over the randomness r^0, r^1. □

3.3 Hinting-Extractors are Somewhere-Extracting

Lemma 3.3 (Somewhere-Extraction Lemma)**.** *Let* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be a (δ, L, h, Q)-hinting extractor. Then, for any $t, \alpha > 0$ such that $Q \geq 2t\frac{2^{2m}}{\delta^2}$, it is also $(t, \alpha, \beta, \varepsilon)$-somewhere extracting with $\varepsilon = 6t\delta$ and $\beta = \alpha - \frac{\log L + h + \log t + \log(1/\varepsilon) + 3}{n}$.*

Proof. Our proof follows a sequence of steps.

Step 0: Relax the Size Requirement. We modify the statement of the lemma as follows. Instead of requiring that $|I_X| \geq \beta \cdot t$ holds with probability 1, we relax this to requiring that $\Pr[|I_X| < \beta \cdot t] \leq \varepsilon/4$. On the other hand, we strengthen the requirement on statistical indisitnguishability from ε to $\varepsilon/2$:

$$(S_1, \ldots, S_t, \mathsf{Ext}(X_1; S_1), \ldots, \mathsf{Ext}(X_1; S_t)) \approx_{\varepsilon/2} (S_1, \ldots, S_t, Z_1, \ldots, Z_t).$$

This modified variant of the lemma implies the original.

To see this, notice that we can replace the set I_X that satisfies the modified variant with I'_X which is defined as $I'_X := I_X$ when $|I_X| \geq \beta t$ and $I'_X := \{1, \ldots, \beta t\}$ else. The set I'_X then satisfies the original variant. In particular, we can prove the indisintinguishability guarantee of the original lemma via a hybrid argument: replace I'_X by I_X ($\varepsilon/4$ statistical distance), switch from the left distribution to right distribution ($\varepsilon/2$ statistical distance), replace I_X back by I'_X ($\varepsilon/4$ statistical distance) for a total distance of ε.

[8] Since the the input to the extractor is interpreted as a polynomial, we will denote it by f rather than the usual x to simplify notation.

Step 1: Change quantifiers. We need to prove that: *for all X with $H_\infty(X) \geq \alpha \cdot tn$, there **exists** some random variable $I_X \subseteq [t]$ with $\Pr[|I_X| < \beta t] \leq \varepsilon/4$ such that **for all** (inefficient) distinguishers D:*

$$\Pr[D(S_1, \ldots, S_t, Y_1, \ldots, Y_t) = 1] - \Pr[D(S_1, \ldots, S_t, Z_1, \ldots, Z_t) = 1] \leq \varepsilon/2 \qquad (1)$$

where we define $Y_i = \mathsf{Ext}(X_i; S_i)$, and the random variables Z_i are defined as in the Lemma. By the min-max theorem, we can switch the order of the last two quantifiers. In particular, it suffices to prove that: *for all X with $H_\infty(X) \geq \alpha \cdot tn$ and **for all** (inefficient, randomized) distinguishers D there **exists** some random variable $I_X \subseteq [t]$ with $\Pr[|I_X| < \beta t] \leq \varepsilon/4$ such that equation (1) holds.*

We can apply min-max because a distribution over inefficient distinguishers D is the same as a single randomized inefficient distinguisher D and a distribution over random variables I_X is the same as a single random variable I_X.

Step 2: Define I_X. Fix a (inefficient/randomized) distinguisher D.

For any fixed value $x \in \{0,1\}^{n \cdot t}$, we define a set $I_x \subseteq [t]$ iteratively as follows. Start with $I_x := \emptyset$. For $i = 1, \ldots, t$ add i to I_x if

$$\left(\begin{array}{ll} & \Pr[D(S_1, \ldots, S_t, Z_1^x \ldots, Z_{i-1}^x, Y_i^x, Y_{i+1}^x, \ldots, Y_t^x) = 1] \\ - & \Pr[D(S_1, \ldots, S_t, Z_1^x, \ldots, Z_{i-1}^x, U_m, Y_{1+1}^x, \ldots, Y_t^x) = 1] \end{array} \right) \leq 3\delta \qquad (2)$$

where S_i is uniform over $\{0,1\}^d$, $Y_j^x = \mathsf{Ext}(x_j; S_j)$ and for $j < i$ we define Z_j^x to be uniformly random over $\{0,1\}^m$ for $j \in I_x$, while $Z_j^x = Y_j^x$ for $j \notin I_x$. Note that $Y_i^x = (Y_i | X = x)$ and $Z_i^x = (Z_i | X = x)$.

Define I_X to be the random variable over the above sets I_x where x is chosen according to X. With the above definition, equation 1 holds since:

$$\begin{aligned}
&\Pr[D(S_1, \ldots, S_t, Y_1, \ldots, Y_t) = 1] - \Pr[D(S_1, \ldots, S_t, Z_1, \ldots, Z_t) = 1] \\
&= \mathbb{E}_{x \leftarrow X} \Pr[D(S_1, \ldots, S_t, Y_1, \ldots, Y_t) = 1 | X = x] - \Pr[[D(S_1, \ldots, S_t, Z_1, \ldots, Z_t) = 1 | X = x] \\
&= \mathbb{E}_{x \leftarrow X} \Pr[D(S_1, \ldots, S_t, Y_1^x, \ldots, Y_t^x) = 1] - \Pr[D(S_1, \ldots, S_t, Z_1^x, \ldots, Z_t^x) = 1] \\
&= \mathbb{E}_{x \leftarrow X} \sum_{i \in [t]} \underbrace{\left(\begin{array}{ll} & \Pr[D(S_1, \ldots, S_t, Z_1^x, \ldots, Z_{i-1}^x, Y_i^x, Y_{i+1}^x, \ldots, Y_t^x) = 1] \\ - & \Pr[D(S_1, \ldots, S_t, Z_1^x, \ldots, Z_{i-1}^x, Z_i^x, Y_{i+1}^x, \ldots, Y_t^x) = 1] \end{array} \right)}_{(*)} \\
&\leq 3t\delta = \varepsilon/2
\end{aligned}$$

The last line follows since, for any x and any $i \in [t]$, if $i \notin I_x$ then $Y_i^x = Z_i^x$ and therefore $(*) = 0$, and if $i \in I_x$ then $(*) \leq 3\delta$ by the way we defined I_x in equation (2).

Step 3: Argue I_X is large. We are left to show that

$$\Pr[|I_X| < \beta \cdot t] \leq \varepsilon/4. \qquad (3)$$

We do this via a proof by contradiction. Assume otherwise that (3) does not hold. Then we show that we can guess X with high probability, which contradicts the fact that X

has high min-entropy. In particular, we define a randomized function $\mathsf{guess}()$ such that, for any x for which $|I_x| < \beta \cdot t$, we have:

$$\Pr_{\hat{x} \leftarrow \mathsf{guess}()}[\hat{x} = x] \geq \frac{1}{4} \left(t^{\beta t+1} 2^{ht} L^t 2^{\beta tn}\right)^{-1}. \tag{4}$$

Then, assuming (3) does not hold, we have

$$\begin{aligned}\Pr_{\hat{x} \leftarrow \mathsf{guess}(), x \leftarrow X}[\hat{x} = x] &\geq \Pr_{x \leftarrow X}[|I_x| < \beta t] \Pr_{\hat{x} \leftarrow \mathsf{guess}(), x \leftarrow X}[\hat{x} = x \;\mid\; |I_x| < \beta t] \\ &\geq \frac{\varepsilon}{16} \left(t^{\beta t+1} 2^{ht} L^t 2^{\beta tn}\right)^{-1}.\end{aligned}$$

which contradicts $H_\infty(X) \geq \alpha tn$.

Before defining the function $\mathsf{guess}()$, we note that by the definition of I_x in equation (2) and the"distinguishing implies predicting" lemma (Lemma 2.2), there exist some predictors P_i (depending only on D), such that, for all $x \in \{0,1\}^n$ and $i \notin I_x$, we have:

$$\Pr[P_i(S_1, \ldots, S_t, Z_1^x, \ldots, Z_{i-1}^x, Y_{i+1}^x, \ldots, Y_t^x) = Y_i^x] \geq \frac{1}{2^m}(1 + 3\delta) \tag{5}$$

The guessing strategy. We define $\mathsf{guess}()$ using these predictors P_i as follows:

1. Sample values $r_1, \ldots, r_t$ with $r_i \leftarrow \{0,1\}^\tau$.
2. Sample a set $\hat{I}_x \subseteq [t]$ of size $|\hat{I}_x| \leq \beta t$ uniformly at random.
3. Sample values $\hat{\sigma}_i \leftarrow \{0,1\}^h$ for $i \notin \hat{I}_x$ uniformly at random.
4. Sample values $\hat{x}_i \leftarrow \{0,1\}^n$ for $i \in \hat{I}_x$ uniformly at random.
5. Let $\{s_i^1, \ldots, s_i^Q\} = \mathsf{rec}_0(r_i)$, and $\{y_i^1, \ldots, y_i^Q\} = \mathsf{rec}_1(\hat{\sigma}_i, r_i)$.
6. Use all of the above values to define, for each $i \notin \hat{I}_x$, a randomized function $\hat{P}_i(s)$ which chooses a random $j^* \leftarrow [Q]$ and outputs:

   $$\hat{P}_i(s) = P_i(s_1^{j^*}, \ldots, s_{i-1}^{j^*}, s, s_{i+1}^{j^*}, \ldots, s_t^{j^*}, z_1^{j^*}, \ldots, z_{i-1}^{j^*}, y_{i+1}^{j^*}, \ldots, y_t^{j^*})$$

   where $z_i^{j^*} := y_i^{j^*}$ if $i \notin \hat{I}_x$ and $z_i^{j^*} \leftarrow \{0,1\}^m$ if $i \in \hat{I}_x$.
7. For each $i \notin \hat{I}_x$, define $\mathsf{cw}_i \in \Sigma^{2^d}$ by setting $\mathsf{cw}_i[s] \leftarrow \hat{P}_i(s)$, where $\Sigma = \{0,1\}^m$. Let $\mathcal{X}_i$ be the list of at most L values $\tilde{x}_i$ such that the Hamming distance between $\mathsf{ECC}(\tilde{x}_i)$ and cw_i is at most $(1+\delta)2^d$, as in Definition 3.2.
8. For each $i \notin \hat{I}_x$, sample $\hat{x}_i \leftarrow \mathcal{X}_i$.
9. Output $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_t)$.

Fix any x such that $|I_x| < \beta t$ and let us analyze $\Pr_{\hat{x} \leftarrow \mathsf{guess}()}[\hat{x} = x]$.

Event E_0. Let E_0 be the event that $\hat{I}_x = I_x$ and, for all $i \in I_x$: $\hat{x}_i = x_i$ and $\hat{\sigma}_i = \mathsf{hint}(x_i, r_i)$. Then $\Pr[E_0] \geq \left(t^{\beta t+1} 2^{ht} 2^{\beta tn}\right)^{-1}$. Let us condition on E_0 occurring for the rest of the analysis. In this case, we can replace all the "hatted" values $\hat{I}_x, \hat{\sigma}_i, \hat{x}_i$ with their "unhatted"

counterparts $I_x, \sigma_i = \mathsf{hint}(x_i, r_i), x_i$ and we have $y_i^j = \mathsf{Ext}(x_i; s_i^j)$. Furthermore, since the "hatted" values were chosen uniformly at random, E_0 is independent of the choice of $r_1, \ldots, r_t$ and of all the "unhatted" values above; therefore conditioning on E_0 does not change their distribution.

