Functional Encryption for Turing Machines with Dynamic Bounded Collusion from LWE

Shweta Agrawal*, Monosij Maitra**, Narasimha Sai Vempati***, and Shota Yamada†

Abstract. The classic work of Gorbunov, Vaikuntanathan and Wee (CRYPTO 2012) and follow-ups provided constructions of bounded collusion Functional Encryption (FE) for circuits from mild assumptions. In this work, we improve the state of affairs for bounded collusion FE in several ways:

1. *New Security Notion.* We introduce the notion of *dynamic* bounded collusion FE, where the declaration of collusion bound is delayed to the time of encryption. This enables the encryptor to dynamically choose the collusion bound for different ciphertexts depending on their individual level of sensitivity. Hence, the ciphertext size grows linearly with its own collusion bound and the public key size is independent of collusion bound. In contrast, all prior constructions have public key and ciphertext size that grow at least linearly with a fixed bound Q.
2. *CPFE for circuits with Dynamic Bounded Collusion.* We provide the first CPFE schemes for circuits enjoying dynamic bounded collusion security. By assuming identity based encryption (IBE), we construct CPFE for circuits of *unbounded* size satisfying *non-adaptive* simulation based security. By strengthening the underlying assumption to IBE with receiver selective opening security, we obtain CPFE for circuits of *bounded* size enjoying *adaptive* simulation based security. Moreover, we show that IBE is a necessary assumption for these primitives. Furthermore, by relying on the Learning With Errors (LWE) assumption, we obtain the first *succinct* CPFE for circuits, i.e. supporting circuits with unbounded size, but fixed output length and depth. This scheme achieves *adaptive* simulation based security.
3. *KPFE for circuits with dynamic bounded collusion.* We provide the first KPFE for circuits of unbounded size, but bounded depth and output length satisfying dynamic bounded collusion security from LWE. Our construction achieves *adaptive* simulation security improving security of [20].
4. *KP and CP FE for TM/NL with dynamic bounded collusion.* We provide the first KPFE and CPFE constructions of bounded collusion functional encryption for Turing machines in the public key setting from LWE. Our constructions achieve non-adaptive simulation based security. Both the input and the machine in our construction can be of *unbounded* polynomial length. We provide a variant of the above scheme that satisfies *adaptive* security, but at the cost of supporting a smaller class of computation, namely Nondeterministic Logarithmic-space (NL). Since NL contains Nondeterministic Finite Automata (NFA), this result subsumes *all* prior work of bounded collusion FE for uniform models from standard assumptions [7, 9].

* IIT Madras, shweta.a@cse.iitm.ac.in
** TU Darmstadt, monosij.maitra@tu-darmstadt.de
*** IIT Madras, narasimhasai07@gmail.com
† AIST Japan, yamada-shota@aist.go.jp

1 Introduction

Functional encryption [29, 13] is a generalization of public key encryption which allows fine grained control on disclosure of encrypted data. In functional encryption (FE), a secret key is associated with a function f, a ciphertext is associated with an input $\mathbf{x}$ and decryption reveals $f(\mathbf{x})$ and nothing more. Security requires that any collusion of users cannot learn more about the ciphertext beyond what they are individually authorized to learn – this property is known as *collusion resistance*.

In a classic work, Gorbunov, Vaikuntanathan and Wee [21] provided the first construction of Functional Encryption for circuits in the *bounded collusion* model – namely in a model where the scheme may generate an unbounded number of keys but security holds only against an adversary who obtains at most an a-priori bounded number of keys, say Q. Their construction supports all polynomial sized circuits, and is based on the existence of public key encryption (PKE) and pseudorandom generators (PRG) in NC^1. Since circuits are a powerful model of computation, this work provides a strong feasibility result, and moreover, from weak assumptions. Subsequent work provided other useful improvements: the work of Ananth and Vaikuntanathan [12] removed the assumption of PRGs so that the resulting scheme relies on the minimal assumption of PKE, while a series of works [8, 1, 12] improved the dependence of the public key and ciphertext on the collusion bound Q. A parallel line of work has studied the construction of functional encryption schemes supporting unbounded collusions [2, 10, 23, 24, 17], from standard assumptions, culminating in the recent breakthrough of Jain, Lin and Sahai [24] which achieves this much sought-after goal. However, this intricate construction relies on several assumptions including pairings making it quantum insecure, and has many complex reductions that incur significant loss in efficiency. Hence, it remains meaningful to consider simpler, plausibly post-quantum constructions, even in weaker security models.

Limitations of Prior Work. Despite their success, existing constructions of FE in the bounded collusion model suffer from at least three major drawbacks: i) the collusion bound Q must be declared at setup time and is fixed once and for all, ii) in the public key setting, these constructions support only the circuit model of computation and iii) all constructions that we are aware of are in the *key policy* setting (KPFE), where the function f is embedded in the secret key and the input $\mathbf{x}$ is embedded in the ciphertext of the FE scheme. The dual, ciphertext policy setting (CPFE), where the roles of f and $\mathbf{x}$ are swapped, is more natural in several applications but has received much less attention. We discuss each of these limitations in turn.

The first limitation pervades all prior work to the best of our knowledge and is quite a significant drawback in our opinion. Since the collusion bound must be declared at setup time, all data encrypted under the scheme must necessarily be subject to the same level of collusion-resistance. Thus, the practioner is forced to choose a collusion bound which is strong enough for the *most sensitive* information that may ever be encrypted under the scheme – this implies that Q is an upper bound on collusion resistance. However, not all information has the same level of sensitivity. Consider an organization: messages involving a potential merger with another organization, or a case of harassment which

must be investigated are significantly more delicate than routine exchanges. It is desirable that such sensitive messages are protected even if a large number of key holders collude. On the other hand, for routine ciphertexts decrypted via function keys (such as checking whether some encrypted message is spam or not), such a strong level of collusion resistance is unnecessary. Moreover, the collusion bound Q impacts the size of the public key and ciphertext of the scheme. Thus, if Q is large, then the public key as well as every ciphertext generated by the scheme is forced to grow at least linearly in Q leading to a prohibitive impact on efficiency.

Regarding the second limitation, it is well known that being non-uniform, the circuit model is ill-suited for several applications [19, 11, 9, 5]. In particular, circuits force the size of the input to be fixed a-priori which in turn necessitates instantiation of the scheme with an upper bound on data size. This again leads to loss in efficiency and is ill-suited to datasets of dynamic size. Moreover, circuits incur *worst case* running time over all inputs, which is also clearly undesirable in practice. Finally, all constructions of CPFE for circuits that we are aware of, proceed via the universal circuit route, i.e. consider the universal circuit $U(f, \mathbf{x}) = f(\mathbf{x})$ and construct KPFE with the circuit $U(\cdot, \mathbf{x})$ in the secret key and f represented as a string, in the ciphertext. Aside from the loss of efficiency that results by this transformation (now, both the input and the function grow with the maximum circuit size), this transformation restricts the CPFE scheme to *bounded* size circuits. This is dissatisfying, as much in practice as in theory. The ciphertext policy variant of FE is desirable in many applications. Even in the special case of attribute based encryption (ABE), the ciphertext policy model allows an access control policy f to be embedded against a secret message m in the ciphertext. The secret key contains user attributes $\mathbf{x}$ which represent the various roles of a user, such as institute, department, date of joining and such others. Decryption succeeds to recover m if and only if the user's attributes satisfy the access control policy in the ciphertext. It is arguably more natural to have an access control policy apply to a secret message than to a user. Another application scenario is when the function f is proprietary and must be hidden via encryption, while the data can be made publicly available. For instance, suppose a government wants to enable citizens to run useful algorithms developed by different research labs on some public data, eg. census data. The government publishes a secret key for the data x, the labs publish ciphertexts with their respective algorithms f_i (of arbitrary size) and anyone can compute $f_i(x)$. The research labs want to keep their algorithms secret from competitors (but the government agency is trusted) so the function is encrypted. The data is public but becomes available asynchronously and independently of the function(s), so the labs cannot compute the function outputs each time. For example, new census data may be published every year but the same programs can be used every year. Computing private programs on public data is analogous to the setting of obfuscation, except that here the government agency can be trusted, making it simpler than obfuscation. Given the many natural applications of CPFE, it is undesirable to settle for a limited generic transformation via KPFE.

State of the Art. While FE for uniform models of computation has been studied [19, 11, 9, 5], direct constructions from standard assumptions, supporting unbounded length inputs, are few and far-between. In the public key setting, the FE scheme by Agrawal

and Singh [9] supports Turing machines and is based on the Learning With Errors assumption. However, this scheme only supports a single key request by the adversary in the security game. In the symmetric key setting, the recent work of Agrawal, Maitra and Yamada [7] provided a construction of FE for non-deterministic finite automata (NFA) which is secure against bounded collusions of arbitrary size. However, generalizing this construction to the public key setting and to stronger models like Turing machines was left open. To the best of our knowledge, all bounded collusion FE schemes suffer from the fixed collusion bound, and the only CPFE scheme that supports unbounded sized circuits is that by Sahai and Seyalioglu [28], which is only single key secure.

1.1 Our Results

In this work, we improve the state of affairs in several ways:

1. *New Security Notion.* We introduce the notion of *dynamic* bounded collusion FE, where the declaration of collusion bound is delayed to the time of encryption. This enables the encryptor to dynamically choose the collusion bound for different ciphertexts depending on their individual level of sensitivity. Hence, the ciphertext size grows linearly with its own collusion bound and the public key size is independent of collusion bound. In contrast, all prior constructions have public key and ciphertext size that grow at least linearly with a fixed bound Q. All our constructions satisfy our new security notion – we also refer to our new notion as achieving *delayed* collusion resistance.
2. *CPFE for circuits with Dynamic Bounded Collusion.* We provide the first CPFE schemes for circuits enjoying dynamic bounded collusion security. In more detail:
   - By relying on the assumption of identity based encryption (IBE), we construct CPFE for circuits of *unbounded* size, output length and depth satisfying *non adaptive* simulation based security. Recall that non-adaptive simulation security refers to a game where the attacker must make all it's key requests before obtaining the challenge ciphertext [13].
   - By strengthening the underlying assumption to IBE with receiver selective opening security, we obtain CPFE for circuits of *bounded* size, output length and depth enjoying *adaptive* simulation based security[1]. Moreover, we show that IBE is a necessary assumption for these primitives.
   - By relying on the Learning With Errors (LWE) assumption, we obtain a *succinct* CPFE for circuits – namely, supporting circuits with unbounded size, but fixed output length and depth, which achieves *adaptive* simulation based security.
3. *KPFE for circuits with dynamic bounded collusion.* We provide the first KPFE for circuits of unbounded size, but bounded depth and output length satisfying dynamic bounded collusion. Our construction relies on LWE and achieves adaptive simulation based security. This improves the security of succinct KPFE by Goldwasser et al. [20].

