Incompressible Functional Encryption

Rishab Goyal [*]
UW-Madison
rishab@cs.wisc.edu

Venkata Koppula
IIT Delhi
kvenkata@iitd.ac.in

Mahesh Sreekumar Rajasree
IIT Delhi
srmahesh1994@gmail.com

Aman Verma
IIT Delhi
amanverma1729@gmail.com.

Abstract

Incompressible encryption (Dziembowski, Crypto'06; Guan, Wichs, Zhandry, Eurocrypt'22) protects from attackers that learn the entire decryption key, but cannot store the full ciphertext. In incompressible encryption, the attacker must try to compress a ciphertext within pre-specified memory bound S before receiving the secret key.

In this work, we generalize the notion of incompressibility to functional encryption. In incompressible functional encryption, the adversary can corrupt non-distinguishing keys at any point, but receives the distinguishing keys only after compressing the ciphertext to within S bits. An important efficiency measure for incompressible encryption is the ciphertext-rate (i.e., $\mathsf{rate} = |m|/|\mathsf{ct}|$). We give many new results for incompressible functional encryption:

1. Incompressible attribute-based encryption for circuits from standard assumptions, with ct-rate-$\frac{1}{2}$ and short secret keys,
2. Incompressible functional encryption for circuits from (non-incompressible) functional encryption, with
   (a) ct-rate-$\frac{1}{2}$ and short secret keys,
   (b) ct-rate-1 and large secret keys.

Our results achieve optimal efficiency, as incompressible attribute-based/functional encryption with ct-rate-1 as well as short secret keys has strong implausibility barriers. Moreover, our assumptions are minimal as incompressible attribute-based/functional encryption are strictly stronger than their non-incompressible counterparts.

1 Introduction

A fundamental principle in cryptography is to leverage 'secrets' for differentiating between an honest user legitimately accessing a system and an attacker trying to illegitimately access it. Consider data encryption as an example, where the goal is data secrecy. A central assumption underlying traditional encryption is that *an attacker cannot learn anything about a user's secret decryption key*.

[*]Support for this research was provided by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.

Moreover, this assumption seems inherent as if an attacker learns the secret key, then how can one distinguish between 'capabilities' of an honest user and an attacker!?

In 2006, Dziembowzki [Dzi06] proposed an exquisite approach to *restrict the 'capabilities' of an attacker*, even when the *entire* secret key gets corrupted. The proposal was to view *long-term memory storage* as a limited and expensive resource. In words, the intuition was that no real-world attacker can maintain an unbounded long-term memory, and it has to pick and choose what it wants to store in its memory. Dziembowzki [Dzi06] formalized this in the secret-key setting, and defined it as a *forward-secure storage* system. Recently, Guan, Wichs, and Zhandry (henceforth GWZ) [GWZ22] generalized this to the public-key setting. They labeled cryptography in this model as "incompressible" cryptography. The intuition being that an attacker cannot 'compress' its unbounded short-term memory into its bounded long-term memory. GWZ provided multiple constructions for 'incompressible' public-key encryption and signatures, from a variety of cryptographic assumptions with different efficiency tradeoffs.

Beyond Public-Key Encryption. Functional encryption (FE) [SW05, BSW07, KSW08, BSW11] is a powerful generalization of public-key encryption (PKE) [DH76]. FE generalizes PKE by providing fine-grained access to encrypted data. In FE, the master decryption key holder can create partial 'functional' decryption keys sk_f for any supported function f of its choice. Such a key holder can recover $f(m)$ from any ciphertext ct, encrypting data m. Standard indistinguishability-based FE security states that no polytime attacker cannot distinguish between encryptions $\mathsf{ct}_0, \mathsf{ct}_1$ of messages m_0, m_1, unless it receives a secret key sk_{f^*} s.t. $f^*(m_0) \neq f^*(m_1)$. That is, unless an attacker receives a 'distinguishing key' sk_{f^*}, it cannot distinguish between any two ciphertexts.[1] The ability to learn only function evaluation of encrypted data and nothing more has had fascinating consequences in cryptography, both theoretically and practically [Sha84, BF01, Coc01, SW05, GPSW06, BW07, BSW07, KSW08, BSW11, O'N10, GVW15, GPS16, GKW17, WZ17, CDG$^+$17, GKRW18, KMUW18, QWW18, GKW18, JLS21, JLS22, BS23, DGM23].

While functional keys offer fine-grained access over encrypted data, enabling many superior applications, they also enable far more attack opportunities compared to plain encryption. In PKE, there is just one secret key, and while that implies an all-or-nothing functionality, storing the key securely for the entire system lifetime is much easier. On the contrary, in FE, there are master secret keys as well as functional decryption keys. Each decryption key must be securely generated, distributed, and stored by the appropriate user. While it might be reasonable to expect the central trusted authority will (in most part) store master key msk securely; expecting this from every other user (holding a functional key) is unrealistic.

Our Contributions: Incompressible FE. In this work, we study incompressible functional encryption systems. We formalize the concept, and provide multiple constructions from a variety of cryptographic assumptions enabling different efficiency tradeoffs. As we elaborate later in Appendix A, incompressibility in FE is extremely interesting, both theoretically and practically. In short, it pushes the traditional perception in FE that leaking an entire decryption key is equivalent to learning the function output on every previously encrypted data.

[1]Throughout the sequel, by the phrase a 'distinguishing key', we mean a secret key that can distinguish between two ciphertexts (by running decryption). Similarly, by the phrase 'non-distinguishing keys', we mean secret keys that do not enable a trivial distinguishing attack between two ciphertexts.

Formalizing incompressible FE. Interestingly, formally defining incompressibility for FE is more nuanced than for PKE. In public-key encryption, there is just one secret key. Thus, to define incompressibility for PKE, one considers a simple two-stage attacker: in stage 1, the attacker receives a ciphertext that it must 'compress' down to S bits[2]; and stage 2, where the attacker receives the secret key (along with the S-bit 'compressed' ciphertext). As long as such an attacker cannot learn anything about the message, the PKE scheme is a secure incompressible scheme. In functional encryption setting, there are many more secret keys! Thus, depending upon the secret key(s) that a *stage 2* attacker learns, there are different formulations of incompressible FE security. Below, we briefly summarize three such formulations, and discuss them in further detail in Section 2.1.[3]

– Standard security: the adversary receives *one* (or bounded) number of *distinguishing* secret key(s), after compressing challenge ciphertext (i.e., in stage 2). Moreover, it receives unbounded number of *non-distinguishing* secret keys before and after compressing the ciphertext.

– Semi-strong security: this is a strengthening of standard security, where the adversary can get an *unbounded* number of distinguishing keys in stage 2.

– Strong security: this further strengthens as now adversary gets *master secret key* in stage 2.

Measuring incompressibility efficiency. Clearly, for the above to not be trivially impossible, the ciphertext size should be larger than S, the incompressibility limit. Further, the ciphertext should be larger than $|m|$ to uniquely encode m and guarantee correctness. However, it is not clear if there are any other necessary constraints on parameter sizes, beyond $|\mathsf{ct}| = \max(S, |m|) + \mathsf{poly}(\lambda)$. Here the $\max(S, |m|)$ term ensures both ciphertext constraints (i.e., $|\mathsf{ct}| > S$ and $|\mathsf{ct}| > |m|$).

Thus, an ideal goal is to design incompressible FE where $|\mathsf{mpk}|, |\mathsf{msk}|, |\mathsf{sk}_f|$ are all independent of S, and $|\mathsf{ct}| = \max(S, |m|) + \mathsf{poly}(\lambda)$. Such an incompressible FE scheme would have asymptotically optimal-sized parameters. As we discuss later in Section 2.1, achieving optimality in all parameter sizes is unachievable from standard falsifiable assumptions. Therefore, a standard approach in the literature [GWZ22, BDD22a] is to design encryption with highest ciphertext-rate. Ciphertext-rate is defined as the ratio of message length and ciphertext length, i.e. $\frac{|m|}{|\mathsf{ct}|}$.

Prior works [GWZ22, BDD22a] built incompressible PKE with optimal ct-rate of 1. Unfortunately, that comes at the cost of large secret keys, i.e. $|\mathsf{sk}| = \mathsf{poly}(S, \lambda)$. In this work, we design incompressible ABE/FE in two incomparable parameter regimes – ct-rate of 1 with large secret keys, and ct-rate of $\frac{1}{2}$ with short secret keys. We believe both parameter regimes to be equally interesting. In FE applications, where an honest user stores only secret keys and not ciphertexts in its persistent long-term storage, it might be beneficial to have short keys with constant ct-rate. While, in applications where an honest user stores only a few secret keys and a large number of ciphertexts, having a ct-rate of 1 might be more useful.

Our results

We summarize our main results below. All our schemes are in the standard model and under standard assumptions. We do not make any ideal-cipher, random oracle, or any other non-standard/non-falsifiable cryptographic assumption. Broadly, we split our results in two categories.

[2] S is typically considered to be the bound on the adversary's long-term storage.

[3] We can also consider joint compression of ciphertexts and secret keys in FE. The focus of this work is only on ciphertext incompressibility as discussed in Appendix A. It is an interesting open question of whether new tradeoffs and applications can be enabled, if one considers joint incompressibility of ciphertexts and secret keys.

Incompressible ABE. Our first result is an incompressible attribute-based encryption (ABE) scheme for polynomial-sized circuits. This scheme achieves (standard) incompressible ABE security, with ciphertext size $|\mathsf{ct}| = S + |m| + \mathsf{poly}(\lambda)$ and all other parameter sizes, $|\mathsf{mpk}|$, $|\mathsf{msk}|$, $|\mathsf{sk}_f|$, are independent of S. That is, this scheme has ct-rate of $\frac{1}{2}$, and short keys.

We provide two separate ABE constructions. The first construction relies on the hardness of Learning with Errors assumption [Reg05], while the second is a generic construction that upgrades any (standard) ABE scheme to an incompressible ABE scheme. For technical reasons discussed later, we provide the two separately.

Incompressible FE. Our second set of results contains new constructions for incompressible FE for polynomial-sized circuits. We provide three separate and incomparable FE constructions as each provides a unique efficiency-security tradeoff. All three construction rely on a standard public-key FE scheme for polynomial-sized circuits.

1. Our first incompressible FE scheme is proven secure in the *semi-strong* incompressibility model. If the underlying FE scheme has ct-rate of $r \in [0, 1]$, then our incompressible FE scheme has an $\frac{r}{2}$ ct-rate.
2. Our second scheme is also proven secure in the *semi-strong* incompressibility model, except we can only prove selective security.[4] Moreover, this construction is a ct-rate preserving construction. That is, we obtain ct-rate r incompressible FE scheme from any ct-rate r standard FE scheme. However, the secret keys are large to avoid implausibility results.
3. Our third scheme is an extension of our second FE scheme. This FE scheme is proven secure only in the *selective standard* incompressibility model, but it provides a rather non-trivial efficiency guarantee. We prove that if the underlying FE scheme has short keys, then so does our incompressible FE scheme (without lowering the ct-rate). As we elaborate later in the overview, this does not contradict the implausibility result (see discussion in Section 2.4), but rather tightly matches it.

By plugging in known optimal-ciphertext-rate FE schemes [GWZ22, JLL23], we obtain incompressible FE schemes under the assumption of polynomially secure FE with the following efficiency and security:

1. a ct-rate-$\frac{1}{2}$, short keys, and selective semi-strong security,
2. a ct-rate-$\frac{1}{4}$, short keys, and *adaptive* semi-strong security,
3. a ct-rate-1, large keys, and selective semi-strong security,
4. a ct-rate-1, *short* keys, and selective *standard* security.

2 Technical Overview

In this section, we provide a high level overview of our main contributions. First, we formally introduce the concept of incompressible FE. Second, we present the main technical ideas behind our incompressible ABE and FE constructions. Lastly, we provide a discussion on alternate approaches, and discuss known implausibility results and some related work.

[4]Here selective means that the adversary must submit challenge messages before receiving secret keys in stage 1.

2.1 Defining Incompressible FE

Syntactically, incompressible FE is identical to (regular) FE as it contains the same set of algorithms: Setup, Enc, KeyGen, Dec. The difference is that the security property is superior. Recall that the traditional IND-based FE security states an attacker that receives a challenge ciphertext ct (encrypting one of two messages m_0, m_1) along with a polynomial number of decryption keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_q}$ (for functions $f_1, \ldots, f_q$) *cannot* guess which of m_0 or m_1 was encrypted so long as $f_i(m_0) = f_i(m_1)$ for all q functions. Clearly, if the attacker receives a key for some function f^* such that $f^*(m_0) \neq f^*(m_1)$, then an attacker can trivially distinguish by decrypting the challenge ciphertext.[5]

Incompressibility of FE states that the indistinguishability should hold even when the attacker receives a secret key for such a distinguishing function f^* so long as the adversary reduces its state to a pre-specified bound S, before receiving the distinguishing key sk_{f^*}. It is straightforward to see that this is no longer trivially impossible whenever $|\mathsf{ct}| > S$ (i.e., compressed local state cannot contains full ciphertext). A bit more formally, the incompressibility game can be abstracted out via the following phases:

Phase I: regular FE-style game**.** The challenger and attacker play the regular FE indistinguishability security game [BSW11]. (The restriction about querying only non-distinguishing keys still holds.) That is, the attacker can use arbitrary polynomial time and space, and can query function f as long as $f(m_0) = f(m_1)$. Here m_0, m_1 are the challenge messages.

State Reduction. The attacker outputs a 'compressed' state st of size at most S.

Phase II: querying the distinguishing key**.** The attacker is reset with the above state st. It also gets the challenge messages (m_0, m_1) sent during Phase I.[6] The attacker then makes a key query for any distinguishing function f^* s.t. $f^*(m_0) \neq f^*(m_1)$. The attacker is also allowed to make polynomially many secret key queries for non-distinguishing functions f s.t. $f(m_0) = f(m_1)$.

Finally, the attacker outputs its guess of which message was encrypted.

We say an FE scheme is a secure incompressible FE scheme if no PPT attacker can successfully guess the message. Clearly, the above implies the regular FE security (since any successful attacker in the regular FE game can also win in the above game by submitting its guess right after Phase I). The definition is a natural generalization of incompressible encryption security [GWZ22] to FE. The distinction is that, in PKE, there is a single (distinguishing) key; while, in FE, there could be many distinguishing keys. Also, note that the above definition is oblivious to function class, thus incompressible PKE/IBE/ABE etc. security can be defined as a special case of the above.

Stronger incompressibility security. Viewing incompressibility via the lens of FE, we observe that we could even consider stronger corruption models. For instance, an attacker could ask for more than one distinguishing key in Phase II or, more generally, an attacker might corrupt the master secret key in Phase II (thereby obtain any key it wants)? It is immediate that the above notions are increasingly more powerful. We refer to these notions of semi-strong incompressibility

[5]We can also consider stronger simulation style security definitions [BSW11, O'N10]. However, we know that general simulation-secure FE [AGVW13] is impossible, and there exists generic compilers [GKW17] to boost IND-based ABE security to SIM-based security. Thus, we stick to IND-based security in this work, and leave SIM-based security for future.

[6]Giving the challenge messages to the Phase-II adversary could be useful in settings where the message size itself is bigger than S.

and strong incompressibility, respectively. In this work, we provide new constructions for, both, standard and semi-strong incompressibility security models. It is an interesting open problem to design FE achieving strong incompressible security.

2.2 Incompressible ABE with ciphertext-rate-$\frac{1}{2}$

Attribute-based encryption (ABE) [SW05, GPSW06] is an extremely popular sub-class of FE. The two common variants of ABE are key-policy ABE and ciphertext-policy ABE. In key-policy ABE, the ciphertext is associated with an attribute attr in addition to a payload message m, and the functionality is that given a secret key sk_f, for some predicate f, one can learn m if $f(\mathsf{attr}) = 1$ and otherwise the payload is hidden. Ciphertext-policy is a dual where the predicate and attribute switch places. In this work, we keep our focus on the key-policy variant of ABE. Our results can be easily generalized to ciphertext-policy ABE as well.

Why doesn't hybrid encryption work? A natural first idea is to use the folklore hybrid encryption technique to combine a (regular) ABE scheme with an incompressible SKE scheme. Basically, the encryption algorithm for the candidate incompressible ABE scheme does the following– (1) sample an incompressible secret key $\mathsf{inc.sk}$, (2) encrypt message m using $\mathsf{inc.sk}$ to create an incompressible SKE ciphertext $\mathsf{inc.ct}$, (3) encrypt $\mathsf{inc.sk}$ w.r.t. attribute attr under the base ABE scheme. The incompressible ABE ciphertext contains, both ABE and incompressible SKE ciphertexts.

The intuition behind above template is that maybe just encoding the payload message m using an incompressible encoding could be enough. While it seems a solid candidate template to boost to incompressible ABE, it does not work! Further, it highlights a key technical difficulty in building incompressible encryption. The issue is that to rely on incompressible SKE security, we need to remove any information about $\mathsf{inc.sk}$ from the challenge ciphertext and somehow put it inside the ABE (distinguishing) secret key. This type of ciphertext/secret key shuffling is not immediate for an ABE scheme, especially when the goal is to achieve good ciphertext-rate.

Our initial approach is to encrypt messages using ABE in a "non-committing" way. To make our task easier, we start with a simpler goal of poor ciphertext-rate. Recall that ciphertext-rate is the ratio of message over ciphertext length, i.e. $\frac{|m|}{|\mathsf{ct}|}$.

Poor ct-rate via deferred encryption. With the simpler target of poor-rate incompressible ABE, our first attempt is to use *deferred encryption* techniques [GKW16] developed originally in the context of boosting selective to semi-adaptive security for FE. At a high level, deferred encryption paradigm relies on the canonical Yao-style garbled-circuit-based 2PC (two-party computation) protocol [Yao82] to defer computation of actual ciphertext to decryption process. We use the same core principle for obtaining the desired ciphertext/secret key shuffling.

A bit formally, to encrypt a message m for attribute attr, the encryptor samples randomness r and writes down the encryption circuit $D_{m,r}$ that has m, r hardwired and, on input an incompressible SKE key $\mathsf{inc.sk}$, outputs $\mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m; r)$. The encryptor garbles $D_{m,r}$, resulting in a garbled circuit $\hat{D}$ and wire keys $\{\mathsf{lab}_{i,b}\}_{i,b}$. These keys are themselves encrypted using the base ABE scheme, where label $\mathsf{lab}_{i,b}$ is encrypted for attribute (attr, i, b). The point is by ensuring that a decryptor only learns half of the wire keys, we can simulate the garbled circuit, rather than actually putting the message in the garbled circuit explicitly.

To achieve this, an incompressible ABE secret key for function f includes two keys – an incompressible SKE key $\mathsf{inc.sk}_f$ and an ABE key $\mathsf{sk}_{\hat{f}}$ for the base ABE scheme. The key generator samples a fresh incompressible SKE secret key $\mathsf{inc.sk}_f$ for each predicate function f, and then creates the ABE scheme for function $\hat{f}$, where $\hat{f}(\mathsf{attr}, i, b) = 1$ iff $f(\mathsf{attr}) = 1$ and $\mathsf{inc.sk}_f[i] = b$. Clearly, the above scheme satisfies correctness. More importantly, one could show that, by simulating the garbled circuit, the ABE portion of the ciphertext can be programmed to contain an incompressible SKE ciphertext rather than the actual hardwired (m, r). And, this is enough to reduce incompressible ABE security to incompressible SKE security[7].

Why does it give poor rate? Unfortunately, the above strategy leads to very poor ciphertext-rate (i.e., $1/\mathsf{poly}(\lambda)$). This is because it garbles an incompressible SKE encryption circuit, and even when encrypt a single bit, the ciphertext size grows as $\mathsf{poly}(S, \lambda)$. There are two main sources of inefficiency:

- Our template uses a generic incompressible SKE scheme. Even if use an incompressible SKE with optimal ciphertext-rate-1 (i.e., $|\mathsf{ct}| = \max(|m|, S) + \mathsf{poly}(\lambda)$), its encryption circuit might still be as large as $\mathsf{poly}(\lambda, S, |m|)$.

- Suppose that we have an incompressible SKE, where encryption circuit is of size $\max(|m|, S) + \mathsf{poly}(\lambda)$. This is still not enough since garbling introduces more overhead. Note that the garbled circuit size typically is at least $\lambda\times$ larger than the original circuit size.

To get around the above technical hurdles, we develop a new 'two-level' deferred encryption technique. We believe this might be of independent interest. Recently, deferred encryption techniques were used in the context of bounded collusion FE [GGLW22, AMVY21, GGL24] to build more efficient FE schemes. We believe that one might be able to enable newer efficiency tradeoffs in bounded collusion FE by applying our two-level deferred encryption techniques.

Rate-$\frac{1}{2}$ ABE via 'two-level' deferred encryption. At a high level, our solution to improving ciphertext-rate to $\frac{1}{2}$ can be partitioned into a two-step approach. The first step is to design an incompressible ABE scheme for encrypting short (λ-bit) messages, with optimal ciphertext size. That is, we want the ciphertext size to be $S + \mathsf{poly}(\lambda)$, where S is the compression parameter. In other words, the goal is to design an optimal incompressible ABE scheme for short messages. Our second step is to generically upgrade any optimal incompressible ABE supporting λ-bit messages to a ct-rate-$\frac{1}{2}$ incompressible ABE scheme. That is, the resulting scheme can encrypt unbounded length messages and has ciphertext size: $|m| + S + \mathsf{poly}(\lambda)$. This implies the ciphertext-rate to be $\frac{1}{2}$, which is computed as the fraction $\frac{|m|}{|\mathsf{ct}|}$ when $|m| \to S \to \infty$ (i.e., message size approaches the compression parameter, and it approaches ∞).