Event E_1. Now, for any fixed choice of $r_1, \ldots, r_t$, define the corresponding procedure $\hat{P}_i$ to be "good" if

$$\Pr_{s \leftarrow \{0,1\}^d}[\hat{P}_i(s) = \mathsf{Ext}(x_i; s)] \geq (1 + 2\delta)\frac{1}{2^m},$$

where the probability is over the choice of $s \leftarrow \{0,1\}^d$ and the internal randomness of $\hat{P}_i$ (i.e., the choice of the index $j^* \leftarrow [Q]$ and the values $z_i^{j^*} \leftarrow \{0,1\}^m$ for $i \in I_x$). Let E_1 be the event that for all $i \notin I_x$ we have $\hat{P}_i$ is good, where the event is over the choice of $r_1, \ldots, r_t$. Define random variables V_i^j over the choice of $r_1, \ldots, r_t$ where

$$\begin{aligned} V_i^j &= \Pr_{s \leftarrow \{0,1\}^d}[\hat{P}_i(s) = \mathsf{Ext}(x_i; s) \mid j^* = j] \\ &= \Pr_{s \leftarrow \{0,1\}^d}[P_i(s_1^j, \ldots, s_{i-1}^j, s, s_{i+1}^j, \ldots, s_t^j, z_1^j, \ldots, z_{i-1}^j, y_{i+1}^j, \ldots, y_t^j) = \mathsf{Ext}(x_i; s)]. \end{aligned}$$

and $V_i := \sum_{j \in Q} V_i^j$. Then $\hat{P}_i$ is good iff $V_i \geq Q(1 + 2\delta)\frac{1}{2^m}$. By equation (5), we have $E[V_i] = \sum_j E[V_i^j] \geq Q(1 + 3\delta)\frac{1}{2^m}$. Furthermore, for any fixed i, the variables V_i^j are pairwise independent by Definition 3.2 and the fact that V_i^j only depends on s_i^j. Therefore $Var[V_i] = \sum_j Var[V_j] \leq Q$. We can apply the Chebyshev inequality to get:

$$\begin{aligned} \Pr[E_1|E_0] &\geq 1 - \Pr\left[\exists i \notin I_x \ : \ V_i < Q(1 + 2\delta)\frac{1}{2^m}\right] \\ &\geq 1 - \sum_{i \notin I_x} \Pr\left[V_i < Q(1 + 2\delta)\frac{1}{2^m}\right] \\ &\geq 1 - \sum_{i \notin I_x} \Pr\left[|V_i - E[V_i]| > Q\delta\frac{1}{2^m}\right] \geq 1 - t\frac{2^{2m}}{\delta^2 Q} \geq \frac{1}{2} \end{aligned}$$

Event E_2. Now fix any choice of the values in steps (1)-(6) such that E_0, E_1 hold. Let cw_i be the values sampled in step 7. Define the event E_2 to hold if for all $i \notin I_x$ the value cw_i agrees with $\mathsf{ECC}(x_i)$ is at least $(1+\delta)2^{d-m}$ coordinates, where the probability is only over the internal randomness used to sample the components $\mathsf{cw}_i(s) \leftarrow \hat{P}_i(s)$. We can define random variables W_i^s which are 1 if $\mathsf{cw}_i(s) = \mathsf{Ext}(x_i; s)$ and 0 otherwise. These variables are mutually independent (since each invocation of $\hat{P}_i$ uses fresh internal randomness) and $E[\sum_s W_i^s] = 2^d \Pr_s[\hat{P}_i(s) = \mathsf{Ext}(x_i; s)] \geq (1 + 2\delta)2^{d-m}$. Therefore, by the Chernoff bound:

$$\begin{aligned} \Pr[E_2|E_1 \wedge E_0] &= 1 - \Pr[\exists i \notin I_x \ : \ \sum_s W_i^s \leq (1 + \delta)2^{d-m}] \\ &\geq 1 - \sum_{i \notin I_x} \Pr[\sum_s W_i^s \leq (1 + \delta)2^{d-m}] \\ &\geq 1 - t \cdot e^{-\delta^2 2^{d-m}/8} \geq \frac{1}{2} \end{aligned}$$

Event E_3. Finally, fix any choice of the values in steps (1)-(7) such that E_0, E_1, E_2 hold. Let E_3 be the event that for each $i \notin \hat{I}_x$ if $\hat{x}_i \leftarrow \mathcal{X}_i$ is the value sampled in step (8) then $\hat{x}_i = x_i$. Then $\Pr[E_3|E_2 \wedge E_1 \wedge E_0] \geq \left(\frac{1}{L}\right)^t$. Therefore, our guess is correct if E_0, E_1, E_2, E_3 all occur, which gives us the bound in equation (4). □

Corollary 3.4. *For any* $n, m, t, \varepsilon > 0, \alpha > 0$*, there exist extractors* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *that are* $(t, \alpha, \beta, \varepsilon)$*-somewhere extracting with either:*

1. *seed length* $d = n$ *and* $\beta = \alpha - \frac{O(m + \log t + \log(1/\varepsilon))}{n}$*, or*
2. *seed length* $d = O((\log n)(m + \log\log n + \log t + \log(1/\varepsilon)))$ *and* $\beta = \alpha - \frac{O(d)}{n}$.

4 Multi-User Security for Incompressible Encryption

Utilizing somewhere randomness extractors, we can now explore the multi-user setting for incompressible encryptions. But first, we need to formally define what it means for an incompressible PKE or SKE scheme to be multi-user secure.

We propose a simulation-based security definition. Roughly, the simulator first needs to simulate all the ciphertexts for all the instances *without* seeing any of the message queries. So far, this is akin to the standard semantic security notion for encryption. But we need to now model the fact that the adversary can store ciphertexts for later decryption, at which point it has all the private keys. We therefore add a second phase where the simulator can query for a *subset* of the messages, and then must simulate *all* the private keys. We require that no space-bounded distinguisher can distinguish between receiving real encryptions/real private keys vs receiving simulated encryptions/keys. The number of messages the simulator can query is related to the storage bound of the distinguisher.

Put formally, let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme, to define simulation-based incompressible ciphertext security for the multiple-instance setting, consider the following two experiments:

- In the real mode experiment, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ interacts with the challenger $\mathcal{C}$, who has knowledge of all the adversary's challenge messages.

  Real Mode $\mathsf{ExpReal}^{\Pi}_{\mathcal{C},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(\lambda, \eta, \ell, S)$:

  1. For $i \in [\eta]$, the challenger $\mathcal{C}$ runs $\mathsf{Gen}(1^\lambda, 1^S)$ to sample $(\mathsf{pk}_i, \mathsf{sk}_i)$.
  2. The challenger $\mathcal{C}$ sends all the pk_i's to $\mathcal{A}_1$.
  3. For each $i \in [\eta]$, $\mathcal{A}_1$ can produce up to ℓ message queries $\{m_{i,j}\}_{j\in[\ell]}$. The adversary submits all of the message queries *in one single batch* $\{m_{i,j}\}_{i,j}$ and receives $\{\mathsf{ct}_{i,j}\}_{i,j}$ where $\mathsf{ct}_{i,j} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_{i,j})$.
  4. $\mathcal{A}_1$ produces a state st of size at most S.
  5. On input of $\mathsf{st}, \{m_{i,j}\}_{i,j}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- In the ideal mode experiment, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ interacts with a simulator $\mathcal{S}$, which needs to simulate the view of the adversary with no/partial knowledge of the challenge messages.

  Ideal Mode $\mathsf{ExpIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(\lambda, \eta, \ell, q, S)$:

1. For $i \in [\eta]$, the simulator $\mathcal{S}$ samples pk_i.
2. The simulator $\mathcal{S}$ sends all the pk_i's to $\mathcal{A}_1$.
3. For each $i \in [\eta]$, and $j \in [\ell]$, $\mathcal{A}_1$ produces $m_{i,j}$. All of the queries $\{m_{i,j}\}_{i,j}$ are submitted in one batch and the simulator $\mathcal{S}$ produces $\{\mathsf{ct}_{i,j}\}_{i,j}$ *without* seeing $\{m_{i,j}\}_{i,j}$.
4. $\mathcal{A}_1$ produces a state st of size at most S.
5. The simulator now submits up to q number of (i,j) index pairs, and receives the corresponding messages $m_{i,j}$'s. Then $\mathcal{S}$ simulates all the secret keys sk_i's.
6. On input of $\mathsf{st}, \{m_{i,j}\}_{i,j}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

Notice that the simulator needs to simulate the ciphertexts first without knowing the corresponding messages, and then sample the secret keys so that the ciphertexts appear appropriate under the given messages.

Definition 4.1 (Multi-Instance Simulation-Based CPA Security)**.** *For security parameters $\lambda, \eta(\lambda), \ell(\lambda), q(\lambda)$ and $S(\lambda)$, a public key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is (η, ℓ, q, S)-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a simulator $\mathcal{S}$ such that:*

$$\left|\Pr\left[\mathsf{ExpReal}^{\Pi}_{\mathcal{C},\mathcal{A}}(\lambda, \eta, \ell, S) = 1\right] - \Pr\left[\mathsf{ExpIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}}(\lambda, \eta, \ell, q, S) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Remark 4.1. *If $\ell = 1$, we say that the scheme has only single-ciphertext-per-user security. For $\ell > 1$, we say that the scheme has multi-ciphertext-per-user security.*

Remark 4.2. *Notice that by replacing the underlying PKE scheme with a Symmetric Key Encryption (SKE) scheme and modifying corresponding syntaxes (sample only sk's instead of $(\mathsf{pk}, \mathsf{sk})$ pairs, and remove step 2 of the experiments where the adversary receives the pk's), we can also get a $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security definition for SKE schemes.*

5 Symmetric Key Incompressible Encryption

In this section, we explore the multi-user security of incompressible SKEs, both in the low-rate setting and the rate-1 setting. We also present a generic lifting technique to obtain an SKE with multi-ciphertext-per-user security from an SKE with single-ciphertext-per-user security.

5.1 Low Rate Incompressible SKE

For low rate incompressible SKE, it follows almost immediately from somewhere randomness extractors that the forward-secure storage by Dziembowski [Dzi06] is $\mathsf{MULT\text{-}SIM\text{-}CPA}$ secure (by using somewhere randomness extractors as the "BSM function" and using One Time Pad (OTP) as the underlying SKE primitive).

First, let us recall the construction by Dziembowski [Dzi06], with the somewhere randomness extractors and OTP plugged in.

Construction 1 (Forward-Secure Storage [Dzi06])**.** *Let λ and S be security parameters. Given $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ a $(t, \alpha, \beta, \epsilon)$-somewhere randomness extractor as defined in Definition 3.1 where the seed length $d = \mathsf{poly}(\lambda)$, output length $w = \mathsf{poly}(\lambda)$ and $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$, the construction $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}^w$ works as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Sample a seed $s \leftarrow \{0,1\}^d$ for the randomness extractor, and a key $k' \leftarrow \{0,1\}^w$. Output $k = (s, k')$.*
- $\mathsf{Enc}(k, m)$*: To encrypt a message m, first parse $k = (s, k')$ and sample a long randomness $R \leftarrow \{0,1\}^n$. Compute the ciphertext as $\mathsf{ct} = (R, \mathsf{ct}' = \mathsf{Ext}(R; s) \oplus k' \oplus m)$.*
- $\mathsf{Dec}(k, \mathsf{ct})$*: First, parse $\mathsf{ct} = (R, \mathsf{ct}')$ and $k = (s, k')$. Then compute $m = \mathsf{Ext}(R; s) \oplus k' \oplus \mathsf{ct}'$.*

Correctness is straightforward. Construction 1 is also MULT-SIM-CPA secure. Essentially, the simulator simply sends ct_i's as uniformly random strings. Then when the simulator sends the keys, it would use the simulator for the somewhere randomness extractor to get the index subset $I \subset [\eta]$, and for $i \in I$, simply send k_i as a uniformly random string. For $i \notin I$, it samples the extractor seed s_i and then compute $k'_i = m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i$. Notice that for $i \notin I$, $\mathsf{ct}'_i = m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus k'_i$, and for $i \in I$, $\mathsf{ct}'_i = m_i \oplus u_i \oplus k'_i$ where u_i is a w-bit uniform string. This is now just the definition of somewhere randomness extractors.

We prove below the MULT-SIM-CPA security of Construction 1 formally through a sequence of hybrids.

In hybrid 0, we start with the ideal mode experiment $\mathsf{ExpIdeal}^{\Pi}_{\mathcal{S},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}$ with a specific simulator plugged in, and through the sequence of hybrids, we gradually move towards the real mode experiment $\mathsf{ExpReal}^{\Pi}_{\mathcal{S},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}$.

Sequence of Hybrids

- Hybrid H_0: The ideal mode experiment $\mathsf{ExpIdeal}^{\Pi}_{\mathcal{C},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(t, 1, (1-\beta)t, S)$.
  1. For each $i \in [t]$, $\mathcal{A}_1$ produces m_i. All of the queries $\{m_i\}_i$ are submitted in a single batch and *not* available to the simulator $\mathcal{S}$. $\mathcal{S}$ samples uniformly random ciphertexts $\mathsf{ct}_i = (R_i, \mathsf{ct}'_i)$, and hence is able to produce $\{\mathsf{ct}_i\}_i$ *without* seeing $\{m_i\}_i$.
  2. $\mathcal{A}_1$ produces a state st of size at most S.
  3. The simulator $\mathcal{S}$ runs the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. The simulator now submits the set $[t] \backslash I$, and receives the corresponding messages $\{m_i\}_{i \notin I}$. Then $\mathcal{S}$ simulates all the keys k_i's. For $i \in I$, sample a uniform $k_i \leftarrow \{0,1\}^w$. For $i \notin I$, sample a uniform seed s_i, and compute $k_i = (s_i, m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i)$.
  4. On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_1: The same as H_0, except that in step 3, for $i \in I$, sample a uniform s_i and compute $k_i = (s_i, m_i \oplus u_i \oplus \mathsf{ct}'_i)$, where u_i is a uniformly random w-bit string.