[1] For the knowledgeable reader, the lower bound from [13] does not apply because there is only one challenge ciphertext, with bounded output length in the security game.

4. *KP and CP FE for TM/NL with dynamic bounded collusion.* We provide the first KPFE and CPFE constructions of bounded collusion functional encryption for Turing machines in the public key setting from LWE. Such a result was not known even with fixed collusion resistance to the best of our knowledge. Our constructions achieve non-adaptive simulation based security. In terms of functionality: a secret key in our KPFE construction encodes an TM M, a ciphertext encodes a message $\mathbf{x}$ and decryption allows recovery of $M(\mathbf{x})$ and nothing else. Both the input $\mathbf{x}$ and the machine M in our construction can be of *unbounded* polynomial length but the ciphertext size grows with the upper bound on the running time of the Turing machine on the given input $\mathbf{x}$. Given RAM access to the ciphertext, the scheme enjoys input specific decryption time.
5. *Adaptive Bounded Collusion FE for NL with dynamic bounded collusion.* The above construction guarantees non-adaptive security while supporting general Turing machines. We also consider a variant of the above scheme that satisfies stronger adaptive security, but at the cost of supporting smaller class of computation Nondeterministic Logarithmic-space (NL). Since NL contains Nondeterministic Finite Automata (NFA), this result subsumes *all* prior work of bounded collusion FE for uniform models from standard assumptions [7, 9].

1.2 Our Techniques

In this section, we provide an overview of our techniques. At a high level, our work addresses two broad challenges: obtaining stronger security guarantees via dynamic collusion and achieving more powerful functionality via general TM or NL. We examine each of these in turn.

Dynamic Bounded Collusion. Observe that the problem of constructing FE with delayed collusion bounds has not been studied until our work, even for bounded size circuits. A first observation is that in such an FE scheme, it is *necessary* that the efficiency of the setup and key generation algorithms are independent of (or dependent only poly-logarithmically on) the collusion bound Q. To the best of our knowledge, previous bounded FE schemes such as [21, 8, 1] do not satisfy this property. However, the recent construction by Ananth and Vaikuntanathan [12] (AV19) does satisfy one half of this requirement – it enjoys a key generation algorithm which is efficient in this sense. Unfortunately, the setup algorithm of their construction runs in time that grows with Q. Our first step will be to remove the dependency on Q from the setup algorithm.

Improving AV19 to remove setup dependence on Q. To begin, we recap some relevant ideas from their construction. Their construction is generic: they construct Q-bounded FE scheme from a single key FE scheme. For concreteness, we instantiate the single key FE scheme with the concrete one by Sahai and Seyalioglu [28]. While their construction is key policy, we adapt it to the ciphertext policy setting in what follows. At a very high level, their Q bounded FE scheme runs Q subsystems in parallel, such that each subsystem in turn runs N instances of the SS10 scheme. Since their construction is optimized using an elegant combinatorial argument, they obtain a secret key size of poly rather than $O(Q \cdot \mathrm{poly})$. This optimized construction forms the starting point of our

work. The specific details of their final construction are not relevant to this overview: we note only some salient features. Their construction makes use of an "MPC style" secret sharing scheme, where an input circuit C is divided into shares $\widehat{C}_1, \ldots, \widehat{C}_N$. Now, n out of N parties, without any interaction, perform some computation on their shares corresponding to input x, to obtain partial outputs $\widehat{y}_1, \ldots, \widehat{y}_n$. These n partial outputs are then combined to obtain output $y = C(x)$.

Using the above secret sharing scheme, the AV19 construction may be summarized as follows. The setup algorithm generates several (the exact number is not relevant for us) public and secret key pairs of a PKE scheme. To encrypt a circuit C, the encryptor computes many shares of C as described above. These shares $\widehat{C}_i$ are then hardwired into garbled circuits which, given input x, compute $\widehat{y}_i$ using the above method. The labels of these garbled circuits are encrypted using the public keys provided by the setup algorithm. The function key for x is a set of PKE secret keys that depend on x (again, the precise dependence is not required here). The decryptor first uses the PKE secret keys to recover the appropriate labels of the garbled circuit corresponding to x. Then the garbled circuit is evaluated to obtain shares $\widehat{y}_i$ of the decryption result. These shares are then combined to obtain $C(x)$.

The reason why their setup algorithm takes time linear in the collusion bound Q is that the number of public keys required by their scheme is linear in Q. Having unrolled their construction when instantiated with [28], our approach to removing this dependency is simple: we replace these public keys of PKE with a single master public key of an identity based encryption (IBE) scheme. Then, encryption with PK_i is replaced by an encryption with $\mathsf{IBE.Enc}(\mathsf{IBE.mpk}, i, \cdot)$, where i is the identity. The intuition for security is the same as the original construction. Thus, we use the power of the IBE to hide the labels of the garbled circuits, use the power of the garbled circuits to hide information other than the decryption shares, and finally use the power of the secret sharing scheme to hide the circuit C.

We show that if we desire adaptive simulation (AD-SIM) security for the resultant construction, we need an IBE satisfying the stronger security notion of *receiver selective opening security* [25]. This is for a reason similar to why [21, 12] required non-committing security for the underlying public key encryption. Since the length of the message that can be encrypted using an IBE with receiver selective opening security is bounded, so is the size of the circuits that can be encrypted in our CPFE scheme. If we relax the security notion and consider non-adaptive (NA-SIM) security, we can use IBE with standard IND-CPA security. This allows us to encrypt a message of unbounded length, which in turn allows us to encrypt circuits with unbounded size. A simple adaptation of the lower bound of Boneh et al. [13] shows that to support unbounded sized circuits, NA-SIM security is optimal[2].

[2] Consider a circuit C^* with unbounded size and unbounded output length. Let us say that this circuit has hardwired with random string s of length ℓ, and upon an input x, the circuit ignores the input and outputs s. Here, ℓ is unbounded. Now if the attacker makes even a single key request for some x^* *after* seeing the challenge ciphertext corresponding to C^*, the simulator is faced with the impossible task of embedding a random string of length ℓ into a fixed sized secret key. Hence, the adversary must not be allowed post-challenge key requests when circuits of unbounded output length are supported.

Supporting Delayed Collusion in Security. So far, we have constructed bounded CPFE schemes whose setup and key generation algorithms run in time independent of Q. However, this is only necessary and not sufficient to construct FE with delayed collusion bound. Once the system is set up with the bound Q, this will still only be secure against a collusion of size Q. Our next step is to remove this restriction so that the encryptor can choose the bound flexibly, and in particular, differently for each ciphertext.

Here, our crucial observation is that the setup and key generation algorithms of the scheme can be run even for super polynomial Q, thanks to their efficiency properties. Hence, we may use the "powers of two trick" [19], where we run the system with different collusion bounds $Q = 2, 2^2, \ldots, 2^\lambda$. The setup and key generation algorithms are run for these λ subsystems in parallel. When we encrypt the message, the encryptor chooses the smallest 2^i that exceeds the bound Q it wants and encrypts the message using the i-th subsystem. Decryption is performed using the secret key for the i-th instance of the subsystem. The resulting scheme inherits the efficiency and security properties of the subsystem. Namely, if the subscheme is NA-SIM (respectively AD-SIM) secure and supports unbounded (respectively bounded) circuits, so does the resulting scheme. Thus, we obtain two schemes with incomparable properties. Please see Section 3 for details.

Observe that the construction of bounded collusion FE in AV19 can be based on the minimal assumption of plain PKE [12]. On the other hand, our constructions described above rely on the stronger primitive of IBE. It is natural to ask whether we can base the security of the construction to weaker primitives such as PKE. We answer this question negatively. In our full version [6] we argue that the usage of IBE is unavoidable, by showing that FE with dynamic bounded collusion for very small class of functionalities already implies IBE.

Supporting More General Function Classes. Next, we describe our techniques for supporting more flexible models of computation, namely Turing machines or NL. The main difficulty of constructing FE for these function classes is to handle unbounded length inputs and unbounded size machines simultaneously. To address this, we borrow a trick from the work of Agrawal, Maitra and Yamada [7]: instead of trying to handling them at once, we construct intermediate schemes that can handle an unbounded size object on one side, but bounded size object on the other. Towards this, we construct KPFE and CPFE schemes with dynamic bounded collusion that support unbounded size circuits, but with bounded output length and depth (note that input length is always fixed). Later, we will see how to compile these to construct FE for more general function classes. Note that in the previous step we already constructed a CPFE scheme that can handle circuits with unbounded size even without restrictions on depth and output length. However, this construction was only NA-SIM secure. Here, we aim to construct schemes with AD-SIM security.

Succinct KPFE and CPFE. Let us start with the construction of KPFE that can support unbounded size circuits. Note that such FE schemes have already been constructed by the previous works under the name of succinct FE [20, 1]. However, they do not satisfy the security requirement that we want. Namely, they are (conventional) bounded collusion schemes and do not satisfy the delayed collusion property. In addition, they only satisfy NA-SIM security. Here, we upgrade the security of existing succinct FE schemes so that

they satisfy the delayed collusion property and AD-SIM security with the help of our CPFE scheme for *bounded* circuits that already satisfies the desired security properties. In more detail, we combine succinct single-key KPFE, denoted by $1\mathsf{KPFE}$ with our AD-SIM secure CPFE for bounded circuits, to obtain a new succinct KPFE with the desired security properties.

At a high level, the construction works as follows. The master public key and master secret key of the final KPFE scheme are those of the CPFE. To encrypt a message x for a collusion bound 1^Q, the encryptor first constructs a circuit $1\mathsf{KPFE.Enc}(\cdot, x)$[3], which is an encryption algorithm of the single-key KPFE that takes as input a master public key of the single-key KPFE and outputs an encryption of the message x under the key. The encryptor then encrypts the circuit using the CPFE scheme with respect to the bound 1^Q. To generate a secret key for a circuit C, we first freshly generate a master key pair of the single-key KPFE $(1\mathsf{KPFE.mpk}, 1\mathsf{KPFE.msk})$. We then generate a CPFE secret key $\mathsf{CPFE.sk}$ corresponding to the string $1\mathsf{KPFE.mpk}$ and then generate secret key $1\mathsf{KPFE.sk}_C$ for the circuit C of the single-key KPFE scheme. The final secret key is $(\mathsf{CPFE.sk}, 1\mathsf{KPFE.sk}_C)$. Decryption is done by first decrypting the CPFE ciphertext using the CPFE secret key to recover $1\mathsf{KPFE.Enc}(1\mathsf{KPFE.mpk}, x)$ and then decrypting it using the secret key $1\mathsf{KPFE.sk}_C$ of the single-key KPFE scheme to recover $C(x)$.