The first step of our approach relies on our two-level deferred encryption technique, and the second step relies on a rather simple yet highly consequential observation about incompressible all-or-nothing encryption systems. First, let us dive into the design of our incompressible ABE scheme for short messages, and later we will extend it to encrypt arbitrary length messages.

[7]Technically, one needs to permute the wire keys to ensure the ABE ciphertext contains no information about $\mathsf{inc.sk}_f$, but we ignore this for simplicity. For the technical overview, one could instead use incompressible PKE and the permutation trick won't be needed

Incompressible ABE for λ-bit messages. Recall the two sources of inefficiency that we discussed earlier– large encryption circuit and garbling overhead. To get around the first issue, our plan is to use Dziembowski's information-theoretic scheme [Dzi06]. As we explain next, it has many useful structural properties that help in simplifying our encryption circuit that we want to garble. Next, to overcome the second issue, we use a special FE scheme that efficiently garbles our specialized encryption circuit. One of our core observations for the second part is that we only need a *partially hiding FE* scheme [GVW15] with short keys, and we know how to build these [GKP+13, BGG+14]. We provide more details next.

The main ingredients behind our two-level deferred encryption approach are as follows:

- First, we rely on a special 'partially hiding' FE scheme. In a typical partially hiding FE (PHFE) scheme, a ciphertext encrypts an input x which is viewed as two disjoint components $(x_{\mathsf{priv}}, x_{\mathsf{pub}})$. That is, input x has a private component x_{priv} and a public component x_{pub}. Each secret key sk_f is still associated with a single function f, but it enables the functionality that decryptor can compute $f(x) = f(x_{\mathsf{priv}}, x_{\mathsf{pub}})$ as well as it can learn x_{pub}. That is, x_{pub} is no longer hidden, and treated as a public component.

  The two special properties that we need from PHFE are that the secret key size is short, and the encryption circuit size only grows with the input length. That is, for any function f, representable as a fixed depth (say λ) circuit, the size of the corresponding secret key sk_f is at most $\mathsf{poly}(\lambda)$. Moreover, we need the encryption circuit size to just grow as $\mathsf{poly}(\lambda, |x|)$. In words, both the secret key and encryption circuit size, $|\mathsf{sk}_f|$ and $|\mathsf{Enc}|$, can grow with depth, but not the size, of circuit representation of f. While such PHFE is a very strong assumption in general, it is much easier to design if only a single secret key gets corrupted. In short, we need the following:

  - (public/private input splittability) $x = (x_{\mathsf{priv}}, x_{\mathsf{pub}})$
  - (key succinctness) $|\mathsf{sk}_f| = \mathsf{poly}(\lambda, \mathsf{depth}(f))$.
  - (Enc succinctness) $|\mathsf{Enc}| = \mathsf{poly}(\lambda, |x|, \mathsf{depth}(f))$.
  - (1-key, 1-ciphertext security) given f, sk_f and an encryption of x, an adversary learns nothing more than $f(x)$ and x_{pub}.

  Such PHFE schemes can be based on the Learning with Errors assumption [GKP+13, BGG+14]. Moreover, we believe that we might be able to design them from simple assumptions such as standard IBE/ABE. We leave further analysis of above PHFE for future works.

- Second, we open up Dziembowski's (information-theoretic) incompressible SKE for λ-bit messages. The secret key consists of two $\lambda-$bit strings (k, w). The scheme uses a strong extractor $\mathsf{Ext} : \{0,1\}^{\lambda} \times \{0,1\}^{S+\mathsf{poly}(\lambda)} \to \{0,1\}^{\lambda}$. To encrypt a message $m \in \{0,1\}^{\lambda}$, the encryptor chooses a random source $R \leftarrow \{0,1\}^{S+\mathsf{poly}(\lambda)}$, and outputs $(R, \mathsf{Ext}_k(R) \oplus w \oplus m)$ as the incompressible ciphertext. Using extractor security, we have that given a uniformly random seed sd, and an S-bit digest computed from $R \leftarrow \{0,1\}^{S+\mathsf{poly}(\lambda)}$, the output of $\mathsf{Ext}_{\mathsf{sd}}(R)$ still looks uniformly random. This is enough to prove SKE incompressibility.

Now our intuition is that if the strong extractor can be implemented using a fixed-depth circuit (say $\mathsf{poly}(\lambda)$), then Dziembowski's encryption algorithm can also be implemented by a $\mathsf{poly}(\lambda)$-depth circuit (as all other operations are just $\oplus$). It is well-known [RRV99, GVW15, CL16] that strong extractors with the desired efficiency properties exist. Thus, combining this with the fact that we have PHFE with short keys and short encryption circuit, we can bypass the two sources of inefficiency in our basic garbled circuit based construction.

At a high level, our approach is to add an "extra" level of deferred encryption via PHFE. That is, instead of directly garbling the incompressible SKE encryption circuit (as we did earlier), we will garble an FE encryption circuit and give an FE secret key in the clear. The purpose of doing this is to ensure that the incompressible SKE encryption circuit is no longer "garbled", but instead the computation of incompressible SKE ciphertext is performed under the PHFE hood. This makes the garbled circuit efficient, and independent of the compression parameter S, since it is just performing PHFE encryption which is a short computation. Moreover, since PHFE encodes the functions optimally within the functional secret key, thus Dziembowski's encryption circuit can be encoded far more efficiently. Combining the above ideas, the decryptor still recovers an incompressible SKE ciphertext eventually (which it decrypts as before), but the difference is that computation of the incompressible SKE ciphertext happens more efficiently.

We refer the above as a two-level deferred encryption technique. This is because we view garbled circuits as performing one level of *outer* deferring, and by using PHFE internally, we can further defer the encryption process to outside of the garbled circuit. Our two-level deferring technique might be of independent interest. Next, we explain our ABE scheme more formally.

Structure of our incompressible ABE ciphertext and secret keys: An incompressible ABE ciphertext contains the following components:

- a long ($S + \mathsf{poly}(\lambda)$ bit) string R.
- a garbled circuit $\hat{C}$ of size $\mathsf{poly}(\lambda)$.
- an FE secret key, of size $\mathsf{poly}(\lambda)$.
- 2λ (regular) ABE ciphertexts, each encrypts a garbled circuit wire label. Hence, overall, these contribute $\mathsf{poly}(\lambda)$ bits.

The incompressible ABE secret key contains two λ-bit strings k, w and a (regular) ABE secret key. Note that the ciphertext size is $S + \mathsf{poly}(\lambda)$, and the secret key has size $\mathsf{poly}(\lambda)$.

How these components are computed:

- To encrypt a message m, we first choose a sufficiently long string R. Let D_R be a function that takes as input (k, w) and outputs $\mathsf{Ext}_k(R) \oplus w$. Note that D_R has size $\mathsf{poly}(\lambda, S)$, but depth is bounded by λ (because of extractor properties).
- Next, we sample $\mathsf{fe.msk}$, and compute $\mathsf{sk}_R \leftarrow \mathsf{FE.KeyGen}(\mathsf{fe.msk}, D_R)$. This key has size $\mathsf{poly}(\lambda)$ (due to PHFE key succinctness).
- We then compute the garbled circuit and wire labels. Let C_{ct} be a circuit that has the λ-bit message m, $\mathsf{fe.msk}$ and randomness r_{Enc} hardwired, takes as input two λ-bit strings (k, w) and outputs $\mathsf{FE.Enc}(\mathsf{fe.msk}, (k, m \oplus w); r_{\mathsf{Enc}})$. Here k is viewed as the public component, and $m \oplus w$ as the private component. This produces garbled circuit $\hat{C}$ and wire labels $\{\mathsf{lab}_{i,b}\}$. (Since PHFE encryption algorithm is succinct.)
- Finally, we encrypt the label $\mathsf{lab}_{i,b}$ using ABE public key for attribute (attr, i, b).

Unsurprisingly, the ABE secret key for a function f is same as for our poor-rate design. That is, it contains an incompressible SKE key (for Dziembowski's scheme) $\mathsf{inc.sk}_f = (k_f, w_f)$ and a regular ABE key for circuit $\mathsf{sk}_{\hat{f}}$, where $\hat{f}(\mathsf{attr}, i, b) = 1$ iff $f(\mathsf{attr}) = 1$ and $\mathsf{inc.sk}_f[i] = b$.

To decrypt an incompressible ABE ciphertext, one must follow a four-step process– (i) decrypt the (regular) ABE ciphertexts using the (regular) ABE secret key to recover wire labels corresponding to $\mathsf{inc.sk}_f = (k_f, w_f)$, (ii) evaluate the garbled circuit to obtain a PHFE ciphertext, (iii) decrypt the PHFE ciphertext using the PHFE secret key to obtain an incompressible SKE ciphertext (for Dziembowski's scheme), (iv) decrypt the incompressible SKE ciphertext using $\mathsf{inc.sk}_f$ to recover message m.

The proof of incompressibility security is similar to the security proof for our poor-rate scheme, with a few extra hybrid steps. Crucially, after simulating the garbled circuit, we have to use PHFE security before reducing to the incompressible SKE security. We refer to Section 5 and Appendix C for a detailed description of our construction and its security analysis. This concludes the high level overview of our incompressible ABE scheme for small messages with optimal efficiency.[8]

Improving ct-rate for incompressible ABE via hybrid encryption. At the beginning of this section, we briefly summarized why hybrid encryption does not work to combine a regular ABE with incompressible SKE to build an incompressible ABE. However, it turns out that we can use a hybrid encryption idea to combine an incompressible ABE/IBE/PKE for fixed-length messages and PRFs to encrypt unbounded length messages efficiently.

Basically, we use deterministic hybrid encryption to encrypt long messages. The idea is that an encryptor now samples a random PRF key $K \leftarrow \{0,1\}^\lambda$, and encrypts K using the incompressible ABE scheme. Now to encrypt the actual message m, it uses K to deterministically encrypt the message as follows: $m \oplus (F_K(1)||\cdots||F_K(|m|))$. The final ciphertext contains the one-time-padded message and the incompressible ABE ciphertext. Whenever the ciphertext size of the incompressible ABE scheme grows as $S + \mathsf{poly}(\lambda)$ (since $|K| = \lambda$), we get that the total ciphertext size grows as $S + |m| + \mathsf{poly}(\lambda)$. This gives us the desired efficiency, and security follows from a simple hybrid argument since incompressible ABE security can be used to argue that K is hidden.

Combining the above two steps, we obtain our result of incompressible ABE with short keys, and ciphertext size $|\mathsf{ct}| = S + |m| + \mathsf{poly}(\lambda)$. Next, we move to our incompressible FE constructions.

2.3 Incompressible FE

In this section, we provide a high level overview of our incompressible FE constructions. Our overview is split into two parts– first, we explain our main ideas behind our constant-rate incompressible FE scheme with short keys; and second, we discuss our optimal-rate incompressible FE scheme. Both our constructions share some common ideas but, due to diverging goals, their exact execution is significantly different.

Part I: rate-$\frac{1}{2}$ with short keys. Although we could start by generalizing the deferred encryption paradigm to the functional encryption setting, we find it easier to design incompressible FE from scratch. This is because deferred encryption already leads to a poor rate (which we optimized using two-level deferring). And, since the existence of a (regular) FE scheme is already necessary assumption for designing incompressible FE, and (regular) FE is an extremely powerful and versatile cryptographic object, thus we provide a direct construction with optimal rate in a single step.

[8]Recall, for short messages, the ciphertext size must be at least $S + \mathsf{poly}(\lambda)$. Thus, our incompressible ABE scheme achieves optimal efficiency of all parameters for the small-message setting.

Our core idea is that, to ensure incompressibility of FE ciphertexts, we will use an incompressible PKE scheme[9] inside of a (regular) FE scheme. The plan is to use incompressible PKE scheme to encrypt the actual challenge message, and put that inside the base FE scheme such that only distinguishing keys decrypt the incompressible PKE ciphertext, while non-distinguishing keys keep on using the message that is just directly encrypted within the FE ciphertext.

In a bit more detail, during system setup, we sample an incompressible PKE public-secret key pair $(\mathsf{inc.pk}, \mathsf{inc.sk})$ along with an FE master public-secret key pair. To encrypt a message m, we run the base FE encryption algorithm to encrypt a message m along with a flag bit b and an incompressible PKE ciphertext $\mathsf{inc.ct}$. Typically, the encryptor sets the flag bit $b = 0$ (to denote that this is a real-world ciphertext) and encrypts some number of zeros inside the incompressible PKE ciphertext $\mathsf{inc.ct}$. Now to create functional key for some function f, the key generator defines an expanded function $\hat{f}$ on the input string $(m, b, \mathsf{inc.ct})$ as follows:

$$\hat{f}(m, b, \mathsf{inc.ct}) = \begin{cases} f(m) & \text{if } b = 0 \text{ or } f \text{ is a non-distinguishing function} \\ f(\mathsf{IncPKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})) & \text{otherwise.} \end{cases}$$

The incompressible FE secret key for function f consists of an FE secret key for function $\hat{f}$. Clearly, the above scheme is correct. To prove security, our plan is that to switch an honest/real-world encryption of challenge message m_β from $\mathsf{FE.Enc}(m_\beta, 0, \mathsf{inc.ct}(0))$ to $\mathsf{FE.Enc}(m_0, 1, \mathsf{inc.ct}(m_b))$. This should follow from regular FE security as for any distinguishing and/or non-distinguishing secret key, both ciphertexts would give the same result. Once this happens, then we hope to reduce security to incompressibility security of the PKE scheme.

The above template has two issues– (1) how to *secretly* encode whether f is a distinguishing or a non-distinguishing function inside $\hat{f}$, (2) how to ensure that a non-distinguishing key does not contain any information about $\mathsf{inc.sk}$. If we can resolve these issues, then it appears that this would be enough to prove incompressibility security of our FE scheme.

Hidden triggers via another layer of 'flagging'. Our approach to get around this is to introduce another flag bit as a hidden trigger. That is, each expanded function $\hat{f}$ no longer contains $\mathsf{inc.sk}$ in the clear, but in an encrypted form. Moroever, the ciphertext encrypting $\mathsf{inc.sk}$ also contains another flag bit $\hat{b}$ that indicates whether the function f is a distinguishing or a non-distinguishing function for challenge messages m_0, m_1. With this modification, we also need to update how we perform encryption. Our incompressible FE ciphertext now is a regular FE ciphertext encrypting a message m, flag bit b, an incompressible PKE ciphertext $\mathsf{inc.ct}$, and an SKE secret key $\mathsf{ske.sk}$. And, the expanded function $\hat{f}$ is defined as:

$$\hat{f}(m, b, \mathsf{inc.ct}, \mathsf{ske.sk}) = \begin{cases} f(m) & \text{if } b = 0 \text{ or } \hat{b} = 0 \\ f(\mathsf{IncPKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})) & \text{otherwise.} \end{cases}$$

where $(\hat{b}, \mathsf{inc.sk}) = \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$. Here $\mathsf{ske.ct}$ is computed individually for each function f, and it contains a flag $\hat{b}$ and an incompressible PKE key $\mathsf{inc.sk}$. Now in the real-world, $\mathsf{ske.ct}$ encrypts zeros, but during the security proof, it transitions to encryption of a bit $\hat{b_f}$ (signalling f is distinguishing/non-distinguishing) as well as incompressible SKE key $\mathsf{inc.sk}$.

[9]Here, we use incompressible PKE for simplicity of exposition. In our formal construction, we use incompressible SKE.

One final layer of redirection. The above FE encryption of m has four slots in the ciphertext: the first slot contains the message m (to be used by the non-distinguishing functions), the second slot contains a flag bit b (which indicates that we are in the 'proof' mode), the third contains an incompressible PKE ciphertext (which, in the above overview, was an encryption of m and is to be used by the distinguishing functions) and the last slot contains a (regular) SKE secret key. The FE secret key for f is a secret key for function $\hat{f}$ which has f and SKE encryption of flag bit $\hat{b}$ and $\mathsf{inc.sk}$ hardwired. The flag bit $\hat{b}$ indicates whether this key is for a non-distinguishing function or a distinguishing function, and $\mathsf{inc.sk}$ is included only in the distinguishing functions' keys.

Unfortunately, the above solution still does not have good ct-rate as the incompressible PKE scheme could have a bad rate. Furthermore, even if we plug in a ct-rate-1 incompressible PKE in the above design, we could possibly achieve a ct-rate of $\frac{1}{2}$ for our FE scheme, but it increases the secret key size (as the incompressible PKE secret would be longer).

To achieve ct-rate-$\frac{1}{2}$, we add one final layer of redirection. Our incompressible FE ciphertext encrypts an additional SKE secret key. We hardwire an SKE ciphertext, for this key, inside $\hat{f}$ and this ciphertext encodes function evaluation $f(m_b)$ for every distinguishing function f. This ensures that we no longer need to encrypt a long message under the incompressible PKE system (as we encrypt this SKE secret key) as well as it still guarantees that the appropriate function value gets hardwired in each distinguishing key. We provide our full construction and proof in detail in Section 6. Combining our results with state-of-the-art (regular) FE schemes with constant/optimal ct-rate [JLL23], we obtain either: (i) a $\frac{1}{4}$ $|\mathsf{ct}|$-rate incompressible FE scheme with short keys and *adaptive* semi-strong incompressibility security, or (ii) a $\frac{1}{2}$ $|\mathsf{ct}|$-rate incompressible FE scheme with short keys and *selective* semi-strong incompressibility security. Any further improvement in the ct-rate or selective/adaptive security of (regular) FE schemes would automatically lead to an incompressible FE scheme with short keys and better efficiency/security.

Part II: rate-1 with semi-strong security. When the desired goal is rate-1 incompressible FE, we can no longer afford to FE encrypt a string that has one slot for the message and a separate slot for an incompressible ciphertext. It is essential that the message slot and the slot containing the randomness part for the core incompressibility/extractor-based argument overlap significantly. At a high level, in the real-world, the encryption of m would be an FE encryption of $(m, 0)$. In the proof, we switch this to an encryption of $(x, 1)$ where x is drawn uniformly at random, and the '1' in second component indicates that ciphertext is generated in 'proof' mode. As a result, the information about m_0 or m_1 is contained solely in the secret keys. For any non-distinguishing key query f, we hardwire $z = f(m_0) = f(m_1)$ in the secret key.[10] In the second phase, the adversary can query for polynomially many distinguishing functions. The challenger samples a random seed s, samples bit β and sets $v = m_\beta \oplus \mathsf{Ext}_s(x)$ (the same s and v are used for all distinguishing key queries). For any such key query f, the adversary receives an FE key for a related function $\hat{f}_{(s,v)}$ where $\hat{f}_{s,v}(x, 1) = f\left(v \oplus \mathsf{Ext}_s(x)\right)$. As before, we need to secretly encode whether a key is for a distinguishing function or a non-distinguishing one, and we use the same hidden triggers mechanism for this.

Our solution in detail. The master public key/secret key are generated using the regular FE scheme's setup. To encrypt a message m, we FE encrypt $(m, 0, \mathbf{0})$ where $\mathbf{0}$ is of length $\mathsf{poly}(\lambda)$. The secret key

[10]Looking ahead, since the function output needs to be included in the secret keys, we can achieve only selective security with this approach (that is, all key queries must come after the challenge messages are sent).

for a function f is the FE secret key for function $\hat{f}$ which has f and a random string $\mathsf{ske.ct}$ hardwired, and is defined as follows:

$$\hat{f}_{\mathsf{ske.ct}}(m, b, \mathsf{ske.sk}) = \begin{cases} f(m) & \text{if } b = 0 \text{ (used by real-world ciphertexts)} \\ z & \text{if } \hat{b} = 0 \text{ (used in proof for non-distinguishing keys)} \\ f(v \oplus \mathsf{Ext}_s(m)) & \text{otherwise (used in proof for distinguishing keys)} \end{cases}$$

where $\left(\hat{b}, z, s, v\right) = \mathsf{SKE.Dec}\left(\mathsf{ske.sk}, \mathsf{ske.ct}\right)$.

First, note that if the base FE scheme is rate-1, then so is the above scheme. Next, we briefly discuss why this scheme satisfies *selective* semi-strong security. In the proof, at the start of the experiment, the challenger samples a bit $\beta \leftarrow \{0, 1\}$, an SKE secret key $\mathsf{ske.sk}$, a sufficiently long string x, and an extractor seed s. On receiving the challenge messages (m_0, m_1), the challenger sends an FE encryption of $(x, 1, \mathsf{ske.sk})$. For the non-distinguishing secret keys, it computes $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}\left(\mathsf{ske.sk}, (0, f(m_0), \perp, \perp)\right)$ and gives an FE secret key for $\hat{f}_{\mathsf{ske.ct}}$. For the distinguishing secret keys, $\mathsf{ske.ct}$ is an encryption of $\left(1, \perp, s, m_\beta \oplus \mathsf{Ext}_s(x)\right)$. Using the security of the base FE scheme and the SKE scheme, we can show that this experiment is indistinguishable from the selective semi-strong incompressibility experiment, and note that the information about β is only contained in the distinguishing keys. However, m_β is masked by $\mathsf{Ext}_s(x)$, and if x is sufficiently longer than the adversary's state size S, then using the strong extractor's guarantee, we can conclude that $\mathsf{Ext}_s(x)$ looks uniformly random to the second-phase adversary, and therefore the bit β is hidden.

