Incompressible Cryptography

Jiaxin Guan[*]
Princeton University

Daniel Wichs[†]
Northeastern University and NTT Research

Mark Zhandry[‡]
Princeton University and NTT Research

December 21, 2021

Abstract

Incompressible encryption allows us to make the ciphertext size flexibly large and ensures that an adversary learns nothing about the encrypted data, even if the decryption key later leaks, unless she stores essentially the entire ciphertext. *Incompressible signatures* can be made arbitrarily large and ensure that an adversary cannot produce a signature on *any* message, even one she has seen signed before, unless she stores one of the signatures essentially in its entirety.

In this work, we give simple constructions of both incompressible public-key encryption and signatures under minimal assumptions. Furthermore, large incompressible ciphertexts (resp. signatures) can be decrypted (resp. verified) in a streaming manner with low storage. In particular, these notions strengthen the related concepts of *disappearing encryption and signatures*, recently introduced by Guan and Zhandry (TCC 2021), whose previous constructions relied on sophisticated techniques and strong, non-standard assumptions. We extend our constructions to achieve an optimal "rate", meaning the large ciphertexts (resp. signatures) can contain almost equally large messages, at the cost of stronger assumptions.

[*]`E-mail:jiaxin@guan.io`. Part of this research was conducted while this author was a research intern at NTT Research, Inc.

[†]`E-mail:danwichs@gmail.com.`

[‡]`E-mail:mzhandry@gmail.com.`

1 Introduction

Security breaches are ubiquitous. Therefore, it is natural to wonder: will encrypted messages remain secure, even if the secret decryption key is later leaked? Forward secrecy deals exactly with this problem, but requires either multi-round protocols or key updates, both of which may be undesirable in many scenarios. And in the usual time-bounded adversary model, unfortunately, such limitations are inherent: an adversary can simply store the ciphertext and wait for the secret key to leak, at which point it can easily decrypt.

Incompressible encryption. In this work we ask: can we force a would-be "save-it-for-later adversary" to actually store the ciphertext in its entirety, for the entire length of time it is waiting for the secret key to leak? At a minimum such storage may be inconvenient, and for very large files or long time frames, it may be prohibitively costly. Even for short messages, one may artificially increase the ciphertext size, hopefully forcing the adversary to use much more storage than message length. We may therefore hope that such an *incompressible encryption scheme* maintains the privacy of messages even if the secret key is later revealed.

Remark 1. *For an illustrative example, an individual with a gigabit internet connection can transmit ∼10TB per day, potentially much more than their own storage. Of course many entities will have 10TB or even vastly more, but an incompressible scheme would force them to devote 10TB to storing a particular ciphertext for potentially years until the key is revealed. Across millions or billions of people, even powerful adversaries like state actors would only be able to devote such storage to a small fraction of victims.*

Unfortunately, traditional public key encryption schemes are not incompressible; an adversary may be able to store only a short digest of the ciphertext and still obtain non-trivial information about the plaintext once the secret key is leaked. For example, for efficiency reasons, hybrid encryption is typically used in the public key setting, where the encryption of a message m may look like:

$$(\; \mathsf{Enc}(\mathsf{pk}, s) \; , \; G(s) \oplus m \;) \; .$$

Here, s is a short seed, and G is a pseudorandom generator used to stretch the random seed into a pseudorandom pad for the message m. A save-it-for-later adversary need not store the entire ciphertext; instead, they can store just $\mathsf{Enc}(\mathsf{pk}, s)$ as well as, say, the first few bits of $G(s) \oplus m$. Once the secret key is revealed, they can learn s and then recover the first few bits of m. This may already be enough to compromise the secrecy of m. Such an attack is especially problematic if we wanted to artificially increase the ciphertext size by simply padding the message and appending dummy bits, since then the first few bits of m would contain the entire secret plaintext.

The compressibility issue is not limited to the scheme above: we could replace $G(s) \oplus m$ with a different efficient symmetric key encryption scheme such as CBC-mode encryption, and essentially the same attack would work. The same goes for bit encryption as well.

Incompressible public key encryption instead requires that if the adversary stores anything much smaller than the ciphertext, the adversary learns absolutely nothing about the message, even if the secret key later leaks.

Remark 2. *We note that plain public key encryption does have* some *incompressibility properties. In particular, it is impossible, in a plain public key encryption scheme, for the adversary to significantly compress the ciphertext and later be able to reconstruct the original ciphertext. However, this guarantee implies nothing about the privacy of the underlying message should the key leak.*

Incompressible Signatures. A canonical application of signatures is to prevent man-in-the-middle attacks: by authenticating each message with a signature, one is assured that the messages were not tampered with. However, a man-in-the-middle can always *delay* sending an authenticated message, by storing it for later. The only way to block such attacks in the usual time-bounded adversary model is to use multi-round protocols, rely on synchronized clocks and timeouts, or have the recipients keep state, all of which may be undesirable. We therefore also consider the case of incompressible *signatures*, which force such a delaying adversary to actually store the entire signature for the duration of the delay.

In slightly more detail, in the case of plain signatures, a forgery is a signature on any *new* message, one the adversary did not previously see signed. The reason only new signed messages are considered forgeries is because an adversary can simply store a valid signature it sees, and later reproduce it. An *incompressible* signature, essentially, requires that an adversary who produces a valid signature on an existing message must have actually stored a string almost as large as the signature. By making the signatures long, we may hope to make it prohibitively costly to maintain such storage. As in the case of encryption, existing signature schemes do not appear to offer incompressible security; indeed, it is usually desired that signatures are very short.

Feature: Low-storage for streaming honest users. Given that communication will be inconveniently large for the adversary to store, a desirable feature of incompressible ciphertexts and signatures is that they can be sent and received with low storage requirements for the honest users. In such a setting, the honest users would never store the entire ciphertext or signature, but instead generate, send, and process the communication bit-by-bit in a streaming fashion.

Feature: High rate. With incompressible ciphertexts and signatures, communication is set to be deliberately large. If the messages themselves are also large, it may be costly to further blow up the communication in order to achieve incompressibility. Therefore, a desirable feature is to have the rate—the ratio of the maximum message length to the communication size—be as close to 1 as possible. In this way, for very large messages, there is little communication overhead to make the communication incompressible.

1.1 Related work: Disappearing Cryptography

Very recently, Guan and Zhandry [GZ21] define and construct what they call *disappearing* public key encryption and digital signatures. Their notions are very similar to ours, except with an important distinction: they assume both honest and malicious parties operate as space-bounded streaming algorithms throughout their operation. Honest users are assumed to have a somewhat lower storage bound than the adversary's.

In terms of the functionality requirement for honest users, their model corresponds to the low-storage streaming variant of incompressible cryptography. However, in terms of the security requirement, disappearing cryptography is somewhat weaker, since it restricts the adversary to also be space-bounded throughout its entire operation, and observe the ciphertexts/signatures produced by the cryptosystem in a streaming manner. On the other hand, incompressible cryptography allows the adversary to observe each ciphertext/signature in its entirety and compute on it using an unrestricted amount of local memory, but then store some small compressed version of it afterwards. Some disappearing schemes may be insecure in the incompressible threat model: for example, one of the disappearing ciphertext schemes from [GZ21] could potentially even be based on *symmetric key* cryptography, despite being a public key primitive.[1] Yet public key incompressible ciphertexts easily imply public key encryption, which is believed to be stronger than symmetric key cryptography [IR90].

In summary, incompressible cryptography with low-storage streaming is also disappearing, but the reverse direction does not hold.

Guan and Zhandry explain several interesting applications of disappearing ciphertexts and signatures, including deniable encryption [CDNO97]. Here, one imagines that the secret key holder is coerced into revealing their key. In order to protect the contents of an encrypted message, traditional deniable encryption allows the key holder to generate a fake key that causes the ciphertext to decrypt to any desired value. Unfortunately, such receiver-deniable encryption is impossible in the standard model [BNNO11]. Disappearing ciphertexts offer a solution, since the contents are protected without even faking the key, as the space-bounded attacker is unable to store the ciphertext.

However, in addition to achieving a weaker security model than incompressible cryptography, the schemes of [GZ21] have some significant limitations:

- Their schemes are built from a novel object called *online obfuscation*, a very strong proposed form of program obfuscation in the bounded storage setting. While [GZ21] gives plausible candidate constructions, the constructions are complex and it is unclear how to prove security. It is even plausible that the notion of online obfuscation is *impossible.*

- One of their candidates requires, at a minimum, standard-model virtual grey box (VGB) obfuscation [BCKP14], which is stronger even than indistinguishability obfuscation [BGI$^+$01], already one of the strongest known assumptions in cryptography. And even assuming VGB, the security remains unproven. Their other candidate could plausibly be information-theoretic (but again, currently not proven), but is limited to a quadratic separation between the ciphertext/signature size and the honest users' storage.

- Their encryption and signature schemes involve ciphertexts/signatures that are significantly larger than the messages, and so their schemes are low "rate" when the messages are large.

[1] It's not hard to see that one-way functions, and therefore symmetric key cryptography, are implied by disappearing ciphertexts, since the secret key can be information-theoretically recovered from the public key.

1.2 Our Results

We give new positive results for incompressible cryptography:

- Under the minimal assumption of standard-model public key encryption, we construct a simple incompressible public key encryption scheme. The scheme supports streaming with constant storage, independent of the ciphertext size. As a special case, we achieve provably secure disappearing ciphertexts with optimal honest-user storage and under mild assumptions, significantly improving on [GZ21]. The ciphertext size is $|c| = |S| + |m| \times \mathsf{poly}(\lambda)$, where $|S|$ is the adversary's storage, $|m|$ the message size, and λ the security parameter.

- Under the minimal assumption of one-way functions, we construct incompressible signatures. Our scheme supports streaming with constant storage, independent of the signature size. Thus we also achieve provably secure disappearing signatures under minimal assumptions, again significantly improving on [GZ21]. The total communication (message length plus signature size) is $|S| + |m| + \mathsf{poly}(\lambda)$.

- Under standard-model indistinguishability obfuscation (iO), we construct "rate 1" incompressible public-key encryption, where $|c| = |S| + \mathsf{poly}(\lambda)$ and the message length can be as large as roughly $|S|$. In particular, for very large messages, the ciphertext size is roughly the same as the message size.

  The public keys of our scheme are small, but the secret keys in this scheme are at least as large as the message, which we explain is potentially inherent amongst provably-secure high-rate schemes.

  Along the way, we give the first rate-1 construction of functional encryption for circuits, where $|c| = |m| + \mathsf{poly}(\lambda)$.

- We consider a notion of "rate-1" incompressible signatures, where the total communication is only $|S| + \mathsf{poly}(\lambda)$, and the message can be as large as roughly $|S|$. Note that the signature by itself must have size at least $|S|$ for incompressibility (since m may be compressible), and so if we separately send the message and signature, the total communication would be at least $|S| + |m|$, which is not rate 1. Instead, we just send a signature and require the message to be efficiently extractible from the signature.

  We show that rate-1 incompressible signatures are *equivalent* to incompressible encodings, defined by Moran and Wichs [MW20]. By relying on the positive results of [MW20], we obtain such signatures under either the Decisional Composite Residuosity (DCR) or Learning With Errors (LWE) assumption, in either the CRS or random oracle model. The random oracle version supports low-space streaming, as does the CRS model if we assume the (large) CRS is streamed. On the other hand, by relying on the negative results of [MW20], we conclude that a provably secure rate-1 construction in the standard model is unlikely.