We discuss the efficiency of the above scheme. First we claim that the above scheme is succinct (or equivalently, can deal with unbounded size of circuits). This is the case even though underlying CPFE can only deal with bounded size circuits, since the size of circuits that should be supported by the encryption algorithm of CPFE is $|1\mathsf{KPFE.Enc}(\cdot, x)| = \mathrm{poly}(\lambda, |x|)$, which is independent of the size of circuits by the succinctness of the underlying $1\mathsf{KPFE}$ scheme. Next, we discuss the security of the scheme. Intuitively speaking, by the security of the underlying CPFE, the adversary can obtain no information beyond the decryption result of the CPFE. This means that, the adversary only obtains the information $\{1\mathsf{KPFE.Enc}(1\mathsf{KPFE.mpk}^{(i)}, x), 1\mathsf{KPFE.sk}^{(i)}_{C^{(i)}}\}_{i\in[Q]}$ for Q freshly generated, independent instances. In turn, this implies that the adversary can only obtain the information of $\{C^{(i)}(x)\}_{i\in[Q]}$ by the single-key security of the underlying KPFE, as desired. A formal argument shows that the resulting KPFE scheme inherits AD-SIM security of the CPFE, even if the underlying single-key KPFE is only NA-SIM secure. We refer to the Section 4 for details.

Next, we discuss the adaptation to the CPFE setting. In this construction, we use a reusable garbled circuit scheme [20] instead of single-key succinct KPFE. Recall that a reusable garbled circuit is a symmetric key variant of single key succinct KPFE, where the circuit in the secret key is also hidden. Now, to encrypt a circuit C in our CPFE, we run the garbling algorithm of the reusable garbled circuit scheme on input C to obtain the garbled version of C as well as some secret information for input garbling. We then construct an input encoding circuit that takes as input x and outputs an encoded version of it using the secret information generated above. Then, we encrypt this input encoding circuit using the underlying CPFE. The final ciphertext consists of the garbled circuit and the encrypted circuit. To generate a secret key for x, we use the key generation algorithm of the underlying CPFE scheme to obtain a secret key for x. Decryption requires running

[3] The description here is oversimplified – in fact we need a PRF to derive the randomness for the encryption.

the decryption algorithm of the underlying CPFE scheme to obtain the encoded version of input x and then using this result with the garbled version of the circuit to recover $C(x)$. The resulting scheme is AD-SIM secure and supports unbounded size circuits. Please see our full version [6] for details.

Handling Unbounded Inputs. So far, we have constructed KPFE and CPFE schemes that can handle unbounded size circuits but with fixed input size. However, for obtaining FE for Turing machines, we have to be able to encrypt unbounded length inputs and this is clearly insufficient. To enable this, we use the "delayed encryption" technique by Goyal, Koppula, and Waters (GKW16) [22]. Intuitively, the technique uses the power of garbled circuits and IBE to transport us to a world where there are infinitely many instances of FE $\{\mathsf{mpk}_i\}_{i\in\mathbb{N}}$ – the encryptor can choose a master public key for some index j and encrypt the message. A secret key is associated with some index k and the decryption is possible iff $j = k$.

In more detail, the GKW16 scheme works as follows. During setup, the master public key and master secret key of an IBE scheme are generated. To encrypt a message x using the j-th instance of the FE scheme as described above without knowing the corresponding mpk_j, the encryptor first constructs a circuit $\mathsf{Enc}(\cdot, x)$ that takes as input the master public key mpk_j for the j-th instance and outputs the ciphertext $\mathsf{Enc}(\mathsf{mpk}_j, x)$. The encryptor then garbles this encryption circuit to obtain the corresponding garbled circuit and set of labels. Then, the encryptor encrypts each pair of labels with the IBE, where the identity for which a label is encrypted encodes the index j, the position of the label and a single bit. Note that the above step can be done without knowing mpk_j. Correspondingly, the key generation algorithm computes IBE secret keys for the correct bits of mpk_k, which together with the IBE ciphertexts allow the decryptor to recover the labels corresponding to mpk_j in the j^{th} garbled circuit when $j = k$. This lets the decryptor evaluate the garbled circuit to retrieve the ciphertext $\mathsf{Enc}(\mathsf{mpk}_j, x)$ as desired.

With this technique, we make progress towards our goal, because we can encrypt a message of any length by the scheme, rather than only being able to encrypt a message of fixed length. However, this is still not enough, since the decryption is possible only when the index j and k match. For example, let us imagine that we want to construct FE for Turing machine using infinitely many instances of FE for circuits, where the i-th instance of FE supports circuits with input length i. Let t be an upper bound on the runtime of the TM on input x. If we encrypt a message $(x, 1^t)$ as an input to a Turing machine, we may encrypt it using $|(x, 1^t)|$-th instance of the FE. On the other hand, to generate a secret key for a Turing machine M, we convert the machine into a circuit $C_M(\cdot)$ and generate a secret key for it. However, it is unclear how to define the input length of the circuit. If the input length does not match $|(x, 1^t)|$, the decryption is impossible. Meanwhile, the entity who generates the secret key does not know the input length $|(x, 1^t)|$, so is stuck.

To resolve the above problem, we incorporate a trick by Agrawal, Maitra and Yamada [7] used to support unbounded inputs for an NFA machine in the context of ABE. They construct two restricted ABE schemes for NFA: one that supports decryption in the case where the length $|x|$ of the input x is larger than the size $|M|$ of the machine M and one that supports the case where $|x| \leq |M|$. Then, they run the restricted schemes in parallel. In the decryption algorithm, these sub-schemes complement each other. Namely, we use the first sub-scheme to decrypt a ciphertext if $|x| > |M|$ and the second otherwise.

Though they introduce the trick in the context of ABE, this perfectly works in the context of FE as well. A hurdle is that their technique works only in the secret key setting, since the encryptor is required to know a master secret in order to generate unbounded instances of FE (proportional to its input length) on the fly. However, we show that in conjunction with the technique from [22] described above, this idea can be made to work in the public key setting as well. In a nutshell, this technique lets us encode x and M in multiple slots of the FE instances so that they always intersect, instead of encoding them on a single slot like in [22].

Onward to FE for TM. Armed with these techniques, let us try constructing FE for TM. As discussed above, our construction handles the cases $|(x, 1^t)| \leq |M|$ and $|(x, 1^t)| > |M|$ separately. Let us begin with the former using our KPFE that supports unbounded size circuits.

By the technique of [22], we can assume that we are in a world where there are infinitely many KPFE instances available and the i-th instance supports circuits with input length i. To encrypt a message x with respect to the time bound 1^t, we use the $|(x, 1^t)|$-th instance of the KPFE. To generate the secret key for a Turing machine M on the other hand, we encode M into a set of circuits $C_{i,M}$ for $i = 1, \ldots, |M|$, where $C_{i,M}$ is a circuit that takes as input a string $(x, 1^t)$ and then run the machine M for t steps and outputs the result. We then generate secret keys for $C_{i,M}$ using the i-th instance of KPFE for all of $i \in [|M|]$. This is possible even for unbounded M, because each KPFE instance supports unbounded size circuits. The decryption is possible when $|(x, 1^t)| \leq |M|$ by using the $|(x, 1^t)|$-th instance. However, it is evident that this is an incomplete scheme, since the decryption is not possible when $|(x, 1^t)| > |M|$.

To complement this, we next construct a scheme that deals with the case of $|(x, 1^t)| > |M|$ using our unbounded CPFE scheme. To encrypt a message $(x, 1^t)$ we convert it into a circuit $\{U_{i,x,t}\}_{i \in [|(x,1^t)|]}$, where $U_{i,x,t}$ is a circuit that takes as input a string M of length i, interprets it as a description of a Turing machine, and then runs it on input x for t steps to obtain the result. We then encrypt the circuit $U_{i,x,t}$ using the i-th instance of the FE for all of $i \in [|(x, 1^t)|]$. This requires the underlying CPFE scheme to support unbounded size of circuits. Since our NA-SIM secure CPFE construction supports such circuits, we can use this here. On the other hand, our CPFE scheme with AD-SIM security cannot be used here, because the scheme can only support circuits with bounded depth, which prohibits us from running the Turing machine inside the circuit for t steps, where t may be arbitrarily large. To generate a secret key for a Turing machine M, we use the $|M|$-th instance of the FE. It can be seen that the decryption is possible in this scheme when $|(x, 1^t)| > |M|$. Having the construction for the cases of $|(x, 1^t)| > |M|$ and $|(x, 1^t)| \leq |M|$, we can obtain the final construction by running them in parallel. The final scheme is NA-SIM secure, because the underlying CPFE scheme is NA-SIM secure (even though the KPFE scheme satisfies the stronger AD-SIM security). Please see Section 5 for details.

Above, the ciphertext size grows with the t, the upper bound on the runtime of the TM on a given input x. We emphasize that t is not a global bound but can vary with each input x, and is therefore unbounded. While we do not know how to remove this dependence using our current techniques, we remark that decryption time can still

be input specific using the "powers of two" trick from Goldwasser et al. [19]. This requires the decryption algorithm to have RAM access to the ciphertext. In more detail, the encryptor may repeat the encryption procedure for $\lceil \log_2 t \rceil$ possible values of TM runtime, with values 2^i for $i \in [\lceil \log_2 t \rceil]$. The decryptor can start with the ciphertext corresponding to the smallest value and proceed to the next ciphertext only if the previous decryption did not result in a valid output[4]. This ensures that the scheme enjoys input specific decryption time.

FE for NL with Adaptive Security. The above construction guarantees NA-SIM security while supporting general Turing machines. We also consider a variant of the above scheme that satisfies stronger AD-SIM security, at the cost of supporting smaller class of computation NL. This is achieved by using AD-SIM secure FE for both the building blocks of KPFE and CPFE. Recall that the reason why we cannot use AD-SIM secure CPFE in the above construction was that it was not possible to run the Turing machine for an unbounded number of steps inside a circuit of fixed depth. This issue arises because the Turing machine is run sequentially. In a nutshell, our next idea is to parallelize computation so that the depth for the corresponding circuit can be bounded by some fixed polynomial. Such a parallelization of the computation is not known to be possible for general Turing machines, but we can do it for NL, which is more restrictive. To do so, we represent the computation of NL as a multiplication of matrices as was done in [26]. First, we enumerate all the possible internal configurations of the Turing machine M on input x that may appear during the computation. The number of such internal configurations can be bounded by some polynomial, since the length of the working tape is logarithmic. We then construct the transition matrix $\mathbf{M}$ for the configurations. Then, one can determine whether M accepts the input x within time t by computing $\mathbf{M}^t$ due to the properties of the transition matrix. Since the matrix exponentiation can be done by $O(\log t)$ multiplications of the matrix, this can be performed by fixed polynomial depth even for unbounded t (assuming $t < 2^\lambda$). We refer to the full version [6] for details.