- Hybrid H_2: The same as H_1, except that in step 3, for all i, sample a uniform seed s_i, and compute $k_i = (s_i, m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i)$. Notice that the game is now identical to the real mode experiment $\mathsf{ExpReal}^{\Pi}_{\mathcal{C},\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}$, where we send the adversary faithful encryptions of the message queries.

Proof of Hybrid Arguments

Lemma 5.1. *No adversary can distinguish between H_0 and H_1 with non-negligible probability.*

Proof. Notice that the only difference between H_0 and H_1 is that in H_0, for $i \in I$, we sample a uniform s_i and a uniform k'_i, and in H_1, we sample a uniform s_i and compute k'_i as $m_i \oplus u_i \oplus \mathsf{ct}'_i$, where u_i is a uniform w-bit string. This is just an One Time Pad (OTP) encryption of $m_i \oplus \mathsf{ct}'_i$, and hence should be indistinguishable from a uniformly random k'_i by the information-theoretic security of OTP. □

Lemma 5.2. *If $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ is a $(t, \alpha, \beta, \epsilon)$-somewhere randomness extractor with $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$, then no adversary can distinguish between H_1 and H_2 with non-negligible probability.*

Proof. First, notice the difference between H_1 and H_2. In H_2, for all i, we have $k_i = (s_i, m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i)$. In H_1, for $i \in I$, $k_i = (s_i, m_i \oplus u_i \oplus \mathsf{ct}'_i)$. For $i \notin I$, $k_i = (s_i, m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i)$ is the same.

Notice that each R_i is a uniformly random n-bit string independent of m_i. So by lemma 2.3, $H_\infty(\{R_i\}_i | \mathsf{st}, \{m_i\}_i) = H_\infty(\{R_i\}_i | \mathsf{st}) \geq nt - n(1-\alpha)t = \alpha \cdot tn$, i.e. $\{R_i\}_i$ has at least $\alpha \cdot tn$ bits of min-entropy conditioned on the adversary's view. And recall that I is the set of indices output by the somewhere randomness extractor simulator. We can invoke the property of the somewhere randomness extractor, and hence have

$$(s_1, \ldots, s_t, \mathsf{Ext}(R_1; s_1), \ldots, \mathsf{Ext}(R_t; s_t)) \approx_\epsilon (s_1, \ldots, s_t, Z_1, \ldots, Z_t),$$

where $Z_i = u_i$ for all $i \in I$, and $Z_i = \mathsf{Ext}(R_i; s_i)$ for all $i \notin I$. Notice that in H_1, we equivalently have $k_i = (s_i, m_i \oplus Z_i \oplus \mathsf{ct}'_i)$, and in H_2, we have $k_i = (s_i, m_i \oplus \mathsf{Ext}(R_i; s_i) \oplus \mathsf{ct}'_i)$. The only difference is that in H_1 we have the Z_i's instead of the $\mathsf{Ext}(R_i; s_i)$'s in H_2, and these are indistinguishable by the extractor property. Hence, no adversary can distinguish between H_1 and H_2 with non-negligible probability. □

Theorem 5.3. *Let λ, S be security parameters. If $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ is a $(t, \alpha, \beta, \epsilon)$-somewhere randomness extractor with $d, w = \mathsf{poly}(\lambda)$ and $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$, then Construction 1 is $(t, 1, (1-\beta)t, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ secure.*

Proof. The lemmas above show a sequence of a polynomial number of hybrid experiments where no adversary can distinguish one from the next with non-negligible probability. Notice that the first hybrid H_0 corresponds to the ideal mode experiment of multi-user security, and the last hybrid H_2 corresponds to the real mode one. The simulation-based security follows. □

Remark 5.1. *While* $\mathsf{MULT\text{-}SIM\text{-}CPA}$ *security only requires that no* PPT *adversaries can distinguish between the real mode and the ideal mode experiments, what we have proved for construction 1 here is that it is actually* $\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure against* all (potentially computationally unbounded) *adversaries, and hence is information theoretically* $\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure.*

5.2 Rate-1 Incompressible SKE

Branco, Döttling and Dujmovic [BDD22] construct rate-1 incompressible SKE from HILL-Entropic Encodings [MW20], extractors and PRGs. We show that by replacing the extractors with somewhere randomness extractors and slightly modifying the scheme, we get $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security.

First, we recall the definitions and security requirements of a HILL-Entropic Encoding scheme [MW20].

Definition 5.1 (HILL-Entropic Encoding [MW20])**.** *Let λ be the security parameter. An (α, β)-HILL-Entropic Encoding in the common random string setting is a pair of PPT algorithms $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ that works as follows:*

- $\mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m) \to c$*: On input the common random string* crs*, the security parameter, and a message, outputs a codeword c.*
- $\mathsf{Dec}_{\mathsf{crs}}(c) \to m$*: On input the common random string and a codeword, outputs the decoded message m.*

It satisfies the following properties.

Correctness. For all $\lambda \in \mathbb{N}$ and $m \in \{0,1\}^*$, $\Pr[\mathsf{Dec}_{\mathsf{crs}}(\mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)) = m] \geq 1 - \mathsf{negl}(\lambda)$.

α-Expansion. For all $\lambda, k \in \mathbb{N}$ and for all $m \in \{0,1\}^k$, $|\mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)| \leq \alpha(\lambda, k)$.

β-HILL-Entropy. There exists a simulator algorithm SimEnc such that for all polynomial $k = k(\lambda)$ and any ensemble of messages $m = \{m_\lambda\}$ of length $k(\lambda)$, consider the following real mode experiment:

- $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}$
- $c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m_\lambda)$

and let CRS, C denote the random variables for the corresponding values in the real mode experiment. Also consider the following simulated experiment:

- $(\mathsf{crs}', c') \leftarrow \mathsf{SimEnc}(1^\lambda, m_\lambda)$

and let CRS', C' be the corresponding random variables in the simulated experiment. We require that $(\mathsf{CRS}, C) \approx_c (\mathsf{CRS}', C')$ and that $H_\infty(C'|\mathsf{CRS}') \geq \beta(\lambda, k)$.

Moran and Wichs [MW20] show that we can construct HILL-Entropic Encodings in the CRS model from either the Decisional Composite Residuosity (DCR) assumption [Pai99, DJ01] or the Learning with Errors (LWE) problem [Reg05]. Their construction achieves $\alpha(\lambda, k) = k(1 + o(1)) + \mathsf{poly}(\lambda)$ and $\beta(\lambda, k) = k(1 - o(1)) - \mathsf{poly}(\lambda)$, which we call a "good" HILL-entropic encoding.

Now we reproduce the construction from [BDD22] with the somewhere randomness extractors and some other minor changes (highlighted below).

Construction 2 ([BDD22]). *Let λ and S be security parameters. Given* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *a $(t, \alpha, \beta, \epsilon)$-somewhere randomness extractor as defined in Definition 3.1 where the seed length $d = \mathsf{poly}(\lambda)$, $w = \mathsf{poly}(\lambda)$ and $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$,* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *a "good" (α', β')-HILL-Entropic Encoding scheme, and* $\mathsf{PRG} : \{0,1\}^w \to \{0,1\}^n$ *a pseudorandom generator secure against non-uniform adversaries, the construction $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}^n$ works as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Sample a seed $s \leftarrow \{0,1\}^d$ for the randomness extractor, a common random string* $\mathsf{crs} \in \{0,1\}^{\mathsf{poly}(\lambda,n)}$ *for the HILL-Entropic Encoding, and a random pad $r \leftarrow \{0,1\}^n$. Output $k = (s, r, \mathsf{crs})$.*
- $\mathsf{Enc}(k, m)$*: To encrypt a message m, first parse $k = (s, r, \mathsf{crs})$ and sample a random PRG seed $s' \leftarrow \{0,1\}^w$. Compute $c_1 = \mathsf{Code.Enc}_{\mathsf{crs}}(1^\lambda, \mathsf{PRG}(s') \oplus r \oplus m)$ and $c_2 = s' \oplus \mathsf{Ext}(c_1, s)$. The final ciphertext is $\mathsf{ct} = (c_1, c_2)$.*
- $\mathsf{Dec}(k, \mathsf{ct})$*: First, parse $\mathsf{ct} = (c_1, c_2)$ and $k = (s, r, \mathsf{crs})$. Then compute $s' = \mathsf{Ext}(c_1; s) \oplus c_2$ and obtain $m = \mathsf{Code.Dec}_{\mathsf{crs}}(c_1) \oplus \mathsf{PRG}(s') \oplus r$.*

Correctness follows from the original construction and should be easy to verify. Notice that by the α'-expansion of the "good" HILL-entropic encoding, the ciphertexts have length $(1 + o(1))n + w + \mathsf{poly}(\lambda) = (1 + o(1))n + \mathsf{poly}(\lambda)$ (the $\mathsf{poly}(\lambda)$ part is independent of n), while the messages have length n. Hence the scheme achieves an optimal rate of 1 ($(1 - o(1))$ to be exact). The keys are bit longer though, having size $d + n + \mathsf{poly}(\lambda, n) = n + \mathsf{poly}(\lambda, n)$. Furthermore, Moran and Wichs [MW20] show that the CRS needs to be at least as long as the message being encoded. Thus the key has length at least $2n + \mathsf{poly}(\lambda)$.

We prove security of Construction 2 through a sequence of hybrids.

Sequence of Hybrids

- Hybrid H_0:
  - Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    - Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    - Sample $r_i \leftarrow \{0,1\}^n$ uniformly at random.
    - Sample $s'_i \leftarrow \{0,1\}^w$ uniformly at random.
    - Sample crs_i uniformly at random.
    - Let $c_{1,i} \leftarrow \mathsf{Code.Enc}_{\mathsf{crs}_i}(1^\lambda, \mathsf{PRG}(s'_i) \oplus r_i \oplus m_i)$.
    - Let $c_{2,i} \leftarrow s'_i \oplus \mathsf{Ext}(c_{1,i}; s_i)$.
    - Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - Let $\{k_i\}_i = \{(s_i, r_i, \mathsf{crs}_i)\}_i$.
  - On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_1:

- Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$.
- For each $i \in [t]$:
  - Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
  - Sample $r_i \leftarrow \{0,1\}^n$ uniformly at random.
  - Sample $s'_i \leftarrow \{0,1\}^w$ uniformly at random.
  - Let $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, \mathsf{PRG}(s'_i) \oplus r_i \oplus m_i)$.
  - Let $c_{2,i} \leftarrow s'_i \oplus \mathsf{Ext}(c_{1,i}; s_i)$.
  - Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.
- Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
- Let $\{k_i\}_i = \{(s_i, r_i, \mathsf{crs}_i)\}_i$.
- On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_2:
  - Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    - Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    - Sample $u_i \leftarrow \{0,1\}^n$ uniformly at random.
    - Sample $s'_i \leftarrow \{0,1\}^w$ uniformly at random.
    - Let $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, u_i)$.
    - Let $c_{2,i} \leftarrow s'_i \oplus \mathsf{Ext}(c_{1,i}; s_i)$.
    - Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    - Let $r_i = u_i \oplus \mathsf{PRG}(s'_i) \oplus m_i$.
    - Let $k_i = (s_i, r_i, \mathsf{crs}_i)$.
  - On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_3:
  - Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    - Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    - Sample $u_i \leftarrow \{0,1\}^n$ uniformly at random.
    - Let $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, u_i)$.
    - Sample $c_{2,i} \leftarrow \{0,1\}^w$ uniformly at random.
    - Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:

* Let $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus \mathsf{Ext}(c_{1,i}; s_i)) \oplus m_i$.
* Let $k_i = (s_i, r_i, \mathsf{crs}_i)$.

– On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_4:

– Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$.

– For each $i \in [t]$:

* Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
* Sample $u_i \leftarrow \{0,1\}^n$ uniformly at random.
* Let $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, u_i)$.
* Sample $c_{2,i} \leftarrow \{0,1\}^w$ uniformly at random.
* Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.

– Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

– Run the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. For each $i \in [t]$:

* If $i \in I$, let $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus v_i) \oplus m_i$ where v_i is a uniformly sampled w-bit string.
* If $i \notin I$, let $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus \mathsf{Ext}(c_{1,i}; s_i)) \oplus m_i$.
* Let $k_i = (s_i, r_i, \mathsf{crs}_i)$.

– On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_5:

– Run the adversary $\mathcal{A}_1$ to receive $\{m_i\}_i$ for $i \in [t]$. Discard $\{m_i\}_i$ without looking at it.

– For each $i \in [t]$:

* Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
* Sample $u_i \leftarrow \{0,1\}^n$ uniformly at random.
* Let $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, u_i)$.
* Sample $c_{2,i} \leftarrow \{0,1\}^w$ uniformly at random.
* Let $\mathsf{ct}_i = (c_{1,i}, c_{2,i})$.

– Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

– Run the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. Submit the set $[t] \backslash I$, and receive the corresponding messages $\{m_i\}_{i \notin I}$. For each $i \in [t]$:

* If $i \in I$, sample a uniform $r_i \leftarrow \{0,1\}^n$.
* If $i \notin I$, let $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus \mathsf{Ext}(c_{1,i}; s_i)) \oplus m_i$.
* Let $k_i = (s_i, r_i, \mathsf{crs}_i)$.

– On input of $\mathsf{st}, \{m_i\}_i, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

Proof of Hybrid Arguments

Lemma 5.4. *If* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *has* β'*-HILL-entropy, then no PPT adversary can distinguish between* H_0 *and* H_1 *with non-negligible probability.*

Proof. The only difference between H_0 and H_1 is that in H_0, crs_i is sampled uniformly random and $c_{1,i} \leftarrow \mathsf{Code.Enc}_{\mathsf{crs}_i}(1^\lambda, \mathsf{PRG}(s'_i) \oplus r_i \oplus m_i)$, while in H_1, we get $(\mathsf{crs}_i, c_{1,i}) \leftarrow \mathsf{SimEnc}(1^\lambda, \mathsf{PRG}(s'_i) \oplus r_i \oplus m_i)$. By the β'-HILL-entropy, the crs_i and $c_{1,i}$ in H_0 are computationally indistinguishable from the ones in H_1. Hence, no PPT adversary can distinguish between H_0 and H_1 with non-negligible probability. □

Lemma 5.5. *No adversary can distinguish between* H_1 *and* H_2 *with non-negligible probability.*

Proof. Here we are just changing the ways the variables are sampled. In H_1, we sample a uniform r_i and compute $u_i = \mathsf{PRG}(s'_i) \oplus r_i \oplus m_i$, while in H_2, we sample a uniform u_i, and then compute $r_i = \mathsf{PRG}(s'_i) \oplus u_i \oplus m_i$. These two ways of sampling are equivalent, and hence no adversary can distinguish between H_1 and H_2 with non-negligible probability. □

Lemma 5.6. *No adversary can distinguish between* H_2 *and* H_3 *with non-negligible probability.*

Proof. This step is similar to the previous one, another change of variables. In H_2, we sample a uniform s'_i, and compute $c_{2,i} = s'_i \oplus \mathsf{Ext}(c_{1,i}; s_i)$, while in H_2, we sample a uniform $c_{2,i}$ and compute $s'_i = c_{2,i} \oplus \mathsf{Ext}(c_{1,i}; s_i)$. These are equivalent and hence no adversary can distinguish. □

Lemma 5.7. *If* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *is a "good" HILL-entropic encoding with* β'*-HILL-entropy, and* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *is a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor with* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$*, then no adversary can distinguish between* H_3 *and* H_4 *with non-negligible probability.*

Proof. By the β'-HILL-entropy and the goodness of the encoding scheme, $H_\infty(c_{1,i}|\mathsf{crs}_i) \geq \beta'(\lambda, n) = n(1-o(1)) - \mathsf{poly}(\lambda)$. With all the $c_{1,i}$'s combined, we have $H_\infty(\{c_{1,i}\}_i|\{\mathsf{crs}_i\}_i) \geq tn(1-o(1)) - t\mathsf{poly}(\lambda)$. Then, by the fact that $c_{1,i}$'s are sampled independent of the m_i's and lemma 2.3, $H_\infty(\{c_{1,i}\}_i|\{\mathsf{crs}_i\}_i, \{m_i\}_i, \mathsf{st}) = H_\infty(\{c_{1,i}\}_i|\{\mathsf{crs}_i\}_i, \mathsf{st}) \geq tn(1-o(1)) - (1-\alpha)nt = \alpha \cdot tn$. Therefore, we can invoke the somewhere randomness extraction property and have

$$(s_1, \ldots, s_t, \mathsf{Ext}(c_{1,1}; s_1), \ldots, \mathsf{Ext}(c_{1,t}; s_t)) \approx_\epsilon (s_1, \ldots, s_t, Z_1, \ldots, Z_t),$$

where $Z_i = v_i$ for all $i \in I$, and $Z_i = \mathsf{Ext}(c_{1,i}; s_i)$ for all $i \notin I$. Notice that in H_3, we have $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus \mathsf{Ext}(c_{1,i}; s_i)) \oplus m_i$, and in H_4, we equivalently have $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus Z_i) \oplus m_i$. The only difference is that in H_4 we have the Z_i's instead of the $\mathsf{Ext}(c_{1,i}; s_i)$'s in H_3, and these are indistinguishable by the extractor property. Hence, no adversary can distinguish between H_3 and H_4 with non-negligible probability. □

Lemma 5.8. *If* PRG *is a pseudorandom generator secure against non-uniform adversaries, then no PPT adversary can distinguish between* H_4 *and* H_5 *with non-negligible probability.*

Proof. First, notice that in H_4, for $i \in I$, we compute $r_i = u_i \oplus \mathsf{PRG}(c_{2,i} \oplus v_i) \oplus m_i$, where v_i is a uniformly random string. This is equivalent as $r_i = u_i \oplus \mathsf{PRG}(v_i') \oplus m_i$ where v_i' is a uniformly random string. So here we are running the PRG on a uniformly random seed. Although we do need to run the *inefficient* simulator for the somewhere randomness extractor earlier, we can still replace the PRG output with random if the PRG is secure *against non-uniform adversaries*. Hence we have $r_i = u_i \oplus u_i' \oplus m_i$ where u_i' is a uniformly random n-bit string, and this is just equivalent as having a uniformly random r_i, which is the exact case in H_5. Therefore, no PPT adversary can distinguish between H_4 and H_5 with non-negligible probability. □

Theorem 5.9. *If* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *is a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor with* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$*,* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *is a "good" HILL-entropic encoding with* β'*-HILL-entropy, and* PRG *is a pseudorandom generator secure against non-uniform adversaries, then Construction 2 is* $(t, 1, (1-\beta)t, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure.*

Proof. The lemmas above show a sequence of a polynomial number of hybrid experiments where no PPT adversary can distinguish one from the next with non-negligible probability. Notice that the first hybrid H_0 corresponds to the real mode experiment of multi-user security, and the last hybrid H_5 corresponds to the ideal mode one. The simulation-based security follows. □

5.3 Dealing with Multiple Messages per User

Above we have showed $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security for SKE schemes where the number of messages per user ℓ is equal to 1. Here, we show how we can generically lift a SKE scheme with single-message-per-user $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security to multiple-messages-per-user $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security.

Construction 3. *Let* λ, S *be security parameters. Given* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *a* $(\eta, 1, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure SKE with key space* $\{0,1\}^n$ [9] *and* $\mathcal{F}$ *a class of* ℓ*-wise independent functions with range* $\{0,1\}^n$*, we construct* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *as follows.*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Sample a random function* $f \leftarrow \mathcal{F}$*. Output* $k = f$*.*
- $\mathsf{Enc}(k = f, m)$ *: Sample a short random string* r *with* $|r| = \mathsf{polylog}(\ell)$*, compute* $k' = f(r)$*, and get* $c \leftarrow \mathsf{SKE.Enc}(k', m)$*. Output* $\mathsf{ct} = (r, c)$*.*
- $\mathsf{Dec}(k = f, \mathsf{ct} = (r, c))$ *: Compute* $k' = f(r)$*, and output* $m \leftarrow \mathsf{SKE.Dec}(k', c)$*.*

Correctness should be easy to verify given the correctness of the underlying SKE scheme and the deterministic property of the ℓ-wise independent functions.

Lemma 5.10. *If* SKE *is a* $(\eta, 1, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure SKE with key space* $\{0,1\}^n$ *and* $\mathcal{F}$ *is a class of* ℓ*-wise independent functions with range* $\{0,1\}^n$*, then Construction 3 is* $(\eta/\ell, \ell, q, S - \eta \cdot \mathsf{polylog}(\ell))$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure.*

[9] Here we assume SKE's keys are uniformly random n-bit strings. This is without loss of generality since we can always take the key to be the random coins for Gen.

Proof. We prove this through a reduction. We show that if there is an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the $(\eta/\ell, \ell, q, S - \eta \cdot \mathsf{polylog}(\ell))$-MULT-SIM-CPA security of Π, then we can construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that breaks the $(\eta, 1, q, S)$-MULT-SIM-CPA security of SKE. $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ works as follows:

- $\mathcal{A}'_1$: First, run $\mathcal{A}_1$ to get a list of message queries $\{m_{i,j}\}_{i\in[\eta/\ell], j\in[\ell]}$. Let $m'_i = m_{(i/\ell)+1,((i-1) \bmod \ell)+1}$ for $i \in [\eta]$. Notice that here we are essentially flattening the list of messages. Submit the list $\{m'_i\}_{i\in[\eta]}$ and receive $\{\mathsf{ct}'_i\}_{i\in[\eta]}$. Reconstruct $\mathsf{ct}_{i,j} = (r_{i,j}, \mathsf{ct}'_{(i-1)\cdot\ell+j})$ for $i \in [\eta/\ell]$ and $j \in [\ell]$, where $r_{i,j}$ is a uniformly random string sampled from $\{0,1\}^{\mathsf{polylog}(\ell)}$. Notice that the $r_{i,j}$'s have no collisions under the same i with overwhelming probability. Send the list of ciphertexts $\{\mathsf{ct}_{i,j}\}_{i,j}$ back to $\mathcal{A}_1$ and receive a state st. Output the state $\mathsf{st}' = (\mathsf{st}, \{r_{i,j}\}_{i,j})$. The size of the state is $|\mathsf{st}| + \eta \cdot \mathsf{polylog}(\ell) \leq S - \eta \cdot \mathsf{polylog}(\ell) + \eta \cdot \mathsf{polylog}(\ell) = S$.
- $\mathcal{A}'_2$: First receive $\mathsf{st}' = (\mathsf{st}, \{r_{i,j}\}_{i,j}), \{m'_i\}_{i\in[\eta]}, \{k'_i\}_{i\in[\eta]}$ from the challenger / simulator. Reorganize $m_{i,j} = m'_{(i-1)\cdot\ell+j}$ for $i \in [\eta/\ell]$ and $j \in [\ell]$. Construct k_i as an ℓ-wise independent function f_i s.t. for all $i \in [\eta/\ell]$ and $j \in [\ell]$, $f_i(r_{i,j}) = k'_{(i-1)\cdot\ell+j}$. Send $\mathsf{st}, \{m_{i,j}\}_{i\in[\eta/\ell], j\in[\ell]}, \{k_i = f_i\}_{i\in[\eta/\ell]}$ to $\mathcal{A}_2$ and receive a bit b. Output b.

Notice that $\mathcal{A}'$ perfectly simulates the view for $\mathcal{A}$. If $\mathcal{A}$ says it is in the real mode, this means the ciphertexts are faithful encryptions of the message queries, and hence $\mathcal{A}'$ should be in the real mode as well, and vice versa. Therefore, construction 3 is $(\eta/\ell, \ell, q, S - \eta \cdot \mathsf{polylog}(\ell))$-MULT-SIM-CPA secure. □

6 Public Key Incompressible Encryption

Here we explore multi-user security of incompressible Public Key Encryptions (PKEs), considering constructions from [GWZ22, BDD22]. Unlike the SKE setting, where we can generically lift single-ciphertext-per-user security to multi-ciphertext-per-user security, here we show how to obtain multi-ciphertext security by modifying each construction specifically.