Conclusion. Note that once we replace m with the extractor randomness, for the non-distinguishing function queries, we must hardwire $f(m)$ in the secret key. As a result, we can only achieve selective security. Next, note that the above construction achieves semi-strong security. The second-phase adversary can query for polynomially many distinguishing functions. For each function f, we compute $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}\left(\mathsf{ske.sk}, (1, \perp, s, v)\right)$ (where s and v are defined during setup and are common for all queries), and give an FE secret key for $\hat{f}_{\mathsf{ske.ct}}$.

Rate-1 incompressible FE with standard security and short keys. The solution discussed above achieves semi-strong security, and has large secret keys (note that the secret key has v hardwired, and v has same length as the message). However, there is a closely related approach where we can have v to be of same length as the output size of f (and therefore, if the output of f is short, then the FE secret key is short). The idea is that the function f does not need to be computed within $\hat{f}_{\mathsf{ske.ct}}$. Instead, we can define $\hat{f}_{\mathsf{ske.ct}}$ as follows:

$$\hat{f}_{\mathsf{ske.ct}}(m, b, \mathsf{ske.sk}) = \begin{cases} f(m) & \text{if } b = 0 \\ z & \text{if } \hat{b} = 0 \\ v \oplus \mathsf{Ext}_s(m) & \text{otherwise} \end{cases}$$

where $\left(\hat{b}, z, s, v\right) \leftarrow \mathsf{SKE.Dec}\left(\mathsf{ske.sk}, \mathsf{ske.ct}\right)$.

This approach gives us a incompressible FE scheme with regular security, having rate-1 ciphertexts and short keys! In the proof, we choose x during setup and the challenge ciphertext is encryption of $(x, 1, \mathsf{ske.sk})$. For the non-distinguishing function queries, we SKE encrypt $(0, f(m_0), \perp, \perp)$ and give an FE key for $\hat{f}_{\mathsf{ske.ct}}$. When we receive the distinguishing function f in the second phase, we sample a seed s, set $v = \mathsf{Ext}_s(x) \oplus f\left(m_\beta\right)$, compute $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}\left(\mathsf{ske.sk}, (1, \perp, s, v)\right)$ and give an FE key for $\hat{f}_{\mathsf{ske.ct}}$. Using the extractor's guarantee, we can conclude that v is indistinguishable from a uniformly random string, and therefore β is hidden. This approach can handle a bounded number of distinguishing key queries. When instantiated with a rate-1 (regular) FE scheme (such as the semi-adaptive FE construction of [JLL23]), this gives us a rate-1 incompressible FE scheme with short keys.

At first sight, rate-1 incompressible FE with short keys may look surprising (see discussion in Section 2.4). The crucial thing to note here is that the function output size is small, and we are aiming for standard incompressible security (and not semi-strong security). This allows the feasibility of rate-1 incompressible FE with short secret keys.

2.4 Rate-optimality, related work, and alternate approaches

Implausibility result and rate-optimality. In a follow-up work [BGK+24], it was shown that constructing a ct-rate-1 incompressible public-key encryption scheme with a *short* secret key is impossible if the goal is to prove security under the hardness of standard falsifiable assumptions [Nao03] via black-box reductions [Nao03, GW11, Wic13]. In more detail, the authors proved that a ct-rate-1 incompressible PKE with large keys and a $|\mathsf{ct}|$-rate-$\frac{1}{2}$ incompressible PKE with short keys to be asymptotically optimal. Since incompressible ABE implies incompressible PKE, thus the above implausibility extends to incompresssible ABE as well. This suggests our incompressible ABE scheme with ct-rate-$\frac{1}{2}$ matches the lower bound. Moreover, we also know FE is strictly more powerful than PKE, thus it seems reasonable to assume that the above lower bound holds for incompressible FE as well. While this is mostly accurate, there is a slight technical nuance. In this work, for simplicity, we consider FE for boolean-value functions. That is, each FE secret key is associated with a function with one-bit output. Thus, for such FE schemes, we can argue that a ct-rate-$\frac{1}{2}$ incompressible FE scheme with short keys is optimal if the adversary can corrupt an unbounded number of FE secret keys (i.e., we consider adversaries in the semi-strong setting). Otherwise, it turns out ct-rate-1 incompressible FE scheme with short keys is not ruled by [BGK+24] if we only consider standard incompressibility security (i.e., where an adversary corrupts only one distinguishing key). Thus, our final FE construction (overview in Section 2.3, full construction in Section 8) does not bypass the implausibility result, but rather it tightly matches it.

Related works. In the space of incompressible encryption, Dziembowski [Dzi06] gave the first incompressible SKE constructions, an information-theoretic rate-$\frac{1}{3}$ construction, and a rate-$\frac{1}{2}$ construction using one-way functions. GWZ [GWZ22] presented two constructions for incompressible PKE– the first construction is based on the existence of PKE schemes but had poor rate, and the second used indistinguishable obfuscators [GGH+13, SW14, GGH+16] to construct a rate-1 scheme. Following GWZ, Branco *et al.* [BDD22b] constructed a rate-1 incompressible PKE scheme that is resistant against chosen ciphertext attacks (CCA) using a rate-1 incompressible SKE and (programmable) hash proof systems. Most recently, GWZ [GWZ23] introduced multi-user incompressible encryption schemes, where the adversary initially has access to arbitrarily many ciphertext for different messages encrypted under different secret keys. After compressing these ciphertexts to an α-fraction and receiving all the secret keys, it can guess at most α-fraction of the messages. They also defined a simulation-based incompressible security notion.

In addition to incompressible encryption, GWZ defined and constructed incompressible signature schemes. Incompressible signatures state that no adversary can forge a valid signature, nor reconstruct a signature from its compressed version. A closely related concept is the notion of incompressible encodings [DGO19, GLW20, MW20]. These encodings have the property that it is computationally infeasible to reconstruct a codeword for a message from its compressed form, even in the presence of the message. Prior works [DGO19, GLW20, MW20] provided positive results for incompressible encodings in the random oracle model as well as the CRS model. Another related concept is the bounded storage model (BSM) [Mau92, CM97, AR99, Lu02, Raz17, GZ21, DQW23], where the adversary has unbounded computational power, but a restricted memory capacity. While BSM appears to be closely related to the incompressible cryptography paradigm, note that the adversary in the latter setting is not space-bounded; only its long-term storage space is bounded.

Alternate approaches. As suggested by an anonymous reviewer, one can generalize the GWZ [GWZ22] PKE constructions to build incompressible ABE. A direct generalization would give us two constructions: (1) an incompressible ABE scheme with poor rate, starting with any ABE scheme (2) an optimal-rate incompressible ABE scheme using indistinguishability obfuscation. We observe that our technique for improving ct-rate for incompressible ABE via hybrid encryption can still be applied to the first construction and this gives us a rate-$\frac{1}{2}$ ABE from minimal assumption of ABE with short keys and standard security. We provide this in Appendix B for completeness. One might try to extend it to FE as well, but it is unclear due to the functionality provided by general FE. Any straightforward adaption either seems insufficient even for proving standard incompressibility security, or it needs the recursive hidden trigger type techniques we use in our FE constructions already.

Lastly, remark that the reason we keep our PHFE based as our main ABE construction is because it contains a new two-level deferred encryption technique which we believe could be of independent interest. Moreover, state that we believe our ABE construction relies on a very weak PHFE for instantiation. We hope that in the future it can be designed from simpler assumptions such as IBE etc.

3 Preliminaries and Notations

Throughout this paper, we will use λ to denote the security parameter and $\text{negl}(\cdot)$ to denote a negligible function in the input. We will use the short-hand notation PPT for "probabilistic polynomial time". For any finite set X, $x \leftarrow X$ denotes the process of picking an element x from X uniformly at random. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from the distribution $\mathcal{D}$. For any natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For any two binary string x and y, $x||y$ denotes the concatenation of x and y. We use the following two notations to denote a family of circuits - $\{\mathcal{C}_n\}_n$ is a set of families of circuits indexed by some parameter n and $\{\mathcal{C}_{d,\ell}\}_{d,\ell}$ is a set of families of circuits indexed by the depth of the circuits d and the number of inputs to the circuits ℓ.

3.1 Randomness Extractors

Definition 1 (Strong Average Min-Entropy Extractor)**.** *A (k, ϵ)-strong average min-entropy extractor is an efficient function* $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ *such that for all jointly distributed random variable X, Y where X takes values $\{0,1\}^d$ and $H_\infty(X|Y) \geq k$, we have $(U_d, \mathsf{Ext}(X, U_d), Y) \approx_\epsilon (U_d, U_m, Y)$ where U_d, U_m are uniformly random strings of length d, m respectively. Here $H_\infty(X|Y) = -\log \mathop{\mathbb{E}}_{y \leftarrow Y}(\max_x$ $\Pr(X = x|Y = y))$ is the average min-entropy of X conditioned on Y.*

Theorem 2. *There exists an explicit efficient* $(k, 2^{-\lambda})$*-strong average min-entropy extractor* $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^\lambda$ *such that* $k = \mathsf{poly}(\lambda)$, $d = \mathsf{poly}(\lambda)$, $n = S + k$ *and the depth of the extractor circuit is* $\mathsf{poly}(\lambda, \log(n))$.

3.2 Pseudorandom Functions

A family of pseudorandom functions $\mathsf{PRF} = (\mathsf{KeyGen}, \mathsf{Eval})$ with key space $\{\mathcal{K}_\lambda\}_\lambda$, input space $\{\mathcal{X}_\lambda\}_\lambda$ and output space $\{\mathcal{Y}_\lambda\}_\lambda$ consists of the following algorithms.

- $\mathsf{KeyGen}(1^\lambda)$: The key generation algorithm is a randomized algorithm that takes as input the security parameter 1^λ and outputs a key $k \in \mathcal{K}_\lambda$.
- $\mathsf{Eval}(k, x)$: The evaluation algorithm is a deterministic algorithm that takes as input a key $k \in \mathcal{K}_\lambda$ and $x \in \mathcal{X}_\lambda$ and outputs $y \in \mathcal{Y}_\lambda$.

Definition 3. *A PRF scheme* PRF *is secure if for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}^{\mathsf{Eval}(k,\cdot)}(1^\lambda) = 1 \ : \ k \leftarrow \mathsf{KeyGen}(1^\lambda)] - \Pr[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1 \ : \ R \leftarrow \mathcal{U}_\lambda]| \leq \text{negl}(\lambda)$$

where $\mathcal{U}_\lambda$ *is the set of all functions from* $\mathcal{X}_\lambda$ *to* $\mathcal{Y}_\lambda$.

3.3 Garbling Scheme

A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\{\mathcal{C}_\lambda\}_\lambda$ consists of the following algorithms.

- $\mathsf{Garble}(1^\lambda, C)$: The garbling algorithm is a randomized algorithm that takes as input the security parameter 1^λ and a circuit $C \in \mathcal{C}_\lambda$ such that $C : \{0,1\}^n \to \{0,1\}$ and outputs a garbled circuit $\hat{C}$ and a set of labels $\{\mathsf{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}$.
- $\mathsf{Eval}(\hat{C}, \{\mathsf{lab}_i\}_{i \in [n]})$: The evaluation algorithm takes as input a garbled circuit $\hat{C}$ and a set of n labels $\{\mathsf{lab}_i\}_{i \in [n]}$ and outputs $y \in \{0,1\}$.

Correctness. For correctness of a garbling scheme GC for a class of circuits $\{\mathcal{C}_\lambda\}_\lambda$, we require that for all $\lambda \in \mathbb{N}, C \in \mathcal{C}_\lambda$, $x \in \{0,1\}^n$,

$$\mathsf{Eval}(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i \in [n]}) = C(x)$$

where $(\hat{C}, \{\mathsf{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$.

Security. For security, we define simulation based security [Yao86, BHR12].

Definition 4. *A garbling scheme* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ *for a class of circuits* $\{\mathcal{C}_\lambda\}_\lambda$ *is said to be secure if there exists a PPT algorithm* Sim *such that for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds.*

$$\left| \Pr\left[\mathcal{A}_2(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i \in [n]}, \mathsf{aux}) = 1 : \begin{array}{c} (C, x, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda), \\ (\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i \in [n]}) \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^{|C|}, C(x)) \end{array} \right] \right.$$
$$\left. - \Pr\left[\mathcal{A}_2(\hat{C}, \{\mathsf{lab}_{i,x_i}\}_{i \in [n]}, \mathsf{aux}) = 1 : \begin{array}{c} (C, x, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda), \\ (\hat{C}, \{\mathsf{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)) \end{array} \right] \right| \leq \frac{1}{2} + \text{negl}(\lambda).$$

3.4 Incompressible Secret Key Encryption

An incompressible secret key encryption scheme $\mathsf{IncSKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\{\mathcal{M}_\lambda\}_\lambda$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, 1^S, 1^n)$: The setup algorithm is a randomized algorithm that takes as input the security parameter λ, a parameter 1^S, the length of the message 1^n and outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{sk}, m)$: The encryption algorithm is a randomized algorithm that takes as input a secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \mathcal{M}_\lambda$ or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}, n \in \mathbb{N}, m \in \mathcal{M}_\lambda$ and $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S)$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m] = 1$$

where the probability is over the random bits used in the encryption algorithm.

Incompressible SKE Security. Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S and the length of the message 1^n. The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^S, 1^n)$.
- **Challenge Phase:** $\mathcal{A}_1$ outputs a message m, along with an auxiliary information aux. The challenger randomly chooses $b \in \{0, 1\}$. If $b = 0$, it samples a truly random string ct^*. Else, it computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, m)$ and sends it to $\mathcal{A}_1$.[11]
- **First Response Phase:** $\mathcal{A}_1$ computes a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:** $\mathcal{A}_2$ receives $(\mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 5. *An SKE scheme is said to be incompressible secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}, S \in \mathbb{N}, n \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

CPA-SKE Security. Consider the following experiment with an adversary $\mathcal{A}$ where Setup algorithm takes only 1^λ and 1^n as input.

- **Initialization Phase:** The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$.

[11] In the incompressible security definition presented in prior works [BDD22b, GWZ23], the adversary sends two messages m_0, m_1 and receives an encryption of one of these message. Note that this is a weaker notion, which is implied by the incompressible security notion defined in this paper.

- **Pre-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Challenge Phase:** $\mathcal{A}$ outputs a message m^*. The challenger randomly chooses $b \in \{0,1\}$. If $b = 0$, it samples a truly random string ct^*. Else, it computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, m^*)$ and sends it to $\mathcal{A}$.[12]
- **Post-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query m, the challenger computes $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}, m)$ and returns ct to Adv.
- **Response Phase:** $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the experiment if $b = b'$.

Definition 6. *An SKE scheme is said to be secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}, n \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Theorem 7 ([Dzi06])**.** *Assuming the existence of one-way functions, there exists an incompressible SKE scheme with $\mathsf{poly}(\lambda)$ secret-key size and $S + n + \mathsf{poly}(\lambda)$ ciphertext size where n is the size of the message and S is the compressibility parameter. The depth of the decryption circuit is $\mathsf{poly}(\lambda, \log(S, n))$.*

3.5 Functional Encryption

A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the function space $\{\mathcal{F}_n : \mathcal{X}_n \to \mathcal{Y}_n\}_n$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm is a randomized algorithm that takes as input the security parameter 1^λ and an index 1^n and outputs a master public key mpk and a secret key msk.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: The key generation algorithm is a randomized algorithm that takes as input the master secret msk and a function $f \in \mathcal{F}_n$ and outputs a secret sk_f.
- $\mathsf{Enc}(\mathsf{mpk}, m)$: The encryption algorithm is a randomized algorithm that takes as input a public key mpk and a message $m \in \mathcal{X}_n$ and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: The decryption algorithm takes as input a secret key sk_f and a ciphertext ct and outputs either a $y \in \mathcal{Y}_n$ or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}, n \in \mathbb{N}, m \in \mathcal{X}_n, f \in \mathcal{F}_n$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$,

$$\Pr[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, m)) = f(m)] = 1$$

where the probability is over the random bits used in the encryption and key generation algorithm.

[12]Note that this security notion implies the standard indistinguishability security notion where the adversary sends two messages m_0, m_1 and receives an encryption of one of the messages.

Adaptive IND-based Security. Consider the following experiment with an adversary $\mathcal{A}$.

- **Initialization Phase:** The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query f, the challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ outputs two message m_0, m_1. If there exists a function f queried by $\mathcal{A}$ such that $f(m_0) \neq f(m_1)$, the challenger aborts the game. Else, it randomly chooses $b \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{mpk}, m_b)$ and sends it to $\mathcal{A}$.
- **Post-Challenge Query Phase:** $\mathcal{A}$ is allowed to make polynomially many queries. For each query f, if $f(m_0) \neq f(m_1)$, the challenger sends $\perp$. Else, computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 8. *An FE scheme satisfies adaptive indistinguishability-based security if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

If the FE is secure against an unbounded number of queries from the adversary, then we say that the scheme is collision-resistant.

A relaxed notion of the FE indistinguishability-based security is the well-known selective security model. An FE scheme is said to be *selectively* secure if it is secure against PPT adversaries $\mathcal{A}$ that do not make any key queries during the pre-challenge query phase, and have to submit the challenge message before obtaining the public parameters.

The following two theorems addresses the optimal ciphertext-rate and secret-key size FE schemes for both adaptive and selective settings. It is important to note that in these FE scheme, the decryption algorithm requires the complete description of the function f in addition to the secret key sk_f to perform decryption.

Theorem 9 ([JLL23])**.** *Assuming selectively secure FE for circuits, there exists an adaptively secure FE such that*

$$|\mathsf{mpk}| = O_\lambda(1), \quad |\mathsf{sk}_f| = O_\lambda(1)^{13}, \quad |\mathsf{ct}| = 2|x| + O_\lambda(1)$$

where $O_\lambda(\cdot)$ hides factors of $\mathsf{poly}(\lambda)$*, λ is the security parameter,* ct *is a ciphertext and* sk_f *is a secret key for the function f, x is any element from the input space of f and* mpk *is the master public key generated by the scheme.*

Theorem 10 ([JLL23])**.** *Assuming selectively secure FE for circuits, there exists an adaptively secure FE such that*

$$|\mathsf{mpk}| = O_\lambda(1), \quad |\mathsf{sk}_f| = O_\lambda(1)^{14}, \quad |\mathsf{ct}| = |x| + O_\lambda(1)$$

where $O_\lambda(\cdot)$ hides factors of $\mathsf{poly}(\lambda)$*, λ is the security parameter,* ct *is a ciphertext and* sk_f *is a secret key for the function f that outputs a single bit, x is any element from the input space of f and* mpk *is the master public key generated by the scheme.*

[13] Size is independent of the function description length $|f|$, but the decryption algorithm requires the entire description of f.

[14] The decryption algorithm requires the entire description of f.

Simulation security. Consider the following real and simulated experiment with an adversary $\mathcal{A}$ where Sim is a PPT simulator.

Real Experiment

- **Initialization Phase:** The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ outputs a message m and a function f. The challenger computes a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m)$ and $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and sends $(\mathsf{ct}, \mathsf{sk}_f)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

Simulated Experiment

- **Initialization Phase:** The simulator runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends mpk to $\mathcal{A}$.
- **Challenge Phase:** $\mathcal{A}$ outputs a message m and a function f. The simulator computes $(\mathsf{ct}, \mathsf{sk}_f) \leftarrow \mathsf{Sim}(\mathsf{mpk}, f, f(x), 1^{|x|})$ and sends $(\mathsf{ct}, \mathsf{sk}_f)$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

Definition 11. *An FE scheme is said to be **single-key** simulation secure if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator* Sim *and a negligible function* $\mathrm{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}, n \in \mathbb{N}$,*

$$\Big| \Pr[\mathcal{A} \textit{ outputs 0 in the real experiment}] - \Pr[\mathcal{A} \textit{ outputs 0 in the simulated experiment}] \Big| \leq \mathrm{negl}(\lambda)$$

Theorem 12 ([GKP+13])**.** *Assuming the hardness of* LWE*, there exists a single-key simulation secure* FE *for a class of circuits $\{\mathcal{C}_{n,d}\}_{n,d}$ that takes input of size n, outputs a single bit and has depth d such that*

$$|\mathsf{ct}| = O_\lambda(n \cdot d^2), \quad |\mathsf{sk}_f| = O_\lambda(d^2), \quad |mpk| = O_\lambda(n \cdot d^2)$$

where $O_\lambda(\cdot)$ hides factors of $\mathsf{poly}(\lambda)$*, λ is the security parameter,* ct *is a ciphertext and* sk_f *is a secret key for the circuit $f \in \mathcal{C}_{n,d}$ and* mpk *is the master public key generated by the scheme.*

Corollary 13. *Assuming the hardness of* LWE*, there exists a single-key simulation secure* FE *for a class of circuits $\{\mathcal{C}_{n,d}\}_{n,d}$ that takes input of size n, outputs m-bits and has depth d such that*

$$|\mathsf{ct}| = O_\lambda(m \cdot n \cdot d^2), \quad |\mathsf{sk}_f| = O_\lambda(m \cdot d^2), \quad |mpk| = O_\lambda(m \cdot n \cdot d^2)$$

where $O_\lambda(\cdot)$ hides factors of $\mathsf{poly}(\lambda)$*, λ is the security parameter,* ct *is a ciphertext and* sk_f *is a secret key for the circuit $f \in \mathcal{C}_{n,d}$ and* mpk *is the master public key generated by the scheme.*

Specializing FE. An attribute based encryption scheme (ABE) is a function encryption scheme where the message space is $\{\mathcal{M}_n \times \mathcal{X}_n\}_n$, i.e., it is a tuple of a message and an attribute and the function class is a set of function $\{\mathcal{F}_n\}_n$ such that any function $f_C \in \mathcal{F}_n$ is associated with a predicate function C with the following definition.

$$\mathcal{F}_C(m, x) = \begin{cases} m & if \;\; C(x) = 1 \\ \perp & if \;\; C(x) = 0 \end{cases}$$

In the security experiment, the adversary has to output m_0, m_1, x^* in the challenge phase with the restriction that it has never queried a key for a function f_C such that $C(x^*) = 1$.