1.3 Concurrent Work

A concurrent work [16] independently introduced the notion of dynamic collusion bound for KPFE schemes, which is the same as what we consider in this paper. They obtain simulation secure KPFE schemes for circuits with dynamic collusion resistance. Further, their techniques also significantly overlap with our CPFE constructions in Section 3. In particular, they compile existing bounded collusion KPFE schemes [21, 12] with IND-CPA secure IBE to obtain KPFE for circuits satisfying dynamic collusion bound property. However, we also extend our results further to obtain succinct CP/KP-FE schemes for circuits with dynamic collusion property, and also to support Turing machines and NL with various security tradeoffs as explained in Section 1.2. All these constructions support dynamic collusion resistance. Furthermore, we also argue about the necessity of IBE to achieve dynamic collusion bound property for FE schemes, which is left as an open problem in [16].

[4] The definition of TM can be easily modified to output "unfinished" if the computation did not conclude in a given number of steps.

2 Preliminaries

In this section, we define some notation and preliminaries that we require. Please see the full version [6] for some additional preliminaries.

2.1 Functional Encryption

Functional encryption (FE) [29, 13, 27] has been traditionally defined in a setting where a trusted key generator holding a master secret key provides authorized users with secret keys corresponding to functions. Such a key, when used to decrypt ciphertexts, reveals only the function of the plaintexts and nothing else. In this subsection, we define the notion of functional encryption (FE) more generally so that it captures the above notion as well as other types of functionalities as special cases.

2.1.1 Syntax and Correctness. Let $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}^*$ be a two-input function where $\mathcal{X}$ and $\mathcal{Y}$ denote "message space" and "key attribute space", respectively. Ideally, we would like to have an FE scheme that handles the relation R directly, where we can encrypt any message $x \in \mathcal{X}$ and can generate a secret key for any key attribute $y \in \mathcal{Y}$. However, in many cases, we are only able to construct a scheme that poses restrictions on the message space and key attribute space. To capture such restrictions, we introduce a parameter prm and consider subsets of the domains $\mathcal{X}_{\mathsf{prm}} \subseteq \mathcal{X}$ and $\mathcal{Y}_{\mathsf{prm}} \subseteq \mathcal{Y}$ specified by it and the function R_{prm} defined by restricting the function R on $\mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}}$. An FE (FE) scheme for $\{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}^*\}_{\mathsf{prm}}$ is defined by the following PPT algorithms:

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm takes as input the unary representation of the security parameter λ and a parameter prm that restricts the domain and range of the function and outputs the master public key mpk and a master secret key msk.

$\mathsf{Encrypt}(\mathsf{mpk}, x) \to \mathsf{ct}$: The encryption algorithm takes as input a master public key mpk and a message $x \in \mathcal{X}_{\mathsf{prm}}$. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}$: The key generation algorithm takes as input the master secret key msk, and a key attribute $y \in \mathcal{Y}_{\mathsf{prm}}$. It outputs a secret key sk. We assume that y is included in sk.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) \to m$ or $\perp$: The decryption algorithm takes as input a ciphertext ct and a secret key sk. It outputs the message m or $\perp$ which represents that the ciphertext is not in a valid form.

Remark 1 (Bounded collusion variants). In this paper, we mainly focus on FE with bounded collusion security, where the security is guaranteed only when the number of secret keys that the adversary obtains during the security game is below collusion bound Q. To do so, we have to slightly change the syntax above. We consider two different types of syntax for FE depending on when Q is declared.

- The first notion we consider is *bounded collusion FE* [21, 12], where Q is fixed when the system is setup. In more details, we change the syntax of FE defined above so that all the algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ take 1^Q as additional input.

– The second notion we consider is a new notion that we call *dynamic bounded collusion FE*. In this notion, the bound Q is specified by the encryptor, not by the setup. Namely, we change the syntax of FE above so that only the encryption algorithms Enc takes 1^Q as input, whereas other algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Dec})$ do not.

We note that the requirement that the algorithms run in polynomial time along with the fact that all algorithms in bounded collusion FE take 1^Q as additional input imply that all the algorithms run in polynomial in Q. In the case of FE with dynamic bounded collusion, similar implication holds for the encryption algorithm. However, the running time of Setup and KeyGen should be dependent only on λ and $|\mathsf{prm}|$, whereas the running time of Dec may indirectly depend on Q if the size of the input ciphertext is dependent on Q.

Definition 1 (Correctness). *An FE scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is correct if for all* prm*,* $x \in \mathcal{X}_{\mathsf{prm}}$*, and* $y \in \mathcal{Y}_{\mathsf{prm}}$*,*

$$\Pr\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}) : \\ \mathsf{Dec}\big(\mathsf{Enc}(\mathsf{mpk}, x), \mathsf{KeyGen}(\mathsf{msk}, y)\big) \neq R(x, y)\end{array}\right] = \mathrm{negl}(\lambda)$$

where probability is taken over the random coins of Setup*,* KeyGen *and* Enc*.*

2.1.2 Security Notions. As security notions for FE, we define simulation-based notions. We are mainly interested in the bounded collusion settings because the security notion without the collusion bound is shown to be impossible [4]. We first provide the description of the security game for FE with dynamic bounded collusion and then for bounded collusion FE.

Definition 2 (AD-SIM and NA-SIM Security for FE with Dynamic Bounded Collusion). *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a (public key) FE scheme with dynamic bounded collusion for the function family* $\{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}^*\}_{\mathsf{prm}}$*. For every stateful PPT adversary* A *and a stateful PPT simulator* $\mathsf{Sim} = (\mathsf{SimEnc}, \mathsf{SimKG})$ *consider the following experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)}$**:**	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{Sim}}(1^\lambda)}$**:**
1: $\mathsf{prm} \leftarrow \mathsf{A}(1^\lambda)$	*1:* $\mathsf{prm} \leftarrow \mathsf{A}(1^\lambda)$
2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$	*2:* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$
3: $(x, 1^Q) \leftarrow \mathsf{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	*3:* $(x, 1^Q) \leftarrow \mathsf{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$ – *Let* $(y^{(1)}, \ldots, y^{(Q_1)})$ *be* A*'s oracle queries.* – *Let* $\mathsf{sk}^{(q)}$ *be the oracle reply to* $y^{(q)}$*.* – *Let* $\mathcal{V} := \left\{\left(z^{(q)} := R(x, y^{(q)}), y^{(q)}, \mathsf{sk}^{(q)}\right)\right\}_{q \in [Q_1]}$.
4: $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, 1^Q)$	*4:* $(\mathsf{ct}, \mathsf{st}) \leftarrow \mathsf{SimEnc}(\mathsf{mpk}, \mathcal{V}, 1^{\lvert x\rvert}, 1^Q)$
5: $b \leftarrow \mathsf{A}^{\mathcal{O}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct})$	*5:* $b \leftarrow \mathsf{A}^{\mathcal{O}'(\mathsf{st},\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct})$
6: Output b	*6: Output* b

We emphasize that the adversary A *is stateful, even though we do not explicitly include the internal state of it into the output above for the simplicity of the notation. On the other hand, the above explicitly denotes the internal state of the simulator* Sim *by* st*. We distinguish between two cases of the above experiment:*

1. *The adaptive case, where:*
   - *The oracle* $\mathcal{O}(\mathsf{msk}, \cdot) = \mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *with* $1 \leq Q_1 < Q$*, and*
   - *The oracle* $\mathcal{O}'(\mathsf{st}, \mathsf{msk}, \cdot)$ *takes as input the* q*-th key query* $y^{(q)}$ *for* $q \in [Q_1 + 1, Q_1 + Q_2]$ *and returns* $\mathsf{SimKG}(\mathsf{st}, \mathsf{msk}, R(x, y^{(q)}), y^{(q)})$*, where* $Q_1 + Q_2 \leq Q$

   The FE scheme FE *is then said to be simulation secure for one message against adaptive adversaries (*$\mathsf{AD\text{-}SIM}$*-secure, for short) if there is a* PPT *simulator* Sim *such that for every* PPT *adversary* A*, the following holds:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}\left(1^\lambda\right) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{Sim}}\left(1^\lambda\right) = 1]\right| = \mathrm{negl}(\lambda) \tag{2.1}$$

2. *The non-adaptive case, where* $Q_1 \leq Q$ *and the oracles* $\mathcal{O}(\mathsf{msk}, \cdot)$ *and* $\mathcal{O}'(\mathsf{msk}, \cdot)$ *are both the "empty" oracles that return nothing: The FE scheme* FE *is then said to be simulation secure for one message against non-adaptive queries (*$\mathsf{NA\text{-}SIM}$*-secure, for short) if there is* PPT *simulator* $\mathsf{Sim} = (\mathsf{SimEnc}, \perp)$ *such that for every* PPT *adversaries* A*, Eq. (2.1) holds. Note that in the non-adaptive case, we can ignore* st *since* SimKG *is not present in the above game and it is never used by other algorithm.*

Definition 3 (AD-SIM **and** NA-SIM **Security for bounded collusion FE [21]).** *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a (public key) bounded collusion FE scheme for the function family* $\{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}^*\}_{\mathsf{prm}}$*. We define* $\mathsf{AD\text{-}SIM}$ *and* $\mathsf{NA\text{-}SIM}$ *security for* FE *by considering the same game as Definition 2 with the following changes:*

- *We change* A *to output* 1^Q *in addition to* prm *at the beginning of the game.*
- *All the algorithms run in the experiment* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SimEnc}, \mathsf{SimKG})$ *take* 1^Q *as additional input.*

We also define weaker notion of the security where the adversary is restricted to always choose $Q = 1$ *in the non-adaptive case. If the scheme only satisfies this weaker security notion, we say the scheme is* 1-$\mathsf{NA\text{-}SIM}$ *secure.*

Remark 2. The above definition allows to argue security of a single instance of FE. For the security proof of our constructions in Section 5.1, it is convenient to consider multi-instance version of the above notion. In the multi-instance security variant, the adversary declares the number of instances M at the beginning of the game and interact with each instance as above. In the real world, the adversary interacts with real algorithms in all instances, while in the ideal world, it interacts with simulators in all instances. The adversary can make queries in arbitrary order and make them arbitrarily correlated as long as it respects the restriction for each instance. This multi-instance security notion is easily shown to be equivalent to the single-instance security notion above by simple hybrid argument.