6.1 Low Rate Incompressible PKE

For low rate incompressible PKE, we show that the construction from [GWZ22] is MULT-SIM-CPA secure by plugging in the somewhere randomness extractor. Then, we upgrade the construction to have multi-ciphertext-per-user security by upgrading the functionality of the underlying functional encryption scheme.

Construction by [GWZ22]. We recall the low rate incompressible PKE construction by [GWZ22], with the somewhere randomness extractor plugged in.

Construction 4 ([GWZ22])**.** *Given* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *a single-key selectively secure functional encryption scheme and a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$*, with* $d = \mathsf{poly}(\lambda)$*,* $w = \mathsf{poly}(\lambda)$ *and* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$*, the construction* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\{0,1\}^w$ *works as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$: *First, obtain* $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. *Then, generate the secret key for the following function* f_v *with a hardcoded* $v \in \{0,1\}^{d+w}$*:*

$$f_v(s' = (s, \mathsf{pad}), \mathsf{flag}) = \begin{cases} s' & \text{if } \mathsf{flag} = 0 \\ s' \oplus v & \text{if } \mathsf{flag} = 1 \end{cases}.$$

  Output $\mathsf{pk} = \mathsf{FE.mpk}$ *and* $\mathsf{sk} = \mathsf{FE.sk}_{f_v} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_v)$.

- $\mathsf{Enc}(\mathsf{pk}, m)$: *Sample a random tuple* $s' = (s, \mathsf{pad})$ *where* $s \in \{0,1\}^d$ *is used as a seed for the extractor and* $\mathsf{pad} \in \{0,1\}^w$ *is used as a one-time pad. The ciphertext consists of three parts:* $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (s', 0))$*, a long randomness* $R \in \{0,1\}^n$*, and* $z = \mathsf{Ext}(R; s) \oplus \mathsf{pad} \oplus m$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{FE.ct}, R, z))$: *First, obtain* $s' \leftarrow \mathsf{FE.Dec}(\mathsf{FE.sk}_{f_v}, \mathsf{FE.ct})$*, and then use the seed* s *to compute* $\mathsf{Ext}(R; s) \oplus z \oplus \mathsf{pad}$ *to recover* m.

The correctness follows from the original construction.

Theorem 6.1. *If* FE *is a single-key selectively secure functional encryption scheme and* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *is a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor with* $d, w = \mathsf{poly}(\lambda)$ *and* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$*, then Construction 4 is* $(t, 1, (1-\beta)t, S)$-MULT-SIM-CPA *secure.*

We prove Theorem 6.1 through a sequence of hybrids, starting with H_0 being the real mode experiment and ending with H_3 being the ideal mode experiment. The proofs of the hybrid arguments are identical to those from [GWZ22] (except for the extractor step, which is analogous to the proof of Lemma 5.2), so we will not reproduce them here and instead point the reader to the original [GWZ22] paper.

Sequence of Hybrids

- Hybrid H_0:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and sample a uniform random $v_i \leftarrow \{0,1\}^{d+w}$. Set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ and $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i})$.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    - Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    - Sample $\mathsf{pad}_i \leftarrow \{0,1\}^w$ uniformly at random.
    - Let $s'_i = (s_i, \mathsf{pad}_i)$.
    - Let $\mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (s'_i, 0))$.
    - Sample $R_i \leftarrow \{0,1\}^n$ uniformly at random.
    - Let $z_i = \mathsf{Ext}(R_i; s_i) \oplus \mathsf{pad}_i \oplus m_i$.
    - Let $\mathsf{ct}_i = (\mathsf{FE.ct}_i, R_i, z_i)$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

– On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_1:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and sample a uniform random $v_i \leftarrow \{0,1\}^{d+w}$. Set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ and $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i})$.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    * Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    * Sample $\mathsf{pad}_i \leftarrow \{0,1\}^w$ uniformly at random.
    * Let $s'_i = (s_i, \mathsf{pad}_i)$.
    * Let $\mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (s'_i \oplus v_i, 1))$.
    * Sample $R_i \leftarrow \{0,1\}^n$ uniformly at random.
    * Let $z_i = \mathsf{Ext}(R_i; s_i) \oplus \mathsf{pad}_i \oplus m_i$.
    * Let $\mathsf{ct}_i = (\mathsf{FE.ct}_i, R_i, z_i)$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_2:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Set only $\mathsf{pk}_i = \mathsf{FE.mpk}_i$.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$:
    * Sample $u_i \leftarrow \{0,1\}^{d+w}$ uniformly at random.
    * Let $\mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (u_i, 1))$.
    * Sample $R_i \leftarrow \{0,1\}^n$ uniformly at random.
    * Sample $z_i \leftarrow \{0,1\}^w$ uniformly at random.
    * Let $\mathsf{ct}_i = (\mathsf{FE.ct}_i, R_i, z_i)$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    * Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
    * Let $\mathsf{pad}_i = \mathsf{Ext}(R_i; s_i) \oplus z_i \oplus m_i$.
    * Let $s'_i = (s_i, \mathsf{pad}_i)$ and compute $v_i = s'_i \oplus u_i$.
    * Obtain $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i})$.
  - On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_3:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Set only $\mathsf{pk}_i = \mathsf{FE.mpk}_i$.

– Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$. Discard $\{m_i\}_i$ without looking at it.

– For each $i \in [t]$:

  * Sample $u_i \leftarrow \{0,1\}^{d+w}$ uniformly at random.
  * Let $\mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (u_i, 1))$.
  * Sample $R_i \leftarrow \{0,1\}^n$ uniformly at random.
  * Sample $z_i \leftarrow \{0,1\}^w$ uniformly at random.
  * Let $\mathsf{ct}_i = (\mathsf{FE.ct}_i, R_i, z_i)$.

– Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

– Run the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. Submit the set $[t] \backslash I$, and receive the corresponding messages $\{m_i\}_{i \notin I}$. For each $i \in [t]$:

  * Sample $s_i \leftarrow \{0,1\}^d$ uniformly at random.
  * If $i \in I$, sample $\mathsf{pad}_i \leftarrow \{0,1\}^w$ uniformly at random.
  * If $i \notin I$, let $\mathsf{pad}_i = \mathsf{Ext}(R_i; s_i) \oplus z_i \oplus m_i$.
  * Let $s'_i = (s_i, \mathsf{pad}_i)$ and compute $v_i = s'_i \oplus u_i$.
  * Obtain $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i})$.

– On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

Upgrading to Multiple Ciphertexts Per User. Additionally, We show that the constructions from [GWZ22] can be upgraded to have multi-ciphertext-per-user security. Essentially, all we need is to upgrade the functionality of the underlying functional encryption scheme to work for a slightly more generalized class of functions. We will need functions $f_{\{v_i\}_i}(s, \mathsf{flag}) = s \oplus v_{\mathsf{flag}}$ for hard coded values $v_1, \ldots, v_\ell$ and a special v_0 being the all 0 string. Notice that the original GWZ construction [GWZ22] can be viewed as using functions that are a special case where $\ell = 1$. We show how to construct FE schemes for such $f_{\{v_i\}_i}$ functions from plain PKE below. With this new class of functions, we can achieve $(t, \ell, (1-\beta)\ell t, S)$-MULT-SIM-CPA security. In the hybrid proof where we replace $\mathsf{FE.Enc}(\mathsf{FE.mpk}, (s', 0))$ with $\mathsf{FE.Enc}(\mathsf{FE.mpk}, (s' \oplus v, 1))$, now for the j-th message query for the i-th user where $i \in [t]$ and $j \in [\ell]$, we replace $\mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (s'_{i,j}, 0))$ with $\mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (s'_{i,j} \oplus v_{i,j}, j))$. The rest of the hybrid proof follows analogously.

Instantiating FE for $f_{\{v_i\}_i}$ Functions. Here we show how to construct the FE scheme for the new class of functions that we need to upgrade construction 4 to have multi-ciphertext-per-user security. We only need plain PKE for the construction. Recall that our functions $f_{\{v_i\}_i}$ have the form $f_{\{v_i\}_i}(s, \mathsf{flag}) = s \oplus v_{\mathsf{flag}}$, where $\mathsf{flag} \in \{0, 1, \ldots, \ell\}$.

Construction 5. *Let* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *be a public key encryption scheme. Our scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a single message bit* s *is defined as:*

- $\mathsf{Setup}(1^\lambda)$*: For* $i \in \{0, 1, \ldots, n\}, b \in \{0,1\}$*, run* $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{Gen}'(1^\lambda)$*. Output* $(\mathsf{mpk} = \{\mathsf{pk}_{i,b}\}_{i,b}\ ,\ \mathsf{msk} = \{\mathsf{sk}_{i,b}\}_{i,b})$.
- $\mathsf{KeyGen}(\mathsf{msk}, f_{\{v_i\}_i}) = \{\mathsf{sk}_{i,v_i}\}_i$*. Notice that we hardcode* $v_0 = 0$.

- $\mathsf{Enc}(\mathsf{mpk}, (s, \mathsf{flag}))$*: Sample uniformly random bits* $s^{(0)}, s^{(1)}, \ldots, s^{(n)}$ *s.t.* $s^{(0)} \oplus s^{(1)} \oplus \cdots \oplus s^{(n)} = s$*. For* $i \in \{0, 1, \ldots, n\} \backslash \{\mathsf{flag}\}, b \in \{0,1\}$*, compute* $c_{i,b} = \mathsf{Enc}'(\mathsf{pk}_{i,b}, s^{(i)})$*. For* $b \in \{0,1\}$*, compute* $c_{\mathsf{flag},b} = \mathsf{Enc}'(\mathsf{pk}_{i,b}, s^{(\mathsf{flag})} \oplus b)$*. Output* $c = (c_{i,b})_{i,b}$*.*

- $\mathsf{Dec}(\mathsf{sk}_{\{v_i\}_i}, c)$*: Output* $x = x^{(0)} \oplus x^{(1)} \oplus \cdots \oplus x^{(n)}$ *where* $x^{(i)} = \mathsf{Dec}'(\mathsf{sk}_{i,v_i}, c_{i,v_i})$

For correctness, note that for $i \neq \mathsf{flag}$, $x^{(i)} = s^{(i)}$, and that $x^{(\mathsf{flag})} = s^{(\mathsf{flag})} \oplus v_{\mathsf{flag}}$, therefore $x = s^{(0)} \oplus s^{(1)} \oplus \cdots \oplus s^{(n)} \oplus v_{\mathsf{flag}} = s \oplus v_{\mathsf{flag}}$.

Lemma 6.2. *If* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *is a CPA secure public key encryption scheme, then Construction 5 is single key semi-adaptively secure for the functions* $f_{\{v_i\}_i}$*.*

Proof. Consider a single key semi-adaptive adversary for Construction 5. Let $m_0 = (s_0, \mathsf{flag}_0), m_1 = (s_1, \mathsf{flag}_1)$ be the challenge messages. For a fixed flag, $f_{\{v_i\}_i}$ is injective. Therefore, if $m_0 \neq m_1$, it must be that $\mathsf{flag}_0 \neq \mathsf{flag}_1$. Then if the adversary's secret key query is on $f_{\{v_i\}_i}$, we must have $s_0 \oplus v_{\mathsf{flag}_0} = s_1 \oplus v_{\mathsf{flag}_1}$. Therefore the c_{i,v_i}'s always encrypt an instance of a secret share of the same value $s_0 \oplus v_{\mathsf{flag}_0} = s_1 \oplus v_{\mathsf{flag}_1}$. Hence, for $i \notin \{\mathsf{flag}_0, \mathsf{flag}_1\}$, $\hat{b} \in \{0,1\}$, $c_{i,\hat{b}}$'s follow the same distribution in both cases and do not depend on the challenge bit b. The only dependence on the challenge bit b is that $c_{\mathsf{flag}_b,0}$ always encrypts the opposite bit that $c_{\mathsf{flag}_b,1}$ encrypts, whereas $c_{\mathsf{flag}_{1-b},0}$ and $c_{\mathsf{flag}_{1-b},1}$ always encrypt the same bit. However, since the adversary never gets to see the secret key $\mathsf{sk}_{\mathsf{flag}_b,1-v_{\mathsf{flag}_b}}$, a simple hybrid argument shows that flipping the challenge bit is indistinguishable. □

6.2 Rate-1 Incompressible PKE

For rate-1 incompressible PKE, we first show that we can easily plug in the somewhere randomness extractor to the construction by Guan, Wichs and Zhandry [GWZ22]. We also provide a generalization on the construction by Branco, Döttling and Dujmovic [BDD22] using a Key Encapsulation Mechanism (KEM) with a special *non-committing* property. For both constructions, we show how to adapt them to allow for multi-ciphertext-per-user security.