Theorem 14 ([BGG+14]). *Assuming the hardness of* LWE*, there exists an selectively secure ABE scheme for any family of circuits $\{\mathcal{C}_{d,\ell}\}_{d,\ell}$ that has ℓ-length input and d-depth satisfying*

$$|\mathsf{ct}| = O_\lambda(\ell \cdot d^2), \qquad |\mathsf{sk}_f| = O_\lambda(d^2), \qquad |mpk| = O_\lambda(\ell \cdot d^2)$$

where $O_\lambda(\cdot)$ hides factors of $\mathsf{poly}(\lambda)$*, λ is the security parameter,* ct *is a ciphertext and* sk_f *is a secret key for the predicate $f \in \mathcal{C}_{d,\ell}$ and* mpk *is the master public key generated by the scheme.*

An identity based encryption scheme (IBE) is a attribute based encryption scheme where the message space is $\{\mathcal{M}_\lambda \times \mathcal{ID}_\lambda\}_\lambda$, i.e., it is a tuple of a message and an identity and the function class is a set of function $\{\mathcal{F}_\lambda\}_\lambda$ such that any function $f_{\mathsf{id}} \in \mathcal{F}_\lambda$ is associated with an identity id with the following definition.

$$\mathcal{F}_{\mathsf{id}}(m, \mathsf{id}') = \begin{cases} m & if \ \ \mathsf{id} = \mathsf{id}' \\ \perp & if \ \ \mathsf{id} \neq \mathsf{id}' \end{cases}$$

In the security experiment, the adversary has to output m_0, m_1, id^* in the challenge phase with the restriction that it has never queried a key for id^*.

4 Incompressible Functional Encryption: Definitions

In this section, we will proceed to define the incompressible version of the security game for FE scheme where Setup takes an additional input 1^S. Similar to the incompressible SKE schemes, we will consider two adversaries $\mathcal{A}_1, \mathcal{A}_2$. The first adversary $\mathcal{A}_1$ will be provided with the complete challenge ciphertext and produce a compressed version of it. The second adversary $\mathcal{A}_2$ is provided with the master public key, compressed challenge ciphertext which was created by $\mathcal{A}_1$ and certain secret keys.

Definition 15. (*Incompressible FE Security*). *Let* FE = (Setup, KeyGen, Enc, Dec) *be an FE scheme in which the setup algorithm takes an additional parameter 1^S as input. Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.*

- ***Initialization Phase:*** *$\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the state size 1^S and 1^n. The challenger runs* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^S, 1^n)$ *and sends* mpk *to $\mathcal{A}_1$.*
- ***Pre-Challenge Query Phase:*** *In this phase, $\mathcal{A}_1$ is allowed to make polynomially many key queries. For each query f sent to the challenger, the challenger computes* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ *and returns* sk_f *to $\mathcal{A}_1$.*
- ***Challenge Phase:*** *$\mathcal{A}_1$ outputs two messages m_0, m_1, along with an auxiliary information aux. If there exists a function f queried by $\mathcal{A}_1$ such that $f(m_0) \neq f(m_1)$, the challenger aborts the game. Else, it randomly chooses $b \in \{0,1\}$ and computes a ciphertext* $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{mpk}, m_b)$ *and sends it to $\mathcal{A}_1$.*
- ***Post-Challenge Query Phase:*** *This is similar to the pre-challenge query phase. The adversary $\mathcal{A}_1$ is allowed to send polynomially many key queries. For each query f, if $f(m_0) \neq f(m_1)$, the challenger sends $\perp$. Else, computes* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ *and returns* sk_f *to $\mathcal{A}_1$.*
- ***First Response Phase:*** *$\mathcal{A}_1$ computes a state st such that $|st| \leq S$.*

- ***Second Response Phase:*** *$\mathcal{A}_2$ receives (mpk, aux, st) and can make the following query.*
    - ***Key Query Phase:*** *In this phase, $\mathcal{A}_2$ is allowed to make* ***a single distinguishing key*** *query but polynomially many queries for non-distinguishing keys. The challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ and returns sk_f to $\mathcal{A}_2$.*

Finally, $\mathcal{A}_2$ outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

An FE scheme is said to be incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

We consider two stronger variants of the above security - strongly incompressible and semi-strongly incompressible. In the strongly incompressible version, $\mathcal{A}_2$ is provided with the master secret msk whereas in the semi-strongly incompressible version, it is allowed to make polynomially many distinguishing key queries.

Definition 16. (*Strongly Incompressible* FE *Security*). *An FE scheme is said to be strongly incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the Incompressible } \mathsf{FE} \textit{ experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

provided the second adversary $\mathcal{A}_2$ is also given the ***master secret key*** msk *at the beginning of Second Response Phase.*

Definition 17. (*Semi-Strongly Incompressible* FE *Security*). *An FE scheme is said to be semi-strongly incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the Incompressible } \mathsf{FE} \textit{ experiment}] \leq \frac{1}{2} + \text{negl}(\lambda)$$

provided the second adversary $\mathcal{A}_2$ is allowed to make ***polynomially many distinguishing key queries*** *in the Key Query Phase.*

We can consider similar incompressible security notions for IBE and ABE schemes.

5 Rate-$\frac{1}{2}$ Incompressible ABE with Short Keys

In this section, we present an *regular incompressible* ABE scheme with ct-rate-$\frac{1}{2}$, using a regular ABE scheme, a single-key simulation secure secret key FE scheme along with other primitives.

Our Construction. Our ABE construction supports attribute spaces $\{\mathcal{X}_n\}_n$ using the following primitives.

- $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ is a PRF scheme with keys of length λ and output space $\{0,1\}$.

- $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^\ell \to \{0,1\}^\lambda$ is a strong average-min entropy randomness extractor where $d = d(\lambda)$ and $\ell = \ell(\lambda, S)$.

- $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ is a functional encryption scheme with message spaces $\{\mathcal{N}_n\}_n$ and function spaces $\{\mathcal{F}_n\}_n$ that contains the extractor circuit Ext.

- $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ is a garbling scheme for Boolean circuits with label spaces $\{\mathcal{M}_n\}_n$.

- $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ is an ABE scheme that supports attribute spaces $\{\tilde{\mathcal{X}}_n\}_n$, predicate classes $\{\mathcal{C}_n : \tilde{\mathcal{X}}_n \to \{0,1\}\}_n$ and message spaces $\{\mathcal{M}_n\}_n$.

Let $\kappa = \lceil \log(d+\lambda) \rceil$ and $\kappa' = \kappa + 1$. Let $\tilde{n} = \tilde{n}(\lambda, n)$ be the lexicographically smallest functionality index such that any element $(x, (i,b)) \in \mathcal{X}_n \times \{0,1\}^{\kappa'}$ can be uniquely represented in $\tilde{\mathcal{X}}_{\tilde{n}}$. Let $\hat{s} = \hat{s}(\lambda, S)$ be the lexicographically smallest functionality index such that every function $f_R(x,y) = \mathsf{Ext}_x(R) \oplus y$ where $R \in \{0,1\}^\ell, x \in \{0,1\}^d , y \in \{0,1\}^\lambda$ can be uniquely represented in $\mathcal{F}_{\hat{s}}$. We describe the algorithms for our incompressible ABE scheme below.

- $\mathsf{Setup}(1^\lambda, 1^S, 1^n)$: The setup algorithm runs ABE setup algorithm to generate a master public and secret key $(\mathsf{abe.mpk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\tilde{n}})$. It sets $\mathsf{mpk} = (\mathsf{abe.mpk}, S)$ and $\mathsf{msk} = \mathsf{abe.msk}$.

- $\mathsf{KeyGen}(\mathsf{msk}, F)$: Let $\mathsf{msk} = \mathsf{abe.msk}$ and $F : \mathcal{X}_n \to \{0,1\}$ be a predicate function. The key generation algorithm first randomly generates $(k_F, w_F) \in \{0,1\}^{d+\lambda}$. Let $\hat{F}$ be a circuit that has $(F, (k_F, w_F))$ hardwired, takes as input $(x, (i,b)) \in \mathcal{X}_n \times \{0,1\}^{\kappa'}$, and outputs 1 if and only if $(F(x) = 1) \wedge (b = (k_F || w_F)[i])$. The algorithm generate an ABE key for $\hat{F}$ by computing $\mathsf{abe.sk}_F \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F})$. It returns $(k_F, w_F, \mathsf{abe.sk}_F)$.

- $\mathsf{Enc}(\mathsf{mpk}, m, x)$: Let $\mathsf{mpk} = (\mathsf{abe.mpk}, S)$ and $x \in \mathcal{X}_n$. The encryption algorithm generates a fresh master secret key of the functional encryption scheme $\mathsf{fe.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$, a PRF key $\mathsf{prf.key} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$ and randomly generates two strings $R \in \{0,1\}^\ell$ and r_{Enc} which will be used as the random bits in an FE encryption.

  Let $C_{\mathsf{fe.msk}', m', r'}$ be a circuit that has $\mathsf{fe.msk}', m'$ and r' hardwired and takes as input (k, w) and outputs $\mathsf{FE.Enc}(\mathsf{fe.msk}', (k, w \oplus m'); r')$. The encryption algorithm constructs the circuit $C_{\mathsf{fe.msk}, \mathsf{prf.key}, r_{\mathsf{Enc}}}$ and generates a garbled circuit and its corresponding labels by computing $(\hat{C}, \{\mathsf{lab}_{i,b}\}_{i \in [\kappa], b \in \{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{\mathsf{fe.msk}, \mathsf{prf.key}, r_{\mathsf{Enc}}})$.

  For all $i \in [\kappa], b \in \{0,1\}$, it encrypts each label $\mathsf{lab}_{i,b}$ with the attribute $(x, (i,b))$ by computing $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,b}, (x, (i,b)))$.

  It then generates an FE key $\mathsf{fe.sk} \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, D_R)$ where $D_{R'}$ takes as input (k, v) and outputs $\mathsf{Ext}_k(R') \oplus v$.

  It generates $t_i = \mathsf{PRF.Eval}(\mathsf{prf.key}, i)), \forall i \in [|m|]$ and sets $t = t_1 || t_2 || \ldots || t_m$. It finally outputs $(\{\mathsf{abe.ct}_{i,b}\}_{i \in [\kappa], b \in \{0,1\}}, \hat{C}, \mathsf{fe.sk}, R, t \oplus m)$.

- $\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct})$: Let $\mathsf{sk}_F = (k_F, w_F, \mathsf{abe.sk}_F)$ and $\mathsf{ct} = (\{\mathsf{abe.ct}_{i,b}\}_{i \in [\kappa], b \in \{0,1\}}, \hat{C}, \mathsf{fe.sk}, R, z)$. The decryption algorithm first decrypts ciphertexts $\mathsf{abe.ct}_{i, (k_F || w_F)[i]}$, for all $i \in [\kappa]$ using $\mathsf{abe.sk}_F$ and obtains the labels $\mathsf{lab}_{i, (k_F || w_F)[i]} = \mathsf{ABE.Dec}(\mathsf{abe.sk}_F, \mathsf{abe.ct}_{i, (k_F || w_F)[i]})$.

Then, it evaluates the garbled circuit $\hat{C}$ using labels $\left\{\mathsf{lab}_{i,(k_F||w_F)[i]}\right\}_{i\in[\kappa]}$ to obtain $\mathsf{fe.ct} = \mathsf{FE.Enc}(\mathsf{fe.msk}, (k_F, w_F \oplus \mathsf{prf.key}); r_{\mathsf{Enc}})$. It decrypts fe.ct to obtain $y = \mathsf{FE.Dec}(\mathsf{fe.sk}, \mathsf{fe.ct})$. It computes $\mathsf{prf.key} = \mathsf{Ext}_{k_F}(R) \oplus w_f \oplus y$. It generates $t_i = \mathsf{PRF.Eval}(\mathsf{prf.key}, i)), \forall i \in [|m|]$ and sets $t = t_1||t_2||\dots||t_m$. Finally, it outputs $m = z \oplus t$.

Correctness. An encryption of a message m using attribute x in the above scheme consists of $(\{\mathsf{abe.ct}_{i,b}\}_{i\in[\kappa],b\in\{0,1\}}, \hat{C}, \mathsf{fe.sk}, R, z)$ where $(\hat{C}, \{\mathsf{lab}_{i,b}\}_{i\in[\kappa],b\in\{0,1\}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{\mathsf{fe.msk},\mathsf{prf.key},r_{\mathsf{Enc}}})$, $\mathsf{abe.ct}_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,b}, (x, (i, b)))$, $\mathsf{fe.sk} \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, D_R)$ where D_R takes as input (k, v) and outputs $\mathsf{Ext}_k(R) \oplus v$. Also, $z = t \oplus m$ where $t = t_1||t_2||\dots||t_m$ where $t_i = \mathsf{PRF.Eval}(\mathsf{prf.key}, i))$.

From the correctness of the ABE scheme, we have $\mathsf{lab}_{i,(k_F||w_F)[i]} = \mathsf{ABE.Dec}(\mathsf{abe.sk}_F, \mathsf{abe.ct}_{i,(k_F||w_F)[i]})$. From the correctness of the garbling scheme, the output of evaluating the garbled circuit $\tilde{C}$ on the labels $\{\mathsf{lab}_{i,(k_F||w_F)[i]}\}_i$ is $\mathsf{fe.ct} = \mathsf{FE.Enc}(\mathsf{fe.msk}, (k_F, w_F \oplus \mathsf{prf.key}); r_{\mathsf{Enc}})$. From the correctness of the FE scheme, we get $y = \mathsf{FE.Dec}(\mathsf{fe.sk}, \mathsf{fe.ct})$ where $y = \mathsf{Ext}_{k_F}(R) \oplus w_F \oplus \mathsf{prf.key}$. Therefore, the PRF key can be obtained using y, k_F, R, w_F. Finally, $m = t \oplus z$ where $t = t_1||t_2||\dots||t_m$ where $t_i = \mathsf{PRF.Eval}(\mathsf{prf.key}, i)$.

Size of the master public key. The master public key of our scheme is $\mathsf{mpk} = (\mathsf{abe.mpk}, S)$. Therefore, the size is $|\mathsf{abe.mpk}| + \log(S)$.

Size of the ciphertext. The size of a ciphertext is $|\hat{C}| + 2 \cdot (|k| + |w|) \cdot |\mathsf{abe.ct}| + |R| + |\mathsf{fe.sk}| + |\mathsf{ske.ct}|$. Recall that k is the seed to the extractor and w has the length equal to the output length of the extractor which has the same length as the PRF key. The length of the PRF key prf.key is λ. Using Theorem 2, we have $k = \mathsf{poly}(\lambda)$, $|w| = \lambda$ and $|R| = S + \mathsf{poly}(\lambda)$.

$\hat{C}$ is a garbled circuit of $\mathsf{FE.Enc}(\mathsf{fe.msk}', (\cdot, \cdot); r')$ that takes as input two $\mathsf{poly}(\lambda)$ bit strings. Hence, we have $|\hat{C}| = \mathsf{poly}(\lambda)$. Adding all the values, we get that the size of the entire ciphertext is $\mathsf{poly}(\lambda) \cdot |\mathsf{abe.ct}| + |\mathsf{fe.sk}| + |m| + S$. Therefore, the rate of our scheme is

$$\frac{|m|}{|\mathsf{ct}|} = \frac{|m|}{\mathsf{poly}(\lambda) \cdot |\mathsf{abe.ct}| + |\mathsf{fe.sk}| + |m| + S} = \frac{1}{\frac{\mathsf{poly}(\lambda) \cdot |\mathsf{abe.ct}| + |\mathsf{fe.sk}|}{|m|} + 1 + \frac{S}{|m|}}$$

Size of the secret key. A secret key in our scheme is of the form $(k_F, w_F, \mathsf{abe.sk}_F)$. Therefore, its size is $|\mathsf{abe.sk}| + \mathsf{poly}(\lambda)$.

Theorem 18. *Assuming* $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ *is an adaptively secure ABE scheme,* $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ *is a secure garbling scheme,* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *is a 1-key simulation secure functional encryption scheme,* $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ *is a secure PRF scheme and* Ext *is a strong average-min entropy randomness extractor, then the above construction is an incompressible ABE scheme. Also,*

$$|\mathsf{mpk}| = |\mathsf{abe.mpk}| + O(\lambda), \quad |\mathsf{sk}| = |\mathsf{abe.sk}| + \mathsf{poly}(\lambda)$$

$$|\mathsf{ct}| = \mathsf{poly}(\lambda) \cdot |\mathsf{abe.ct}| + |\mathsf{fe.sk}| + |m| + |S|$$

Proof-Sketch: The proof proceeds via a sequence of hybrid experiments.

- H_0: This corresponds to the adaptive incompressibility security game. The adversary first receives the master public key mpk. It queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. Then, it sends its challenge messages m_0, m_1 and receives the challenge ciphertext which is encryption of m_b. Again, the adversary queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. After compressing its state, it queries for polynomially many non-distinguishing functions but only a single *distinguishing* function. Upon receiving the secret keys, it must guess which message was encrypted.

- H_1 : In this hybrid, the challenger modifies the challenge ciphertext according to the *distinguishing* key. Specifically, it generates (k^*, w^*) in the challenge phase and will use this to generate the distinguishing key. It then encrypts $\mathbf{0}$ instead of $\mathsf{lab}_{i,1-(k^*,w^*)[i]}$ with attributes $(x^*, i, 1-(k^*, w^*)[i])$ to generate $\mathsf{abe.ct}^*_{i,1-(k^*,w^*)[i]}$.

  Indistinguishability of H_0 and H_1 follows from standard indistinguishability-based security of ABE scheme. This is because the attacker does not gain access to any ABE key that can decrypt these ciphertext. For any non-distinguishing key for F, i.e., $F(x^*) = 0$, the secret key cannot decrypt any $\mathsf{abe.ct}^*_{i,1-(k^*,w^*)[i]}$ because the first component of the attribute is x^*. Whereas, the distinguishing key for F^* generated in the second phase, the key can decrypt a ciphertext if its attribute is of the form $(x^*, i, (k^*, w^*)[i])$ which is not the case for $\mathsf{abe.ct}^*_{i,1-(k^*,w^*)[i]}$.

- H_2 : In this hybrid game, the challenger uses the simulator to generate a simulated garbled circuit and the partial set of labels in the challenge phase. That is, it generates $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,(k_{F^*},w_{F^*})[i]}\right\}\right)$ $\leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\kappa, 1^{|C_{\mathsf{fe.msk},m,r}|}, \mathsf{fe.ct}^*)$.

  Note that H_1 and H_2 are indistinguishable due to the simulation security of the garbling scheme.

- H_3 : In this hybrid, the challenger generates a simulated FE ciphertext using $D_{R^*}(k^*, w^* \oplus \mathsf{prf.key})$ rather than encrypting $(k^*, w^* \oplus \mathsf{prf.key})$ using the real encryption algorithm.

  Indistinguishability of H_2 and H_3 follows from standard simulation-based security of FE scheme.

- H_4 : In this hybrid, the challenger generates the simulated FE ciphertext using $D_{R^*}(k^*, w^* \oplus \mathbf{0})$ instead of $D_{R^*}(k^*, w^* \oplus \mathsf{prf.key})$.

  Note that H_3 and H_4 are indistinguishable due to the incompressible indistinguishability security of Dziembowki's scheme. In the Dziembowski's scheme, a ciphertext is of the form $(R^*, \mathsf{Ext}_{k^*}(R^*) \oplus w^* \oplus m^*)$ where m^* is the message and (k^*, w^*) is the secret key. Recall that part of the ciphertext of our ABE scheme is the FE key for D_{R^*} where D_{R^*} is a circuit that takes as input (k, w) and outputs $\mathsf{Ext}_k(R^*) \oplus w$.

  To generate the simulated FE ciphertext, we require $D_{R^*}(k^*, w^* \oplus \mathbf{0})$ (or $D_{R^*}(k^*, w^* \oplus \mathsf{prf.key})$). So, using the Dziembowski's ciphertext, we generate both $\mathsf{fe.sk}^*$ and $\mathsf{fe.ct}^*$. To generate the distinguishing key in the second phase, (k^*, w^*) is provided by the challenger in the incompressible security game.

- H_5 : In this hybrid, the challenger randomly samples t^* instead of using the PRF outputs. The indistinguishability of H_0 and H_1 follows from pseudorandom property of the PRF scheme.

Finally, note that in H_5, the bit b is only present in the challenge ciphertext in the form of $t^* \oplus m_b$ where t^* is a truly random string that is not utilized anywhere else in the game. Therefore, $t^* \oplus m_b$ is also a truly random string, make it impossible for an adversary to guess b with non-negligible probability.

The proof for the above theorem is provided in Appendix C.

Using Theorem 13 and Theorem 14, we get the following corollary.

Corollary 19. *Assuming the hardness of* LWE *problem, the above construction is a selective incompressible ABE scheme for predicate classes with circuit of depth D with the following parameters.*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}| = \mathsf{poly}(\lambda) \cdot D, \quad |\mathsf{ct}| = \mathsf{poly}(\lambda) \cdot (D + \log(|m|)) + |m| + |S|$$

Using a standard complexity leveraging argument (for instance, see [BB11]), we can obtain an adaptively secure version of the ABE scheme in Theorem 14 which results in the following corollary.