2.1.3 Special Classes of FE We then define various kinds of FE by specifying the relation.

KPFE for circuits. To define KPFE for circuits, we set $\mathcal{X} = \{0,1\}^*$ and $\mathcal{Y}$ as the set of all circuits and define $R(x, C) = C(x)$ if the length of the string x and the input length of C match and otherwise $R(x, C) = \perp$. In this paper, we will consider the circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$, depth $\mathsf{dep} := \mathsf{dep}(\lambda)$, and output length $\mathsf{out} := \mathsf{out}(\lambda)$. To do so, we set $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}}, 1^{\mathsf{out}})$, $\mathcal{X}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$, and $\mathcal{Y}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$.

CPFE for circuits. To define CPFE for circuits, we set $\mathcal{X}$ to be the set of all circuits and $\mathcal{Y} = \{0,1\}^*$ and define $R(C, x) = C(x)$ if the length of the string x and the input length of C match and otherwise $R(x, C) = \perp$. In this paper, we will consider the circuit class $\mathcal{C}_{\mathsf{inp}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$. To do so, we set $\mathsf{prm} = 1^{\mathsf{inp}}$, $\mathcal{X}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp}}$, and $\mathcal{Y}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$.

Remark 3. In the definition of KPFE for circuits, even though the input length, output length, and depth of the circuits in $\mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$ are bounded, the size of the circuits is unbounded. Similar comments hold true for $\mathcal{C}_{\mathsf{inp}}$ in the definition of CPFE.

Remark 4. Note that our definition of the KPFE requires that the running time of the encryption algorithm is bounded by $\mathrm{poly}(\lambda, |\mathsf{prm}|) = \mathrm{poly}(\lambda, \mathsf{inp}, \mathsf{dep}, \mathsf{out})$. In particular, the running time should be independent from the size of the circuit being supported by the scheme, which is unbounded. This property is called succinctness in [20].

FE for Turing Machines. To define FE for Turing machines, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all Turing machine, and define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\perp\}$ as

$$R((x, 1^t), M) = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ in } t \text{ steps} \\ 0 & \text{otherwise.} \end{cases}.$$

FE for NL. To define FE for NL, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all non-deterministic Turing machines with two tapes, one of which encodes the input and can only be read, whereas the other tape can be read as well as written. When we measure the space complexity of the computation, we consider the space being used for the latter tape. We define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\perp\}$ as

$$R((x, 1^t, 1^{2^s}), M) = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s \\ 0 & \text{otherwise.} \end{cases}.$$

Note that here, s is in the exponent to reflect the idea that the space for the computation is logarithmically bounded.

We recall the following result by Goldwasser et al. [20] that we will use later in Section 4.

Theorem 1 ([20]). *There exists a KPFE scheme* KPFE *for the circuit class* $\mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$ *with* 1-NA-SIM *security assuming sub-exponential hardness of the LWE problem.*

Remark 5. Note that [20] defines their security notion as *full-simulation* based security. However, as stated in [20] only, their *full-simulation* security is equivalent to the *non-adaptive simulation* security definition from [21], where no post-challenge key queries are allowed. The 1-NA-SIM security given in Definition 3 is implied by the same that is adopted for this work. Thus, [20] is 1-NA-SIM secure according to our Definition 3.

3 CPFE with Dynamic Bounded Collusion

In this section, we construct public-key, ciphertext-policy functional encryption (CPFE) schemes with delayed collusion bound. The first scheme supports *unbounded* polynomial-size circuits and achieves NA-SIM security. The second scheme only supports bounded polynomial-size circuits, but achieves stronger AD-SIM security. Both schemes are obtained by first constructing a bounded FE with special efficiency property (Sections 3.2 and 3.3) and then converting it into a scheme with delayed collusion bound (Section 3.4).

3.1 Preparations

Here, we define reusable, dynamic multi-party computation (RDMPC) protocol in the client-server framework, which is introduced by Ananth and Vaikuntanathan [12] as a useful notion for the construction of bounded FE. The formal definition of the protocol as a tuple of algorithms with its correctness and security definitions appear in our full version [6]. We adapt the syntax and definitions of [12] to our setting.

We provide some intuition on how these algorithms may be used as a multi-party computation in a client-server framework. Similar to [12], here also the setting consists of a single client and N servers. In particular, the client wants to offload an *apriori bounded* number of computations R to these N servers. Each computation is termed as a session. To this end, the protocol consists of two phases as described below:

- An *offline* phase, where the client takes the session count R and a *secret* circuit $C \in \mathcal{C}_{\mathsf{inp}}$ as input, and encodes it (via CktEnc algorithm) as $(\widehat{C}_1, \ldots, \widehat{C}_N)$. For all $k \in [N]$, it then sends the kth encoding $\widehat{C}_k$ to the kth server.
- An *online* phase, that is performed for R sessions. The client wishes to delegate an input $x^{(j)}$ in the jth session for some $j \in [R]$. For this, it encodes $x^{(j)}$ (via InpEnc algorithm) as $\widehat{x}^{(j)}$ and forwards it to all the N servers. Hereon, some n out of N servers (formalized via any subset $S \subseteq [N]$ of size n) may come to do some *local* computation (via the Local algorithm) on their own inputs to get a *partial* output encoding. In particular, the uth server may compute the partial output encoding $\widehat{y}_u^{(j)} = \mathsf{Local}(\widehat{C}_u, \widehat{x}^{(j)})$. The final output $C(x^{(j)})$ is obtained by combining these partial output encodings $\{\widehat{y}_u^{(j)}\}_{u \in S}$ via the public evaluation algorithm Decode.

Crucially, the R input encodings $\{\widehat{x}^{(j)}\}_{j \in [R]}$ are generated with randomness *independent* from that of the offline phase. Further the terms "*reusable*" and "*dynamic*" stems from the respective requirements that the secret circuit encoding must be reusable across all R sessions and the final output may be recovered dynamically by *any* subset S of the N servers in each session. We then recall the result from [12] adapted to our setting.

Theorem 2 (Adapted from [12]). *Assuming the existence of one-way functions, there exists an RDMPC protocol with parameter $N = \Theta(R^2\lambda)$, $t = \Theta(R\lambda)$, and $n = \Theta(t)$ for $\mathcal{C}_{\mathsf{inp}}$ with any $\mathsf{inp} = \mathrm{poly}(\lambda)$.*

3.2 Basic Construction

Here, we give a construction of bounded CPFE. The construction supports unbounded size circuits and is secure against bounded collusion. A nice feature of the construction is that the running times of the setup and key generation algorithms are independent from the collusion bound. Looking ahead, we leverage this property to upgrade the construction to be an CPFE scheme with *dynamic bounded collusion property* later in Section 3.4.

In more details, we provide the description of CPFE for the circuit class $\mathcal{C}_\ell$ for arbitrary $\ell = \ell(\lambda)$, where $\mathcal{C}_\ell$ is the set of all polynomial size circuits with input length ℓ. Formally, our FE is for the relation $R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}^*$ where $\mathsf{prm} = 1^\ell$, $\mathcal{X}_{\mathsf{prm}} := \mathcal{C}_\ell$, and $\mathcal{Y}_{\mathsf{prm}} := \{0,1\}^\ell$ and $R_{\mathsf{prm}}(C, x) = C(x)$, for all $C \in \mathcal{C}_\ell$ and $x \in \{0,1\}^\ell$.

Ingredients. We now describe the underlying building blocks used to obtain our CPFE construction:

1. A reusable, dynamic MPC protocol for $\mathcal{C}_\ell$ denoted by $\mathsf{RDMPC} = (\mathsf{CktEnc}, \mathsf{InpEnc}, \mathsf{Local}, \mathsf{Decode})$ with $N = \Theta(R^2\lambda)$, $t = \Theta(R\lambda)$, and $n = \Theta(t)$. We can instantiate this by the protocol by Ananth and Vaikuntanathan [12], which can be based on any one-way function (See Theorem 2.). Looking ahead, we will set $R = \lambda$ in our construction and therefore ignore the dependence on R when considering the size of the parameters in the following. We denote the length of an encoding $\widehat{x}$ of $x \in \{0,1\}^\ell$ by $\widehat{\ell} = \mathrm{poly}(\lambda, \ell)$. We also denote the size of the circuit $\mathsf{Local}(\widehat{C}_j, \cdot)$ by $\widehat{s}(\lambda, R, |C|)$, where $\widehat{C}_j$ is an output of CktEnc on input a circuit C. By the efficiency properties of RDMPC, we have $\widehat{s}(\lambda, |C|) = \mathrm{poly}(\lambda, |\widehat{C}_j|) = \mathrm{poly}(\lambda, |C|)$.
2. A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ for circuit class $\mathcal{C}_{\widehat{\ell}}$. We can instantiate this by Yao's scheme [30], which can be based on any one-way function. We assume that a label is represented by a binary string. The length of a label depends on the size of circuits that are garbled. We denote the length of the labels obtained by garbling a circuit of size $\widehat{s}(\lambda, |C|)$ by $L(\lambda, |C|)$. By the efficiency property of the garbled circuit, we have $L(\lambda, |C|) = \mathrm{poly}(\lambda, \widehat{s}(\lambda, |C|)) = \mathrm{poly}(\lambda, |C|)$.
3. An IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.Enc}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Dec})$ with IND-CPA security whose identity space and message space are $\{0,1\}^*$. We can instantiate IBE from various standard assumptions including LWE [3, 14], CDH, and Factoring [15].

Construction. Let $N := N(\lambda, R)$, $n := n(\lambda, R)$, and $t := t(\lambda, R)$ be the parameters associated with RDMPC. In the following construction, we run the protocol with $R = \lambda$. The basic CPFE scheme $\mathsf{BCPFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the circuit class $\mathcal{C}_\ell$ works as follows. Note that the scheme below deviates from the syntax of bounded collusion FE because Setup and KeyGen take the collusion bound Q as binary form,

rather than unary form as defined in Remark 1. This change in syntax is to reflect the fact that these algorithms run in polylogarithmic time in Q rather than in polynomial time. Even with this change, we can consider the same correctness requirement and security notions for it.

$\mathsf{Setup}(1^\lambda, 1^\ell, Q)$: On input the security parameter λ, an input length $\ell = \mathrm{poly}(\lambda)$ of the circuit family to be supported, and the upper bound for the collusion $1 \leq Q \leq 2^\lambda$ in *binary* form and compute $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. Output the master key pair as $(\mathsf{mpk}, \mathsf{msk}) := (\mathsf{IBE.mpk}, \mathsf{IBE.msk})$.