Construction by [GWZ22]. We first reproduce the rate-1 PKE construction from [GWZ22], with the somewhere randomness extractors plugged in.

Construction 6 ([GWZ22])**.** *Given* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *a rate-1 functional encryption scheme satisfying single-key semi-adaptive security,* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor with* $d, w = \mathsf{poly}(\lambda)$*,* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$ *and* $\mathsf{PRG} : \{0,1\}^w \to \{0,1\}^n$ *a secure PRG against non-uniform adversaries, the construction* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for message space* $\{0,1\}^n$ *works as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: First, obtain* $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$*. Then, generate the secret key for the following function* $f_{v,s}$ *with a hardcoded large random pad* $v \in \{0,1\}^n$ *and a small extractor seed* $s \in \{0,1\}^d$*:*

$$f_{v,s}(x, \mathsf{flag}) = \begin{cases} x & \text{if } \mathsf{flag} = 0 \\ \mathsf{PRG}(\mathsf{Extract}(x; s)) \oplus v & \text{if } \mathsf{flag} = 1 \end{cases}.$$

  Output $\mathsf{pk} = \mathsf{FE.mpk}$ *and* $\mathsf{sk} = \mathsf{FE.sk}_{f_{v,s}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_{v,s})$*.*

- $\mathsf{Enc}(\mathsf{pk}, m)$*: The ciphertext is simply an encryption of* $(m, 0)$ *using the underlying* FE *scheme, i.e.* $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (m, 0))$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: Decryption also corresponds to* FE *decryption. The output is simply* $\mathsf{FE.Dec}(\mathsf{FE.sk}_{f_{v,s}}, \mathsf{ct}) = f_{v,s}(m, 0) = m$ *as desired.*

Correctness easily follows from the original construction. The rate of the construction is the rate of the underlying FE multiplied by $\frac{n}{n+1}$. If the FE has rate $(1 - o(1))$, the construction has rate $(1 - o(1))$ as desired.

Theorem 6.3. *If* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a single-key semi-adaptively secure functional encryption scheme,* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^w$ *is a* $(t, \alpha, \beta, \epsilon)$*-somewhere randomness extractor, with* $d, w = \mathsf{poly}(\lambda)$ *and* $n = \frac{S}{(1-\alpha)t} + \mathsf{poly}(\lambda)$*, and* $\mathsf{PRG} : \{0,1\}^w \to \{0,1\}^n$ *is a PRG secure against non-uniform adversaries, then Construction 6 is* $(t, 1, (1-\beta)t, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *secure.*

We prove Theorem 6.3 through a sequence of hybrids, starting with H_0 being the real mode experiment where we play the role of the challenger and ending with H_6 being the ideal mode experiment where we play the role of the simulator. For the proofs of each hybrid argument, see the original [GWZ22] paper, since they are identical except for the extractor step (analogous to Lemma 5.2) and the PRG against non-uniform attackers step (analogous to Lemma 5.8).

Sequence of Hybrids

- Hybrid H_0:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and sample uniformly random $v_i \leftarrow \{0,1\}^n$ and $s_i \leftarrow \{0,1\}^d$. Set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ and $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$, let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (m_i, 0))$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit $1/0$.
- Hybrid H_1:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$, let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (m_i, 0))$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    - Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    - Sample a uniformly random $u_i \leftarrow \{0,1\}^n$, and let $v_i = u_i \oplus m_i$.

* Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.

– On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_2:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$, let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (m_i, 0))$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    * Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    * Sample a uniformly random PRG key $k_i \leftarrow \{0,1\}^w$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.
    * Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.
  - On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_3:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$, sample a uniformly random $R_i \leftarrow \{0,1\}^n$, and let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (m_i, 0))$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    * Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    * Let $k_i = \mathsf{Ext}(R_i; s_i)$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.
    * Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.
  - On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.
- Hybrid H_4:
  - For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.
  - Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.
  - For each $i \in [t]$, sample a uniformly random $R_i \leftarrow \{0,1\}^n$, and let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (R_i, 1))$.
  - Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.
  - For each $i \in [t]$:
    * Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    * Let $k_i = \mathsf{Ext}(R_i; s_i)$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.

* Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.

– On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_5:

  – For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.

  – Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$.

  – For each $i \in [t]$, sample a uniformly random $R_i \leftarrow \{0,1\}^n$, and let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (R_i, 0))$.

  – Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

  – Run the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. For each $i \in [t]$:

    * Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    * If $i \in I$, sample a uniformly random PRG key $k_i \leftarrow \{0,1\}^w$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.
    * If $i \notin I$, let $k_i = \mathsf{Ext}(R_i; s_i)$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.
    * Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.

  – On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

- Hybrid H_6:

  – For each $i \in [t]$, obtain $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Only set $\mathsf{pk}_i = \mathsf{FE.mpk}_i$ for now.

  – Send $\{\mathsf{pk}_i\}_i$ to the adversary $\mathcal{A}_1$ and receive $\{m_i\}_i$ for $i \in [t]$. Discard $\{m_i\}_i$ without looking at it.

  – For each $i \in [t]$, sample a uniformly random $R_i \leftarrow \{0,1\}^n$, and let $\mathsf{ct}_i = \mathsf{FE.ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_i, (R_i, 0))$.

  – Send $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ and receive a state st.

  – Run the simulator for the somewhere randomness extractor to get a set of indices $I \subseteq [t]$ with $|I| \geq \beta t$. Submit the set $[t] \backslash I$, and receive the corresponding messages $\{m_i\}_{i \notin I}$. For each $i \in [t]$:

    * Sample a uniformly random $s_i \leftarrow \{0,1\}^d$.
    * If $i \in I$, sample a uniformly random $v_i \leftarrow \{0,1\}^n$.
    * If $i \notin I$, let $k_i = \mathsf{Ext}(R_i; s_i)$, and let $v_i = \mathsf{PRG}(k_i) \oplus m_i$.
    * Let $\mathsf{sk}_i = \mathsf{FE.sk}_{f_{v_i,s_i}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, f_{v_i,s_i})$.

  – On input of $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

Upgrading to Multiple Ciphertexts Per User. Upgrading Construction 6 to multi-ciphertext-per-user security is rather straightforward. Since the construction already requires a full functionality FE scheme, we just modify the class of functions that the underlying FE scheme uses, without introducing any new assumptions. Specifically, we use the class of functions $f_{\{v_j\}_j,\{s_j\}_j}$ with hard-coded values $v_j \in \{0,1\}^n$ and $s_j \in \{0,1\}^d$ for $j \in [\ell]$ that behaves as follows:

$$f_{\{v_j\}_j,\{s_j\}_j}(x, \mathsf{flag}) = \begin{cases} x & \text{if } \mathsf{flag} = 0 \\ \mathsf{PRG}(\mathsf{Extract}(x; s_{\mathsf{flag}})) \oplus v_{\mathsf{flag}} & \text{if } \mathsf{flag} \in [\ell] \end{cases}.$$

This gives us $(t, \ell, (1-\alpha)\ell t, S)$-MULT-SIM-CPA security. Notice that this modification does slightly harm the rate of the scheme, since the flag is now $\log(\ell)$ bits instead of one bit, but asymptotically the rate is still $(1 - o(1))$.

The hybrid proof works analogously to that of Theorem 6.3, except that in the hybrid proof where we swap the FE encryption of $(m, 0)$ to $(R, 1)$, we now swap from $(m_{i,j}, 0)$ to $(R_{i,j}, j)$ for the j-th ciphertext from the i-th user.

Generalization of Construction by [BDD22]. [BDD22] show how to lift a rate-1 incompressible SKE scheme to a rate-1 incompressible PKE scheme using a Key Encapsulation Mechanism [CS03] built from programmable Hash Proof Systems (HPS) [CS02, Kal05]. Their construction satisfy CCA2 security. We show that if we are to relax the security notion to only CPA security, all we need for the lifting is a Key Encapsulation Mechanism with a *non-committing* property, defined as follows.

Definition 6.1 (Key Encapsulation Mechanism [CS03])**.** *Let λ be the security parameters, a Key Encapsulation Mechanism (KEM) is a tuple of algorithms* $\Pi = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *that works as follow:*

- $\mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k}) \to (\mathsf{pk}, \mathsf{sk})$*: The key generation algorithm takes as input the security parameter and the desired symmetric key length $\mathcal{L}_k$, outputs a pair of public key and private key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Encap}(\mathsf{pk}) \to (k, c)$*: The encapsulation algorithm takes the public key* pk*, produces a symmetric key $k \in \{0,1\}^{\mathcal{L}_k}$, and a header c that encapsulates k.*
- $\mathsf{Decap}(\mathsf{sk}, c) \to k$*: The decapsulation algorithm takes as input the private key* sk *and a header c, and decapsulates the header to get the symmetric key k.*

We require *correctness* of the KEM.

Definition 6.2 (Correctness)**.** *A key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *is said to be correct if:*

$$\Pr\left[k' = k : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k}) \\ (k, c) \leftarrow \mathsf{Encap}(\mathsf{pk}) \\ k' \leftarrow \mathsf{Decap}(\mathsf{sk}, c) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 6.3 (Non-Committing)**.** *A key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *is said to be* non-committing *if there exists a pair of simulator algorithm* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that* $\mathsf{Sim}_1(1^\lambda, 1^{\mathcal{L}_k})$ *outputs a simulated public key* pk'*, a header* c' *and a state* st *with* $|\mathsf{st}| = \mathsf{poly}(\lambda, \mathcal{L}_k)$*, and for any given target key* $k' \in \{0,1\}^{\mathcal{L}_k}$*,* $\mathsf{Sim}_2(\mathsf{st}, k')$ *outputs the random coins* r^{KeyGen} *and* r^{Encap}*. We require that if we run the key generation and encapsulation algorithm using these random coins, we will get the desired* pk', c'*, and* k'*, i.e.:*

$$\Pr\left[\begin{array}{c}\mathsf{pk}' = \mathsf{pk} \\ k' = k \\ c' = c\end{array} : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k}; r^{\mathsf{KeyGen}}) \\ (k, c) \leftarrow \mathsf{Encap}(\mathsf{pk}; r^{\mathsf{Encap}})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Kindly notice that by the correctness property, $\mathsf{Decap}(\mathsf{sk}, c') \to k'$*.*

This *non-committing* property allows us to commit to a public key and header first, but then later able to reveal it as an encapsulation of an arbitrary symmetric key in the key space. And it will be impossible to distinguish the simulated public key and header from the ones we get from faithfully running KeyGen and Encap.

Using this non-committing KEM, we are able to construct rate-1 incompressible PKE from rate-1 incompressible SKE, with multi-user security in mind. This is a generalization of the construction by [BDD22].

Construction 7 (Generalization of [BDD22])**.** *Let* λ, S *be security parameters. Given* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *a non-commiting KEM and* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *a rate-1 incompressible SKE for message space* $\{0,1\}^n$*, we construct rate-1 incompressible PKE* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for message space* $\{0,1\}^n$ *as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: First, run* $\mathsf{SKE.Gen}(1^\lambda, 1^S)$ *to determine the required symmetric key length* $\mathcal{L}_k$ *under security parameters* λ, S*. Then run* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda, 1^{\mathcal{L}_k})$ *and output* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk}, m)$*: First, run* $(k, c_0) \leftarrow \mathsf{KEM.Encap}(\mathsf{pk})$ *to sample a symmetric key* k*, and encapsulate it into a header* c_0*. Then compute* $c_1 \leftarrow \mathsf{SKE.Enc}(k, m)$*. The ciphertext is the tuple* (c_0, c_1)*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (c_0, c_1))$*: First, decapsulate* c_0 *using* sk *to obtain* $k \leftarrow \mathsf{KEM.Decap}(\mathsf{sk}, c_0)$*, and then use* k *to decrypt* c_1 *and get* $m \leftarrow \mathsf{SKE.Dec}(k, c_1)$*.*

Correctness follows from the correctness of the underlying incompressible SKE and the KEM scheme. In terms of the rate, to achieve a rate-1 incompressible PKE, we would require the KEM to produce "short" headers, i.e. $|c_0| = \mathsf{poly}(\lambda)$ independent of $\mathcal{L}_k$ (notice that $\mathcal{L}_k = \mathsf{poly}(\lambda, n)$ and needs to be at least as large as n). We can build such KEMs using various efficient encapsulation techniques [BBB+17, ACP+19, BCL+17]. With the short header and an incompressible SKE with rate $(1 - o(1))$, the ciphertext length is $n/(1 - o(1)) + \mathsf{poly}(\lambda)$, yielding an ideal rate of $(1 - o(1))$ for the construction. However, these KEMs require long public keys, as opposed to the short public keys in Construction 6.