1.3 Other Related Work

Bounded Storage Model. The work by Guan and Zhandry [GZ21] is set in the Bounded Storage Model (BSM) due to Maurer [Mau92], which leverages bounds on the adversary's storage to enable applications. Most of the related work in the BSM is about achieving unconditionally secure schemes for the types of scenarios for which we already have computationally secure schemes in the standard model (CPA encryption [CM97, AR99, Lu02, Raz17, GZ19], Key Agreement [CM97, GZ19, DQW21], Oblivious Transfer [CCM98, Din01, DHRS04, GZ19, DQW21], etc.). In contrast, time-stamping in the bounded storage model [MST04] is perhaps the first application of the bounded storage model beyond achieving information-theoretic security by assuming additional computational assumptions. Similarly, our work, as well as the work by Guan and Zhandry [GZ21], considers scenarios for which computationally secure schemes in the standard model are impossible and which only make sense in the BSM (public-key encryption where the adversary gets the secret key after seeing the ciphertext, signature schemes where the adversary cannot sign messages whose signatures she has previously observed). Our results necessarily rely on computational assumptions.

Big-Key Cryptography in the Bounded Retrieval Model. The study of big-key cryptography in the Bounded Retrieval Model (BRM) has evolved through a series of works [Dzi06, DLW06, CDD$^+$07, ADW09, ADN$^+$10, BKR16]. The high-level difference is that in the BRM, the secret keys are made large to prevent exfiltration, while the communication (e.g., ciphertexts, signatures) are kept small. In incompressible cryptography, we do the reverse and want to make the communication (e.g., ciphertexts, signatures) large to prevent an adversary from being able to remember it in its entirety, while the secret key is ideally small. On a technical level, while there are some high-level similarities such as relying on a combination of computational and information-theoretic techniques, the concrete schemes are quite different.

1.4 Technical Overview

Incompressible Encryption. We first consider incompressible public key encryption. The syntax is identical to that of standard-model encryption, but the security game is different:

1. The challenger first gives the adversary the public key.
2. The adversary then produces two messages m_0, m_1.
3. The challenger encrypts one of the two messages, as the ciphertext c.
4. Now the adversary must produce a small state s of size somewhat smaller than c.
5. The challenger then reveals the secret key.
6. The adversary, given only the small state s but also the secret key, now makes a guess for which message was encrypted.

Note that, except for the size of the state s being bounded between Steps 4 and 6, the size of the adversary's storage is unbounded. It is also easy to see that this definition implies standard semantic security of public-key encryption.

Remark 3. *Note that this security definition is quite similar to that of disappearing public key encryption by Guan and Zhandry [GZ21] with two distinctions. Firstly, in the disappearing encryption security experiment, there is no Step 4 as above. Instead, the adversary is bounded by some space* throughout the entire experiment. *Additionally, functionality wise, disappearing encryption requires the protocol to be executable by honest parties with some space bound lower than the adversary's storage. In our setting, we do not consider this to be an inherent requirement, but rather a desirable feature that some of our schemes satisfy. As we will see in Remark 4, this feature is incompatible with rate-1 schemes, and hence we will drop it in that setting.*

Our Solution. We give a construction of incompressible encryption in Section 3, under the minimal assumption of generic public key encryption.

We describe our solution using *functional* encryption (FE), which is a form of public key encryption where the secret key holder can give out function secret keys for functions f; a function secret key allows for learning $f(m)$ but nothing else about the message. For our application, we only need a very special case of single-key functional encryption, which we instantiate with a simple and potentially practical construction from generic public key encryption scheme. Our incompressible encryption scheme works as follows:

- The public key is just the public key for the underlying FE scheme. The secret key is a function secret key for the function f_v defined as

$$f_v(s, b) = \begin{cases} s & \text{if } b = 0 \\ s \oplus v & \text{if } b = 1 \end{cases}$$

  where the value v is chosen uniformly at random and hard-coded into f_v. Here, s, v are reasonably short strings, whose length will be discussed shortly.

- To encrypt m, choose a random s, and compute $c \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (s, 0))$ as an encryption of $(s, 0)$ under the FE scheme. Then choose a large random string R. Interpret s as the pair (s', t), where t is a string of length equal to the message length, and s' is the seed for a strong extractor. Then compute $z = \mathsf{Extract}(R; s') \oplus t \oplus m$. The final ciphertext is (c, R, z).

- To decrypt, use the FE secret key to recover $s = (s', t)$ from c. Then recover $m = z \oplus \mathsf{Extract}(R; s') \oplus t$.

We can generate and transmit the string R in a streaming fashion. We can then use an online extractor [Vad03] so that $\mathsf{Extract}(R; s')$ can be computed without having to store R in its entirety. Note that R is the only "big" component of the ciphertext, so encryption and decryption therefore require small space.

We prove security through a hybrid argument. First, we use FE security to switch to c being generated as $c \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (s \oplus v, 1))$. Since this c decrypts equivalently under the secret key, this change is indistinguishable.

We then observe that the string $u = s \oplus v$ being encrypted under the FE scheme, as well as the string z included in the final ciphertext, are both just uniformly random strings. We can therefore delay the generation of the secret key and v, until the very end of the experiment. Now we can think of the adversary's state (as well as some other small values needed to complete the simulation) as a leakage on the large random string R. Since the adversary's storage is required to be sufficiently small compared to R, R still has min-entropy conditioned on this leakage. This means we can invoke the randomness guarantee of the randomness extractor to replace $\mathsf{Extract}(R; s')$ with a uniform random string. But at this point, m is one-time-padded with a uniform string, and therefore information-theoretically hidden.

We explain how to instantiate the functional encryption scheme. Since the adversary only ever sees a single secret key, we can build such a functional encryption scheme generically from public key encryption, using garbled circuit techniques [GVW12]. On the other hand, our functional encryption scheme only needs to support an extremely simple linear function. We show a very simple and potentially practical solution from any public key encryption scheme.

Remark 4. *We note that our scheme has a less-than-ideal rate, since the ciphertext size is at least as large as the adversary's storage* plus *the length of the message. Low rates, however, are inherent to schemes supporting low-storage streaming. Indeed, the storage requirements of the honest users must be at least as large as the message, and in the high-rate case this means the honest users must be capable of storing the entire ciphertext. This remains true* even if the message itself is streamed bit-by-bit*, which can be seen as follows: by incompressibility, the decrypter cannot start outputting message bits until essentially the entire stream has been sent. Otherwise, an attacker can store a short prefix of the ciphertext, and then when it gets the secret key mimic the decrypter until it outputs the first message bit. Now, at the point right before the decrypter outputs the first message bit, the entire contents of the message must be information-theoretically contained within the remaining communication (which is short) and the decrypter's state, since the decrypter ultimately outputs the whole message. Thus the decrypter's state must be almost as large as the message.*

A rate-1 solution. We now discuss how we achieve a rate-1 scheme, using indistinguishability obfuscation. This is our most complicated construction, and we only give a brief overview here with the full construction in Section 4.

The central difficulty in achieving a rate-1 scheme is that we cannot guarantee a ciphertext with large information-theoretic entropy. Indeed, the ciphertext must be almost as small as the message, so there is little room for added entropy on top of the message. But the message itself, while large, many not have much entropy. Therefore, our approach of using randomness extraction to extract a random string from the ciphertext will not work naively.

Our solution, very roughly, is to have the large random value in the *secret key.* Using a delicate argument, we switch to a hybrid where the ciphertext is just an encryption of large randomness R, and the secret key contains the message, masked by a string extracted from R. Now we can mimic the low-rate case, arguing that given the small state produced by the adversary, R still has min-entropy. Thus, the message m is information-theoretically hidden.

The result is that we achieve an incompressible encryption scheme whose rate matches the rate of the underlying functional encryption scheme. Unlike the low-rate case, our FE scheme appears to need the full power of FE for circuits, since it will be evaluating cryptographic primitives such as PRGs and extractors. Unfortunately, all existing FE schemes for general circuits, even using iO, have poor rate. For example, if we look at the original iO scheme of [GGH$^+$13], the ciphertext contains *two* plain public key encryption encryptions of the message, *plus* a NIZK proof of consistency. The result is that the rate is certainly at most 1/3. Another construction due to [BCP14] sets the ciphertext to be an obfuscated program containing the message; since known obfuscation schemes incur a large blowup, the scheme is not rate-1.

We give a novel rate-1 FE scheme (with many key security), by building on ideas from [BZ14]. They build an object called private linear broadcast encryption (PLBE), which can be seen as a special case of FE for simple comparison functionalities. However, their approach readily generalizes to more complex functionalities. The problem with their construction is that their proof incurs a security loss proportional to the domain size. In their case, the domain is polynomial and this is not a problem. But in our case, the domain is the message space, which is exponential. One may hope to use complexity leveraging, but this would require setting the security parameter to be at least as large as the message. However, this will not give a rate-1 scheme since the ciphertext is larger than the message by an additive factor linear in the security parameter.

We therefore devise new techniques for proving security with just a polynomial loss, even for large messages, thus giving the first rate-1 FE scheme for general circuits, from iO and one-way functions. Details in Section 7.

Remark 5. *We note that the final construction of rate-1 incompressible encryption has very short public keys, but large secret keys. We therefore leave as an interesting open question devising a scheme that also has short secret keys. However, achieving such a scheme with provable security under standard assumptions appears hard. Indeed, cryptographic assumptions typically make no restrictions on the adversary's storage. The issue is that the message itself may have little entropy, and so to prove that a ciphertext is incompressible it seems the computational assumptions will be used to transition to a hybrid where the ciphertext has nearly full entropy (indeed, this is how our proof works). But this transition happens without space bounds, meaning the reduction actually is capable of decrypting the ciphertext and recovering the message once the key is revealed. Yet in this hybrid the ciphertext was "used up" in order to make it high-entropy, and it seems the only place left to embed the message is the secret key (again, this is how our proof works). If the message is large, it therefore seems the secret key must be large as well. We believe this intuition can be formalized as a black-box separation result, similarly to analogous results of [Wic13], but we leave this for future work.*

Incompressible Signatures. An incompressible signature scheme is defined by the following experiment:

1. The challenger first gives the adversary the public key.
2. The adversary makes repeated signing queries on arbitrary messages. In response, the challenger produces a signature on the message.
3. After observing many signatures, the adversary must produce a small state s of size somewhat smaller than a single signature.

4. Next, the adversary, is given the small state s, and wins if it produces a valid signature on *any* message, potentially even one used in a prior signing query.

Note that, except for the size of the state s being bounded between Steps 3 and 4, the size of the adversary's storage is unbounded.

Remark 6. *This definition is also quite similar to that of disappearing signature due to Guan and Zhandry [GZ21] except for two differences. For disappearing signatures, the security experiment does not have Step 3 as above, and instead requires the adversary to be bounded by some space* throughout the entire experiment*. Functionality wise, disappearing signature requires the scheme can be run by honest parties with a space bound somewhat lower that the adversary's storage, whereas we don't require that for incompressible signatures.*

Our Solution. We give a very simple construction of incompressible signatures in Section 5. To sign m, first choose a large uniformly random string R, and then compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},\ (R, m)\ )$, where Sign is a standard-model signature scheme. The overall signature is then (R, σ). Verification is straightforward.

Both signing and verification can be evaluated in a low-space streaming fashion, provided Sign can be evaluated as such. One can always assume this property of Sign: first hash the message using a streaming-friendly hash function such as Merkle-Damgård, and then sign the hash. Since the hash is small and computing the hash requires low-space, the overall signing algorithm is low space.

For security, consider an adversary which produces a small state s somewhat smaller than the length of R. Since R is random, it will be infeasible for the adversary to reproduce R in Step 4. Therefore, any valid signature must have an R different than any of the messages previously signed. But this then violates the standard unforgeability of Sign.

A rate-1 solution. In Section 6, we modify the above construction to get a rate-1 solution. We note that "rate" here has to be defined carefully. In the above solution, the signature size is independent of the message size, and so it seems that the signature has good rate. However, communication will involve both the signature *and* the message, and so the total length of the communication will be significantly larger than the message. We therefore want that the *total communication* length is only slightly longer than the message being signed.

On the other hand, if the message is very long, one may naturally wonder whether we can just sign the message using any standard-model signature scheme, and have the resulting communication be rate-1. However, a long message may in fact be compressible. What we want is to achieve rate-1 total communication, and incompressibility, even if the message may be compressed.