$\mathsf{KeyGen}(\mathsf{msk}, x, Q)$: On input master secret key $\mathsf{msk} = \mathsf{IBE.msk}$, an input $x \in \{0,1\}^\ell$ and the upper bound for the collusion $1 \leq Q \leq 2^\lambda$ in *binary* form and do the following:

1. Compute $\widehat{x} \leftarrow \mathsf{InpEnc}(1^\lambda, 1^\lambda, 1^\ell, x)$.
2. Sample $u \leftarrow [Q]$.
3. Sample random set $\Delta \subset [N]$ such that $|\Delta| = n$.
4. For all $j \in \Delta$ and $k \in [\widehat{\ell}]$, generate a secret key as

$$\mathsf{IBE.sk}_{u,j,k,\widehat{x}_k} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (u, j, k, \widehat{x}_k)),$$

   where $\widehat{x}_k$ is the k-th bit of $\widehat{x}$.
5. Output

$$\mathsf{sk} = \left(u, \Delta, \{\mathsf{IBE.sk}_{u,j,k,\widehat{x}_k}\}_{j\in\Delta, k\in[\widehat{\ell}]}\right). \tag{3.1}$$

$\mathsf{Enc}(\mathsf{mpk}, C, 1^Q)$: On input the master public key $\mathsf{mpk} = \mathsf{IBE.mpk}$, a circuit $C \in \mathcal{C}_\ell$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Compute $(\widehat{C}_{i,1}, \ldots, \widehat{C}_{i,N}) \leftarrow \mathsf{CktEnc}(1^\lambda, 1^\lambda, 1^\ell, C)$ for $i \in [Q]$.
2. Define the circuit $\mathsf{L}_{i,j}(\cdot) := \mathsf{Local}(\widehat{C}_{i,j}, \cdot)$ with input length $\widehat{\ell} := |\widehat{x}|$.
   For all $i \in [Q]$ and $j \in [N]$, do the following:
   (a) Run the garbling algorithm

$$\{\mathsf{lab}_{i,j,k,b}\}_{k\in[\widehat{\ell}], b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{L}_{i,j}).$$

   (b) For all $k \in [\widehat{\ell}]$ and $b \in \{0,1\}$, compute

$$\mathsf{IBE.ct}_{i,j,k,b} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,k,b), \mathsf{lab}_{i,j,k,b}\right).$$

3. Output

$$\mathsf{ct} = \{\mathsf{IBE.ct}_{i,j,k,b}\}_{i\in[Q], j\in[N], k\in[\widehat{\ell}], b\in\{0,1\}}. \tag{3.2}$$

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}, 1^Q)$: On input a secret key sk, a ciphertext ct, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Parse the secret key as Eq. (3.1) and the ciphertext as Eq. (3.2).
2. For all $j \in \Delta$, do the following:
   (a) For all $k \in [\widehat{\ell}]$, compute $\mathsf{lab}'_{u,j,k} := \mathsf{IBE.Dec}(\mathsf{IBE.sk}_{u,j,k,\widehat{x}_k}, \mathsf{IBE.ct}_{u,j,k,\widehat{x}_k})$, where $\widehat{x}_k$ is the k-th bit of $\widehat{x}$.

(b) Compute $\widehat{y}'_{u,j} := \mathsf{GC.Eval}(\{\mathsf{lab}'_{u,j,k}\}_{k\in[\widehat{\ell}]})$

3. Compute and output $z = \mathsf{Decode}(\{\widehat{y}'_{u,j}\}_{j\in\Delta})$.

Remark 6 (Intuition for the construction.). The above scheme can be seen as a variant of the FE scheme by Ananth and Vaikuntanathan [12], who showed a generic construction of Q-bounded FE scheme from a single-key FE scheme. Here, we instantiate the single-key FE scheme with the construction by Sahai and Seyalioglu [28] and then replace the PKE used for encrypting the labels of the garbled circuits inside their construction with IBE. In more details, in our construction above, we run QN instances of the single-key FE scheme. These QN instances are grouped into Q groups each consisting of N instances. In the key generation algorithm of BCPFE, one chooses a group $u \in [Q]$ and then generates n out of N secret keys of the single-key FE instances in the group. To encrypt a message, one generates shares of the message (in our case, a circuit) using RDMPC and then encrypts them using N instances of FE for each group $i \in [Q]$. By the correctness of RDMPC, n secret keys of the single-key FE from a single group can recover the decryption results.

Security. The following theorem asserts the security of our CPFE scheme.

Theorem 3. *Assume that* IBE *satisfies* IND-CPA *security,* GC *is a secure garbled circuit scheme, and* RDMPC *is secure. Then,* BCPFE *for the circuit class* $\mathcal{C}_\ell$ *is* NA-SIM*-secure.*

We refer to the full version [6] for the detailed proof of Theorem 3.

3.3 A Variant of Basic Construction with AD-SIM Security

Here, we provide a variant of the basic construction in Section 3.2 that satisfies stronger AD-SIM security rather than NA-SIM security at the cost of only supporting circuits with bounded size. Similarly to the construction in Section 3.2, the construction can be upgraded into a construction with delayed collusion bound in Section 3.4. In more details, our construction is for the circuit class $\mathcal{C}_{\ell,s}$, where $\mathcal{C}_{\ell,s}$ is the set of circuits with input length ℓ and size at most s. Formally, our FE is for the relation $R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}^*$ where $\mathsf{prm} = (1^\ell, 1^s)$, $\mathcal{X}_{\mathsf{prm}} := \mathcal{C}_{\ell,s}$, and $\mathcal{Y}_{\mathsf{prm}} := \{0,1\}^\ell$ and $R_{\mathsf{prm}}(C, x) = C(x)$, for all $C \in \mathcal{C}_{\ell,s}$ and $x \in \{0,1\}^\ell$.

Ingredients and Parameters. Our construction is the same as that in Section 3, but we require stronger SIM-RSO security for the IBE [25]. As shown in [25], IBE with SIM-RSO security can be constructed only from IBE with more standard IND-CPA security. Therefore, our construction here can be based on the same set of assumptions as the construction in Section 3. However, since the IBE scheme with SIM-RSO security can only encrypt messages with bounded length, we will have a bound on the length of the labels of GC. This in turn implies that we can only support circuits with bounded size as messages of the CPFE. In the construction, we encrypt circuits with fixed size $s = s(\lambda)$. We denote the size of the circuit $\mathsf{Local}(\widehat{C}_j, \cdot)$ by $\widehat{s}$, where $\widehat{C}_j$ is an output of CktEnc on input a circuit C of size s. We also denote the length of the labels output by GC on input a circuit of size $\widehat{s}$ as L. While $\widehat{s}$ and L are polynomial function in λ and s,

we treat them as functions only depend on λ here, since the size of the circuit s is fixed. We assume that the message space of IBE is $\{0,1\}^L$.

We observe that Setup and KeyGen run in time $\mathrm{poly}(\lambda, \ell, s, \log Q)$ by the efficiency properties of IBE and RDMPC. This means that these algorithms can be run even for super polynomial Q. Looking ahead, this property will be used crucially for the full-fledged construction that we show in Section 3.4. We also observe that Enc and Dec run in time $Q \cdot \mathrm{poly}(\lambda, \ell, s)$ by the efficiency properties of IBE, GC, and RDMPC.

The following theorem asserts the security of our CPFE scheme.

Theorem 4. *Assume that* IBE *satisfies* IND-CPA *security and* SIM-RSO *security,* GC *is a secure garbled circuit scheme, and* RDMPC *is secure. Then,* BCPFE *for the circuit class $\mathcal{C}_{\ell,s}$ is* AD-SIM*-secure.*

Overview for the proof. Before going to the formal proof, we provide its overview. The proof is similar to that for Theorem 3. However, here, we have to consider secret key queries after the encryption query. Since we consider simulation-based security definition for BCPFE, this intuitively means that we have to be able to "program" a decryption result into a secret key issued after the encryption query. RDMPC already has this type of capability, since this is designed for constructing FE with AD-SIM security [12]. Using this property and the security of the garbled circuits, we can "program" the decryption results into labels of garbled circuits. What remains is to simulate the IBE ciphertexts and secret keys so that the decryption results correspond with the labels generated as above. For this purpose, we use IBE with SIM-RSO security [25].

We refer to the full version [6] for the detailed proof of Theorem 4.

3.4 Full-fledged Construction

In Sections 3.2 and 3.3, we construct bounded collusion CPFE schemes. Here, we show that these schemes can be converted into CPFE schemes with dynamic bounded collusion property. The resulting schemes satisfy the same level of the security and support the same class of circuits as the original schemes. The conversion is the same for both schemes. Let $\mathsf{BCPFE} = (\mathsf{BCPFE.Setup}, \mathsf{BCPFE.KeyGen}, \mathsf{BCPFE.Enc}, \mathsf{BCPFE.Dec})$ be our FE scheme in either Section 3.2 or 3.3 and let $\mathcal{C}_{\mathsf{prm}}$ be the supported circuit class. We have $\mathsf{prm} = 1^\ell$ or $\mathsf{prm} = (1^\ell, 1^s)$. An input to a circuit $C \in \mathcal{C}_{\mathsf{prm}}$ has fixed length $\ell(\lambda)$ in both cases.

Construction. We now construct a CPFE scheme $\mathsf{CPFE} = (\mathsf{CPFE.Setup}, \mathsf{CPFE.KeyGen}, \mathsf{CPFE.Enc}, \mathsf{CPFE.Dec})$ with delayed collusion bound as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm})$: On input the security parameter λ, the parameter prm that specifies the circuit family to be supported, do the following:

1. Run $(\mathsf{BCPFE.mpk}_i, \mathsf{BCPFEmsk}_i) \leftarrow \mathsf{BCPFE.Setup}(1^\lambda, \mathsf{prm}, 2^i)$ for $i \in [\lambda]$, where 2^i is represented as a binary number here.
2. Output $(\mathsf{mpk}, \mathsf{msk}) := (\{\mathsf{BCPFE.mpk}_i\}_{i\in[\lambda]}, \{\mathsf{BCPFE.msk}_i\}_{i\in[\lambda]})$.