For security, we prove that if the underlying SKE has $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security, then Construction 7 has $\mathsf{MULT\text{-}SIM\text{-}CPA}$ security as well.

Theorem 6.4. *If* KEM *is a non-commiting KEM, and* SKE *is a* $(\eta, 1, q, S)$-MULT-SIM-CPA *secure SKE with message space* $\{0,1\}^n$*, then Construction 7 is* $(\eta, 1, q, S - \eta \cdot \mathsf{poly}(\lambda, n))$-MULT-SIM-CPA *secure.*

Proof. We prove this through a reduction. We show that if there is an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the $(\eta, 1, q, S - \eta \cdot \mathsf{poly}(\lambda, n))$-MULT-SIM-CPA security of Π, then we can construct an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that breaks the $(\eta, 1, q, S)$-MULT-SIM-CPA security of SKE. $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ works as follows:

- $\mathcal{A}'_1$: Use the security parameters λ, S to determine the key length $\mathcal{L}_k$ for the underlying SKE[10]. For each $i \in [\eta]$, obtain $(\mathsf{pk}_i, c_{0,i}, \mathsf{KEM.st}_i) \leftarrow \mathsf{KEM.Sim}_1(1^\lambda, 1^{\mathcal{L}_k})$. Send $\{\mathsf{pk}_i\}_i$ to $\mathcal{A}_1$ to get a list of message queries $\{m_i\}_i$. Then, forward the list $\{m_i\}_i$ to the challenger / simulator and receive a list of ciphertexts $\{\mathsf{ct}'_i\}_i$. Construct $\mathsf{ct}_i = (c_{0,i}, \mathsf{ct}'_i)$, and send all $\{\mathsf{ct}_i\}_i$ to $\mathcal{A}_1$ to receive a state st. Output the state $\mathsf{st}' = (\mathsf{st}, \{\mathsf{KEM.st}_i\}_i)$. The size of the state is $|\mathsf{st}| + \eta \cdot \mathsf{poly}(\lambda, \mathcal{L}_k) \leq S - \eta \cdot \mathsf{poly}(\lambda, n) + \eta \cdot \mathsf{poly}(\lambda, n) = S$.
- $\mathcal{A}'_2$: First receive $\mathsf{st}' = (\mathsf{st}, \{\mathsf{KEM.st}_i\}_i), \{m_i\}_i, \{k_i\}_i$ from the challenger / simulator. For each $i \in [\eta]$, run $(r_i^{\mathsf{KeyGen}}, r_i^{\mathsf{Encap}}) \leftarrow \mathsf{KEM.Sim}_2(\mathsf{KEM.st}_i, k_i)$, and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda, 1^{\mathcal{L}_k}; r_i^{\mathsf{KeyGen}})$. Notice that pk_i matches the pk_i produced previously by $\mathcal{A}'_1$ due to the non-committing property of the KEM. Send $\mathsf{st}, \{m_i\}_i, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$ to $\mathcal{A}_2$ and receive a bit b. Output b.

Notice that $\mathcal{A}'$ perfectly simulates the view for $\mathcal{A}$. If $\mathcal{A}$ says it is in the real mode interacting with the challenger, this means the ciphertexts ct_i's are faithful encryptions of the message queries m_i's, i.e. $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i) = \mathsf{SKE.Dec}(\mathsf{KEM.Decap}(\mathsf{sk}_i, c_{0,i}), \mathsf{ct}'_i) = m_i$ for all $i \in [\eta]$. This implies that $\mathsf{SKE.Dec}(k_i, \mathsf{ct}'_i) = m_i$, and hence $\mathcal{A}'$ is also in the real mode. The converse also holds true. Therefore, construction 7 is $(\eta, 1, q, S - \eta \cdot \mathsf{poly}(\lambda, n))$-MULT-SIM-CPA secure. □

Upgrading to Multiple Ciphertexts Per User. Next we show how to upgrade Construction 7 to have multi-ciphertext-per-user security. All we need is to upgrade the KEM to be ℓ-*strongly non-committing*, defined as below.

Definition 6.4 (ℓ-Strongly Non-Committing)**.** *A key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *is said to be* ℓ-strongly non-committing *if there exists a pair of simulator algorithm* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that* $\mathsf{Sim}_1(1^\lambda, 1^{\mathcal{L}_k})$ *outputs a simulated public key* pk'*, a set of simulated headers* $\mathcal{C}' = \{c'_1, c'_2, \ldots, c'_\ell\}$ *and a state* st *with* $|\mathsf{st}| = \mathsf{poly}(\lambda, \mathcal{L}_k, \ell)$*, and for any given set of target keys* $\mathcal{K}' = \{k'_1, k'_2, \ldots, k'_\ell\}$ *where* $k'_i \in \{0,1\}^{\mathcal{L}_k}$ *for all* $i \in [\ell]$*,* $\mathsf{Sim}_2(\mathsf{st}, \mathcal{K}')$ *outputs a set of random coin pairs* $\{(r_i^{\mathsf{KeyGen}}, r_i^{\mathsf{Encap}})\}_{i \in [\ell]}$*. We require that if we run the key generation and encapsulation algorithm using the* i*-th pair of these random coins, we will get the desired* pk', c'_i*, and* k'_i*, i.e. for all* $i \in [\ell]$*:*

$$\Pr\left[\begin{matrix}\mathsf{pk}' = \mathsf{pk} \\ k'_i = k \\ c'_i = c\end{matrix} : \begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k}; r_i^{\mathsf{KeyGen}}) \\ (k, c) \leftarrow \mathsf{Encap}(\mathsf{pk}; r_i^{\mathsf{Encap}})\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Kindly notice that by the correctness property, $\mathsf{Decap}(\mathsf{sk}, c'_i) \to k'_i$.

[10]For the ease of syntax, we imagine the security parameters to be part of the public parameters always accessible to the adversary.

We show how to construct ℓ-strongly non-committing KEMs by composing plain non-committing KEMs below.

To get multi-ciphertext security, we simply plug in the ℓ-strongly non-committing KEM in place of the plain non-committing KEM in construction 7. The resulting construction has $(\eta/\ell, \ell, q, S - \eta \cdot \mathsf{poly}(\lambda, n, \ell))$-MULT-SIM-CPA security. The security proof follows analogous from that of Theorem 6.4.

Instantiating ℓ-Strongly Non-Committing KEM. We give a simple construction of ℓ-strongly non-committing KEM by composing 2ℓ plain non-committing KEMs.

Construction 8. *Let $\mathsf{KEM}_1, \mathsf{KEM}_2, \ldots, \mathsf{KEM}_n$ be $n = 2\ell$ instances of non-committing KEMs, we construct an ℓ-strongly non-committing KEM $\Pi = (\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ as follows:*

- $\mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k})$*: For each $i \in [n]$, run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KEM}_i.\mathsf{KeyGen}(1^\lambda, 1^{\mathcal{L}_k})$. Publish $\mathsf{pk} = \{\mathsf{pk}_i\}_i$ and $\mathsf{sk} = \{\mathsf{sk}_i\}_i$.*
- $\mathsf{Encap}(\mathsf{pk})$*: First sample a random subset $I \subseteq [n]$. Then for all $i \in I$, get $(k_i, c_i) \leftarrow \mathsf{KEM}_i.\mathsf{Encap}(\mathsf{pk}_i)$. Output $k = \bigoplus_{i \in I} k_i$, and $c = (I, \{c_i\}_i)$.*
- $\mathsf{Decap}(\mathsf{sk}, c)$*: First parse $c = (I, \{c_i\}_i)$. Then for all $i \in I$, get $k_i \leftarrow \mathsf{KEM}_i.\mathsf{Decap}(\mathsf{sk}_i, c_i)$. Output $k = \bigoplus_{i \in I} k_i$.*

Correctness is trivial given the correctness of the underlying KEMs. The public key, private key and header sizes all blow up by a factor of n.

Lemma 6.5. *If $\mathsf{KEM}_1, \mathsf{KEM}_2, \ldots, \mathsf{KEM}_n$ are non-committing KEMs, then construction 8 is ℓ-strongly non-committing.*

Proof. We show how to construct the pair of simulator algorithms $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ for Π:

- $\mathsf{Sim}_1(1^\lambda, 1^{\mathcal{L}_k})$: For all $i \in [n]$, get $(\mathsf{pk}'_i, c'_i, \mathsf{st}_i) \leftarrow \mathsf{KEM}_i.\mathsf{Sim}_1(1^\lambda, 1^{\mathcal{L}_k})$. For all $j \in [\ell]$, sample a random subset $I_j \subseteq [n]$, and let $\hat{c}_j = (I_j, \{c'_i\}_{i \in I_j})$. Output $\mathsf{pk}' = \{\mathsf{pk}'_i\}_{i \in [n]}, \mathcal{C}' = \{\hat{c}_j\}_{j \in [\ell]}$, and $\mathsf{st} = (\{I_j\}_{j \in [\ell]}, \{\mathsf{st}_i\}_{i \in [n]})$.
- $\mathsf{Sim}_2(\mathsf{st}, \mathcal{K}' = \{k'_j\}_{j \in [\ell]})$: First, parse $\{I_j\}_{j \in [\ell]}$ as a $\ell \times n$ bit matrix $\mathbf{M}$. $\mathbf{M}_{\alpha,\beta} = 1$ if and only if $\beta \in I_\alpha$. Solve for the vector $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ where each $\mathbf{v}_i \in \{0,1\}^{\mathcal{L}_k}$ such that
$$\mathbf{M} \cdot \mathbf{v}^\top = (k'_1, k'_2, \ldots, k'_\ell)^\top.$$
Assume for now that there exists a satisfying solution for $\mathbf{v}$. Notice that this means for all $j \in [\ell]$, $k'_j = \bigoplus_{i \in I_j} \mathbf{v}_i$, i.e. $\mathbf{v}$ gives an assignment of keys for $\mathsf{KEM}_1, \mathsf{KEM}_2, \ldots, \mathsf{KEM}_n$ that satisfies the target key set $\mathcal{K}'$ for Π. Now we just to run $(r_i^{\mathsf{KeyGen}}, r_i^{\mathsf{Encap}}) \leftarrow \mathsf{KEM}_i.\mathsf{Sim}_2(\mathsf{st}_i, \mathbf{v}_j)$ for all $i \in [n]$. Output $\{(r_i^{\mathsf{KeyGen}}, r_i^{\mathsf{Encap}})\}_{i \in [n]}$.

By the non-committing property of the underlying KEMs, it is easy to see that these random coins yield the simulated public keys, headers and the target keys.

The only remaining thing to show is that $\mathbf{M} \cdot \mathbf{v}^\top = (k'_1, k'_2, \ldots, k'_\ell)^\top$ has a satisfying solution for $\mathbf{v}$. Notice that $\mathbf{v}$ has at least one solution if $\mathbf{M}$ has rank ℓ. Having rank ℓ essentially says that all the rows of $\mathbf{M}$ are linearly independent. This ensures that the

linear equation system generated by $\mathbf{M}\cdot\mathbf{v}^\top = (k'_1, k'_2, \ldots, k'_\ell)^\top$ is consistent. Notice having rank ℓ also means that $\mathbf{M}$ has at least ℓ non-zero columns. This gives us a consistent linear equation system with ℓ equations and at least ℓ variables, which is guaranteed to have a (not necessarily unique) solution. Notice that if we choose $n = 2\ell$, then $\mathbf{M}$ is an $\ell \times 2\ell$ matrix, which has full rank (rank ℓ) with overwhelming probability $1 - O(2^{-\ell})$. □

7 Incompressible Encryption in the Random Oracle Model

7.1 Rate-1 Incompressible SKE from Random Oracles

We show how to build rate-1 incompressible SKE in the random oracle model.

Construction 9. *Let λ, S be security parameters. Given $G : \{0,1\}^{\mathsf{poly}(\lambda)} \times \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}^n, H : \{0,1\}^{\mathsf{poly}(\lambda)} \times \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(\lambda)}$ two hash functions modelled as random oracles, we construct $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}^n$ as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Sample a uniformly random key $k \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Output k.*
- $\mathsf{Enc}(k, m)$*: First, choose a random $r \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$. Let $d = G(k, r) \oplus m$. Then let $c = H(k, d) \oplus r$. Output $\mathsf{ct} = (c, d)$.*
- $\mathsf{Dec}(k, \mathsf{ct} = (c, d))$*: First, Compute $r = H(k, d) \oplus c$, and then $m = G(k, r) \oplus d$.*

Correctness is easy to verify given that G and H are deterministic. The ciphertext has length $|c| + |d| = n + \mathsf{poly}(\lambda)$, which gives an ideal rate of $(1 - o(1))$. The secret key size is $\mathsf{poly}(\lambda)$, which is also optimal.