Corollary 20. *Assuming the sub-exponential hardness of* LWE *problem, the above construction is an adaptive incompressible ABE scheme for predicate classes with circuit of depth D with the following parameters.*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}| = \mathsf{poly}(\lambda) \cdot D, \quad |\mathsf{ct}| = \mathsf{poly}(\lambda) \cdot (D + \log(|m|)) + |m| + |S|$$

6 Rate-$\frac{1}{2}$ Incompressible FE with Short Keys and Semi-Strong Security

In this section, we present a *semi-strong incompressible* FE scheme, using a regular FE scheme along with an incompressible SKE and regular SKE scheme. If the underlying FE has ct-rate of $r \in [0, 1]$, then our incompressible FE scheme has an $\frac{r}{2}$ ct-rate.

Our Construction. Our FE scheme supports functions from the class $\{\mathcal{F}_n : \mathcal{N}_n \to \{0, 1\}\}_n$ such that any function in this class has a $\mathsf{polylog}(|\mathcal{N}_n|)$ depth circuit using the following primitives. Throughout S denotes the (in)compressibility parameter.

- $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible SKE. We define $\alpha = \alpha(\lambda, S)$ as the length of the secret key of this scheme and $\beta = \beta(\lambda, S)$ as the length of the ciphertext for a message of length λ encrypted using this scheme.
- $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be an SKE scheme with secret keys of length λ.
- $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be an FE scheme with message spaces $\{\tilde{\mathcal{N}}_n\}_n$ and function space $\{\tilde{\mathcal{F}}_n\}_n$ that contains the decryption circuits of IncSKE and SKE.

Let $\tilde{n} = \tilde{n}(\lambda, n, S)$ be the lexicographically smallest functionality index such that $\tilde{\mathcal{F}}_{\tilde{n}}$ contains $\mathcal{F}_n$, the decryption circuit of SKE and IncSKE. We describe the algorithms for our incompressible FE scheme below.

- $\mathsf{Setup}(1^\lambda, 1^S, 1^n)$: The setup algorithm samples a master public key and a master secret key of the FE scheme by computing $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$. It generates a secret key of the incompressible SKE scheme by computing $\mathsf{inc.sk} \leftarrow \mathsf{IncSKE.Setup}(1^\lambda, 1^S, 1^\lambda)$ and two secret keys of the SKE scheme by computing $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and $\mathsf{ske.sk}' \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. It sets $\mathsf{mpk} = \mathsf{fe.mpk}$ and $\mathsf{msk} = (\mathsf{fe.msk}, \mathsf{inc.sk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$.

- $\mathsf{KeyGen}(\mathsf{msk}, f)$: Let $\mathsf{msk} = (\mathsf{fe.msk}, \mathsf{inc.sk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$. The key generation algorithm first computes ciphertext $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^{\alpha}))$ and $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$. Then, it generates a key of the FE scheme by computing $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ where the function $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ is described in Figure 1.

```
Input: Messages $m$, SKE key $\mathsf{ske.sk}$, incompressible ciphertext $\mathsf{inc.ct}$ and a flag $b$
Hardwired: Function $f$, two SKE ciphertexts $\mathsf{ske.ct}, \mathsf{ske.ct}'$
Output: $y$

1. Check if $b = 0$ :
    – If yes, output $y = f(m)$.
2. Otherwise, compute $(flag, \mathsf{inc.sk}) \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$.
3. Check if $flag = 0$ :
    – If yes, output $y = f(m)$.
4. Otherwise, compute $\mathsf{ske.sk}' \leftarrow \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})$.
5. Output $y = \mathsf{SKE.Dec}(\mathsf{ske.sk}', \mathsf{ske.ct}')$.
```

Figure 1: Description of $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$

- $\mathsf{Enc}(\mathsf{mpk}, m)$: Let $\mathsf{mpk} = \mathsf{fe.mpk}$. The encryption algorithm randomly samples $\mathsf{inc.ct} \leftarrow \{0,1\}^{\beta}$ and generates $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m, \perp, \mathsf{inc.ct}, 0))$[15]. It returns $\mathsf{fe.ct}$.

- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: Let $\mathsf{sk}_f = \mathsf{fe.sk}_f$. The decryption algorithm decrypts the ciphertext by computing $m = \mathsf{FE.Dec}(\mathsf{fe.sk}_f, \mathsf{ct})$. It returns m.

Correctness. A ciphertext corresponding to a message m in the above scheme is an encryption of $(m, \perp, \mathsf{inc.ct}, 0)$ using the FE scheme, i.e., $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m, \perp, \mathsf{inc.ct}, 0))$ where $\mathsf{inc.ct}$ is a truly random string.

Consider that a decryptor possesses the secret key sk_f for some function f. The secret key sk_f is a key $\mathsf{fe.sk}_f$ of the underlying FE scheme for the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ which takes as input in $(m, \mathsf{ske.sk}, \mathsf{inc.ct}, b)$ and outputs $f(m)$ if $b = 0$. The decryption algorithm simply computes the decryption algorithm of the underlying FE scheme on the ciphertext ct using $\mathsf{fe.sk}_f$ and returns the output. Since, b is set to 0 in the ciphertext ct, decrypting ct using $\mathsf{fe.sk}_f$ gives $f(m)$ due to the correctness of the underlying FE scheme.

Size of the master public key. The master public key of our scheme is $\mathsf{fe.mpk}$. Therefore, the size is $|\mathsf{fe.mpk}|$.

Size of the ciphertext. Recall that the ciphertext is an FE encryption of $(m, \perp, \mathsf{inc.ct}, 0)$ where the third component has size β. From the definition of β, it is the length of an incompressible ciphertext which is intended encrypt an SKE secret key, therefore, $\beta = S + \mathsf{poly}(\lambda)$ using Dziembowski's construction (see Theorem 7). So, the size of $(m, \perp, \mathsf{inc.ct}, 0)$ is $|m| + S + \mathsf{poly}(\lambda)$. Assuming that FE

[15]By $\perp$, we mean it sets a null secret key.

has ct-rate of r, the size of fe.ct is also $r^{-1} \cdot (|m| + S + \mathsf{poly}(\lambda))$. Therefore, the rate of the proposed scheme is

$$\frac{r \cdot |m|}{(|m| + S + \mathsf{poly}(\lambda))} = \frac{r}{\left(1 + \frac{S}{|m|} + \frac{\mathsf{poly}(\lambda)}{|m|}\right)}$$

Size of the secret keys. The size of the secret key associated with a function f is $|\mathsf{fe.sk}_f|$ where $\mathsf{fe.sk}_f$ is an FE secret key for the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$. To use this secret key, the decryption algorithm would require the entire description of $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ (see Theorem 9 and Theorem 10).

In addition to the description of f and the decryption circuits of SKE and IncSKE, the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ contains two SKE ciphertexts generated during the key generation process. Therefore, $\mathsf{fe.sk}_f$ should contain both an FE secret key for the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ and the two ciphertexts $\mathsf{ske.ct}, \mathsf{ske.ct}'$. The first SKE ciphertext is an encryption of an incompressible SKE scheme's secret key inc.sk along with a bit $flag$, and the second is just a bit y. Using Dzeimbowski's construction (see Theorem 7), we have $|\mathsf{inc.sk}| = \mathsf{poly}(\lambda)$. Therefore, $|\mathsf{fe.sk}_f|$ is the size of an FE secret key associated with $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$, with an additional $\mathsf{poly}(\lambda)$.

Theorem 21. *Assuming* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *is an* ct*-rate-r adaptively (selectively) secure FE scheme,* $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *is a secure SKE scheme against unbounded number of ciphertexts and* $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ *be a secure incompressible SKE scheme, then the above construction is an adaptively (selectively) secure semi-strongly incompressible FE scheme. Also,*

$$|\mathsf{mpk}| = |\mathsf{fe.mpk}|, \quad |\mathsf{sk}| = |\mathsf{fe.sk}|, \quad |\mathsf{ct}| = (|m| + S + \mathsf{poly}(\lambda))/r$$

Proof-Sketch: The proof proceeds via a sequence of hybrid experiments.

- H_0: This corresponds to the adaptive semi-strong incompressibility security game. The adversary first receives the master public key mpk. It queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. Then, it sends its challenge messages m_0, m_1 and receives the challenge ciphertext which is encryption of m_b. Again, the adversary queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. It then compresses its state, and queries for both distinguishing and non-distinguishing functions. Upon receiving the secret keys, it must guess which message was encrypted.

- H_1 : In this hybrid, the challenger keeps the challenge ciphertext same as before, but modifies the *distinguishing* secret keys. For every *distinguishing* key query for some function f, it encrypts $f(m_b)$ instead of 0 to generate $\mathsf{ske.ct}'$.

  Note that H_0 and H_1 are indistinguishable due to SKE security. This is because the attacker does not get access to the SKE secret key, except in the form of ciphertexts as part of functional secret keys. We highlight we know the value $f(m_b)$ because the adversary is allowed to make distinguishing key queries in the second phase only.

- H_2 : In this hybrid, the challenger generates $\mathsf{inc.ct}^*$ by encrypting $\mathsf{ske.sk}'$ instead of sampling a truly random string. The rest of the game proceeds as before.

Indistinguishability of hybrids H_1 and H_2 follows from standard indistinguishability-based security of the incompressible SKE scheme. This is because the secret key $\mathsf{inc.sk}$ is used only during the generation of $\mathsf{inc.ct}^*$.

- H_3 : In this hybrid experiment, the challenger keeps the challenge ciphertext same as before, but modifies the *distinguishing* secret keys. For every *distinguishing* key query for some function f, it encrypts $(1, \mathsf{inc.sk})$ instead of $(0, 0^\alpha)$ to generate $\mathsf{ske.ct}$.

  Note that H_2 and H_3 are indistinguishable due to SKE security. This is because the attacker does not get access to the SKE secret key, except in the form of ciphertexts as part of functional secret keys.

- H_4 : In this hybrid, the challenger generates modifies the challenge ciphertext ct^* by encrypting $(m_0, \mathsf{ske.sk}, \mathsf{inc.ct}^*, 1)$ instead of $(m_b, \perp, \mathsf{inc.ct}^*, 0)$. The rest of the game proceeds as before.

  Indistinguishability of hybrids H_3 and H_4 follows from adaptive indistinguishability-based security of FE, as the output of the function stays the same for all key queries. This is because in H_4, the non-distinguishing keys would decrypt the challenge ciphertext to $f(m_0)$ (since b is set to 1 and $flag = 0$), which is equal to $f(m_b)$. Whereas, the distinguishing keys would decrypt to $f(m_b)$ (since b is set to 1 and $flag = 1$, see Figure 1).

- H_5 : In this hybrid, the challenger reverts to generating $\mathsf{inc.ct}^*$ by sampling a truly random string. The rest of the game proceeds as before.

  Indistinguishability of hybrids H_4 and H_5 follows from the *incompressible* indistinguishability-based security of the incompressible SKE scheme. This is because the information $\mathsf{inc.sk}$ is needed to generate distinguishing keys in the second phase.

  Finally, note that in H_5, the bit b is used only in the second phase while generating distinguishing keys. That is, $\mathsf{ske.ct}'$ is generated by encrypting $f(m_b)$. Therefore, the winning probability for any adversary in G_5 depends on the standard indistinguishability-based security of the incompressible SKE scheme.

The proof for the above theorem is provided in Appendix D.

Using Theorem 9, we have the following corollary.

Corollary 22. *Assuming the existence of selectively secure FE for circuits, there exists a* ct*-rate-*$\frac{1}{4}$ *adaptively secure semi-strongly incompressible FE scheme for circuits. Also,*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}_f| = \mathsf{poly}(\lambda), \quad |\mathsf{ct}| = 2(|m| + S) + \mathsf{poly}(\lambda)$$

Using Theorem 10, we have the following corollary.

Corollary 23. *Assuming the existence of selectively secure FE for circuits, there exists a* ct*-rate-*$\frac{1}{2}$ *selectively secure semi-strongly incompressible FE scheme for circuits. Also,*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}_f| = \mathsf{poly}(\lambda), \quad |\mathsf{ct}| = |m| + S + \mathsf{poly}(\lambda)$$

7 Rate-1 Incompressible FE with Large Keys and Semi-Strong Security

In this section, we present a *selective semi-strong incompressible* FE scheme, using a (regular) public key FE, together with other standard cryptographic primitives. If the underlying FE scheme has ct-rate-1, then our incompressible FE scheme also has ct-rate-1. Although the functional key sizes are larger.

Our construction. For simplicity, we consider boolean-valued functions. To design an incompressible FE supporting function class $\{\mathcal{F}_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$, we use the following primitives. Throughout S denotes the (in)compressibility parameter.

- $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^\ell \to \{0,1\}^\lambda$ be a strong average-min entropy randomness extractor, where $d = d(\lambda)$ and $\ell = \ell(\lambda, S)$.
- $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be an SKE scheme that can encrypt messages of length $n + d + 1$. (Here n is the FE message length and d is the seed length. W.l.o.g., we assume the secret key length to be λ.)
- $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be an FE scheme with message spaces $\{\{0,1\}^n\}_n$ and function space $\{\tilde{\mathcal{F}}_n\}_n$.

  Let $\tilde{n} = \max(n, S) + \lambda + 1$. We require the function class $\tilde{\mathcal{F}}_{\tilde{n}}$ to be such that it contains $\mathcal{F}_n$, and the decryption circuit of SKE and extractor Ext.

We describe the algorithms for our incompressible FE scheme below.

- $\mathsf{Setup}(1^\lambda, 1^S, 1^n)$: Let $\tilde{n} = \max(n, S) + \lambda + 1$. The setup algorithm samples a master public and secret key of the FE scheme by sampling $(\mathsf{fe.msk}, \mathsf{fe.mpk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$. It also samples an SKE secret key $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. It sets $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$ and $\mathsf{mpk} = \mathsf{fe.mpk}$.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: Parse master key as $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$. The key generation algorithm first computes ciphertext $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$. That is, it encrypts $n + d + 1$ zeros. Then, it generates an FE secret key by computing $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$, where C is described in Fig. 2.
- $\mathsf{Enc}(\mathsf{msk}, m)$: The encryption algorithm generates the ciphertext by computing $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{msk}, (m, \perp, 0))$. It outputs $\mathsf{fe.ct}$. (We point out that the message m is padded to be of length S in case $S > n$. Also, by $\perp$, we mean it sets a null secret key.)
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: The decryption algorithm decrypts the ciphertext by computing $m = \mathsf{FE.Dec}(\mathsf{sk}_f, \mathsf{ct})$. It returns m.

Correctness. Follows immediately from the construction.

Size of the master public key. The master public key of our scheme is $\mathsf{fe.mpk}$. Therefore, the size is $|\mathsf{fe.mpk}|$.

Size of the ciphertext. The size of the message encrypted by $\mathsf{FE.Enc}$ is $\tilde{n} = \max(n, S) + \lambda + 1$. Now if the base FE scheme has ct-rate-1, then ciphertext size in our scheme is also $\max(S, n) + \mathsf{poly}(\lambda)$.

```
Input: $x \in \{0,1\}^{\max(n,S)}$, an SKE secret key $\mathsf{ske.sk} \in \{0,1\}^{\lambda}$, bit $\mathsf{flag} \in \{0,1\}$.
Hardwired: Function $f$, an SKE ciphertext $\mathsf{ske.ct}$.
Output: $y \in \{0,1\}$.

1. Check if $\mathsf{flag} = 0$ :
     – If yes, parse $x$ as an $n$-bit message $m$, and output $f(m)$.
       That is, if $S > n$, then $m = x[1:n]$, else $m = x$.
2. Otherwise, compute $(b,s,v) = \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$.
   Here $b \in \{0,1\}$, $s \in \{0,1\}^d$, $v \in \{0,1\}^n$.
3. Check if $b = 0$ :
     – If yes, then output first bit of $s$, i.e. $s[1]$.
4. Otherwise, parse $x$ as an $S$-bit source $R$, and output $f(v \oplus \mathsf{Ext}_s(R))$.
   That is, if $S < n$, then set $R = x[1:S]$, else set $R = x$.
   Then compute message $m = v \oplus \mathsf{Ext}_s(R)$ and output $f(m)$.
```

Figure 2: Description of $C_{f,\mathsf{ske.ct}}$

Size of the secret key. The size of the secret key associated with a function f is $|\mathsf{fe.sk}_f|$ where $\mathsf{fe.sk}_f$ is an FE secret key for the circuit $C_{f,\mathsf{ske.ct}}$. To use this secret key, the decryption algorithm would require the entire description of $C_{f,\mathsf{ske.ct}}$ (see Theorem 10).

In addition to the description of f and the decryption circuit of SKE, the circuit $C_{f,\mathsf{ske.ct}}$ contains an SKE ciphertext generated during the key generation process. So, $\mathsf{fe.sk}_f$ should contain both an FE secret key for the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ and the ciphertext $\mathsf{ske.ct}$. The SKE ciphertext is an encryption of a message of length $n + d + 1$ where d is the seed length of the extractor[16] and n is the length of the message. Therefore, $|\mathsf{fe.sk}_f|$ is equal to $n + d + 1$ plus the size of an FE secret key associated with $C_{f,\mathsf{ske.ct}}$.

Theorem 24. *Assuming* $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *is IND-CPA secure,* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *is a selectively secure FE with* ct*-rate-r, then the above FE construction is a selective semi-strong incompressible FE scheme. Also,*

$$|\mathsf{mpk}| = |\mathsf{fe.mpk}|, \quad |\mathsf{sk}| = |\mathsf{fe.sk}|, \quad |\mathsf{ct}| = (\max(n,S) + \mathsf{poly}(\lambda))/r$$

Proof-Sketch: The proof proceeds via a sequence of hybrid experiments.

- H_0: This corresponds to the selective semi-strong incompressibility security game. The adversary first sends its challenge messages m_0, m_1 and receives the challenge ciphertext which is encryption of m_b. Next, the adversary queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. It then compresses its state, and queries for both distinguishing and non-distinguishing functions. Upon receiving the secret keys, it must guess which message was encrypted.

- H_1 : In this hybrid, the challenger keeps the challenge ciphertext same as before, but modifies the secret keys. During setup, the challenger samples a *single* random string $R \leftarrow \{0,1\}^S$ as

[16] d is equal to $\mathsf{poly}(\lambda)$, see Theorem 2

the extractor source, and a *single* random string $s \leftarrow \{0,1\}^d$ as the extractor seed. Now, instead of using an SKE encryption of $\mathbf{0}$ within each secret key, the challenger does the following:

– For every *non-distinguishing* key query for some function f, it encrypts $(0, f(m_0), \mathbf{0})$. That is, the second message bit is $f(m_0)$ and rest are still all zeros.
– For every *distinguishing* key query for some function f, it sets $v = m_b \oplus \mathsf{Ext}_s(R)$, and computes the SKE ciphertext as to be an encryption of $(1, s, v)$.

Note that H_0 and H_1 are indistinguishable due to SKE security. This is because the attacker does not get access to the SKE secret key, except in the form of ciphertexts as part of functional secret keys. We highlight that since we need to know $f(m_0)$ to answer even a non-distinguishing key query, thus we can only prove selective security of our incompressible FE scheme.

- H_2 : In this hybrid, the challenger encrypts $(R, \mathsf{ske.sk}, 1)$ instead of $(m_b, \perp, 0)$. (Note that here we are overloading notation and consider that there is a fixed deterministic way, e.g. by padding, to write R and m as $\max(n, S)$-bit strings.) The rest of the game proceeds as before.

  Indistinguishability of hybrids H_1 and H_2 follows standard selective indistinguishability-based security of FE, as the output of the function stays the same for all key queries. This is because for non-distinguishing keys we know that $f(m_0) = f(m_b)$, while for non-distinguishing keys, the function output is still $f(m_b)$.

- H_3 : In this hybrid experiment, v is chosen uniformly at random (instead of setting it as $m_b \oplus \mathsf{Ext}_s(R)$. Using the strong extractability guarantee of the extractor, we can argue that H_2 and H_3 are negligibly close.

  Finally, note that the bit b is not used in H_3, and therefore the adversary has no advantage in this experiment.

The detailed proof for the above theorem is provided in Appendix E.

Using Theorem 10, we have the following corollary.

Corollary 25. *Assuming the existence of* ct-*rate*-$\frac{1}{2}$ *selectively secure FE for circuits, there exists a* ct-*rate*-$\frac{1}{2}$ *selectively secure semi-strongly incompressible FE scheme for circuits. Also,*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}_f| = |m| + \mathsf{poly}(\lambda), \quad |\mathsf{ct}| = |m| + S + \mathsf{poly}(\lambda)$$

8 Rate-1 Incompressible FE with Short Keys and Standard Security

In this section, we present a *selective* (*standard*) *incompressible* FE scheme, using a (regular) public key FE, together with other standard cryptographic primitives. If the underlying FE scheme has ct-rate-1, then our incompressible FE scheme also has ct-rate-1. Interestingly, we show that if the underlying FE scheme has short keys, then our resulting incompressible FE scheme has short keys as well. While it might seem to contradict the lower bound [BGK+24], we observe that since we only prove standard incompressibility security (where only one non-distinguishing key gets corrupted) and the functions supported by our FE scheme has single-bit output, thus this does not contradict the result. This is because the lower bound actually states that if an incompressible FE scheme has ct-rate-1, then the size of the functional secret key must grow with the output length of the function, and not necessarily the input/message length.

Our construction. As discussed in the overview, this construction relies on the same ingredients as our FE construction from Section 7. The core difference is that now the functions that we create secret keys for change slightly. Below we describe the algorithms where we make modifications, and $\underline{\text{underline}}$ all the changes.