We therefore define a rate-1 incompressible signature as an incompressible signature where the signature is only slightly longer than the message, and where there is a procedure to extract the message from the signature. In this way, all that needs to be sent is the signature itself, and therefore the total communication remains roughly the same as the message.

Equivalence to incompressible encodings. We next demonstrate that incompressible signatures are equivalent to incompressible encodings [MW20]. These are public encoding schemes where the encoding encodes a message into a codeword c that is only slightly longer than the message. From c, the original message can be recovered using a decoding procedure. For security, the adversary then receives the codeword as well as the message, tries to compress the codeword into a small storage s. Then the adversary, given *s and the message*, tries to recover the exact codeword c.

A rate-1 incompressible signature (with small public keys) gives an incompressible encoding: to encode a message, simply generate a new public/secret key pair, and sign the message. The codeword c is then the public key together with the signature. Decoding and security follow readily from the message extraction procedure and security of the incompressible signature.

In the other direction, to sign a message, first incompressibly encode the message and then sign the result using a standard-model signature scheme. The final signature is the codeword together with the standard-model signature. Extraction follows from the decoding procedure. If the incompressible encoding supports low-space streaming, so does the signature scheme. For security, since the adversary cannot produce the original codeword that was signed due to the security of the incompressible encoding, they must produce some other codeword. But a valid signature would also contain a standard-model signature on this new codeword, violating the security of the signature scheme.

Moran and Wichs [MW20] instantiate incompressible encodings under either the Decisional Composite Residuosity (DCR) or Learning With Errors (LWE) assumptions, in either the CRS or random oracle models. We observe that their incompressible encodings simply break the message into blocks of length $\mathsf{poly}(\lambda)$ and encode each block separately; as such they can be easily streamed in low space, though the CRS-based scheme would need the CRS to be streamed as well. We obtain the incompressible signatures under the same assumptions in the same models, with low-space streaming.

We also note that we can have the signer generate the CRS and include it in the public key, giving a standard-model incompressible encoding scheme with large public keys. Note that such a scheme is not immediately equivalent to incompressible encodings, since the codeword contains the public key, and would therefore be too large.

On the other hand, [MW20] show that a CRS or random oracle is somewhat necessary, by giving a black box separation relative to falsifiable assumptions in the standard model. Due to our equivalence, this implies such a black box impossibility for incompressible signatures in the standard model as well.

2 Preliminaries

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

Definition 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 1 ([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 (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 7. *Any strong randomness extractor is also a strong* average *min-entropy extractor, with a constant loss in* ϵ*.*

Digital Signatures. We also generalize the syntax of a signature scheme, which will ultimately be necessary to achieve a meaningful high "rate". Instead of producing a signature that is sent along side the message, we would implicitly embed or *encode* the message into the signature. The signature is then all that is sent to the receiver, from which the message can be decoded and verified. In this way, the "signature" captures the total communication. Any standard signature scheme can readily be viewed in our generalized syntax by just calling (m, σ) the "signature." Now when we define "rate", it is simply the ratio of the message to signature size.

A public key signature scheme for message space $\{0,1\}^{L_m}$ and signature space $\{0,1\}^{L_\sigma}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ such that:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$ samples a verification key vk, and a signing key sk.
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$ takes as input the signing key sk and a message m, and computes a signature σ *that implicitly contains the message* m*.*
- $\mathsf{Ver}(\mathsf{vk}, \sigma) \to m/\bot$ takes as input the verification key vk and a signature σ, and outputs either the message m or $\bot$. Outputting m means that the signature verifies, and outputting $\bot$ means that the signature is invalid.

Definition 3 (Correctness). *For all* $\lambda \in \mathbb{N}$ *and message* $m \in \{0,1\}^{L_m}$*, let* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, then we have* $\Pr[\mathsf{Ver}(\mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, m)) = m] \geq 1 - \mathsf{negl}(\lambda)$.

We modify the security experiment slightly by asking the adversary to output a single signature σ instead of a message-signature pair, and the adversary wins the game if and only if $\mathsf{Ver}(\mathsf{vk}, \sigma) \neq \bot$ or any of the messages queried before. Notice that the original syntax of a signature scheme easily fulfills this new syntax: simply set the signature to be $\sigma = (m, \sigma')$ where σ' is the signature in the original syntax, and modify Ver to output the original message if the signature verifies. The "rate" of the signature scheme is hence defined to be simply L_m/L_σ.

Functional Encryption. For our constructions we also need single-key game-based functional encryption. 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 4 (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 5 (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. We can also consider *many-time* semi-adaptive/selective security, where the adversary is able to adaptively query for as many sk_C as it would like, provided they all occur after the challenge query.

3 Incompressible Encryption: Our Basic Construction

Here we show how to construct an incompressible public key encryption scheme with low "rate", i.e. the ratio of the message size to the ciphertext size. First, we define what it means for a public key encryption scheme to be *incompressible*.

3.1 Definition

We give the definition of incompressible encryption, which is based on the similar definition of disappearing encryption [GZ21].

For security parameters λ and S, an incompressible public key encryption scheme with message space $\{0,1\}^{L_m}$ and ciphertext space $\{0,1\}^{L_{\mathsf{ct}}}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Remark 8. *For the original disappearing PKE defined in [GZ21], it is additionally required that* Gen*,* Enc*, and* Dec *can be run in space* $N \ll L_{\mathsf{ct}}$*. Here, we will consider schemes that have both large and small space.*

The rest of the syntax of an incompressible PKE scheme is identical to that of a classical PKE scheme. The "rate" of the PKE scheme is simply L_m/L_{ct}.

For the security definition, consider the following indistinguishability experiment for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

Incompressible Encryption Security Experiment $\mathsf{Dist}^{\mathsf{IncomEnc}}_{\mathcal{A},\Pi}(\lambda)$:

1. The adversary $\mathcal{A}_1$, on input 1^λ, 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 then provided the public key pk.
5. The adversary replies with the challenge query consisting of two messages m_0 and m_1, 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 $(\mathsf{pk}, \mathsf{sk}, m_0, m_1, \mathsf{st})$ 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 6 (Incompressible Encryption Security)**.** *For security parameters* λ *and* S*, a public key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has incompressible encryption security if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*:*

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

Remark 9. *The original Disappearing Ciphertext Security [GZ21] has a very similar security notion, except that the adversary has a space bound of* S throughout the entire experiment*, and that the ciphertext is a long stream sent bit by bit. Notice that our definition of Incompressible Encryption Security is a strictly stronger security definition than Disappearing Ciphertext Security.*

3.2 Construction

Construction 1. *Given* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *a single-key selectively secure functional encryption scheme with a rate of* ρ_{FE} *and a strong average min-entropy extractor* $\mathsf{Extract} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{L_m}$*, with* $d = \mathsf{poly}(\lambda)$ *and* $n = S + \mathsf{poly}(\lambda)$ *the construction* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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* $v \in \{0,1\}^{d+L_m}$*:*

$$f_v(s' = (s,t), \mathsf{flag}) = \begin{cases} s' & \textit{if } \mathsf{flag} = 0 \\ s' \oplus v & \textit{if } \mathsf{flag} = 1 \end{cases}.$$

  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* $s' = (s,t)$ *where* $s \in \{0,1\}^d$ *is used as a seed for the extractor and* $t \in \{0,1\}^{L_m}$ *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* $z = \mathsf{Extract}(R; s) \oplus t \oplus m$*.*

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{FE.ct}, R, z))$*: First, obtain* $s' \leftarrow \mathsf{FE.Dec}(\mathsf{FE.sk}_{f_v}, \mathsf{FE.ct})$*, and then use the seed* s *to compute* $\mathsf{Extract}(R; s) \oplus z \oplus t$ *to recover* m*.*

Note that if $\mathsf{Extract}$ is an *online* extractor [Vad03], then encryption and decryption can be run in a low-space streaming fashion, by first sending $\mathsf{FE.ct}$, then streaming R, and then sending z. The rate of this construction is

$$\frac{L_m}{L_{\mathsf{ct}}} = L_m \left(\frac{d + L_m + 1}{\rho_{\mathsf{FE}}} + n + L_m \right)^{-1} = \frac{1}{(1/\rho_{\mathsf{FE}} + 1) + S/L_m} - o(1).$$

Theorem 1. *Assuming the existence of a functional encryption scheme with single-key selective security and a rate of* $1/\mathsf{poly}(\lambda)$*, and a* $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ *average min-entropy extractor, there exists an incompressible PKE with ciphertext size* $S + L_m + \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) L_m$*, public key size* $\mathsf{poly}(\lambda)$ *and secret key size* $\mathsf{poly}(\lambda)$*. Furthermore, it supports streaming decryption using* $L_m + \mathsf{poly}(\lambda)$ *bits of memory.*

3.3 Proof of Security

We organize our proof of security into a sequence of hybrids.

Sequence of Hybrids

- H_0: The original incompressible encryption security experiment $\mathsf{Dist}^{\mathsf{IncomEnc}}_{\mathcal{A},\Pi}$, where the bit b in the experiment is fixed to be 0.

- H_1: In step 5, instead of computing $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (s', 0))$, compute $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (s' \oplus v, 1))$.

- H_2: In step 2, only sample $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. In step 5, after receiving the challenge query, sample uniformly random $z \in \{0,1\}^{L_m}$, $u \in \{0,1\}^{d+L_m}$, $R \in \{0,1\}^n$ and send back $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (u, 1))$, R, and z as the ciphertext. In step 7, sample a uniformly random $s \in \{0,1\}^d$, and compute $t = \mathsf{Extract}(R; s) \oplus z \oplus m_0$, and $v = s' \oplus u$ where s' is the tuple (s, t). Use this v to compute $\mathsf{sk} = \mathsf{FE.sk}_{f_v} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_v)$.

- H_3: In step 7, sample a uniformly random $r \in \{0,1\}^{L_m}$ and compute $t = r \oplus z \oplus m_0$ instead.

- H_4: Swap the bit b in the security experiment to be 1 instead of 0.

- H_5: Switch back to the case where $t = \mathsf{Extract}(R; s) \oplus z \oplus m_1$.

- H_6: Switch back to the case where we produce sk in step 2 instead of step 5.

- H_7: Switch the FE ciphertext back to the real one $\mathsf{FE.Enc}(\mathsf{FE.mpk}, (s', 0))$. Notice here we're at the original incompressible encryption security experiment, where the bit b is fixed to be 1.

Proof of Hybrid Arguments

Lemma 2. *If the functional encryption scheme* FE *has single-key selective security, then no PPT adversary can distinguish between* H_0 *and* H_1 *(respectively* H_6 *and* H_7*) with non-negligible probability.*

Proof. Here we will prove the case for H_0 and H_1. The case for H_6 and H_7 follows analogously. This is by a simple reduction to the single-key selective security of the functional encryption scheme. If an adversary $\mathcal{A}$ is able to distinguish between H_0 and H_1, we show how to construct an adversary $\mathcal{A}'$ that breaks security of the functional encryption scheme FE. The only difference between H_0 and H_1 is that in H_0 the adversary receives an encryption of $(s', 0)$, while in H_1 the adversary receives an encryption of $(s' \oplus v, 1)$. But notice that $f_v(s', 0) = s' = f_v(s' \oplus v, 1)$, so the adversary $\mathcal{A}$ is able to distinguish between two FE ciphertexts that have the same functional output on function f_v, for which it has a secret key. This directly breaks the underlying functional encryption security. Concretely, $\mathcal{A}'$ works as follows by using $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as a subroutine:

- On input 1^λ, sample uniform values s' and v, and submit the challenge query $\mathsf{FE}.m_0 = (s', 0)$ and $\mathsf{FE}.m_1 = (s' \oplus v, 1)$ to the challenger. Receive $\mathsf{FE.mpk}$ and $\mathsf{FE.ct}$ in response.