$\mathsf{KeyGen}(\mathsf{msk}, x)$: On input master secret key $\mathsf{msk} = \{\mathsf{BCPFE.msk}_i\}_{i\in[\lambda]}$ and an input $x \in \{0,1\}^\ell$, do the following:

1. Run $\mathsf{BCPFE.sk}_i \leftarrow \mathsf{BCPFE.KeyGen}(\mathsf{IBE.msk}_i, x, 2^i)$ for $i \in [\lambda]$, where 2^i is represented as a binary number above.
2. Output $\mathsf{sk} := \{\mathsf{BCPFE.sk}_i\}_{i\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, C, 1^Q)$: On input the master public key mpk, a circuit $C \in \mathcal{C}_{\mathsf{prm}}$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Find u such that $2^{u-1} < Q \leq 2^u$.
2. Parse $\mathsf{mpk} \rightarrow \{\mathsf{BCPFE.mpk}_i\}_{i\in[\lambda]}$.
3. Compute $\mathsf{BCPFE.ct}_u \leftarrow \mathsf{BCPFE.Enc}\left(\mathsf{BCPFE.mpk}_u, x, 1^{2^u}\right)$ and output $\mathsf{ct} = \mathsf{BCPFE.ct}_u$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}, 1^Q)$: On input a secret key sk, a ciphertext ct, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Find u such that $2^{u-1} < Q \leq 2^u$.
2. Parse $\mathsf{sk} \rightarrow \{\mathsf{BCPFE.sk}_i\}_{i\in[\lambda]}$ and $\mathsf{ct} = \mathsf{BCPFE.ct}_u$.
3. Compute and output $\mathsf{BCPFE.Dec}(\mathsf{BCPFE.sk}_u, \mathsf{BCPFE.ct}_u)$.

It is clear that the correctness of the above scheme follows from that of the underlying scheme BCPFE. We analyse the efficiency of the above construction in our full version [6].

The following theorem asserts the security of our CPFE scheme.

Theorem 5. *If* BCPFE *satisfies* AD-SIM *security as bounded FE scheme, then so does* CPFE *scheme as FE with delayed collusion bound. Similarly, if* BCPFE *satisfies* NA-SIM *security as bounded FE scheme, then so does* CPFE *scheme as FE with delayed collusion bound.*

The proof of the above theorem appears in our full version [6].

4 Succinct KPFE with Dynamic Bounded Collusion

In this section, we construct new succinct public key KPFE scheme. The construction substantially improves the security of the previous succinct bounded KPFE schemes [20, 1], because it supports delayed collusion bound and satisfies AD-SIM security. Furthermore, our construction can be instantiated only from the LWE assumption, similarly to the previous constructions.

In our full version [6], we also construct a CPFE scheme for unbounded size circuits with AD-SIM security and delayed collusion property. The construction is the dual version of the KPFE scheme in this section in the sense that it satisfies similar properties and the idea for the construction is very similar. Further, the scheme can be instantiated only from the LWE assumption, similar to the construction in this section.

We now describe the parameters for our KPFE scheme in more detail. We construct KPFE scheme for the circuit family $\mathcal{C}_{\mathsf{inp,dep,out}}$ containing circuits with input length $\mathsf{inp} = \mathsf{inp}(\lambda)$, depth $\mathsf{dep} = \mathsf{dep}(\lambda)$, output length $\mathsf{out} = \mathsf{out}(\lambda)$ but with arbitrary polynomial size. Formally, we construct an FE scheme for $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}}, 1^{\mathsf{out}})$, $\mathcal{X}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$, $\mathcal{Y}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp,dep,out}}$ and $R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \rightarrow \{0,1\}^{\mathsf{out}}$, where $R(x, C) = C(x)$, for all $C \in \mathcal{C}_{\mathsf{inp,dep,out}}$ and $x \in \{0,1\}^{\mathsf{inp}}$.

Ingredients. The underlying building blocks for our KPFE construction are as follows:

1. A bounded, public key KPFE scheme, denoted by

$$1\mathsf{KPFE} = (1\mathsf{KPFE.Setup}, 1\mathsf{KPFE.KeyGen}, 1\mathsf{KPFE.Enc}, 1\mathsf{KPFE.Dec}),$$

   for the same functionality (i.e., R_{prm} defined above). We require the scheme to satisfy NA-SIM security against a *single* key query. We can instantiate such a scheme with the succinct KPFE scheme from LWE [20] (See Theorem 1).
2. A CPFE scheme denoted by

$$\mathsf{CPFE} = (\mathsf{CPFE.Setup}, \mathsf{CPFE.KeyGen}, \mathsf{CPFE.Enc}, \mathsf{CPFE.Dec}),$$

   for bounded polynomial sized circuits that already supports delayed collusion bound property and satisfies AD-SIM security. Namely, CPFE supports a circuit class $\mathcal{C}_{\mathsf{inp}',\mathsf{size}'}$ consisting of circuits with input length inp' and size at most size', where the setting of the parameters is deferred until the description of the construction. Formally, we use FE with $\mathsf{prm}' = 1^{\mathsf{inp}'}, \mathcal{X}_{\mathsf{prm}'} = \mathcal{C}_{\mathsf{inp}',\mathsf{size}'}, \mathcal{Y}_{\mathsf{prm}'} = \{0,1\}^{\mathsf{inp}'}$ and $R_{\mathsf{prm}'} : \mathcal{X}_{\mathsf{prm}'} \times \mathcal{Y}_{\mathsf{prm}'} \to \{0,1\}^*$, where $R(C, x) = C(x)$, for all $C \in \mathcal{C}_{\mathsf{inp}',\mathsf{size}'}$ and $x \in \{0,1\}^{\mathsf{inp}'}$. We instantiate it from our construction in Section 3.4, which in turn can be based on any IBE.
3. A pseudorandom function $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$. We assume without loss of generality that the input and output space of PRF is the 1KPFE public key space and the randomness space used in 1KPFE.Enc algorithm respectively.

At a high level, our KPFE construction is a compiler that applies the CPFE scheme already satisfying the delayed collusion property and AD-SIM security on top of the single key, succinct KPFE scheme. This makes the resultant KPFE satisfy the delayed collusion property and AD-SIM security as well. We now proceed to the formal construction below.

4.1 Construction

The KPFE scheme $\mathsf{KPFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with delayed collusion bound for the circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$ is as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm})$: On input the security parameter λ and the parameters $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}}, 1^{\mathsf{out}})$ do the following:

1. Compute $\mathsf{prm}' := (1^{\mathsf{inp}'}, 1^{\mathsf{size}'})$, where $\mathsf{inp}' := \mathsf{inp}'(\lambda, \mathsf{prm})$ is the length of the master public key for 1KPFE output by $1\mathsf{KPFE.Setup}(1^\lambda, \mathsf{prm})$ and $\mathsf{size}' = \mathsf{size}'(\lambda, \mathsf{prm})$ is the size of the circuit $\mathsf{E}_{[x,\mathsf{K}]}$ described in Figure 1.
2. Run $(\mathsf{CPFE.mpk}, \mathsf{CPFE.msk}) \leftarrow \mathsf{CPFE.Setup}(1^\lambda, \mathsf{prm}')$.
3. Output $(\mathsf{mpk}, \mathsf{msk}) := (\mathsf{CPFE.mpk}, \mathsf{CPFE.msk})$.

$\mathsf{KeyGen}(\mathsf{msk}, C)$: On input the master secret key $\mathsf{msk} = \mathsf{CPFE.msk}$ and a circuit $C \in \mathcal{C}_{\mathsf{inp},\mathsf{dep},\mathsf{out}}$, do the following:

1. Compute $(1\mathsf{KPFE.mpk}, 1\mathsf{KPFE.msk}) \leftarrow 1\mathsf{KPFE.Setup}(1^\lambda, \mathsf{prm})$.
2. Generate a secret key under 1KPFE as $1\mathsf{KPFE.sk} \leftarrow 1\mathsf{KPFE.KeyGen}(1\mathsf{KPFE.msk}, C)$.
3. Generate another secret key $\mathsf{CPFE.sk} \leftarrow \mathsf{CPFE.KeyGen}(\mathsf{CPFE.msk}, 1\mathsf{KPFE.mpk})$, where 1KPFE.mpk is interpreted as a string in $\{0,1\}^{\mathsf{inp}'}$.

4. Output the full secret key $\mathsf{sk} := (\mathsf{CPFE.sk}, \mathsf{1KPFE.sk})$.

$\mathsf{Enc}(\mathsf{mpk}, x, 1^Q)$: On input the master public key $\mathsf{mpk} = \mathsf{CPFE.mpk}$, an input $x \in \{0,1\}^{\mathsf{inp}}$ and the query bound $1 \leq Q < 2^\lambda$, do the following:

1. Sample a PRF key $\mathsf{K} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. Using prm, construct the circuit $\mathsf{E}_{[x,\mathsf{K}]}(\cdot)$ defined as Figure 1.
3. Compute a ciphertext $\mathsf{CPFE.ct} \leftarrow \mathsf{CPFE.Enc}(\mathsf{CPFE.mpk}, \mathsf{E}_{[x,\mathsf{K}]}, 1^Q)$.
4. Output the ciphertext $\mathsf{ct} := \mathsf{CPFE.ct}$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$: On input a secret key sk associated with circuit C and a ciphertext ct, do the following:

1. Parse the ciphertext $\mathsf{ct} = \mathsf{CPFE.ct}$ and the secret key $\mathsf{sk} = (\mathsf{CPFE.sk}, \mathsf{1KPFE.sk})$, where $\mathsf{CPFE.sk}$ and $\mathsf{1KPFE.sk}$ are associated with $\mathsf{1KPFE.mpk} \in \{0,1\}^{\mathsf{inp}'}$ and the circuit C respectively.
2. Compute $y = \mathsf{CPFE.Dec}(\mathsf{CPFE.sk}, \mathsf{CPFE.ct})$.
3. Compute and output $z = \mathsf{1KPFE.Dec}(\mathsf{1KPFE.sk}, y)$.

Circuit $\mathsf{E}_{[x,\mathsf{K}]}$

Hardwired constants: $x \in \{0,1\}^{\mathsf{inp}}$, PRF key K.
On input $\mathsf{str} \in \{0,1\}^{\mathsf{inp}'}$, proceed as follows:

1. Compute $\mathsf{R} = \mathsf{PRF.Eval}(\mathsf{K}, \mathsf{str})$.
2. Compute and output $\mathsf{1KPFE.Enc}(\mathsf{str}, x; \mathsf{R})$, where str is interpreted as a master public key for $\mathsf{1KPFE}$ and R is used as a randomness for the encryption.

Fig. 1 : Circuit $\mathsf{E}_{[x,\mathsf{K}]}$.

In our full version [6], we show that the scheme is correct, succinct and satisfies AD-SIM security.

5 FE for Turing Machines with Dynamic Bounded Collusion

In this section, we construct FE for Turing machines from KPFE and CPFE schemes that we have constructed so far. For conciseness, we first provide a generic construction of FE that combines many instance of FE together in Section 5.1. We then instantiate this generic construction to construct FE for Turing machines with NA-SIM security in Section 5.2 and FE for NL with AD-SIM security in our full version [6].