- $\mathsf{Setup}(1^\lambda, 1^S, 1^n)$: This is the same as in Section 7.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: Parse master key as $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$. The key generation algorithm first computes ciphertext $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, \underline{\mathbf{0}})$. $\underline{\text{That is, it encrypts } d+2 \text{ zeros.}}$ Then, it generates an FE secret key by computing $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C'_{f,\mathsf{ske.ct}})$, where C is described in Figure 3.

Input: $x \in \{0,1\}^{\max(n,S)}$, an SKE secret key $\mathsf{ske.sk} \in \{0,1\}^\lambda$, bit $\mathsf{flag} \in \{0,1\}$.
Hardwired: Function f, an SKE ciphertext $\mathsf{ske.ct}$.
Output: $y \in \{0,1\}$.

```
1. Check if $\mathsf{flag} = 0$ :
      – If yes, parse $x$ as an $n$-bit message $m$, and output $f(m)$.
2. Otherwise, compute $(b, s, v) = \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$.
   Here $b \in \{0,1\}, s \in \{0,1\}^d, \underline{v \in \{0,1\}}$.
3. Check if $b = 0$ :
      – If yes, then output first bit of $s$, i.e. $s[1]$.
4. Otherwise, parse $x$ as an $S$-bit source $R$, and output $\underline{v \oplus \mathsf{Ext}_s(R)}$.
   $\underline{\text{That is, it no longer computes } f \text{ in this case.}}$
```

Figure 3: Description of $C'_{f,\mathsf{ske.ct}}$

- $\mathsf{Enc}(\mathsf{msk}, m), \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: The encryption and decryption algorithms are the same as in Section 7.

Correctness. Follows immediately from the construction.

Size of the master public key and ciphertext. This is the same as in Section 7.

Size of the secret key. The size of the secret key associated with a function f is $|\mathsf{fe.sk}_f|$ where $\mathsf{fe.sk}_f$ is an FE secret key for the circuit $C'_{f,\mathsf{ske.ct}}$. To use this secret key, the decryption algorithm would require the entire description of $C'_{f,\mathsf{ske.ct}}$ (see Theorem 10).

In addition to the description of f and the decryption circuit of SKE, the circuit $C_{f,\mathsf{ske.ct}}$ contains an SKE ciphertext generated during the key generation process. So, $\mathsf{fe.sk}_f$ should contain both an FE secret key for the circuit $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ and the ciphertext $\mathsf{ske.ct}$. The SKE ciphertext is an encryption of a message of length $d+2$ where d is the seed length of the extractor[17]. Therefore, $|\mathsf{fe.sk}_f|$ is the size of an FE secret key associated with $C_{f,\mathsf{ske.ct}}$, with an addition $d+1$.

[17] d is equal to $\mathsf{poly}(\lambda)$, see Theorem 2

Theorem 26. *Assuming* $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ *is IND-CPA secure,* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *is a selectively secure FE with* ct*-rate-*r*, then the above FE construction is a selective (standard) incompressible FE scheme. Also,*

$$|\mathsf{mpk}| = |\mathsf{fe.mpk}|, \quad |\mathsf{sk}| = |\mathsf{fe.sk}|, \quad |\mathsf{ct}| = (\max(n, S) + \mathsf{poly}(\lambda))/r$$

Proof-Sketch: The proof is very similar to the security proof of Theorem 24, except with one change. Since we only prove standard incompressibility of our FE scheme, then we only need to answer a single distinguishing key query. Therefore, rather than programming v as $m_b \oplus \mathsf{Ext}_s(R)$ as in the previous proof, we simply program $v = f(m_b) \oplus \mathsf{Ext}_s(R)$. That is, we only hardwire one function value inside v rather than the full message m_b. Because of this change, we can only prove standard incompressibility security (because we cannot answer more than one non-distinguishing key), but the circuit $C'_{f,\mathsf{ske.ct}}$ (that we have to create a functional key for) now has a succinct description as it only needs f and $\mathsf{ske.ct}$ (which is short anyways). For completeness, we sketch the hybrids below.

- H_0: This corresponds to the selective standard incompressibility security game. The adversary first sends its challenge messages m_0, m_1 and receives the challenge ciphertext which is encryption of m_b. Next, the adversary queries for polynomially many non-distinguishing keys, and receives the corresponding secret keys. It then compresses its state, and queries for one distinguishing function fst and many non-distinguishing functions. Upon receiving the secret keys, it must guess which message was encrypted.
- H_1 : In this hybrid, the challenger keeps the challenge ciphertext same as before, but modifies the secret keys. During setup, the challenger samples a single random string $R \leftarrow \{0,1\}^S$ as the extractor source, and a random string $s \leftarrow \{0,1\}^d$ as the extractor seed. Now, instead of using an SKE encryption of $\mathbf{0}$ within each secret key, the challenger does the following:
  - For every *non-distinguishing* key query for some function f, it encrypts $(0, f(m_0), \mathbf{0})$. That is, the second message bit is $f(m_0)$ and rest are still all zeros.
  - For *distinguishing* key query f, it sets $v = f(m_b) \oplus \mathsf{Ext}_s(R)$, and computes the SKE ciphertext as to be an encryption of $(1, s, v)$.
- H_2 : In this hybrid, the challenger encrypts $(R, \mathsf{ske.sk}, 1)$ instead of $(m_b, \perp, 0)$. The rest of the game proceeds as before.
- H_3 : In this hybrid experiment, v is chosen uniformly at random (instead of setting it as $f(m_b) \oplus \mathsf{Ext}_s(R)$.

  Finally, note that the bit b is not used in H_3, and therefore the adversary has no advantage in this experiment.

The detailed proof for the above theorem is provided in Appendix F.

Using Theorem 10, we have the following corollary.

Corollary 27. *Assuming the existence of* ct*-rate-*1 *selectively secure FE for circuits, there exists a* ct*-rate-*1 *selectively secure regular incompressible FE scheme for circuits. Also,*

$$|\mathsf{mpk}| = \mathsf{poly}(\lambda), \quad |\mathsf{sk}_f| = \mathsf{poly}(\lambda), \quad |\mathsf{ct}| = \max(S, n) + \mathsf{poly}(\lambda)$$

Remark 28 (Handling functions with longer outputs, or a fixed number of distinguishing keys)**.** *The above FE construction can be naturally extended to support functions with longer outputs, or a fixed number of distinguishing keys. The idea is quite simply to hardwire either the multi-bit function output (in the first case) or the function output for every distinguishing function (in the second case). Note that if the secret keys for the underlying FE scheme are short (independent of the function description), then the above scheme achieves incompressible FE with optimal ciphertexts as well as secret keys.*

References

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013. 5

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. In *CRYPTO*, 2021. 7

[AR99] Yonatan Aumann and Michael O Rabin. Information theoretically secure communication in the limited storage space model. In *Annual International Cryptology Conference*, pages 65–79. Springer, 1999. 15

[BB11] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24:659–693, 2011. 26

[BDD22a] Pedro Branco, Nico Döttling, and Jesko Dujmović. Rate-1 incompressible encryption from standard assumptions. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part II*, pages 33–69. Springer, 2022. 3

[BDD22b] Pedro Branco, Nico Döttling, and Jesko Dujmović. Rate-1 Incompressible Encryption from Standard Assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography*, Lecture Notes in Computer Science, pages 33–69, Cham, 2022. Springer Nature Switzerland. 14, 17

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001. 2

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 533–556. Springer, 2014. 8, 21

[BGK$^+$24] Kaartik Bhushan, Rishab Goyal, Venkata Koppula, Varun Narayanan, Manoj Prabhakaran, and Mahesh Sreekumar Rajasree. Possibilities and impossibilities in incompressible cryptography. 2024. Unpublished Manuscript. 14, 32

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS '12*, 2012. 16

[BS23] Nir Bitansky and Tomer Solomon. Bootstrapping homomorphic encryption via functional encryption. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 17:1–17:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 2

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007. 2

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *TCC*, 2011. 2, 5

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007. 2

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65. Springer, 2017. 2

[CL16] Kuan Cheng and Xin Li. Randomness extraction in ac0 and with small locality. *arXiv preprint arXiv:1602.01530*, 2016. 8

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1997. 15

[Coc01] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, IMA International Conference*, volume 2260 of LNCS, pages 360–363, 2001. 2

[DGM23] Nico Döttling, Phillip Gajland, and Giulio Malavolta. Laconic function evaluation for turing machines. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 606–634. Springer, 2023. 2

[DGO19] Ivan Damgård, Chaya Ganesh, and Claudio Orlandi. Proofs of replicated storage without timing assumptions. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 355–380. Springer, 2019. 14, 15

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976. 2

[DQW23] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 86–116. Springer, 2023. 15

[Dzi06] Stefan Dziembowski. On Forward-Secure Storage. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, Lecture Notes in Computer Science, pages 251–270, Berlin, Heidelberg, 2006. Springer. 2, 8, 14, 18

[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 *FOCS*, 2013. 14

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. 14

[GGL24] Rachit Garg, Rishab Goyal, and George Lu. Dynamic collusion functional encryption and multi-authority attribute-based encryption. In *IACR International Conference on Public-Key Cryptography*, pages 69–104. Springer, 2024. 7

[GGLW22] Rachit Garg, Rishab Goyal, George Lu, and Brent Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In *EUROCRYPT*, 2022. `https://ia.cr/2021/847`. 7

[GKP$^+$13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564, 2013. 8, 20

[GKRW18] Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 467–497. Springer, 2018. 2

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, 2016. 6

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 612–621, 2017. 2, 5

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 660–670. ACM, 2018. 2

[GLW20] Rachit Garg, George Lu, and Brent Waters. New techniques in replica encodings with client setup. In *Theory of Cryptography Conference*, pages 550–583. Springer, 2020. 14, 15

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604. Springer, 2016. 2

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98. ACM, 2006. 2, 6

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, 2015. 2, 8

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 99–108, 2011. 14

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible Cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, Lecture Notes in Computer Science, pages 700–730, Cham, 2022. Springer International Publishing. 2, 3, 4, 5, 14, 15, 42, 43

[GWZ23] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Somewhere Randomness Extraction and Security against Bounded-Storage Mass Surveillance, 2023. Report Number: 409. 14, 17

[GZ21] Jiaxin Guan and Mark Zhandry. Disappearing cryptography in the bounded storage model. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 365–396. Springer, 2021. 15

[JLL23] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 479–510. Springer, 2023. 4, 12, 14, 19

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021. 2

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over f p, dlin, and prgs in nc 0. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part I*, pages 670–699. Springer, 2022. 2

[KMUW18] Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Daniel Wichs. Hardness of non-interactive differential privacy from one-way functions, 2018. 2

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008. 2

[Lu02] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2002. 15

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptol.*, 5(1):53–66, 1992. 15

[MW20] Tal Moran and Daniel Wichs. Incompressible Encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, Lecture Notes in Computer Science, pages 494–523, Cham, 2020. Springer International Publishing. 14, 15

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003. 14

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. 2, 5

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 859–870. IEEE Computer Society, 2018. 2

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742. IEEE Computer Society, 2017. 15

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. 4

[RRV99] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 149–158, 1999. 8

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984. 2

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005. 2, 6

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014. 14

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 111–126. ACM, 2013. 14

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 600–611, 2017. 2

[Yao82] Andrew C Yao. Protocols for secure computations. In *FOCS*, 1982. 6

[Yao86] Andrew Yao. How to generate and exchange secrets. In *FOCS*, 1986. 16

A Discussion on Incompressible FE

While the regular ABE/FE definitions offer strong security guarantees, there are many real-world scenarios that are not captured by the existing ABE/FE definitions. We discuss the following motivating scenarios for looking beyond traditional definitions of FE.

- Audit Logs and Big Databases. Suppose an organization wants to use (ciphertext-policy) ABE to provide auditors, customers, and employees fine-grained access over all customer and organizational data. Every user affiliated with this organization will receive a secret key corresponding to their attributes. Each data block will be encrypted under a policy predicate and a user gets access to a data block based on its attributes.

  Most large tech companies (such as Google, Facebook, Amazon, Microsoft) store and generate hundreds to thousands of terabytes of data everyday. Even mid-size organizations and universities routinely process a few terabytes of data everyday. Thus, we assume the daily communication and data storage is in the order of terabytes. A honest user may not want to locally store such large amounts of data forever; they can process it appropriately and then discard it. The organization stores all encrypted data in its (long-term) storage server, for analytics and audit purposes. This storage, which would be in the order of petabytes, is only sparingly accessed by most users. Therefore, we can assume that regular download access to this storage server is limited to at most a few terabytes. We wish to have the following security guarantees:

  - *Daily security:* Suppose an adversary colludes with users with attributes $x_1, x_2, \ldots, x_t$. It can decrypt the daily communication that these users are authorized to access, but it should not be able to learn anything about the messages that these users are not authorized to access.

  - *Security even in presence of storage server:* Suppose, at a later point, the adversary with secret keys for user attributes $x_1, \ldots, x_t$ also learns the secret key for some target user with attribute x^*. While such key breaches are possible, they might be rare. Therefore, it is infeasible for an adversary to store the entire daily communication and data storage ciphertexts, in the hope of learning/corrupting a key for x^* at a later point. However, does the presence of a storage server make the system vulnerable?

    If a traditional ABE system is used then the adversary, that can access $S \approx 10$ gigabytes from the server, could potentially learn $o(S)$[18] gigabytes of information that only x^* was authorized to learn. *Is it possible to prevent any leakage even when the adversary can arbitrarily select any S gigabytes of information from the server?*

- Eternal Forward Security from Bounded 'Long-Term Storage'. Consider any setting where FE is used to provide fine-grained data access. The master public key is used to encrypt, and the central authority can generate secret keys for various members of the organization. Each member i is authorized to learn a particular function f_i of the encrypted data. As long as msk remains hidden, an adversary colluding with users $i_1, \ldots, i_t$ can only learn $f_{i_1}, \ldots, f_{i_t}$ applied on the encrypted data.

  Suppose msk gets compromised. Clearly, all future encrypted communications will no longer be private. The adversary, given an encryption of x, can learn $f(x)$ for any function f of its choice. *However, is the same true for all past encryptions too?* The answer depends on the adversary's long-term storage. Suppose it has bounded long-term storage, and can only store S bits of the ciphertext.

  In traditional FE, the adversary can potentially learn S bits of $f(x)$ by storing just S bits of a ciphertext, even when $|\mathsf{ct}| >> S$ bits. What if there exists an FE scheme where $|\mathsf{ct}|$ is just slightly greater than S, but any adversary with S-bits of memory learn any information about $f(x)$, even given msk? Such an FE gives eternal forward security for bounded long-term storage attackers.

- Imperfect Key Distribution and Generation. While functional secret keys offer fine-grained access over encrypted data, they also lead to a lot more opportunities for an attacker to undermine the security of the system compared to plain encryption. In many applications, this unfortunately is the major deterrent factor preventing a much wider adoption of FE systems in practice.

  Simply put, in PKE, there is just one secret key, and while that implies an 'all-or-nothing' flavor of security, it also makes it far easier to store the key securely for the entire system lifetime. In FE, on the contrary, the central authority is burdened with multiple tasks with strong security implications – (1) storing msk securely, (2) correctly computing functional keys for each user, and (3) securely distributing keys to users. E.g., in identity-based encryption (IBE), there is an exponential number of functional keys (one for each user identity). What if the authority generates Alice's key for the identity '`a1ice@iacr.org`' instead of '`alice@iacr.org`'. (The difference is a typo, using number '1' instead of letter 'l'.)

  Broadly, the argument is that there are a lot more attack avenues and targets in FE systems. While it is reasonable to expect the master key authority will (in most part) store msk securely,

[18] Here we say $o(S)$ instead of S since an ABE ciphertext encrypting a few bits could be as large as a few hundred bytes.

expecting it from every user is unrealistic. Thus, in practice, standard FE security might not be enough! We need to look beyond the traditional security formulations.

B Incompressible ABE from GWZ [GWZ22]

In this section, we provide an alternate construction for rate-1 incompressible ABE by extending the rate-1 incompressible PKE scheme from [GWZ22], as long as the stage two adversary only queries a single distinguishing function. That is, the following construction satisfies the (standard) incompressible ABE security.

Construction. Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ be a rate-1 secure FE scheme, $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^\ell$ be a secure PRG and $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\lambda$ be a strong average min-entropy extractor.

- $\mathsf{Setup}(1^\lambda, 1^S)$: The setup algorithm first runs $\mathsf{FE.Setup}$ to generate a master public and secret key $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. It sets $\mathsf{mpk} = \mathsf{abe.mpk}$ and $\mathsf{msk} = (\mathsf{abe.msk}, S)$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: Let $\mathsf{msk} = (\mathsf{fe.msk}, S)$. The key generation algorithm first randomly generates two string $v \in \{0,1\}^{d\times\lambda}$. It computes $\mathsf{sk}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, f_{C,v})$ where

$$f_{C,v}(m, x, b) = \begin{cases} m & if\ b = 0 \wedge C(x) = 1 \\ m \oplus v & if\ b = 1 \wedge C(x) = 1 \\ \perp & otherwise \end{cases}$$

  It returns sk_C.
- $\mathsf{Enc}(\mathsf{mpk}, m, x)$: Let $\mathsf{mpk} = \mathsf{fe.mpk}$. The encryption algorithm first randomly generate $s \in \{0,1\}^d, t \in \{0,1\}^\lambda, R \in \{0,1\}^n$ and computes $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, ((s,t), x, 0))$. It then computes $c = \mathsf{PRG}(\mathsf{Ext}(R; s) \oplus t) \oplus m$ and return $(\mathsf{fe.ct}, R, c)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = (\mathsf{sk}_C)$ and $\mathsf{ct} = (\mathsf{fe.ct}, R, c)$. It first computes $(s,t) = \mathsf{FE.Dec}(\mathsf{sk}_C, \mathsf{fe.ct})$ and $m = c \oplus \mathsf{PRG}(\mathsf{Ext}(R; s) \oplus t)$. It returns m.

Correctness. A ciphertext for a message m and an attribute x in the above scheme is $(\mathsf{fe.ct}, R, c)$ where $c = \mathsf{PRG}(\mathsf{Ext}(R; s) \oplus t) \oplus m$, $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, ((s,t), x, 0))$ for some randomly generated s, t, R. The decryption algorithm computes $\mathsf{FE.Dec}(\mathsf{sk}_C, \mathsf{fe.ct})$ which by the correctness of FE is equal to (s,t) when sk_C is a secret key associated to a circuit C such that $C(x) = 1$. Then, the decryption algorithm proceeds by computing $c \oplus \mathsf{PRG}(\mathsf{Ext}(R; s) \oplus t)$ which is equal to m.

Security. Since the proof is similar to the proof in [GWZ22], we only highlight the changes in sequence of hybrids.

1. H_0 : This the original regular adaptive ABE incompressible security game where the challenge ciphertext is an encryption of m_b.

2. H_1 : The challenge ciphertext is generated by computing $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, ((s,t) \oplus v_{C^*}), x, 1)$ where C^* is the circuit given by the adversary in the query phase of the Second Response Phase.

3. H_2 : The challenge ciphertext is $(\mathsf{fe.ct}, R, z)$ where $z = \mathsf{PRG}(T) \oplus m_b$, $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, (S, x, 1)$ where S, R, T are randomly generated. In the query phase of the Second Response Phase, first s is randomly generated. Then, t is set to $T \oplus \mathsf{Ext}(R; s)$ and finally $v_{C^*} = S \oplus (s,t)$.

4. H_3 : In the query phase of the Second Response Phase, s, t are randomly generated and $v_{C^*} = S \oplus (s,t)$.

5. H_4 : The challenge ciphertext is $(\mathsf{fe.ct}, R, z)$ where $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, (S, x, 1)$ where S, R, z are randomly generated.

Remark. Observe that we can achieve only standard and not semi-strong security because we cannot construct multiple distinguishing keys such that v_{C^*}'s are uniformly random and $\mathsf{fe.ct}$ decrypts to the same (s,t). Additionally, it is worth mentioning that the FE scheme used above can be built from any standard ABE scheme by substituting PKE with ABE in the transformation described in [GWZ22] Section 3.4.

C Proof of Theorem 18

Proof. We will show that the scheme provided in Section 5 is secure using a sequence of hybrid arguments.

G_0: This is the adaptive incompressible ABE security game where the challenger randomly chooses $d \in \{0,1\}$ and encrypts one of m_0, m_1 for the attribute x^* given by $\mathcal{A}_1$.

- **Initialization Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n.
  2. It computes $(\mathsf{abe.mpk}, \mathsf{abe.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\tilde{n}})$.
  3. It sets $\mathsf{mpk} = (\mathsf{abe.mpk}, S)$ and $\mathsf{msk} = \mathsf{abe.msk}$ and sends mpk to $\mathcal{A}_1$.
- **Pre-Challenge Query Phase:**
  1. For each query F from the first adversary $\mathcal{A}_1$, the challenger randomly generates (k_F, w_F).
  2. It generates an ABE secret key by computing $\mathsf{abe.sk}_F \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F})$.
  3. It returns $\mathsf{sk}_F = (k_F, w_F, \mathsf{abe.sk}_F)$ to $\mathcal{A}_1$.
- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly generates $d \in \{0,1\}$.
  3. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.

4. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
5. It randomly generates two strings r^*_{Enc}, R^*.
6. It computes $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,b}\right\}_{i\in[\kappa],b\in\{0,1\}}\right) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{\mathsf{fe.msk}^*,\mathsf{prf.key}^*,r^*_{\mathsf{Enc}}})$.
7. For all $i \in [\kappa], b \in \{0,1\}$, it computes $\mathsf{abe.ct}^*_{i,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,b}, (x^*, i, b))$.
8. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, D_{R^*})$.
9. It computes $t^*_i = \mathsf{PRF.Eval}(\mathsf{prf.key}^*, i)), \forall i \in [|m|]$ and sets $t^* = t^*_1||t^*_2||\dots||t^*_m$.
10. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i\in[\kappa],b\in\{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

- **Post-Challenge Query Phase:**
  1. For each query F from the first adversary $\mathcal{A}_1$, the challenger checks whether $F(x^*) = 1$.
  2. If the check is true, it returns $\perp$.
  3. Else, it randomly generates (k_F, w_F).
  4. It generates an ABE secret key by computing $\mathsf{abe.sk}_F \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F})$.
  5. It returns $\mathsf{sk}_F = (k_F, w_F, \mathsf{abe.sk}_F)$ to $\mathcal{A}_1$.

- **First Response Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\mathsf{st}| \leq S$.

- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For the query F^* such that $F^*(x^*) = 1$ from the second adversary $\mathcal{A}_2$.
       - The challenger randomly generates (k_{F^*}, w_{F^*}).
       - It generates an ABE secret key by computing $\mathsf{abe.sk}_{F^*} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F}^*)$.
       - It returns $\mathsf{sk}_{F^*} = (k_{F^*}, w_{F^*}, \mathsf{abe.sk}_{F^*})$ to $\mathcal{A}_2$.

     (b) For every query F such that $F(x^*) = 0$.
       - The challenger randomly generates (k_F, w_F).
       - It generates an ABE secret key by computing $\mathsf{abe.sk}_F \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F})$.
       - It returns $\mathsf{sk}_F = (k_F, w_F, \mathsf{abe.sk}_F)$ to $\mathcal{A}_2$.
  3. Finally, $\mathcal{A}_2$ outputs $d' \in \{0,1\}$.

G_1: The challenger generates (k_{F^*}, w_{F^*}) in the Challenge Phase and uses it encrypt **0** instead of $\mathsf{lab}_{i,1-(k_{F^*},w_{F^*})[i]}$ under the ABE scheme.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.

2. The challenger randomly generates (k_{F^*}, w_{F^*}).
3. It randomly generates $d \in \{0,1\}$.
4. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.
5. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
6. It randomly generates two strings r^*_{Enc}, R^*.
7. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, D_{R^*})$.
8. It computes $\left(\hat{C}^*, \{\mathsf{lab}^*_{i,b}\}_{i\in[\kappa], b\in\{0,1\}}\right) \leftarrow \mathsf{GC.Garble}(1^\lambda, C_{\mathsf{fe.msk}^*, \mathsf{prf.key}^*, r^*_{\mathsf{Enc}}})$.
9. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,(k_{F^*}, w_{F^*})[i]}, (x^*, i, (k_{F^*}, w_{F^*})[i]))$.
10. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,1-(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathbf{0}, (x^*, i, 1-(k_{F^*}, w_{F^*})[i]))$.
11. It computes $t^*_i = \mathsf{PRF.Eval}(\mathsf{prf.key}^*, i)), \forall i \in [|m|]$ and sets $t^* = t^*_1 || t^*_2 || \ldots || t^*_m$.
12. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i\in[\kappa], b\in\{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For the query F^* such that $F^*(x^*) = 1$ from the second adversary $\mathcal{A}_2$.
        - ~~The challenger randomly generates (k_{F^*}, w_{F^*}).~~
        - It generates an ABE secret key by computing $\mathsf{abe.sk}_{F^*} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F}^*)$.
        - It returns $\mathsf{sk}_{F^*} = (k_{F^*}, w_{F^*}, \mathsf{abe.sk}_{F^*})$ to $\mathcal{A}_2$.

     (b) For every query F such that $F(x^*) = 0$.
        - The challenger randomly generates (k_F, w_F).
        - It generates an ABE secret key by computing $\mathsf{abe.sk}_F \leftarrow \mathsf{ABE.KeyGen}(\mathsf{abe.msk}, \hat{F})$.
        - It returns $\mathsf{sk}_F = (k_F, w_F, \mathsf{abe.sk}_F)$ to $\mathcal{A}_2$.
  3. Finally, $\mathcal{A}_2$ outputs $d' \in \{0,1\}$.

G_2: The challenger generates the garbled circuit and the necessary labels using the simulator.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly generates (k_{F^*}, w_{F^*}).
  3. It randomly generates $d \in \{0,1\}$.

4. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.
5. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
6. It randomly generates two strings r^*_{Enc}, R^*.
7. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}^*, D_{R^*})$.
8. It computes $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,(k_{F^*},w_{F^*})[i]}\right\}_{i\in[\kappa]}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\kappa, 1^{|C_{\mathsf{fe.msk},m,r}|}, \mathsf{FE.Enc}(\mathsf{msk}^*,$ $(k_{F^*}, w_{F^*} \oplus \mathsf{prf.key}^*); r^*_{\mathsf{Enc}}))$.
9. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,(k_{F^*},w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,(k_{F^*},w_{F^*})[i]}, (x^*, i, (k_{F^*}, w_{F^*})[i]))$.
10. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,1-(k_{F^*},w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathbf{0}, (x^*, i, 1-(k_{F^*}, w_{F^*})[i]))$.
11. It computes $t^*_i = \mathsf{PRF.Eval}(\mathsf{prf.key}^*, i)), \forall i \in [|m|]$ and sets $t^* = t^*_1||t^*_2||\ldots||t^*_m$.
12. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i\in[\kappa],b\in\{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

G_3: The challenger simulates the FE ciphertext and secret key in the Challenge Phase.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
2. The challenger randomly generates (k_{F^*}, w_{F^*}).
3. It randomly generates $d \in \{0,1\}$.
4. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.
5. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
6. It randomly generates two strings r^*_{Enc}, R^*.
7. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}^*, D_{R^*})$.
8. It generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Sim}(1^\lambda, D_{R^*}, D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathsf{prf.key}^*), 1^\kappa)$.
9. It computes $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,(k_{F^*},w_{F^*})[i]}\right\}_{i\in[\kappa]}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\kappa, 1^{|C_{\mathsf{fe.msk},m,r}|}, \mathsf{fe.ct}^*)$.
10. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,(k_{F^*},w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,(k_{F^*},w_{F^*})[i]}, (x^*, i, (k_{F^*}, w_{F^*})[i]))$.
11. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,1-(k_{F^*},w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathbf{0}, (x^*, i, 1-(k_{F^*}, w_{F^*})[i]))$.
12. It computes $t^*_i = \mathsf{PRF.Eval}(\mathsf{prf.key}^*, i)), \forall i \in [|m|]$ and sets $t^* = t^*_1||t^*_2||\ldots||t^*_m$..
13. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i\in[\kappa],b\in\{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

G_4: The challenger encrypts a garbled string instead of the PRF key in the Challenge Phase.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
2. The challenger randomly generates (k_{F^*}, w_{F^*}).
3. It randomly generates $d \in \{0,1\}$.
4. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.
5. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
6. It randomly generates two strings r^*_{Enc}, R^*.
7. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}^*, D_{R^*})$.
8. It generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Sim}(1^\lambda, \mathsf{fe.sk}^*, D_{R^*}, D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathbf{0}), 1^\kappa)$.
9. It computes $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,(k_{F^*}, w_{F^*})[i]}\right\}_{i \in [\kappa]}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\kappa, 1^{|C_{\mathsf{fe.msk},m,r}|}, \mathsf{fe.ct}^*)$.
10. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,(k_{F^*}, w_{F^*})[i]}, (x^*, i, (k_{F^*}, w_{F^*})[i]))$.
11. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,1-(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathbf{0}, (x^*, i, 1-(k_{F^*}, w_{F^*})[i]))$.
12. It computes $t^*_i = \mathsf{PRF.Eval}(\mathsf{prf.key}^*, i)), \forall i \in [|m|]$ and sets $t^* = t^*_1 || t^*_2 || \ldots || t^*_m$..
13. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i \in [\kappa], b \in \{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

G_5: The challenger randomly generates t^* in the Challenge Phase.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends $(m_0, m_1, x^*, \mathsf{aux})$ where x^* is the target attribute and aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
2. The challenger randomly generates (k_{F^*}, w_{F^*}).
3. It randomly generates $d \in \{0,1\}$.
4. It generates $\mathsf{fe.msk}^* \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\hat{s}})$.
5. It generates a PRF key $\mathsf{prf.key}^* \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$.
6. It randomly generates two strings r^*_{Enc}, R^*.
7. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}^*, D_{R^*})$.
8. It generates $\mathsf{fe.ct}^* \leftarrow \mathsf{FE.Sim}(1^\lambda, \mathsf{fe.sk}^*, D_{R^*}, D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathbf{0}), 1^\kappa)$.
9. It computes $\left(\hat{C}^*, \left\{\mathsf{lab}^*_{i,(k_{F^*}, w_{F^*})[i]}\right\}_{i \in [\kappa]}\right) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\kappa, 1^{|C_{\mathsf{fe.msk},m,r}|}, \mathsf{fe.ct}^*)$.
10. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathsf{lab}_{i,(k_{F^*}, w_{F^*})[i]}, (x^*, i, (k_{F^*}, w_{F^*})[i]))$.
11. For all $i \in [\kappa]$, it computes $\mathsf{abe.ct}^*_{i,1-(k_{F^*}, w_{F^*})[i]} \leftarrow \mathsf{ABE.Enc}(\mathsf{abe.mpk}, \mathbf{0}, (x^*, i, 1-(k_{F^*}, w_{F^*})[i]))$.
12. It generates $\mathsf{fe.sk}^* \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, D_{R^*})$.
13. It randomly generates t^*.
14. It returns $(\{\mathsf{abe.ct}^*_{i,b}\}_{i \in [\kappa], b \in \{0,1\}}, \hat{C}^*, \mathsf{fe.sk}^*, R^*, t^* \oplus m_d)$ to $\mathcal{A}_1$.

Analysis: Let $p_{\mathcal{A},i}$ denote probability of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputting $d' = d$ in Game G_i. We will show that this probability is almost the same in every game.

Lemma 29. *Assuming* ABE *is adaptively secure ABE scheme, then for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \text{negl}(\lambda)$.*

Proof. The proof of the lemma following from the multi-ciphertext adaptive security of the ABE scheme. We will show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish G_1 and G_0, then there is an adversary $\mathcal{B}$ that wins the adaptive security game of the ABE scheme with equal probability.

$\mathcal{B}$ starts by receiving S and 1^n from $\mathcal{A}_1$ and sends 1^n to its challenger. The challenger returns $\mathsf{abe.mpk}$ and $\mathcal{B}$ simulated G_1 until the challenger phase. To respond to the queries F from $\mathcal{A}_1$, it sends $\hat{F}$ to its challenger. The challenger responds with sk_F and $\mathcal{B}$ relays it to $\mathcal{A}_1$. To generate the ABE ciphertexts, $\mathcal{B}$ sends $(\{\mathsf{lab}_{i,1-(k_{F^*},w_{F^*})}\}_{i\in[\kappa]}, \{0\}_{i\in[\kappa]}, \{(x^*, i, 1-(k_{F^*}, w_{F^*})[i]\}_{i\in[\kappa]})$ to its challenger. It receives $\{\mathsf{abe.ct}_i^*\}_{i\in[\kappa]}$ and uses it to simulate the rest of the game.

Let us carefully analyse the key queries made by $\mathcal{B}$ to its challenger $\mathcal{C}$. Observe that the targeted attributes given by $\mathcal{B}$ to $\mathcal{C}$ are $\{(x^*, i, 1-(k_{F^*}, w_{F^*})[i])\}$. We must ensure that $\mathcal{B}$ does not make key queries for policies $\hat{F}$ such that $\hat{F}(x^*, i, 1-(k_{F^*}, w_{F^*})[i]) = 1$. In the Second Response Phase, $\mathcal{B}$ makes a query for policy of the form $\hat{F^*}$ such that $\hat{F^*}(x, (i, b)) = 1 \iff F(x) = 1 \wedge b = (k_{F^*}, w_{F^*})[i]$. Observe that even though $F^*(x^*) = 1$, we have $\hat{F^*}(x^*, i, 1-(k_{F^*}, w_{F^*})[i]) = 0$ for all i.

□

Lemma 30. *Assuming* GC *is a secure garbling scheme, then for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \text{negl}(\lambda)$.*

Proof. Security follows from the simulation property of the garbling scheme. The reduction simulates the game G_2 exactly, differing only during the challenge phase when it relies on the challenger of the garbling scheme's security game to create the circuit and the labels. If the challenger provides the garbled circuit and labels, then the reduction has simulated G_1, and if the challenger sends over the simulated output, the reduction has simulated G_2. Thus, the advantage with which the reduction wins is exactly equal to the advantage with which an adversary can distinguish between G_1 and G_2. □

Lemma 31. *Assuming* FE *is 1-key 1-ciphertext secure FE scheme, then for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \text{negl}(\lambda)$.*

Proof. Security follows from the simulation security of the FE scheme. The reduction simulates the game G_3 exactly, differing only during the challenge phase when it relies on the challenger/simulator of the FE scheme's security game to create the FE ciphertext and secret key. If the FE challenger provides the ciphertext and secret key, then the reduction has simulated G_2, and if the FE simulator sends over the simulated ciphertext and secret key, the reduction has simulated G_3. Thus, the advantage with which the reduction wins is exactly equal to the advantage with which an adversary can distinguish between G_3 and G_2. □

Lemma 32. *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},4} - p_{\mathcal{A},3}| \leq \text{negl}(\lambda)$.*

Proof. The only distinction between G_3 and G_4 lies in the generation of $\mathsf{fe.ct}^*$. To be precise, in G_4 the FE simulator takes $D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathsf{prf.key}^*)$ as input, whereas in G_5, it is $D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathbf{0})$. Observe that $D_{R^*}(k_{F^*}, w_{F^*} \oplus v) = \mathsf{Ext}_{k_{F^*}}(R^*) \oplus w_{F^*} \oplus v$ and $(R^*, \mathsf{Ext}_{k_{F^*}}(R^*) \oplus w_{F^*} \oplus v)$ is an encryption of v under the Dziembowski's SKE scheme with secret key (k_{F^*}, w_{F^*}). This is an incompressible encryption scheme. Hence, we can trivially reduce the incompressibility security game of Dziembowski's scheme to a distinguishing game between G_5 and G_4. This is because $\mathcal{A}_1$ has access to (k_{F^*}, w_{F^*}) only via the ciphertext $D_{R^*}(k_{F^*}, w_{F^*} \oplus \mathsf{prf.key}^*)$, while the reduction gains access to (k_{F^*}, w_{F^*}) in the second phase and will be able to successfully respond to the distinguishing query F^*. □

Lemma 33. *Assuming* PRF *is a secure PRF scheme, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \text{negl}(\lambda)$.*

Proof. In both the games, the PRF key $\mathsf{prf.key}^*$ is only used for generating t^*. Therefore, it is a straightforward reduction from the PRF security game to distinguishing G_4 and G_5. □

Lemma 34. *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $p_{\mathcal{A},5} = 1/2 + \text{negl}(\lambda)$.*

Proof. The information b is only present in the challenge ciphertext in the form of $t^* \oplus m_d$. Given that t^* is a truly random string, it follows that $t^* \oplus m_d$ is also a truly random string. Therefore, the adversary has negligible advantage in guessing the value of b. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq 1/2 + \text{negl}(\lambda)$.

□

D Proof of Theorem 21

We will show that the scheme provided in Section 6 is secure using a sequence of hybrid arguments.

G_0: This is the real adaptive semi-strongly incompressible FE security game with the challenge bit b.

- **Initialization Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n - index of a function class.
  2. The challenger computes $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$.
  3. It computes $\mathsf{inc.sk} \leftarrow \mathsf{IncSKE.Setup}(1^\lambda, 1^S)$.
  4. It computes $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
  5. It computes $\mathsf{ske.sk}' \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
  6. It sets $\mathsf{mpk} = \mathsf{fe.mpk}$ and $\mathsf{msk} = (\mathsf{fe.msk}, \mathsf{inc.sk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$ and sends mpk to $\mathcal{A}_1$.
- **Pre-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^\alpha))$ and $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_1$.

- **Challenge Phase:**
    1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
    2. The challenger randomly samples $\mathsf{inc.ct}^* \leftarrow \{0,1\}^{\beta}$.
    3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m_b, \perp, \mathsf{inc.ct}^*, 0))$ and sends it to $\mathcal{A}_1$.
- **Post-challenge Query Phase:**
    1. For each query f such that $f(m_0) = f(m_1)$ from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^{\alpha}))$ and $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.
    2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_1$.
- **First Response Phase:**
    1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:**
    1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
    2. **Key Generation Queries**
        (a) For every query f from the first adversary $\mathcal{A}_2$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^{\alpha}))$.
        (b) It computes $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.
        (c) It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_2$.
    3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_1: In this hybrid, $\mathsf{ske.ct}'$ in the generation of distinguishing keys in the Second Response phase is computes as $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', m_b)$ rather than $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.

- **Second Response Phase:**
    1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
    2. **Key Generation Queries**
        (a) For every query f such that $f(m_0) \neq f(m_1)$ from the second adversary $\mathcal{A}_2$,
            - The challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^{\alpha}))$.
            - It computes $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$.
            - It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_2$.
        (b) For other queries f from $\mathcal{A}_2$,
            - The challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^{\alpha}))$.
            - It computes $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.
            - It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_2$.

3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_2: In this hybrid, the incompressible ciphertext is modified to $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger computes $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m_b, \perp, \mathsf{inc.ct}^*, 0))$ and sends it to $\mathcal{A}_1$.

G_3: In this hybrid, $\mathsf{ske.ct}$ in the generation of distinguishing keys in the Second Response Phase is computed as $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, \mathsf{inc.sk}))$.

- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For every query f such that $f(m_0) \neq f(m_1)$ from the second adversary $\mathcal{A}_2$,
        - The challenger first computes $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, \mathsf{inc.sk}))$.
        - It computes $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$.
        - It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_2$.

     (b) For other queries f from $\mathcal{A}_2$,
        - The challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^\alpha))$.
        - It computes $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$.
        - It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ and sends it to $\mathcal{A}_2$.
  3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_4: In this hybrid, the challenge ciphertext is generated as $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m, \mathsf{ske.sk}, \mathsf{inc.ct}, 1))$.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger computes $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m_0, \mathsf{ske.sk}, \mathsf{inc.ct}^*, 1))$ and sends it to $\mathcal{A}_1$.

G_5: In this hybrid, the incompressible ciphertext is generated as $\mathsf{inc.ct}^* \leftarrow \{0,1\}^\beta)$.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger computes $\mathsf{inc.ct}^* \leftarrow \{0,1\}^\beta$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}, (m_0, \mathsf{ske.sk}, \mathsf{inc.ct}^*, 1))$ and sends it to $\mathcal{A}_1$.

Analysis: Let $p_{\mathcal{A},i}$ denote probability of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputting $b' = b$ in Game G_i. We will show that this probability is almost the same in every game.

Lemma 35. *Assuming* SKE *is secure SKE scheme, then for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \text{negl}(\lambda)$.*

Proof. The proof of the lemma follows from the CPA security of the SKE scheme. The only difference between G_0 and G_1 is that in G_1, $\mathsf{ske.ct}'$ used in the generation of the distinguishing keys in the Second Response Phase is computed as $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$ instead of $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$. Observe that $\mathsf{ske.sk}'$ is only used in the generation of SKE ciphertexts in the entire game.

Consider $T+1$ intermediate hybrids $G_{0,i}$ where $0 \leq i \leq T$ and T is the number of distinguishing key queries made by the second adversary $\mathcal{A}_2$. In $G_{0,i}$ hybrid, for the first i distinguishing key queries made by $\mathcal{A}_2$, the challenger computes $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$ and for the rest of the queries, it computes $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$. Observe that $G_{0,0} = G_0$ and $G_{0,T} = G_1$ whereas $G_{0,i}$ and $G_{0,i+1}$ only differ at the key generation for the i^{th} distinguishing key query by $\mathcal{A}_2$.

We will now show that if there is a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish between any $G_{0,i}$ and $G_{0,i-1}$ where $1 \leq i \leq T$, then we can build an adversary $\mathcal{B}$ that breaks the indistinguishability security of the SKE scheme. For the CPA security game of SKE scheme, the adversary $\mathcal{B}$ would simulate the game G_1 till the $(i-1)^{th}$ distinguishing key query in the second response phase. It will send $(0, f(m_b))$ to its challenger where f is the i^{th} distinguishing query by $\mathcal{A}_2$, who will respond with a challenge ciphertext c^*. It will use c^* as $\mathsf{ske.ct}'$ to generate the key in the i^{th} distinguishing key query and simulate the rest of the game. An important point to note is that $\mathcal{B}$ does not have the secret key $\mathsf{ske.sk}$, so it will have to query its challenger for encryptions of 0 or 1 to simulate the key generation algorithm. Observe that if c^* is $\mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$, then $\mathcal{B}$ has exactly simulated $G_{0,i}$. Else, it has simulated $G_{0,i-1}$. □

Lemma 36. *Assuming the pseudorandomness of* IncSKE *scheme, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that for all λ, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \text{negl}(\lambda)$.*

Proof. The proof of the lemma follows from the pseudorandomness of the ciphertexts in the incompressible SKE scheme. The only difference between G_2 and G_1 is in the generation of $\mathsf{inc.ct}^*$. Observe that in G_1, we have $\mathsf{inc.ct}^* \leftarrow \{0,1\}^\beta$ whereas in G_2, it is $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.