- Send 1^λ to $\mathcal{A}_1$ and receive 1^S.

- Send $\mathsf{FE.mpk}$ to $\mathcal{A}_1$, receive challenge query m_0 and m_1, and respond with $\mathsf{FE.ct}$, R and z, where R is a random string of length $S + \mathsf{poly}(\lambda)$, and $z = \mathsf{Extract}(R; s) \oplus t \oplus m_0$. The adversary $\mathcal{A}_1$ produces a state st. Notice that the only component that's different for H_0 and H_1 is $\mathsf{FE.ct}$, and it does not depend on the challenge query from $\mathcal{A}_1$. R and z remain unchanged.

- Send f_v to the challenger and receive $\mathsf{FE.sk}_{f_v}$. Forward $\mathsf{sk} = \mathsf{FE.sk}_{f_v}$ to $\mathcal{A}_2$ together with $(\mathsf{FE.mpk}, m_0, m_1, \mathsf{st})$.
- If $\mathcal{A}_2$ outputs that it is in H_0, output 0. Otherwise, output 1.

It is straightforward to verify that if $\mathcal{A}$ wins the game, $\mathcal{A}'$ wins as well. □

Lemma 3. *No adversary can distinguish between H_1 and H_2 (respectively H_5 and H_6) with non-negligible probability.*

Proof. Here we will prove the case for H_1 and H_2. The case for H_5 and H_6 follows analogously.

Since pk does not depend on sk, and sk is not used until in step 7, now instead of fixing f_v (and thus $\mathsf{sk} = \mathsf{FE.sk}_{f_v}$) in step 2, we can sample it lazily in step 7. Notice that our new sampling procedure in H_2 makes the following two changes to H_1.

First, in H_1, we sample a uniform t and compute $z = \mathsf{Extract}(R; s) \oplus t \oplus m_0$, while in H_2, we sample a uniform z and compute $t = \mathsf{Extract}(R; s) \oplus z \oplus m_0$. This is just a change of variables, and gives two identical distributions.

Similarly, in H_1 we sample a uniform v and encrypt $u = v \oplus s'$, while in H_2 we encrypt a uniform u and compute $v = u \oplus s'$. Again, these are two identical distributions. Therefore, no adversary can distinguish between H_1 and H_2 with non-negligible probability. □

Lemma 4. *If the extractor $\mathsf{Extract}$ is a $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ average min-entropy extractor, then no adversary that produces a state st of size at most S can distinguish between H_2 and H_3 (respectively H_4 and H_5) with non-negligible probability.*

Proof. We prove the case for H_2 and H_3. The other case follows naturally.

Here let the random variables $X = R$, and $Y = (\mathsf{FE.mpk}, \mathsf{FE.msk}, m_0, m_1, u, z)$ and $Z = \mathsf{st}$. By Lemma 1, we have

$$H_\infty(X|Y, Z) \geq \min_y H_\infty(X|Y = y, Z) \geq \min_y H_\infty(X|Y = y) - S = \mathsf{poly}(\lambda).$$

The last equality above follows since $X = R$ is a uniformly random string, independent of Y, of length $S + \mathsf{poly}(\lambda)$. By extractor security, no adversary can distinguish $(s, \mathsf{Extract}(R; s), Y, Z)$ from (s, U_{L_m}, Y, Z) except with $\mathsf{negl}(\lambda)$ probability. Since we now sample $u \leftarrow U_{L_m}$, no adversary can now distinguish between $t = \mathsf{Extract}(R; s) \oplus z \oplus m_0$ and $t = u \oplus z \oplus m_0$, i.e. H_2 and H_3. □

Lemma 5. *No adversary can distinguish H_3 from H_4 with non-zero probability.*

Proof. Notice that the only difference between H_3 and H_4 is that in H_3 we have $t = r \oplus z \oplus m_0$ while in H_4 we have $t = r \oplus z \oplus m_1$, where r is uniformly random. Thus t is uniformly random in both cases, and H_3 and H_4 are identical. □

Theorem 2. *If FE is a functional encryption scheme with single-key selective security, and $\mathsf{Extract}$ is a $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ average min-entropy extractor, then Construction 1 has incompressible encryption security.*

Proof. The lemmas above show a sequence of a polynomial number of hybrid experiments where no PPT adversary that produces a state with size at most S can distinguish one from the next with non-negligible probability. The first hybrid H_0 corresponds to the incompressible encryption security game where $b = 0$, and the last one H_7 corresponds to the case where $b = 1$. The security of the indistinguishability game follows. □

3.4 Instantiating our FE

We now give a simple construction of functional encryption for our needed functionality. Recall that our functions f_v have the form $f_v(s, \mathsf{flag}) = s \oplus (\mathsf{flag} \cdot v)$.

Construction 2. *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 message length* $n + 1$ *is defined as:*

- $\mathsf{Setup}(1^\lambda)$*: For* $i \in \{1, \ldots, n\}, b \in \{0, 1\}$*, run* $(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{Gen}'(1^\lambda)$*. Output* $(\mathsf{mpk} = (\mathsf{pk}_{i,b})_{i,b}\ ,\ \mathsf{msk} = (\mathsf{sk}_{i,b})_{i,b})$*.*
- $\mathsf{KeyGen}(\mathsf{msk}, f_v) = (\mathsf{sk}_{i,v_i})_i$*.*
- $\mathsf{Enc}(\mathsf{mpk}, (s, \mathsf{flag}))$*: For* $i \in \{1, \ldots, n\}, b \in \{0, 1\}$*, compute* $c_{i,b} = \mathsf{Enc}'(\mathsf{pk}_{i,b},\ s_i \oplus (\mathsf{flag} \cdot b)\ )$*. Output* $c = (c_{i,b})_{i,b}$*.*
- $\mathsf{Dec}(\mathsf{sk}_{f_v}, c)$*: Output* $x = x_1 x_2 \cdots x_n$ *where* $x_i = \mathsf{Dec}'(\mathsf{sk}_{i,v_i}, c_{i,v_i})$

For correctness, note that $x_i = s_i \oplus (\mathsf{flag} \cdot v_i)$, and therefore $x = s \oplus (\mathsf{flag} \cdot v) = f_v(s, \mathsf{flag})$. Note that the rate of this scheme is $1/\mathsf{poly}(\lambda)$. Thus the overall rate of our incompressible encryption scheme is $1/\mathsf{poly}(\lambda)$.

Theorem 3. *If* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *is a CPA secure public key encryption scheme, then Construction 2 is single key semi-adaptively secure for the functions* f_v*.*

Proof. Consider a single key semi-adaptive adversary for Construction 2. Let $m_0 = (s_0, \mathsf{flag}_0), m_1 = (s_1, \mathsf{flag}_1)$ be the challenge messages. For a fixed flag bit, f_v 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, we must have $v = s_0 \oplus s_1$. Thus the two possibilities for the challenge ciphertext are the same for c_{i,v_i}, but encrypt opposite bits in $c_{i,1-v_i}$. Since the adversary never gets to see the secret keys $\mathsf{sk}_{i,1-v_i}$, a simple hybrid argument shows that flipping these bits is indistinguishable. □

Corollary 1. *Assuming the existence of a CPA secure public key encryption scheme and a* $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ *average min-entropy extractor, there exists an incompressible PKE with ciphertext size* $S + L_m + \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) L_m$*, public key size* $\mathsf{poly}(\lambda)$ *and secret key size* $\mathsf{poly}(\lambda)$*. Furthermore, it supports streaming decryption using* $L_m + \mathsf{poly}(\lambda)$ *bits of memory.*

4 Rate-1 Incompressible Encryption

Here, we construct incompressible encryption with an optimal rate of $1 - o(1)$, i.e. the message length is (almost) the same as the ciphertext length.

4.1 Construction

For our construction, we require a functional encryption scheme with single-key semi-adaptive security and a rate of 1, a strong average min-entropy extractor, and a secure pseudorandom generator (PRG). Our construction works as follows.

Construction 3. *Given* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *a rate-1 functional encryption scheme satisfying single-key semi-adaptive security,* $\mathsf{Extract} : \{0,1\}^{L_m} \times \{0,1\}^d \to \{0,1\}^n$ *a strong average min-entropy extractor where* $d, n = \mathsf{poly}(\lambda)$*, and* $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{L_m}$ *a secure PRG, the construction* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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* $v \in \{0,1\}^{L_m}$ *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* $\mathsf{sk} = \mathsf{FE.sk}_{f_{v,s}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f_{v,s})$*. Set* $L_m = S + \mathsf{poly}(\lambda)$*.*

- $\mathsf{Enc}(\mathsf{pk}, m)$*: 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.*

Let ρ_{FE} be the rate of FE. Then the ciphertext size is $(L_m + 1)/\rho_{\mathsf{FE}}$ and the rate of our incompressible encryption scheme is $\rho_\Pi = \rho_{\mathsf{FE}}/(1 + L_m^{-1})$. If $\rho_{\mathsf{FE}} = 1 - o(1)$, then $\rho_\Pi = 1 - o(1)$ as well.

Theorem 4. *Assuming the existence of a functional encryption scheme with single-key semi-adaptive security and a rate of* $1 - o(1)$*, and a* $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ *average min-entropy extractor, there exists an incompressible PKE with message size of up to* $S - \mathsf{poly}(\lambda)$*, ciphertext size* $S + \mathsf{poly}(\lambda)$*, public key size* $\mathsf{poly}(\lambda)$ *and secret key size* $\mathsf{poly}(S, \lambda)$*.*

4.2 Proof of Security

We organize our proof of security into a sequence of hybrids.

Sequence of Hybrids

- H_0: The original incompressible encryption security experiment $\mathsf{Dist}^{\mathsf{IncomEnc}}_{\mathcal{A},\Pi}$, where the bit b in the experiment is fixed to be 0.

- H_1: Instead of fixing v and s in step 2 of the security experiment, lazily sample v and s in step 7 where we need to provide sk. Also, instead of sampling v directly, first sample a uniformly random $u \in \{0,1\}^{L_m}$, and then compute $v = u \oplus m_0$.

- H_2: We further modify how we sample v. Now instead of sampling a random u, we sample a random PRG key $k \in \{0,1\}^n$, and set $v = \mathsf{PRG}(k) \oplus m_0$.

- H_3: We once more modify how we sample v. We now sample a long randomness $R \in \{0,1\}^{L_m}$ and use that to compute $v = \mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus m_0$.

- H_4: In step 5, set the ciphertext to be $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (R, 1))$.

- H_5: In step 7, revert to computing $v = \mathsf{PRG}(k) \oplus m_0$ for a uniform k.
- H_6: In step 7, revert to computing $v = u \oplus m_0$ for a uniform u.
- H_7: Switch the bit b of the experiment from 0 to 1.
- H_8: In step 7, sample v as $\mathsf{PRG}(k) \oplus m_1$.
- H_9: In step 7, sample v as $\mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus m_1$.
- H_{10}: In step 5, change the ciphertext back to $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (m_1, 0))$.
- H_{11}: In step 7, sample v as $\mathsf{PRG}(k) \oplus m_1$.
- H_{12}: In step 7, sample v as $u \oplus m_1$.
- H_{13}: Sample a uniform v back at the beginning of the experiment in step 2. Notice that now we're back at the original incompressible encryption security experiment, where the bit b is fixed to be 1.

Proof of Hybrid Arguments

Lemma 6. *No adversary can distinguish between H_0 and H_1 (respectively H_{12} and H_{13}) with non-negligible probability.*

Proof. We prove the case for H_0 and H_1. The case for H_{12} and H_{13} follows analogously. Notice that pk does not depend on sk, and sk is the only value that depends on v and s, but it is not used until in step 7. So we can sample v and s lazily in step 7 instead of fixing it as early as in step 2.