The construction has $(2^\lambda, 2^\lambda, \frac{S}{n}, S)$-MULT-SIM-CPA security. Notice that this security holds for an unbounded (exponential) number of ciphertexts per user.

Theorem 7.1. *If G, H are hash functions modelled as random oracles, then construction 9 is $(2^\lambda, 2^\lambda, \frac{S}{n}, S)$-MULT-SIM-CPA secure.*

Proof. We prove this by limiting the adversary's queries to the random oracles G and H through several steps. First, recall the challenger's behavior in the real mode experiment:

- For $i \in [2^\lambda]$, sample a uniform $k_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$.
- Receive a list of message queries $\{m_{i,j}\}_{i,j \in [2^\lambda]}$ from $\mathcal{A}_1$.
- For each $i, j \in [2^\lambda]$:
  - Sample a uniformly random $r_{i,j} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
  - Let $d_{i,j} = G(k_i, r_{i,j}) \oplus m_{i,j}$.
  - Let $c_{i,j} = H(k_i, d_{i,j}) \oplus r_{i,j}$.
  - Let $\mathsf{ct}_{i,j} = (c_{i,j}, d_{i,j})$.

- Send $\{\mathsf{ct}_{i,j}\}_{i,j}$ to $\mathcal{A}_1$ and receive a state st of size at most S.
- On input of $\mathsf{st}, \{m_{i,j}\}_{i,j}, \{k_i\}_i$, $\mathcal{A}_2$ outputs a bit 1/0.

Now we limit the adversary's queries to G and H.

1. Notice that $\mathcal{A}_1$ can never query the random oracles G or H using some k_i, since the k_i's remain hidden from $\mathcal{A}_1$. The probability of $\mathcal{A}_1$ guessing a k_i correctly is exponentially small.
2. $\mathcal{A}_2$ can only query $G(k_i, r_{i,j})$ after querying $H(k_i, d_{i,j})$. This is because if $\mathcal{A}_2$ has not queried $H(k_i, d_{i,j})$ yet, then $r_{i,j} = H(k_i, d_{i,j}) \oplus c_{i,j}$ is just a uniformly random λ-bit string to the adversary, and the probability of guessing it correctly is exponentially small.
3. $\mathcal{A}_2$ can make at most S/n queries to $H(k_i, d_{i,j})$ with different (i, j) pairs. Notice that the probability of guessing a $d_{i,j}$ correctly is exponentially small. So in order to successfully query $H(k_i, d_{i,j})$, $d_{i,j}$ must be stored in st. But each $d_{i,j}$ is n bits, and $|\mathsf{st}| \leq S$, so $\mathcal{A}_2$ can recover at most S/n such $d_{i,j}$'s and hence make at most S/n valid queries to H.

With these limitations in mind, the simulator for the ideal mode experiment works as follow:

- Receive a list of message queries $\{m_{i,j}\}_{i,j\in[2^\lambda]}$ from $\mathcal{A}_1$. Discard without looking at it.
- For each $i, j \in [2^\lambda]$:
  - Sample a uniformly random $c_{i,j} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
  - Sample a uniformly random $d_{i,j} \leftarrow \{0,1\}^n$.
  - Let $\mathsf{ct}_{i,j} = (c_{i,j}, d_{i,j})$.
- Send $\{\mathsf{ct}_{i,j}\}_{i,j}$ to $\mathcal{A}_1$ and receive a state st of size at most S.
- Sample uniformly random keys $k_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$
- $\mathcal{A}_2$ receives $\mathsf{st}, \{m_{i,j}\}_{i,j}, \{k_i\}_i$.
- Whenever $\mathcal{A}_2$ queries the random oracle on $H(k_i, d_{i,j})$, submit a query for message $m_{i,j}$. There will be at most S/n such queries. Program $H(k_i, d_{i,j})$ to output a uniformly random string $r'_{i,j} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$, and program $G(k_i, r'_{i,j} \oplus c_{i,j}) = m_{i,j} \oplus d_{i,j}$.
- $\mathcal{A}_2$ outputs a bit 1/0.

Notice that this simulator queries a subset of messages that has size at most S/n. It is easy to see that a PPT adversary cannot distinguish between the challenger and the simulator. For a pair of index (i, j) that $\mathcal{A}$ has queried $H(k_i, d_{i,j})$, we have $\mathsf{ct}_{i,j} = (c_{i,j}, d_{i,j}) = (H(k_i, d_{i,j}) \oplus r_{i,j}, G(k_i, r_{i,j}) \oplus m_{i,j})$ is just a faithful encryption of $m_{i,j}$, which is the same thing the challenger in the real mode would output. For a pair of index (i, j)

that $\mathcal{A}$ has *not* queried $H(k_i, d_{i,j})$, then by limitation 2 above, $\mathcal{A}$ has also not queried $G(k_i, r_{i,j})$. Here, $m_{i,j}$ is essentially masked with a random string, so the adversary cannot tell between an encryption of $m_{i,j}$ and a random ciphertext, i.e. the challenger output and the simulator output. □

7.2 Rate-1 Incompressible PKE from Random Oracles

We then show how to construct rate-1 incompressible PKE from random oracles, plain PKE, and rate-1 incompressible SKE. The construction is essentially a hybrid mode PKE with random oracles plugged in. Notice that this construction can be viewed as a generalization of Construction 5 in Section 7.1 of [BDD22].

Construction 10. *Let λ, S be security parameters. Given $\mathsf{PKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ a plain PKE scheme with many-time CPA security, $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a rate-1 incompressible SKE with $(2^\lambda, 1, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ security, message space $\{0,1\}^n$ and key space $\{0,1\}^{\mathcal{L}_k}$, and $H : \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}^{\mathcal{L}_k}$ a hash function modelled as a random oracle, we construct $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}^n$ as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Run* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE}'.\mathsf{Gen}'(1^\lambda)$*. Output* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk}, m)$*: Sample a short random* $r \in \{0,1\}^{\mathsf{poly}(\lambda)}$*. Compute* $c \leftarrow \mathsf{PKE}'.\mathsf{Enc}'(\mathsf{pk}, r)$ *and* $d \leftarrow \mathsf{SKE}.\mathsf{Enc}(H(r), m)$*. Output* $\mathsf{ct} = (c, d)$*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (c, d))$*: Get* $r \leftarrow \mathsf{PKE}'.\mathsf{Dec}'(\mathsf{sk}, c)$*, and output* $m \leftarrow \mathsf{SKE}.\mathsf{Dec}(H(r), d)$*.*

It is easy to see that given the correctness of PKE' and SKE and that H is deterministic, this construction is correct. The ciphertexts have length $|c| + |d| = n + \mathsf{poly}(\lambda)$, yielding an ideal rate of $(1 - o(1))$. The public key and the private key both have size $\mathsf{poly}(\lambda)$, which is optimal.

We show that the construction has $(2^\lambda, 2^\lambda, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ security.

Theorem 7.2. *If PKE' has many-time CPA security, SKE has $(2^\lambda, 1, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ security, and H is a hash function modelled as a random oracle, then construction 10 is $(2^\lambda, 2^\lambda, q, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ secure.*

Proof. We show how to construct the simulator for the ideal mode experiment by using the simulator for the underlying incompressible SKE.

- For $i \in [2^\lambda]$, sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{PKE}'.\mathsf{Gen}'(1^\lambda)$.
- Receive a list of message queries $\{m_{i,j}\}_{i,j \in [2^\lambda]}$ from $\mathcal{A}_1$. Discard without looking at it.
- Run the simulation for the incompressible SKE to obtain a list of ciphertexts $\{d_{i,j}\}_{i,j}$ for $i, j \in [2^\lambda]$.
- For each $i, j \in [2^\lambda]$:
  - Sample a uniformly random $r_{i,j} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
  - Let $c_{i,j} \leftarrow \mathsf{PKE}'.\mathsf{Enc}'(\mathsf{pk}_i, r_{i,j})$.

– Let $\mathsf{ct}_{i,j} = (c_{i,j}, d_{i,j})$.

- Send $\{\mathsf{ct}_{i,j}\}_{i,j}$ to $\mathcal{A}_1$ and receive a state st of size at most S. Forward the state st to the simulator for the incompressible SKE.

- Run the incompressible SKE simulator to obtain the simulated symmetric keys $\{k_{i,j}\}_{i,j}$, and reprogram the random oracle H to output $H(r_{i,j}) = k_{i,j}$. In the process, if the SKE simulator queries for message $m_{i,j}$, also query for $m_{i,j}$. Notice that there will be at most q such queries.

- $\mathcal{A}_2$ receives $\mathsf{st}, \{m_{i,j}\}_{i,j}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_i$ and outputs a bit 1/0.

The security of the underlying PKE' ensures that the reprogramming of H is undetectable to the adversary. This is because for $\mathcal{A}_1$, $r_{i,j}$'s are encrypted under PKE', and the PKE' private keys remain hidden to $\mathcal{A}_1$. Therefore, $\mathcal{A}_1$ is not able to query H on any of the $r_{i,j}$'s before the reprogramming happens, and hence is not able to detect it.

By the property of the incompressible SKE simulator, the rest is easy to see that the simulator constructed above is indistinguishable from a real mode challenger. □

Remark 7.1. *By using construction 9 as the incompressible SKE schme in construction 10, we would get a rate-*1*, random oracle based, incompressible PKE scheme for message space* $\{0,1\}^n$ *that has* $(2^\lambda, 2^\lambda, \frac{S}{n}, S)$-$\mathsf{MULT\text{-}SIM\text{-}CPA}$ *security.*

References

[ACP+19] Martin Albrecht, Carlos Cid, Kenneth G Paterson, Cen Jung Tjhai, and Martin Tomlinson. Nts-kem. *NIST submissions*, 2:4–13, 2019.

[AOR+20] Divesh Aggarwal, Maciej Obremski, João L. Ribeiro, Luisa Siniscalchi, and Ivan Visconti. How to extract useful randomness from unreliable sources. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 343–372. Springer, Heidelberg, May 2020.

[BBB+17] Magali Bardet, Elise Barelli, Olivier Blazy, Rodolfo Canto Torres, Alain Couvreur, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich. Big quake binary goppa quasi–cyclic key encapsulation. *NIST submissions*, 2017.

[BCL+17] Daniel J Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, et al. Classic mceliece: conservative code-based cryptography. *NIST submissions*, 2017.

[BDD22] Pedro Branco, Nico Döttling, and Jesko Dujmovic. Rate-1 incompressible encryption from standard assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 33–69. Springer, Heidelberg, November 2022.

[BJ] Thomas Barnett Jr. The zettabyte era officially begins (how much is that?). https://blogs.cisco.com/sp/the-zettabyte-era-officially-begins-how-much-is-that.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[Dep] Statista Research Department. Data center storage capacity worldwide from 2016 to 2021, by segment. https://www.statista.com/statistics/638593/worldwide-data-center-storage-capacity-cloud-vs-traditional/.

[DFR+07] Ivan Damgård, Serge Fehr, Renato Renner, Louis Salvail, and Christian Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 360–378. Springer, Heidelberg, August 2007.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Heidelberg, February 2001.

[DKZ18] Stefan Dziembowski, Tomasz Kazana, and Maciej Zdanowicz. Quasi chain rule for min-entropy. *Information Processing Letters*, 134:62–66, 2018.

[DQW22] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Authentication in the bounded storage model. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 737–766. Springer, Heidelberg, May / June 2022.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.

[Dzi06] Stefan Dziembowski. On forward-secure storage (extended abstract). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 251–270. Springer, Heidelberg, August 2006.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, 1989.

[Gün90] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37. Springer, Heidelberg, April 1990.

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 700–730. Springer, Heidelberg, May / June 2022.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Heidelberg, May 2005.

[MW20] Tal Moran and Daniel Wichs. Incompressible encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 494–523. Springer, Heidelberg, August 2020.

[Nis90] Noam Nisan. Psuedorandom generators for space-bounded computation. In *22nd ACM STOC*, pages 204–212. ACM Press, May 1990.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

[V+12] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.