We will show that if there exists $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish between G_2 and G_1, then there exists $\mathcal{B}$ that can distinguish a ciphertext from a truly random string. $\mathcal{B}$ receives $1^S, 1^n$ and sends 1^S to its challenger. The challenger returns $\mathsf{inc.sk}$ and $\mathcal{B}$ simulates the game till it receives (m_0, m_1, aux) from $\mathcal{A}_1$. It sends $\mathsf{ske.sk}'$ to its challenger and receives $\mathsf{inc.ct}^*$ using which it simulates the rest of the game. Observe that if $\mathsf{inc.ct}^* \leftarrow\leftarrow \{0,1\}^\beta$, then $\mathcal{B}$ has simulated G_1, otherwise it has simulated G_2 exactly. □

Lemma 37. *Assuming* SKE *is a secure SKE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* λ*,* $|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$*.*

Proof. The proof is similar to Theorem 35. □

Lemma 38. *Assuming* FE *is a adaptively secure FE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* λ*,* $|p_{\mathcal{A},4} - p_{\mathcal{A},3}| \leq \mathrm{negl}(\lambda)$*.*

Proof. The proof of the lemma follows from the adaptive security of the FE scheme. The only difference between G_4 and G_3 is the message used in the challenge ciphertext. To be precise, in G_3, the challenge ciphertext is an encryption of $M_1 = (m_b, \mathbf{0}, \mathsf{inc.ct}^*, 0)$ whereas in G_4, it is an encryption of $M_2 = (m_0, \mathsf{ske.sk}, \mathsf{inc.ct}^*, 1)$ where $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.

We first show that for each query f in the Pre-challenge Query and Post-challenge Query phase in both the games, the secret key generated decrypts their respective challenge ciphertext to the same value. In both the hybrids, the secret key is generated as $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ where $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, 0^\alpha))$ and $\mathsf{ske.ct}' = \mathsf{SKE.Enc}(\mathsf{ske.sk}', 0)$. Since $b = 0$ in the challenge ciphertext of G_3, from the description of $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$, the decryption will produce $f(m_b)$. Whereas, in G_4, $b = 1$ in the challenge ciphertext. Therefore, $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ will first compute $(flag, \mathsf{inc.sk}') \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$. From the description of $\mathsf{ske.ct}$, we have $(flag, \mathsf{inc.sk}') = (0, 0^\alpha)$. Therefore, $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ will output $f(m_0)$ which is equal to $f(m_b)$.

We now show that for each query f in the Second Response Phase in both the games, the secret key decrypts their respective challenge ciphertext to the same value. Observe that for queries f such that $f(m_0) = f(m_b)$, the argument in the above paragraph follows. Let us focus on queries f such that $f(m_0) \neq f(m_1)$. In both the games, the secret key is $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'})$ where $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, \mathsf{inc.sk}))$ and $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}', f(m_b))$. In G_3, since $b = 0$, the decryption would produce $f(m_b)$. In G_4, b is set to 1 in the challenge ciphertext. Therefore, $C_{f,\mathsf{ske.ct},\mathsf{ske.ct}'}$ will first decryption $\mathsf{ske.ct}$ to get $(1, \mathsf{inc.sk}) \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct})$. Since $flag = 1$, it will compute $\mathsf{ske.sk}' \leftarrow \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct}^*)$ using which it will output $y = \mathsf{SKE.Dec}(\mathsf{ske.sk}', \mathsf{ske.ct}')$. From the description of $\mathsf{ske.ct}'$, we have $y = f(m_b)$ are required. □

Lemma 39. *Assuming* IncSKE *is a secure incompressible SKE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* λ*,* $|p_{\mathcal{A},5} - p_{\mathcal{A},4}| \leq \mathrm{negl}(\lambda)$*.*

Proof. The proof of the lemma follows from the incompressible security of the incompressible SKE scheme. The only difference between G_5 and G_4 is in the generation of $\mathsf{inc.ct}^*$. Observe that in G_5, we have $\mathsf{inc.ct}^* \leftarrow \{0,1\}^\beta$ whereas in G_4, it is $\mathsf{inc.ct}^* \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, \mathsf{ske.sk}')$.

We will show that if there exists $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish between G_5 and G_4, then there exists $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that can win the incompressibility security game with equal probability. $\mathcal{B}_1$ receives $1^S, 1^n$ and sends 1^S to its challenger. The challenger returns $\mathsf{inc.sk}$ and $\mathcal{B}$ simulates the game till it receives (m_0, m_1, aux) from $\mathcal{A}_1$. It sends $\mathsf{ske.sk}'$ to its challenger and receives $\mathsf{inc.ct}^*$ using which it simulates the game till the Second Response Phase. On receiving $\mathsf{inc.sk}$ from this challenger, $\mathcal{B}_2$ can response to queries f such that $f(m_0) \neq f(m_1)$.

Observe that if $\mathsf{inc.ct}^* \leftarrow \{0,1\}^\beta$, then $\mathcal{B}$ has simulated G_5, otherwise it has simulated G_4 exactly. □

Lemma 40. *Assuming* SKE *is a secure SKE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* λ*,* $p_{\mathcal{A},5} = 1/2 + \mathrm{negl}(\lambda)$*.*

Proof. The proof is similar to Theorem 35. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq 1/2 + \text{negl}(\lambda)$.

E Proof of Theorem 24

We will show that the construction in section 7 is secure using a sequence of hybrid arguments.

G_0: This is the real selective semi-strongly incompressible PK-FE security game with challenge bit b.

- **Initialization Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n - index of a function class and two message (m_0, m_1).
  2. The challenger computes $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
  3. It computes $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$.
  4. It sets $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$ and $\mathsf{mpk} = \mathsf{fe.mpk}$.
  5. It sends mpk to $\mathcal{A}_1$.
- **Pre-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{n+d+1})$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends aux where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly samples $b \leftarrow \{0, 1\}$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.msk}, (m_b, \perp, 0))$ and sends it to $\mathcal{A}_1$.
- **Post-challenge Query Phase:**
  1. For each query f such that $f(m_0) = f(m_1)$ from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{n+d+1})$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **First Response Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.

2. **Key Generation Queries**
   (a) For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{n+d+1})$.
   (b) It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_1: In this game, the challenger modifies the secret keys generates in the query phases as follows.

- **Initialization Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n - index of a function class and two message (m_0, m_1).
  2. The challenger computes $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
  3. It computes $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$.
  4. It sets $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$ and $\mathsf{mpk} = \mathsf{fe.mpk}$.
  5. It sends mpk to $\mathcal{A}_1$.
  6. It randomly samples $R \leftarrow \{0,1\}^S$ and $s \leftarrow \{0,1\}^d$.
- **Pre-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **Post-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For every query f such that $f(m_0) \neq f(m_1)$ from the second adversary $\mathcal{A}_2$,
        i. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, s, v)))$ where $v = m_b \oplus \mathsf{Ext}_s(R)$.
        ii. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
     (b) For other queries f from $\mathcal{A}_2$
        i. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.

ii. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.

3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_2: The challenger computes the challenge ciphertext by encrypting $(R, \mathsf{ske.sk}, 1)$ instead of $(m_b, \perp, 0)$.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends *aux* where *aux* is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly samples $b \leftarrow \{0,1\}$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.msk}, (R, \mathsf{ske.sk}, 1))$ and sends it to $\mathcal{A}_1$.

G_3: The challenger sets v to a truly random string (consistent across all distinguishing key queries) instead of $m_b \oplus \mathsf{Ext}_s(R)$.

Analysis: Let $p_{\mathcal{A},i}$ denote probability of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputting $b' = b$ in Game G_i. We will show that this probability is almost the same in every game.

Lemma 41. *Assuming* SKE *is secure SKE scheme, then for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *there exists a negligible function* $\text{negl}(\cdot)$ *such that for all* λ, $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \text{negl}(\lambda)$.

Proof. The proof of the lemma follows from the CPA security of the SKE scheme. The only difference between G_0 and G_1 is that in G_1, $\mathsf{ske.ct}$ used in the generation of the distinguishing keys (non-distinguishing keys) is computed as $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, s, v)))$ ($\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))))$ instead of $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$. Observe that $\mathsf{ske.sk}$ is only used in the generation of SKE ciphertexts in the entire game.

Consider $T+1$ intermediate hybrids $G_{0,i}$ where $0 \leq i \leq T$ and T is the number of key queries made by both the adversaries $\mathcal{A}_1, \mathcal{A}_2$. In $G_{0,i}$ hybrid, for the first i key queries, the challenger computes $\mathsf{ske.ct}$ as described in G_1 whereas for the rest of the queries, it computes $\mathsf{ske.ct}$ as described in G_0. Observe that $G_{0,0} = G_0$ and $G_{0,T} = G_1$ whereas $G_{0,i}$ and $G_{0,i+1}$ only differ at the key generation for the i^{th} key query.

We will now show that if there is a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish between any $G_{0,i}$ and $G_{0,i-1}$ where $1 \leq i \leq T$, then we can build an adversary $\mathcal{B}$ that breaks the indistinguishability security of the SKE scheme. For the CPA security game of SKE scheme, the adversary $\mathcal{B}$ would simulate the game G_1 till the $(i-1)^{th}$ key queries. If the i^{th} query is a non-distinguishing query, then it will send $(\mathbf{0}, (0, f(m_0), \mathbf{0}))$ to its challenger (where f is the i^{th} query), who will respond with a challenge ciphertext c^*. It will use c^* as $\mathsf{ske.ct}$ to generate the key in the i^{th} key query and simulate the rest of the game. An important point to note is that $\mathcal{B}$ does not have the secret key $\mathsf{ske.sk}$, so it will have to query its challenger for encryptions to simulate the key generation algorithm. Observe that if c^* is $\mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$, then $\mathcal{B}$ has exactly simulated $G_{0,i-1}$. Else, it has simulated $G_{0,i}$. A similar argument follows if the i^{th} query is a distinguishing query, except that $\mathcal{B}$ will send $(\mathbf{0}, (1, s, v))$ to its challenger. $\square$

Lemma 42. *Assuming* FE *is a selectively secure FE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *there exists a negligible function* $\text{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \text{negl}(\lambda)$.

Proof. The proof of the lemma follows from the selective security of the FE scheme. The only difference between G_2 and G_1 is the message used in the challenge ciphertext. To be precise, in G_1, the challenge ciphertext is an encryption of $M_1 = (m_b, \perp, 0)$ whereas in G_2, it is an encryption of $M_2 = (R, \mathsf{ske.sk}, 1)$.

We first show that for each query f in Pre and Post-challenge Query phase in both the games, the secret key generated decrypts their respective challenge ciphertext to the same value. In both the hybrids, the secret key is generated as $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ where $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$. In G_1, since the challenge ciphertext is encryption of $(m_b, \perp, 0)$, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would give $f(m_b)$ because the the $flag$ bit is set to 0. Whereas in G_2, the challenge ciphertext is encryption of $(R, \mathsf{ske.sk}, 1)$, therefore, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would first decrypt $\mathsf{ske.ct}$ to get $M = (0, f(m_0), \mathbf{0}))$. Since, the first bit of M is 0, it will output the second bit of M (see Figure 2), i.e., $f(m_0)$ which is equal to $f(m_b)$.

We now show that for each distinguishing query f in Second Response Phase in both the games, the secret key decrypts their respective challenge ciphertext to the same value (For non-distinguishing query, the argument is similar to the above paragraph). In both the hybrids, the secret key is generated as $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ where $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, \mathsf{ske.sk}, v))$ where $v = m_b \oplus \mathsf{Ext}_s(R)$. In G_1, since the challenge ciphertext is encryption of $(m_b, \perp, 0)$, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would give $f(m_b)$ because the $flag$ bit is set to 0. Whereas in G_2, the challenge ciphertext is encryption of $(R, \mathsf{ske.sk}, 1)$, therefore, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would first decrypt $\mathsf{ske.ct}$ to get $M = (1, \mathsf{ske.sk}, v))$. Since, the first of M is 1, it computes $f(v \oplus \mathsf{Ext}_s(R))$ (see Figure 2) which is equal to $f(m_b)$. Therefore, the outputs in both the games are identical. □

Lemma 43. *Assuming that* Ext *is a strong average min-entropy extractor, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$.

Proof. From the fact that Ext is a strong average min-entropy extractor and $|\mathsf{st}| \leq S$, we have $(s, m_b \oplus \mathsf{Ext}_s(R), \mathsf{st})$ is statistically close to (s, v, st) where v is a truly random string. □

Lemma 44. *For all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $p_{\mathcal{A},3} \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.

Proof. Observe that the information b is not used anywhere in the entire game. Therefore, an adversary can wins the game with probability at most $1/2 + \mathrm{negl}(\lambda)$. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq 1/2 + \mathrm{negl}(\lambda)$.

F Proof of Theorem 26

We will show that the construction in section 8 is secure using a sequence of hybrid arguments.

G_0: This is the real selective regular incompressible PK-FE security game with challenge bit b.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n - index of a function class and two message (m_0, m_1).
2. The challenger computes $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
3. It computes $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$.
4. It sets $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$ and $\mathsf{mpk} = \mathsf{fe.mpk}$.
5. It sends mpk to $\mathcal{A}_1$.

- **Pre-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{d+2})$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends aux where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly samples $b \leftarrow \{0, 1\}$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.msk}, (m_b, \perp, 0))$ and sends it to $\mathcal{A}_1$.
- **Post-challenge Query Phase:**
  1. For each query f such that $f(m_0) = f(m_1)$ from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{d+2})$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
- **First Response Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\mathsf{st}| \leq S$.
- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, 0^{d+2})$.
     (b) It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
  3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.

G_1: In this game, the challenger modifies the secret keys generates in the query phases as follows.

- **Initialization Phase:**
  1. The first adversary $\mathcal{A}_1$ outputs S - an upper bound on the state size and 1^n - index of a function class and two message (m_0, m_1).
  2. The challenger computes $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

3. It computes $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\tilde{n}})$.
4. It sets $\mathsf{msk} = (\mathsf{ske.sk}, \mathsf{fe.msk})$ and $\mathsf{mpk} = \mathsf{fe.mpk}$.
5. It sends mpk to $\mathcal{A}_1$.
6. It randomly samples $R \leftarrow \{0,1\}^S$ and $s \leftarrow \{0,1\}^d$.

- **Pre-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.

- **Post-challenge Query Phase:**
  1. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.
  2. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.

- **Second Response Phase:**
  1. The second adversary $\mathcal{A}_2$ is given the master public key mpk, the auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
  2. **Key Generation Queries**
     (a) For the *single* query f such that $f(m_0) \neq f(m_1)$ from the second adversary $\mathcal{A}_2$,
        i. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, s, v)))$ where $v = f(m_b) \oplus \mathsf{Ext}_s(R)$.
        ii. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
     (b) For other queries f from $\mathcal{A}_2$
        i. For each query f from the first adversary $\mathcal{A}_1$, the challenger first computes $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$.
        ii. It generates $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ and sends it to $\mathcal{A}_1$.
  3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.

G_2: The challenger computes the challenge ciphertext by encrypting $(R, \mathsf{ske.sk}, 1)$ instead of $(m_b, \perp, 0)$.

- **Challenge Phase:**
  1. The first adversary $\mathcal{A}_1$ sends aux where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
  2. The challenger randomly samples $b \leftarrow \{0,1\}$.
  3. It computes $\mathsf{ct}^* \leftarrow \mathsf{FE.Enc}(\mathsf{fe.msk}, (R, \mathsf{ske.sk}, 1))$ and sends it to $\mathcal{A}_1$.

G_3: The challenger sets v to a truly random string (consistent across all distinguishing key queries) instead of $f(m_b) \oplus \mathsf{Ext}_s(R)$.

Analysis: Let $p_{\mathcal{A},i}$ denote probability of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputting $b' = b$ in Game G_i. We will show that this probability is almost the same in every game.

Lemma 45. *Assuming* SKE *is secure SKE scheme, then for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* λ, $|p_{\mathcal{A},1} - p_{\mathcal{A},0}| \leq \mathrm{negl}(\lambda)$.

Proof. The proof of the lemma follows from the CPA security of the SKE scheme. The only difference between G_0 and G_1 is that in G_1, $\mathsf{ske.ct}$ used in the generation of the distinguishing keys (non-distinguishing keys) is computed as $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, s, v)))$ ($\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))))$ instead of $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$. Observe that $\mathsf{ske.sk}$ is only used in the generation of SKE ciphertexts in the entire game.

Consider $T + 1$ intermediate hybrids $G_{0,i}$ where $0 \leq i \leq T$ and T is the number of key queries made by both the adversaries $\mathcal{A}_1, \mathcal{A}_2$. In $G_{0,i}$ hybrid, for the first i key queries, the challenger computes $\mathsf{ske.ct}$ as described in G_1 whereas for the rest of the queries, it computes $\mathsf{ske.ct}$ as described in G_0. Observe that $G_{0,0} = G_0$ and $G_{0,T} = G_1$ whereas $G_{0,i}$ and $G_{0,i+1}$ only differ at the key generation for the i^{th} key query.

We will now show that if there is a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can distinguish between any $G_{0,i}$ and $G_{0,i-1}$ where $1 \leq i \leq T$, then we can build an adversary $\mathcal{B}$ that breaks the indistinguishability security of the SKE scheme. For the CPA security game of SKE scheme, the adversary $\mathcal{B}$ would simulate the game G_1 till the $(i-1)^{th}$ key queries. If the i^{th} query is a non-distinguishing query, then it will send $(\mathbf{0}, (0, f(m_0), \mathbf{0}))$ to its challenger (where f is the i^{th} query), who will respond with a challenge ciphertext c^*. It will use c^* as $\mathsf{ske.ct}$ to generate the key in the i^{th} key query and simulate the rest of the game. An important point to note is that $\mathcal{B}$ does not have the secret key $\mathsf{ske.sk}$, so it will have to query its challenger for encryptions to simulate the key generation algorithm. Observe that if c^* is $\mathsf{SKE.Enc}(\mathsf{ske.sk}, \mathbf{0})$, then $\mathcal{B}$ has exactly simulated $G_{0,i-1}$. Else, it has simulated $G_{0,i}$. A similar argument follows if the i^{th} query is a distinguishing query, except that $\mathcal{B}$ will send $(\mathbf{0}, (1, s, v))$ to its challenger. □

Lemma 46. *Assuming* FE *is a selectively secure FE scheme, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},2} - p_{\mathcal{A},1}| \leq \mathrm{negl}(\lambda)$.

Proof. The proof of the lemma follows from the selective security of the FE scheme. The only difference between G_2 and G_1 is the message used in the challenge ciphertext. To be precise, in G_1, the challenge ciphertext is an encryption of $M_1 = (m_b, \perp, 0)$ whereas in G_2, it is an encryption of $M_2 = (R, \mathsf{ske.sk}, 1)$.

We first show that for each query f in Pre and Post-challenge Query phase in both the games, the secret key generated decrypts their respective challenge ciphertext to the same value. In both the hybrids, the secret key is generated as $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ where $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (0, f(m_0), \mathbf{0}))$. In G_1, since the challenge ciphertext is encryption of $(m_b, \perp, 0)$, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would give $f(m_b)$ because the the $flag$ bit is set to 0. Whereas in G_2, the challenge ciphertext is encryption of $(R, \mathsf{ske.sk}, 1)$, therefore, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would first decrypt $\mathsf{ske.ct}$ to get $M = (0, f(m_0), \mathbf{0}))$. Since, the first bit of M is 0, it will output the second bit of M (see Figure 3), i.e., $f(m_0)$ which is equal to $f(m_b)$.

We now show that the distinguishing query f in Second Response Phase in both the games, the secret key decrypts their respective challenge ciphertext to the same value (For non-distinguishing query, the argument is similar to the above paragraph). In both the hybrids, the secret key is generated as $\mathsf{fe.sk}_f \leftarrow \mathsf{FE.Keygen}(\mathsf{fe.msk}, C_{f,\mathsf{ske.ct}})$ where $\mathsf{ske.ct} = \mathsf{SKE.Enc}(\mathsf{ske.sk}, (1, \mathsf{ske.sk}, v))$ where $v = f(m_b) \oplus \mathsf{Ext}_s(R)$. In G_1, since the challenge ciphertext is encryption of $(m_b, \perp, 0)$, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would give $f(m_b)$ because the $flag$ bit is set to 0. Whereas in G_2, the challenge ciphertext is encryption of $(R, \mathsf{ske.sk}, 1)$, therefore, the decryption of the challenge ciphertext using $\mathsf{fe.sk}_f$ would first decrypt $\mathsf{ske.ct}$ to get $M = (1, \mathsf{ske.sk}, v))$. Since, the first of M is 1, it computes $v \oplus \mathsf{Ext}_s(R)$ (see Figure 3) which is equal to $f(m_b)$. Therefore, the outputs in both the games are identical. □

Lemma 47. *Assuming that* Ext *is a strong average min-entropy extractor, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*$|p_{\mathcal{A},3} - p_{\mathcal{A},2}| \leq \mathrm{negl}(\lambda)$.

Proof. From the fact that Ext is a strong average min-entropy extractor and $|\mathsf{st}| \leq S$, we have $(s, f(m_b) \oplus \mathsf{Ext}_s(R), \mathsf{st})$ is statistically close to (s, v, st) where v is a truly random bit. □

Lemma 48. *For all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*$p_{\mathcal{A},3} \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.

Proof. Observe that the information b is not used anywhere in the entire game. Therefore, an adversary can wins the game with probability at most $1/2 + \mathrm{negl}(\lambda)$. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq 1/2 + \mathrm{negl}(\lambda)$.