Sampling a uniformly random u and XORing it with m_0 is equivalent to using u as a one-time pad. By the statistical security of OTP, H_0 and H_1 are also statistically indistinguishable. □

Lemma 7. *If the underlying* PRG *is a secure pseudorandom generator, then no PPT adversary can distinguish between H_1 and H_2 (as well as H_5 and H_6, H_7 and H_8, H_{11} and H_{12}) with non-negligible probability.*

Proof. Here we prove the case for H_1 and H_2. The other three cases follow naturally. In H_1, we have $v = u \oplus m_0$ with uniformly random u, and in H_2, we have $v = \mathsf{PRG}(k) \oplus m_0$ with a uniformly random PRG key k. Since the key k is random and not used anywhere else, by PRG security, the PRG output should be computationally indistinguishable from a uniform distribution. This directly completes the proof. □

Lemma 8. *If the underlying* $\mathsf{Extract}$ *is a $(L_m, \mathsf{negl}(\lambda))$ average min-entropy extractor, then no adversary can distinguish between H_2 and H_3 (respectively H_{10} and H_{11}) with non-negligible probability.*

Proof. We prove the case for H_2 and H_3. The other case follows.

The randomness R is freshly sampled and not used anywhere else, and hence have full L_m average min-entropy conditioned on the other variables. Therefore, we can easily invoke the extractor security and that gives us $\mathsf{Extract}(R; s)$ is statistically close to a uniform k, and hence also H_2 and H_3. □

Lemma 9. *If the underlying* FE *is a functional encryption scheme with single-key semi-adaptive game-based security, then no PPT adversary can distinguish between H_3 and H_4 (respectively H_9 and H_{10}) with non-negligible probability.*

Proof. We will prove the case for H_3 and H_4. The other one follows analogously. Notice that the only difference between H_3 and H_4 is the message being encrypted by the underlying FE scheme. In H_3, we use the FE scheme to encrypt $(m, 0)$, while in H_4, we encrypt $(R, 1)$. Notice that

$$f_{v,s}(R, 1) = \mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus \mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus m = m = f_{v,s}(m, 0).$$

So we have two ciphertexts with the same functionality under the function $f_{v,s}$. By the single-key semi-adaptive security of the FE scheme, they should be computationally indistinguishable.

More concretely, assume that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that distinguishes between H_3 and H_4, we show how to construct an adversary $\mathcal{A}'$ that wins the semi-adaptive security game of the FE scheme. By using $\mathcal{A}$ as a subroutine, $\mathcal{A}'$ works as follows:

- Receive 1^λ and $\mathsf{FE.mpk}$ from the challenger, send 1^λ to $\mathcal{A}_1$, receive 1^S and set $L_m = S + \mathsf{poly}(\lambda)$.
- Send $\mathsf{FE.mpk}$ to $\mathcal{A}_1$ and receive challenge query m_0 and m_1.
- Sample a uniformly random $R \in \{0, 1\}^{L_m}$, and submit the challenge query $\mathsf{FE}.m_0 = (m_0, 0)$ and $\mathsf{FE}.m_1 = (R, 1)$ to the challenger. Receive $\mathsf{FE.ct}$ in response and forward it to $\mathcal{A}_1$. $\mathcal{A}_1$ produces a state st.
- Sample random seed $s \in \{0, 1\}^d$, compute $v = \mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus m_0$, and send $f_{v,s}$ to the challenger. Receive in response $\mathsf{FE.sk}_{f_{v,s}}$, and forward it to $\mathcal{A}_2$ together with $(\mathsf{FE.mpk}, m_0, m_1, \mathsf{st})$.
- If $\mathcal{A}_2$ outputs that it is in H_3, output 0. Otherwise, output 1.

It should be easy to verify that if $\mathcal{A}$ wins, $\mathcal{A}'$ also wins. □

Lemma 10. *If the underlying* $\mathsf{Extract}$ *is a* $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ *average min-entropy extractor, then no adversary that uses a state of size at most S can distinguish between H_4 and H_5 (respectively H_8 and H_9) with non-negligible probability.*

Proof. Here we prove the case for H_4 and H_5. The other case follows analogously.

Here let the random variables $X = R$, and $Y = (\mathsf{FE.mpk}, \mathsf{FE.msk}, m_0, m_1)$ and $Z = \mathsf{st}$. By Lemma 1, we have

$$H_\infty(X|Y, Z) \geq \min_y H_\infty(X|Y = y, Z) \geq \min_y H_\infty(X|Y = y) - S = \mathsf{poly}(\lambda).$$

The last equation follows from that $X = R$ is a uniformly random string of length $L_m = S + \mathsf{poly}(\lambda)$. Therefore, by extractor security, no adversary can distinguish $(s, \mathsf{Extract}(R; s), Y, Z)$ from (s, U_n, Y, Z) except with $\mathsf{negl}(\lambda)$ probability. And since we now sample $k \leftarrow U_n$, no adversary can now distinguish between $v = \mathsf{PRG}(\mathsf{Extract}(R; s)) \oplus m_0$ and $v = \mathsf{PRG}(k) \oplus m_0$, i.e. H_4 and H_5. □

Lemma 11. *No adversary can distinguish between H_6 and H_7 with non-negligible probability.*

Proof. The only difference between H_6 and H_7 is that in H_6 we have $v = u \oplus m_0$ and in H_7 we have $v = u \oplus m_1$, where u is uniformly random. This is just a one time pad encryption with a uniformly sampled key. By OTP security, H_6 and H_7 are statistically indistinguishable. $\square$

Theorem 5. *If* FE *has single-key semi-adaptive security,* $\mathsf{Extract}$ *is a* $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$ *average min-entropy extractor, and* PRG *is a secure PRG, then Construction 3 has incompressible encryption security.*

Proof. The lemmas above show a sequence of a polynomial number of hybrid experiments where no PPT adversary that produces a state with size at most S can distinguish one from the next with non-negligible probability. The first hybrid H_0 corresponds to the incompressible encryption security game where $b = 0$, and the last one H_{13} corresponds to the case where $b = 1$. The security of the indistinguishability game follows. $\square$

5 Incompressible Signatures: Our Basic Construction

5.1 Definition

Here we give the definition of *incompressible* signatures. An incompressible signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ takes an additional space parameter S, and in addition to the standard model signature security (where the adversary has unbounded space throughout the game), we also require *incompressible signature security* that utilizes the following experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

Signature Forgery Experiment $\mathsf{SigForge}^{\mathsf{IncomSig}}_{\mathcal{A},\Pi}(\lambda)$:

- The adversary $\mathcal{A}_1$, on input 1^λ, outputs a space bound 1^S.
- Run $\mathsf{Gen}(1^\lambda, 1^S)$ to obtain keys $(\mathsf{vk}, \mathsf{sk})$.
- The adversary $\mathcal{A}_1$ is given the public key vk.
- For $q = \mathsf{poly}(\lambda)$ rounds, $\mathcal{A}_1$ submits a message m, and receives $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$. At the end of the last round, $\mathcal{A}_1$ produces a state st of size at most S.
- The adversary $\mathcal{A}_2$ is given the public key vk, the state st, and all the queried messages m, and outputs a signature σ'. If $\mathsf{Ver}(\mathsf{vk}, \sigma')$ outputs $\perp$, output 0. Otherwise, output 1.

Notice that traditionally, we would require $\mathsf{Ver}(\mathsf{vk}, \sigma')$ to be distinct from the messages m's queried before, but here we have no such requirement. With this experiment in mind, we now define the additional security requirement for an incompressible signature scheme.

Definition 7 (Incompressible Signature Security). *For security parameters λ and S, an incompressible signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ has incompressible signature security, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:*

$$\Pr\left[\mathsf{SigForge}^{\mathsf{IncomSig}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

5.2 Construction

We present a very simple construction from classical public key signature schemes.

Construction 4. *Let λ, S be security parameters. Given $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ a classical public key signature scheme with message space $\{0,1\}^{n+L_m}$ where $n = S + \mathsf{poly}(\lambda)$ and rate ρ', we construct an incompressible signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Run* $\mathsf{Sig.Gen}(1^\lambda)$ *to obtain* $(\mathsf{Sig.vk}, \mathsf{Sig.sk})$. *Output* $\mathsf{vk} = \mathsf{Sig.vk}$ *and* $\mathsf{sk} = \mathsf{Sig.sk}$.
- $\mathsf{Sign}(\mathsf{sk}, m)$*: Sample randomness $R \in \{0,1\}^n$, and output* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{Sig.sk}, (R, m))$.
- $\mathsf{Ver}(\mathsf{vk}, \sigma)$*: Run* $M \leftarrow \mathsf{Sig.Ver}(\mathsf{Sig.vk}, \sigma)$. *If $M = \perp$, output $\perp$. Otherwise, if $M = (R, m)$, output m.*

We can always assume Sig can be computed in an low-space streaming fashion, since we can hash the message in low space first using Merkle-Damgård. Then Construction 5 can readily be computed with low space streaming. The rate of this construction is

$$\frac{L_m}{L_\sigma} = \frac{L_m}{(S + L_m)/\rho'} = \rho'(1 + S/L_m)^{-1}.$$

5.3 Proof of Security

Theorem 6. *Assuming the existence of a secure public key signature scheme with rate ρ', there exists an incompressible signature scheme with signature size $\rho'(S + L_m + \mathsf{poly}(\lambda))$, public key size $\mathsf{poly}(\lambda)$ and secret key size $\mathsf{poly}(\lambda)$. Furthermore, it supports streaming computation using $\mathsf{poly}(\lambda)$ bits of memory.*

Proof. We show this through a reduction proof. Concretely, we show how one can use an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the incompressible signature security as a subroutine to build an adversary $\mathcal{A}'$ the breaks the underlying classical Sig scheme. The adversary $\mathcal{A}'$ works as follows:

- Send 1^λ to $\mathcal{A}_1$, receive 1^S, and set $n = S + \mathsf{poly}(\lambda)$.
- Receive vk from the challenger, and forward it to $\mathcal{A}_1$.
- For each signing query m_i made by $\mathcal{A}_1$, sample a random $R_i \in \{0,1\}^n$ and make a query (R_i, m_i) to the challenger. Receive back σ_i and forward it directly to $\mathcal{A}_1$.
- When $\mathcal{A}_1$ produces a state st, send vk, st and all the signing queries $\{m_i\}_i$ to $\mathcal{A}_2$. Output what $\mathcal{A}_2$ outputs as σ'.

Notice that if $\mathcal{A}$ wins, that means $\mathsf{Ver}(\mathsf{vk}, \sigma') = (R', m') \neq \perp$. If $m' \notin \{m_i\}_i$, then (R', m') is a pair not queried before by $\mathcal{A}'$, and thus $\mathcal{A}'$ wins the game. If $m' = m_j$ for some j, then we argue that with overwhelming probability $R' \neq R_j$, and hence $\mathcal{A}'$ wins as well. Indeed this is true since

$$H_\infty(R_j | \mathsf{st}, \mathsf{vk}, \{m_i\}_i) \geq S + \mathsf{poly}(\lambda) - S = \mathsf{poly}(\lambda).$$

Therefore R_j is unpredictable conditioned on $\mathcal{A}_2$'s view, so the probability of $\mathcal{A}_2$ producing some $R' = R_j$ is negligible. □

6 Rate-1 Incompressible Signatures

6.1 Incompressible Encoding

Moran and Wichs [MW20] give the definition for incompressible encodings and show construction based on either the Decisional Composite Residuosity (DCR) or Learning With Errors (LWE) assumptions. We modify the definition slightly to better accommodate the syntax in this paper.

Definition 8 (Incompressible Encodings [MW20])**.** *Let λ be security parameters. An incompressible encoding scheme for message space $\{0,1\}^{L_m}$ and codeword space $\{0,1\}^{L_c}$ is a pair of PPT algorithms $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ that utilizes the following syntax:*

- $\mathsf{Enc}(1^\lambda, m) \to c$ *on input the security parameter and a message, outputs a codeword c.*
- $\mathsf{Dec}(c) \to m$ *on input a codeword, outputs the decoded message m.*

The "rate" of the incompressible encoding is L_m/L_c.[2]

We additionally require correctness and S-incompressibility[3]:

Definition 9 (Correctness)**.** *For all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, m)) = m] \geq 1 - \mathsf{negl}(\lambda)$.

For S-incompressibility, consider the following experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

Codeword Compression Experiment $\mathsf{Comp}^{\mathsf{IncomCode}}_{\mathcal{A},\mathsf{Code}}(\lambda, S)$:

- On input 1^λ, the adversary $\mathcal{A}_1$ submits a message m and auxiliary input aux. It receives $c \leftarrow \mathsf{Enc}(1^\lambda, m)$, and produces a state st of size at most S.
- The adversary $\mathcal{A}_2$ is given the state st, the message m, and the auxiliary information aux; it produces a codeword c'. Output 1 if and only if $c' = c$.

Definition 10 (S-Incompressibility)**.** *For security parameter λ, we require that for all PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:*

$$\Pr\left[\mathsf{Comp}^{\mathsf{IncomCode}}_{\mathcal{A},\mathsf{Code}}(\lambda, S) = 1\right] \leq \mathsf{negl}(\lambda).$$

[2]This is equivalent to the α-expansion property as defined in [MW20] for $\alpha = L_c/L_m$.

[3]This is equivalent to the β-incompressibility property as defined in [MW20] for $\beta = S$.

6.2 Construction

Now we show how we modify Construction 4 to get an incompressible signature scheme with a rate of 1. Essentially we can think of the procedure of attaching a long random string in Construction 4 as a form of an incompressible encoding with a poor rate. Here we just need to replace it with an incompressible encoding with a rate of 1.

Construction 5. *Let λ, S be security parameters. Given* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *a classical signature scheme with rate* 1*, and* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *an incompressible encoding scheme with rate* 1 *and S-incompressibility, we construct an incompressible signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *as follows:*

- $\mathsf{Gen}(1^\lambda, 1^S)$*: Run* $\mathsf{Sig.Gen}(1^\lambda)$ *to obtain* $(\mathsf{Sig.vk}, \mathsf{Sig.sk})$*. Output* $\mathsf{vk} = \mathsf{Sig.vk}$ *and* $\mathsf{sk} = \mathsf{Sig.sk}$.
- $\mathsf{Sign}(\mathsf{sk}, m)$*: First compute the codeword* $c \leftarrow \mathsf{Code.Enc}(1^\lambda, m)$*, and then compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{Sig.sk}, c)$.
- $\mathsf{Ver}(\mathsf{vk}, \sigma)$*: Run* $c \leftarrow \mathsf{Sig.Ver}(\mathsf{Sig.vk}, \sigma)$*. If* $c = \perp$*, output* $\perp$*. Otherwise, output* $m \leftarrow \mathsf{Code.Dec}(c)$.

The rate of our scheme is just the product of the rates of the incompressible encoding and standard-model signature scheme. Notice that using a universal one-way hash function, we can easily construct a classical signature scheme with rate $1 - o(1)$ by first hashing the message and then signing the hash value; such a signature can be built from any one-way function. The resulting incompressible signature scheme therefore has rate $1 - o(1)$, in the CRS or random oracle model.

Theorem 7. *Assuming the existence of a secure public key signature scheme with rate* 1 *and an incompressible encoding scheme with rate* 1*, there exists an incompressible signature scheme with signature size L_m, public key size* $\mathsf{poly}(\lambda)$ *and secret key size* $\mathsf{poly}(\lambda)$*. Furthermore, it supports streaming computation using* $\mathsf{poly}(\lambda)$ *bits of memory, assuming either the random oracle model, or the streaming of the CRS in the CRS model.*

Proof. Assume towards contradiction that there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that wins the incompressible signature game. Let $\{m_i\}_i$ be the message queries made by $\mathcal{A}_1$, $\{\sigma_i\}_i$ the responses, and $\{c_i = \mathsf{Sig.Ver}(\mathsf{vk}, \sigma_i)\}_i$. Let σ' be $\mathcal{A}_2$'s forgery, and $m' = \mathsf{Sig.Ver}(\mathsf{vk}, \sigma')$.

Let p be the probability that $\mathcal{A}$ wins and $m' \notin \{m_i\}_i$. The security of the standard-model signature scheme immediately implies that p is negligible: simply devise a new adversary $\mathcal{A}'$ that is the same as $\mathcal{A}$, except that it encodes every message m_i into $c_i \leftarrow \mathsf{Code.Enc}(1^\lambda, m_i)$ before making a signing query.

Let r be the probability that $\mathcal{A}$ wins and $m' \in \{m_i\}$. The security of the incompressible encoding implies that r is negligible: we construct a new adversary $\mathcal{A}''$ which sets up the standard-model signature for itself and simulates the entire view of $\mathcal{A}$. The exception is that it guesses a random i^*, and forwards the m_{i^*} as it's challenge message; when it receives c_{i^*} from the encoding challenge, it computes $\sigma_{i^*} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, c_{i^*})$. With probability r/q, this adversary is able to reproduce c_{i^*}, despite compressing it. Here, q is the number of queries made.

We therefore have that $\mathcal{A}$ wins with probability $p + r$, which is negligible. □

6.3 Equivalence to Incompressible Encoding

Lastly, we quickly show that incompressible signatures are equivalent to incompressible encodings (plus one-way functions) by showing how to construct an incompressible encoding scheme from an incompressible signature scheme.

Construction 6. *Let λ be a security parameter. Given* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *an incompressible signature scheme with rate* 1 *and small verification keys, we construct an incompressible encoding scheme* $\Pi = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ *as follows:*

- $\mathsf{Enc}(1^\lambda, m)$*: Sample* $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.Gen}(1^\lambda, 1^S)$*, and then compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m)$*. Output* $c = (\mathsf{Sig.vk}, \sigma)$*.*
- $\mathsf{Dec}(c = (\mathsf{Sig.vk}, \sigma))$*: Simply output* $m \leftarrow \mathsf{Sig.Ver}(\mathsf{Sig.vk}, \sigma)$*.*

The codeword length is the signature length (equal to message length if Sig has rate 1) plus the length of the verification length. Hence the rate is 1 if the verification keys are short.

Correctness follows directly from the correctness of the signature scheme. Security also follows directly: if an adversary using a state st of size at most S is able to produce $c' = c$, then it has also produced a valid signature σ and hence wins the incompressible signature security game. Therefore, by Construction 5 and 6, incompressible signatures and incompressible encodings (plus one-way functions) are equivalent.

7 Constructing Rate-1 Functional Encryption

Here, we build rate-1 functional encryption (FE). For our application, we only need one key security. However, our construction satisfies many-key security, though we need indistingishability obfuscation (iO). We leave it as an open question whether such high-rate *single key* FE can be built from standard tools.

Our construction is based on the techniques of Boneh and Zhandry [BZ14], who build from iO something called private linear broadcast encryption, which is a special case of general FE. A number of issues arise in generalizing their construction to general functions, which we demonstrate how to handle.

7.1 Building Blocks

Definition 11 (Indistinguishability Obfuscation [BGI+01])**.** *An* indistinguiability obfuscator *$i\mathcal{O}$ for a circuit class $\{\mathcal{C}_\lambda\}$ is a PPT uniform algorithm satisfying the following conditions:*

- **Functionality***: For any $C \in \mathcal{C}_\lambda$, then with probability 1 over the choice of $C' \leftarrow i\mathcal{O}(1^\lambda, C)$, $C'(x) = C(x)$ for all inputs x.*
- **Security***: For all pairs of PPT adversaries (S, D), if there exists a negligible function α such that*

$$\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow S(\lambda)] > 1 - \alpha(\lambda)$$

then there exists a negligible function β such that

$$\left|\Pr[D(\sigma, i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[D(\sigma, i\mathcal{O}(\lambda, C_1)) = 1]\right| < \beta(\lambda)$$

When $\mathcal{C}_\lambda$ is the class of all polynomial-size circuits, we simply call $i\mathcal{O}$ an indistinguishability obfuscator. There are several known ways to construct indistinguishability obfuscation:

- Garg et al. [GGH+13] build the first candidate obfuscation from cryptographic multilinear maps.
- Provably from novel strong circularity assumptions [BDGM20, GP21, WW20]
- Provably from "standard" assumptions [JLS21]: (sub-exponentially secure) LWE, LPN over fields, bilinear maps, and constant-locality PRGs

Definition 12 (Puncturable PRF [BW13, KPTZ13, BGI14])**.** *A* puncturable PRF *with domain $\mathcal{X}_\lambda$ and range $\mathcal{Y}_\lambda$ is a pair* $(\mathsf{Gen}, \mathsf{Punc})$ *where:*

- $\mathsf{Gen}(1^\lambda)$ *outputs an efficiently computable function* $\mathsf{PRF} : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$
- $\mathsf{Punc}(\mathsf{PRF}, x)$ *takes as input a function* PRF *and an input $x \in \mathcal{X}_\lambda$, and outputs a "punctured" function* $\mathsf{PRF}^{\overline{x}}$*.*
- **Correctness***: With probability 1 over the choice of* $\mathsf{PRF} \leftarrow \mathsf{Gen}(1^\lambda)$*,*

$$\mathsf{PRF}^{\overline{x}}(x') = \begin{cases} \mathsf{PRF}(x') & \text{if } x' \neq x \\ \perp & \text{if } x' = x \end{cases}$$

- **Security***: For all $x \in \mathcal{X}_\lambda$,* $(\mathsf{PRF}^{\overline{x}}, \mathsf{PRF}(x))$ *is computationally indistinguishable from* $(\mathsf{PRF}^{\overline{x}}, y)$*, where* $\mathsf{PRF} \leftarrow \mathsf{Gen}(1^\lambda)$ *and $y \leftarrow \mathcal{Y}_\lambda$.*

Such puncturable PRFs can be built from any one-way function [GGM86].

We now give a new definition of a type of signature scheme with a *single-point binding* (SPB) property. Very roughly, this allows, given a message m, for generating a fake verification key together with a signature on m. The fake verification key and signature should be indistinguishable from the honest case. Yet the fake verification key has the property that there are no signatures on messages other than m. In [BZ14], their security proof implicitly constructs such signatures from iO and one-way functions, but with a logarithmic message space, which was good enough for their special-purpose FE scheme. In our case, we need to handle very large exponential message spaces. The problem with [BZ14]'s approach is that the security loss is proportional to the message space; to compensate we would need to assume (sub)exponential hardness, and also set the security parameter to be somewhat larger than the message length. But this results in the signature size being polynomial in the message rate, resulting in a low-rate FE scheme. By using SPB signatures, we avoid the exponential loss, and can therefore keep the security parameter small, resulting in a rate-1 FE scheme.

Definition 13. *A* single-point binding (SPB) signature *is a quadruple of algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{GenBind})$ *where* $\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}$ *satisfy the usual properties of a signature scheme. Additionally, we have the following:*

- $(\mathsf{vk}, \sigma) \leftarrow \mathsf{GenBind}(1^\lambda, m)$ *takes as input a message m, and produces a verification key* vk *and signature σ.*
- *For any messages m and with overwhelming probability over the choice of* $(\mathsf{vk}, \sigma) \leftarrow \mathsf{GenBind}(1^\lambda, m)$*,* $\mathsf{Ver}(\mathsf{vk}, \sigma') \in \{m, \perp\}$ *for any σ'. That is, there is no message $m' \neq m$ such that there is a valid signature of m' relative to* vk*.*
- *For any m,* $\mathsf{GenBind}(1^\lambda, m)$ *and* $(\mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, m))$ *are indistinguishable, where* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*. Note that this property implies that* $\mathsf{Ver}(\mathsf{vk}, \sigma)$ *accepts and output m, when* $(\mathsf{vk}, \sigma) \leftarrow \mathsf{GenBind}(1^\lambda, m)$*.*

We explain how to construct SPB signatures in Section 7.3, either from leveled FHE (and hence LWE), or from iO and one-way functions.

Our Rate-1 FE Scheme. We now give our rate-1 FE scheme:

Construction 7. *Let $i\mathcal{O}$ be an indistinguishability obfuscator,* Gen *be a PRF,* $(\mathsf{Gen}', \mathsf{Sig}, \mathsf{Ver})$ *a signature scheme, and* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}, \mathsf{PRG}' : \{0,1\}^\lambda \to \{0,1\}^{L_m}$ *be a PRG.*

- $\mathsf{Setup}(1^\lambda)$*: Sample* $\mathsf{PRF} \leftarrow \mathsf{Gen}(1^\lambda)$*. Set* $\mathsf{msk} = \mathsf{PRF}$ *and* $\mathsf{mpk} = i\mathcal{O}(1^\lambda, P_{\mathsf{Enc}})$*, where P_{Enc} is the program given in Figure 1.*
- $\mathsf{KeyGen}(\mathsf{msk}, f)$*: output* $\mathsf{sk}_f \leftarrow i\mathcal{O}(1^\lambda, P_{\mathsf{Dec},f})$*, where $P_{\mathsf{Dec},f}$ is the program given in Figure 2.*
- $\mathsf{Enc}(\mathsf{mpk}, m)$*: Choose a random r, and evaluate* $(t, v) \leftarrow \mathsf{mpk}(r)$*. Then parse $v = (w, u)$. Set* $c = \mathsf{PRG}'(w) \oplus m$*. Next run* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}'(1^\lambda; u)$*, using u as the random coins for* Gen'*. Compute* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, c)$*. Output (t, σ).*
- $\mathsf{Dec}(sk_f,\ (t, \sigma)\ ) = \mathsf{sk}_f(t, \sigma)$

Inputs: r
Constants: PRF

1. $t \leftarrow \mathsf{PRG}(r)$.
2. $v \leftarrow \mathsf{PRF}(t)$.
3. Output (t, v).

Figure 1: The program P_{Enc}.

Correctness follows immediately from the correctness of the various components. Notice that the ciphertext size is $L_m + \mathsf{poly}(\lambda)$, provided the signature scheme outputs short signatures. Therefore, construction 7 has rate $1 - o(1)$.

```
Inputs: $t, \sigma$
Constants: $\mathsf{PRF}$

  1. $(w, u) \leftarrow \mathsf{PRF}(t)$
  2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}'(1^\lambda; u)$.
  3. $c \leftarrow \mathsf{Ver}(\mathsf{vk}, c, \sigma)$. If $c = \perp$, abort and output $\perp$.
  4. Output $f(\mathsf{PRG}'(w) \oplus c)$.
```

Figure 2: The program $P_{\mathsf{Dec},f}$.

Also notice that, provided the random coins for $(\mathsf{Gen}', \mathsf{Sign}, \mathsf{Ver})$ are independent of the message length, P_{Enc} has size $\mathsf{poly}(\lambda)$, independent of the message length. If sk can be evaluated from its random coins in low-space, and Sign can be evaluated in a low-space streaming fashion, then so can Enc.

7.2 Proof of Security

Sequence of Hybrids

- H_0: This is the FE security experiment, where the bit b in the experiment is fixed to be 0. Note that in this hybrid, the challenge ciphertext is generated as (t^*, σ^*), where $r^* \leftarrow \{0,1\}^\lambda$, $t^* \leftarrow \mathsf{PRG}(r^*)$, $(w^*, u^*) \leftarrow \mathsf{PRF}(t^*)$, $x^* \leftarrow \mathsf{PRG}'(w^*)$, $c^* \leftarrow x^* \oplus m_0$, $(\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}'(1^\lambda; u^*)$, and $\sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}^*, c^*)$.

- H_1: This is identical to H_0, except that we now generate t^* uniformly at random: $t^* \leftarrow \{0,1\}^{2\lambda}$.

- H_2: This is the same as H_1, except that we change the way we generate $\mathsf{mpk}, \mathsf{sk}_f$. First compute $\mathsf{PRF}^{\overline{t^*}} \leftarrow \mathsf{Punc}(\mathsf{PRF}, t^*)$, $(w^*, u^*) \leftarrow \mathsf{PRF}(t^*)$. Then let $(\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}'(1^\lambda; u^*)$ and $x^* = \mathsf{PRG}(w^*)$. We now compute $\mathsf{mpk} \leftarrow i\mathcal{O}(1^\lambda, P^{\mathsf{punc}}_{\mathsf{Enc}})$ and answer secret key queries with $\mathsf{sk}_f \leftarrow i\mathcal{O}(1^\lambda, P^{\mathsf{punc}}_{\mathsf{Dec}})$. Here, $P^{\mathsf{punc}}_{\mathsf{Enc}}$ and $P^{\mathsf{punc}}_{\mathsf{Dec},f}$ are the programs in Figures 3 and 4

- H_3: This is identical to H_2, except that now we generate w^*, u^* uniformly at random, instead of $(w^*, u^*) \leftarrow \mathsf{PRF}(t^*)$.

- H_4: This is identical to H_3 except that we now generate x^* uniformly at random instead of $x^* \leftarrow \mathsf{PRG}(w^*)$.

- H_5: This is identical to H_4, except for the following changes:
  - We generate c^* uniformly at random at the beginning of the experiment.
  - After the challenge query, we generate $x^* = c^* \oplus m_0$. Note that x^* is the only place m_0 enters the experiment.

- H_6: This is identical to H_5, except now we generate $(\mathsf{vk}^*, \sigma^*) \leftarrow \mathsf{GenBind}(1^\lambda, c^*)$.

- H_7 through H_{13}: for $i = 0, \cdots, 6$, Hybrid H_{7+i} is identical to H_{6-i}, except that m_0 is replaced with m_1. Thus H_{13} is the FE security experiment where b is fixed to be 1.

```
Inputs: $m; r$
Constants: $\mathsf{PRF}^{\overline{t^*}}$, $t^*$

1. $t \leftarrow \mathsf{PRG}(r)$. If $t = t^*$, immediately abort and output $\perp$.
2. $v \leftarrow \mathsf{PRF}^{\overline{t^*}}(t)$.
3. Output $(t, v)$.
```

Figure 3: The program $P^{\mathsf{punc}}_{\mathsf{Enc}}$. Differences from P_{Enc} highlighted in yellow.

```
Inputs: $t, \sigma$
Constants: $\mathsf{PRF}_1^{\overline{t^*}}$, $\mathsf{PRF}_2^{\overline{t^*}}$, $t^*, x^*, \mathsf{vk}^*$

1. If $t \neq t^*$, skip to Step 2. If $t = t^*$, run $c \leftarrow \mathsf{Ver}(\mathsf{vk}^*, \sigma)$;
   if $c = \perp$, abort and output $\perp$, otherwise abort and output $f(x^* \oplus c)$.
2. $(w, u) \leftarrow \mathsf{PRF}^{\overline{t^*}}(t)$
3. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}'(1^\lambda; u)$.
4. $c \leftarrow \mathsf{Ver}(\mathsf{vk}, c, \sigma)$. If $c = \perp$, abort and output $\perp$.
5. Output $f(\mathsf{PRG}(w) \oplus c)$.
```

Figure 4: The program $P^{\mathsf{punc}}_{\mathsf{Dec},f}$. Differences from $P_{\mathsf{Enc},f}$ highlighted in yellow.

Proofs of Hybrid Steps

Lemma 12. *If* PRG *is a secure PRG, then no PPT adversary can distinguish between* H_0 *and* H_1 *(respectively* H_{12} *and* H_{13}*) except with negligible probability.*

Proof. The only difference between the hybrids is how we generate t^*; in H_0 it is pseudorandom and in H_1 it is uniformly random. Thus indistinguishability follows immediately from the security of PRG. □

Lemma 13. *If* $i\mathcal{O}$ *is a secure indistinguishability obfuscator, then no PPT adversary can distinguish between* H_1 *and* H_2 *(respectively* H_{11} *and* H_{12}*) except with negligible probability.*

Proof. Note that, with overwhelming probability, the uniformly random t^* is not in the sparse image of PRG. Thus, with overwhelming probability, the abort step in $P_{\mathsf{Enc}}^{\mathsf{punc}}$ is never triggered. On all $t \neq t^*$, PRF and $\mathsf{PRF}^{\overline{t^*}}$ behave identically. Thus, P_{Enc} and $P_{\mathsf{Enc}}^{\mathsf{punc}}$ have identical functionalities. Thus their obfuscations are indistinguishable. Likewise, $P_{\mathsf{Dec},f}$, on input (t^*, c, σ), would compute $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}'(1^\lambda; u^*)$, which would exactly output $(\mathsf{vk}^*, \mathsf{sk}^*)$. Provided the signature accepted, it would output $f(m)$ where $m = \mathsf{PRG}(w^*) \oplus c$. Thus $P_{\mathsf{Dec},f}$ and $P_{\mathsf{Dec},f}$ behave identically on all inputs of this form. On inputs (t, c, σ) with $t \neq t^*$, the programs trivially behave identically. Thus they are identical on all inputs, and their obfuscations are indistinguishable. □

Lemma 14. *If* $(\mathsf{Gen}', \mathsf{Punc})$ *is a secure puncturable PRF, then no PPT adversary can distinguish between* H_3 *and* H_4 *(respectively* H_{10} *and* H_{11}*) except with negligible probability.*

Proof. The only difference between these hybrids is that w^*, u^* switch from being outputs of $\mathsf{PRF}(t^*)$ to being uniformly random. But since the rest of the experiment can be simulated using only $\mathsf{PRF}^{\overline{t^*}}$, security follows immediately from punctured PRF security. □

Lemma 15. *If* PRG' *is a secure PRG, then no PPT adversary can distinguish* H_3 *and* H_4 *(respectively* H_9 *and* H_{10}*) except with negligible probability.*

Proof. The only difference between the hybrids is that we switch from x^* being pseudorandomly generated from a random w^* to x^* being uniform. Indistinguishability follows immediately from the security of PRG'. □

Lemma 16. H_4 *and* H_5 *(respectively* H_8 *and* H_9*) are identically distributed.*

Proof. Since x^* is uniform, so is $c^* = x^* \oplus m_b$. In both hybrids, we choose c^* or x^* randomly, and solve for the other. Thus the distributions are identical. □

Lemma 17. *If* $(\mathsf{Gen}', \mathsf{Sign}, \mathsf{Ver}, \mathsf{GenBind})$ *is a secure SPB signature scheme, then no PPT adversary can distinguish between* H_5 *and* H_6 *(respectively* H_7 *and* H_8*) except with negligible probability.*

Proof. Note that neither hybrid requires sk^*, and the only difference is how we generate vk^*, σ^*: in H_5 (respectively H_8) vk^* is generated from Gen' using fresh random coins u^* and σ^* is the signature on c^*, whereas in H_6 (respectively H_7), $(\mathsf{vk}^*, \sigma^*)$ is generated as $\mathsf{GenBind}(1^\lambda, c^*)$. Indistinguishability follows immediately from the security of the signature scheme. □

Lemma 18. *If* $i\mathcal{O}$ *is a secure indistinguishability obfuscator, then no PPT adversary can distinguish between* H_6 *and* H_7.

Proof. The only difference between the two hybrids is whether $x^* = c^* \oplus m_0$ or $x^* = c^* \oplus m_1$, and the only place x^* enters the experiment is in $P_{\mathsf{Dec},f}^{punc}$. Moreover, x^* only affects the output $f(x^* \oplus c)$, and only in the event that the input (t, c, σ) satisfies $t = t^*$ and $\mathsf{Ver}(\mathsf{vk}^*, c, \sigma)$ accepts.

By the single-point binding of vk^*, all inputs (t^*, c, σ) reject, except for $c = c^*$. But in the case $c = c^*$, we have that $f(x^* \oplus c) = f(m_b)$. The FE security experiment guarantees that $f(m_0) = f(m_1)$. Thus the programs $P_{\mathsf{Dec},f}^{punc}$ have identical functionality in both hybrids, and so their obfuscations are indistinguishable. □

Theorem 8. *If $i\mathcal{O}$ is a secure indistinguishability obfuscator,* $\mathsf{PRG}, \mathsf{PRG}'$ *are secure PRGs,* $(\mathsf{Gen}', \mathsf{Sign}, \mathsf{Ver}, \mathsf{GenBind})$ *is a secure SPB signature, and* $(\mathsf{Gen}, \mathsf{Punc})$ *is a secure puncturable pseudorandom function, then Construction 7 is a secure functional encryption scheme.*

7.3 Constructing SPB Signatures

We now show how to construct single-point binding signatures.

A low-rate construction. We first describe a simple low-rate construction. This construction is not good enough for our purposes, as our FE scheme inherits the rate of the signature scheme. But we will later show how to compile any low-rate construction into a high-rate construction.

Our construction is just Lamport one-time signatures, where the underling one-way function is replaced with a PRG:

Construction 8. *Let* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a PRG. Then for a desired message length n, our construction works as follows:*

- $\mathsf{Gen}(1^\lambda)$*: for $i \in \{1, \dots, n\}, b \in \{0,1\}$, sample* $\mathsf{sk}_{i,b} \leftarrow \{0,1\}^\lambda$ *and set* $\mathsf{vk}_{i,b} = \mathsf{PRG}(\mathsf{sk}_{i,b})$*. Output* $(\mathsf{vk} = (\mathsf{vk}_{i,b})_{i,b}\ ,\ \mathsf{sk} = (\mathsf{sk}_{i,b})_{i,b})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$*: output* $\sigma = (m,\ (\mathsf{sk}_{i,m_i})_i\ )$
- $\mathsf{Ver}(\mathsf{vk}, \sigma)$*: Extract m from σ. For each $i \in \{1, \dots, n\}$, check that* $\mathsf{PRG}(\mathsf{sk}_{i,m_i}) = \mathsf{pk}_{i,m_i}$*. If all checks pass, output m. Otherwise output $\perp$.*
- $\mathsf{GenBind}(1^\lambda, m)$*: for each $i \in \{1, \dots, n\}$, sample* $\mathsf{sk}_{i,m_i} \leftarrow \{0,1\}^\lambda$ *and set* $\mathsf{vk}_{i,m_i} = \mathsf{PRG}(\mathsf{sk}_{i,m_i})$*. Then sample* $\mathsf{vk}_{i,1-m_i} \leftarrow \{0,1\}^{2\lambda}$ *uniformly. Output* $(\mathsf{vk} = (\mathsf{vk}_{i,b})_{i,b}\ ,\ \sigma = (m,\ (\mathsf{sk}_{i,m_i})_i)\ )$.

In other words, to bind to a message, simply replace all the public key components that do not correspond to the message with uniform randomness.

Binding follows from the fact that, with overwhelming probability $\mathsf{vk}_{i,1-m_i}$ in binding mode will have no pre-images. Since any message other than m must differ from m on *some* bit i, such messages will not have any signatures. Security follows immediately from the pseudorandomness of PRG.

The problem with this signature scheme is that its raate is poor: the signature on a message is a multiplicative $\mathsf{poly}(\lambda)$ factor larger than the message itself

From low-rate to high-rate using SPB hashes. We now describe a new object, related to somewhere statistically binding (SSB) hashing [HW15], which we call single-point binding (SPB) hashing.

Definition 14. *A* single-point binding (SPB) hash function *is a triple of algorithms* $(\mathsf{Gen}, H, \mathsf{GenBind})$ *where:*

- $\mathsf{Gen}(1^\lambda)$ *produces a hashing key* hk.

- $H(\mathsf{hk}, m)$ *deterministically produces a hash h, with $|h| \ll |m|$.*
- $\mathsf{GenBind}(1^\lambda, m^*)$ *takes as input a message m^*, and produces a hashing key* hk *with the property that, with overwhelming probability over the choice of* $\mathsf{hk} \leftarrow \mathsf{GenBind}(1^\lambda, m^*)$*, for any $m \neq m^*$, $H(\mathsf{hk}, m) \neq H(\mathsf{hk}, m^*)$.*
- *For any message m^*, $(m^*, \mathsf{Gen}(1^\lambda))$ is computationally indistinguishable from $(m^*, \mathsf{GenBind}(1^\lambda, m^*))$.*

We now use a SPB hash to improve the rate of an SPB signature. The construction is the usual hash-and-sign signature scheme: to sign a message m, simply compute the signature $\sigma \leftarrow H(\mathsf{hk}, m)$, and output (m, σ). If the underlying signature is an SPB signature, then $\mathsf{GenBind}$ for the new signature simply binds hk to m, and then binds vk to $H(\mathsf{hk}, m)$.

If H hashes to a size that is independent of the message, then the resulting signature has rate 1, regardless of the rate of the original signature.

Constructing SPB Hashing. It remains to construct an SPB hash function.

We first briefly note that such hash functions can be build from fully homomorphic encryption (FHE), following essentially the same construction of somewhere statistically binding hashing from [HW15]. The hashing key is normally the encryption of a random string r of length equal to the message. To hash a message m, homomorphically compute an encryption of b, the result of comparing m with r. To bind the hashing key to m, simply encrypt the message m. FHE security immediately implies security. For binding, the message m will then hash to an encryption of 1, whereas any other message will hash to an encryption of 0. By the correctness of the FHE scheme, encryptions of 0 and 1 must be disjoint.

Next, we explain how to get a construction from iO and one-way functions. Since we are already using iO and one-way functions to build our FE scheme, these assumptions are redundant.

Construction 9. *Let $i\mathcal{O}$ be an indistinguishability obfuscator,* Gen' *the generation algorithm for a PRF, and* PRG *a pseudorandom generator.*

- $\mathsf{Gen}(1^\lambda)$*: Sample* $\mathsf{PRF} \leftarrow \mathsf{Gen}'(1^\lambda)$*, and output* $\mathsf{hk} = i\mathcal{O}(1^\lambda, P_{\mathtt{hash}})$*, where $P_{\mathtt{hash}}$ is the program given in Figure 5.*
- $H(\mathsf{hk}, m) = \mathsf{hk}(m)$
- $\mathsf{GenBind}(1^\lambda, m^*)$*: Sample* $\mathsf{PRF} \leftarrow \mathsf{Gen}'(1^\lambda)$*, compute* $\mathsf{PRF}^{\overline{m^*}} \leftarrow \mathsf{Punc}(\mathsf{PRF}, m^*)$*, and choose a random string x^*. Output* $\mathsf{hk} = i\mathcal{O}(1^\lambda, P^{\mathtt{bind}}_{tthash})$*, where $P^{\mathtt{bind}}_{tthash}$ is the program given in Figure 6.*

Remark 10. *If we want our rate-1 incompressible encryption to have encryption be computable in low space given a stream of m, then we need our rate-1 FE encryption to be likewise be computable in low space given a message stream. This in turn means we need to be able to evaluate $H(\mathsf{hk}, m)$ in low space given the message stream, and given the random coins used to construct* hk*. We can always assume the random coins are small. In our construction, we cannot compute* hk *itself in low space, since it is a large* *obfuscated program. However, we can nevertheless compute* $H(\mathsf{hk}, m) = \mathsf{PRG}(\mathsf{PRF}(m))$ *in low-space, provided* PRF *has small keys and can be evalauted on a message stream in low space (the output of* PRF *is small, so* PRG *can easily be computed once we have* $\mathsf{PRF}(m)$*. Most PRFs have this property, including the puncturable PRF from one-way functions due to [GGM84]. This gives us the desired rate-1 incompressible encryption with low-space encryption.*

```
Inputs: $m$
Constants: $\mathsf{PRF}$

  1. Output $\mathsf{PRG}(\mathsf{PRF}(m))$
```

Figure 5: The program $P_{\texttt{hash}}$.

```
Inputs: $m$
Constants: $\mathsf{PRF}^{\overline{m^*}}$, $m^*, x^*$

  1. If $m = m^*$, output $x^*$. Otherwise,

  2. Output $\mathsf{PRG}(\mathsf{PRF}(m))$
```

Figure 6: The program $P_{\texttt{hash}}^{\texttt{bind}}$. Differences from $P_{\texttt{hash}}$ are highlighted in yellow.

For binding, note that the random x^* outputted on input m^* is, with overwhelming probability, outside the range of PRG. But all inputs $m \neq m^*$ must output points in the range of PRG. Thus, there are no collisions with m^*.

For security, use the following sequence of hybrids:

- H_0: this is the case where $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda)$.
- H_1: here, we generate $\mathsf{hk} = i\mathcal{O}(P_{\texttt{hash}}^{\texttt{bind}})$, except that x^* is set to $\mathsf{PRG}(\mathsf{PRF}(m^*))$. Note that this x^* is exactly the output of $P_{\texttt{hash}}(m^*)$. Hence in this case $P_{\texttt{hash}}^{\texttt{bind}}$ and $P_{\texttt{hash}}$ have identical functionalities. Indistinguishability follows from iO.
- H_2: here we generate $x^* = \mathsf{PRG}(s^*)$ for a uniform random value s^*. The only difference from H_1 is that we replace $\mathsf{PRF}(m^*)$ with s^*. But since only the punctured PRF $\mathsf{PRF}^{\overline{m^*}}$ is needed, this change follows from punctured PRF security.
- H_3: here, we generate $\mathsf{hk} \leftarrow \mathsf{GenBind}(1^\lambda, m^*)$. The only difference from H_2 is that we replace $x^* = \mathsf{PRG}(s^*)$ with a uniformly random x^*. Since s^* is uniformly random, this follows immediately from the pseudorandomness of PRG.

References

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri

Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 113–134. Springer, Heidelberg, May / June 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 36–54. Springer, Heidelberg, August 2009.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 65–79. Springer, Heidelberg, August 1999.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2014.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 79–109. Springer, Heidelberg, May 2020.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

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

[BNNO11] Rikke Bendlin, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Lower and upper bounds for deniable public-key encryption. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2011.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *39th FOCS*, pages 493–502. IEEE Computer Society Press, November 1998.

[CDD$^+$07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 479–498. Springer, Heidelberg, February 2007.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Heidelberg, August 1997.

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 292–306. Springer, Heidelberg, August 1997.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 446–472. Springer, Heidelberg, February 2004.

[Din01] Yan Zong Ding. Oblivious transfer in the bounded storage model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 155–170. Springer, Heidelberg, August 2001.

[DLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 225–244. Springer, Heidelberg, March 2006.

[DQW21] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. Cryptology ePrint Archive, Report 2021/1270, 2021. `https://ia.cr/2021/1270`.

[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. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 207–224. Springer, Heidelberg, March 2006.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 736–749, 2021.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

[GZ19] Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 500–524. Springer, Heidelberg, May 2019.

[GZ21] Jiaxin Guan and Mark Zhandry. Disappearing cryptography in the bounded storage model. In *Theoretical Cryptography Conference*, 2021. `https://ia.cr/2021/406`.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

[IR90] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 8–26. Springer, Heidelberg, August 1990.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 60–73, 2021.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

[Lu02] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 257–271. Springer, Heidelberg, August 2002.

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

[MST04] Tal Moran, Ronen Shaltiel, and Amnon Ta-Shma. Non-interactive timestamping in the bounded storage model. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 460–476. Springer, Heidelberg, August 2004.

[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.

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In Chris Umans, editor, *58th FOCS*, pages 732–742. IEEE Computer Society Press, October 2017.

[Vad03] Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 61–77. Springer, Heidelberg, August 2003.

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 111–126. ACM, January 2013.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. Cryptology ePrint Archive, Report 2020/1042, 2020. `https://eprint.iacr.org/2020/1042`.