5.1 Generalized Bundling of Functionality

Consider an FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for a parameter $\mathsf{prm} = 1^i$ and a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ for all $i \in \mathbb{N}$. Using such FE, we will construct a new FE with message space $\mathcal{A}$ and key space $\mathcal{B}$. We assume that there

exist efficiently computable maps $\mathcal{S} : \mathbb{N} \to 2^{\mathbb{N}}$ and $\mathcal{T} : \mathbb{N} \to 2^{\mathbb{N}}$ such that $\max \mathcal{S}(n)$ and $\max \mathcal{T}(n)$ can be bounded by some fixed polynomial in n. We also assume that there exist maps f with domain $\mathcal{A}$ and g with domain $\mathcal{B}$ such that

$$f(x) \in \prod_{i \in \mathcal{S}(|x|)} \mathcal{X}_i \quad \text{and} \quad g(y) \in \prod_{i \in \mathcal{T}(|y|)} \mathcal{Y}_i,$$

where $|x|$ and $|y|$ are the lengths of x and y as binary strings. Namely, f and g are maps such that

$$f : \mathcal{A} \ni x \mapsto \{f(x)_i \in \mathcal{X}_i\}_{i \in \mathcal{S}(|x|)}, \quad g : \mathcal{B} \ni y \mapsto \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}.$$

Here, we require that the length of $|f(x)|_i$ and $|g(x)|_i$ can be computed from the length of $|x|$ alone and they do not depend on the actual value of x. In this setting, we can construct an FE scheme $\mathsf{BFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for a two input function $R^{\mathsf{bndl}} : \mathcal{A} \times \mathcal{B} \to \{0,1\}^*$ defined in the following

$$R^{\mathsf{bndl}}(x, y) = \{R_i(f(x)_i, g(y)_i)\}_{i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)}, \tag{5.1}$$

where $f(x)_i \in \mathcal{X}_i$ and $g(y)_i \in \mathcal{Y}_i$ are the i-th entries of $f(x)$ and $g(x)$, respectively.

Ingredients. We now describe the underlying building blocks used to obtain our FE construction:

1. A pseudorandom function $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$. We assume that $\mathsf{PRF.Eval}(\mathsf{K}, \cdot)$ has domain $\{0,1\}^\lambda$ and range $\{0,1\}^\lambda$ for any key K output by $\mathsf{PRF.Setup}(1^\lambda)$. The input space $\{0,1\}^\lambda$ can be regarded as $[2^\lambda]$ by the natural bijection between the two sets. We can instantiate PRF from any one-way function [18].
2. An FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for a parameter $\mathsf{prm} = 1^i$ and a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ for $i \in \mathbb{N}$ with delayed collusion property. We require the scheme to have either NA-SIM or AD-SIM security. We also require that the randomness used by $\mathsf{FE.Setup}(1^\lambda, 1^i)$ is of fixed length λ. This is without loss of generality, since we can generate unbounded length of pseudorandom bits from a binary string R of length λ by regarding R as a PRF key.
3. A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$. We assume that a label is represented by a binary string and denote its length by $L(\lambda, |C|)$, where C is the circuit being garbled. We can instantiate it by Yao's garbled circuit [30], which can be based on any one-way function.
4. An IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.Enc}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Dec})$ with IND-CPA security whose identity space and message space are $\{0,1\}^*$. We assume that the key generation algorithm is deterministic. This is without loss of generality, since we can use PRF to derandomize the key generation algorithm. We can instantiate IBE from various standard assumptions including LWE [3, 14], CDH, and Factoring [15].

Construction. We then provide the description of the construction of $\mathsf{BFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for R^{bndl} above.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, do the following:

1. Run $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Sample a PRF key $\mathsf{K} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
3. Output the master key pair as $(\mathsf{mpk}, \mathsf{msk}) := (\mathsf{IBE.mpk}, (\mathsf{IBE.msk}, \mathsf{K}))$.

$\mathsf{KeyGen}(\mathsf{msk}, y)$: On input master secret key $\mathsf{msk} = \mathsf{IBE.msk}$, a key attribute $y \in \mathcal{B}$, do the following:

1. Compute $\mathcal{T}(|y|) \subseteq \mathbb{N}$, where $|y|$ is the length of y as a binary string.
2. Compute $\mathsf{R}_i = \mathsf{PRF.Eval}(\mathsf{K}, i)$ for $i \in \mathcal{T}(|y|)$, where $i \in \mathbb{N}$ is interpreted as a binary string in $\{0,1\}^\lambda$.
3. Run $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^i; \mathsf{R}_i)$ for $i \in \mathcal{T}(|y|)$.
4. Compute $g(y) = \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}$.
5. For $i \in \mathcal{T}(|y|)$, compute
$$\mathsf{FE.sk}_i \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, g(y)_i).$$
6. Let $\ell_i := |\mathsf{FE.mpk}_i|$. For all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$, generate a secret key as
$$\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{FE.mpk}_{i,j}))$$
where $\mathsf{FE.mpk}_{i,j}$ is the j-th bit of $\mathsf{FE.mpk}_i \in \{0,1\}$ as a binary string.
7. Output
$$\mathsf{sk} = \left(\mathcal{T}(|y|),\ \left\{\mathsf{FE.sk}_i, \{\mathsf{IBE.sk}_{i,j}\}_{j \in [\ell_i]}\right\}_{i \in \mathcal{T}(|y|)}\right). \tag{5.2}$$

$\mathsf{Enc}(\mathsf{mpk}, x, 1^Q)$: On input the encryption key $\mathsf{mpk} = \mathsf{IBE.mpk}$, a message $x \in \mathcal{A}$, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Compute $\mathcal{S}(|x|) \subset \mathbb{N}$, where $|x|$ is the length of x as a binary string.
2. Compute $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.
3. Do the following for $i \in \mathcal{S}(|x|)$.
   (a) Compute the length ℓ_i of $\mathsf{FE.mpk}_i$. This is done without knowing $\mathsf{FE.mpk}_i$.
   (b) Sample a randomness r_i for the encryption algorithm $\mathsf{FE.Enc}(\mathsf{FE.mpk}_i, f(x)_i, 1^Q; r_i)$. This is done without knowing $\mathsf{FE.mpk}_i$.
   (c) Define a circuit
   $$\mathsf{E}_i(\cdot) := \mathsf{FE.Enc}(\cdot, f(x)_i, 1^Q; r_i)$$
   that takes as input a string $\mathsf{str} \in \{0,1\}^{\ell_i}$ and outputs $\mathsf{FE.Enc}(\mathsf{str}, f(x)_i, 1^Q; r_i)$, where str is interpreted as a master public key of the FE.
   (d) Generate a garbled circuit
   $$\{\mathsf{lab}_{i,j,b}\}_{j \in [\ell_i], b \in \{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i).$$
   (e) For all $j \in [\ell_i]$ and $b \in \{0,1\}$, compute
   $$\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,b}\right).$$
4. Output
$$\mathsf{ct} = \left(\mathcal{S}(|x|),\ \{\mathsf{IBE.ct}_{i,j,b}\}_{i \in \mathcal{S}(|x|), j \in [\ell_i], b \in \{0,1\}}\right). \tag{5.3}$$

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$: On input a secret key sk, a ciphertext ct, and the query bound $1 \leq Q \leq 2^\lambda$ in unary form, do the following:

1. Parse the secret key as Eq. (5.2) and the ciphertext as Eq. (5.3).
2. For all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ do the following.
   (a) Retrieve $\mathsf{FE.mpk}_i$ from the $\mathsf{FE.sk}_i$.
   (b) For all $j \in [\ell_i]$ compute $\mathsf{lab}'_{i,j} := \mathsf{IBE.Dec}(\mathsf{IBE.sk}_{i,j,\mathsf{FE.mpk}_{i,j}}, \mathsf{IBE.ct}_{i,j,\mathsf{FE.mpk}_{i,j}})$.
   (c) Compute $c_i := \mathsf{GC.Eval}(\{\mathsf{lab}'_{i,j}\}_{j\in[\ell_i]})$.
   (d) Compute $z_i := \mathsf{FE.Dec}(\mathsf{FE.sk}_i, c_i)$
3. Output $\{z_i\}_{i\in\mathcal{S}(|x|)\cap\mathcal{T}(|y|)}$.

In the full version [6],we show that the scheme is correct, satisfies efficiency in that all the algorithms run in time polynomial in their respective input lengths and satisfies AD-SIM (resp., NA-SIM) security, if the same holds for the FE.

5.2 FE for Turing Machines with NA-SIM Security

Here, we provide the construction of FE for Turing machines defined as in Section 2.1.3, which satisfies NA-SIM security. Recall that in FE for Turing machines, a ciphertext is associated with $(x, 1^t)$ and a secret key is for a Turing machine M, and the decryption results to 1 if the machine accepts the input within t steps and 0 otherwise. To construct such a scheme, we start with constructing two schemes with partial functionality and then combine them. The one scheme takes care of the case where $|(x, 1^t)| > |M|$, while the other takes care of the case where $|(x, 1^t)| \leq |M|$. Both schemes are obtained by applying the generic conversion in Section 5.1 to the schemes we constructed so far.

5.2.1 The Case of $|(x, 1^t)| > |M|$ We first show that by applying the conversion in Section 5.1 to the CPFE scheme in Section 3.2, we can obtain an FE scheme for Turing machines for the case where $|(x, 1^t)| > |M|$. Formally, we construct an FE for $R^> : \mathcal{A} \times \mathcal{B} \to \{0, 1\}$, where $\mathcal{A} = \{0, 1\}^*$, $\mathcal{B}$ is the set of all Turing machines, and

$$R^>((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \big(|(x, 1^t)| > |M|\big) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we recall that the scheme in Section 3.2 is an FE for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0, 1\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, $\mathcal{Y}_i = \{0, 1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1, 2, \ldots, i-1\}, \quad \mathcal{T}(i) = \{i\}, \quad f(x, 1^t) = \{U_{i,x,t}(\cdot)\}_{i\in[|(x,1^t)|-1]}, \quad g(M) = M$$

where $U_{i,x,t}(\cdot)$ is defined as Figure 2. The circuit (in particular, Step 2 of the computation) is padded so that the circuit size only depends on $|(x, 1^t)|$. Note that $U_{i,x,t}$ is in $\mathcal{X}_i$ even for x and t with unbounded length, since $\mathcal{X}_i$ contains circuits of unbounded size.

Circuit $U_{i,x,t}$

Hardwired constants: x, t.
On input $M = (Q, \delta, F) \in \{0,1\}^i$, proceed as follows:

1. Parse the input $M \in \{0,1\}^i$ as a description of a Turing machine.
2. Run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Fig. 2 : Circuit $U_{i,x,t}$.

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Equation (5.1) is equivalent to $R^{>}$ except for the case of $|(x, 1^t)| \leq |M|$. In this case, the decryption result in FE for R^{bndl} is an empty set $\emptyset$, whereas it should be 0 in FE for $R^{>}$. However, the former can be converted into the latter very easily. To do so, we add the extra step to the decryption algorithm of the former where it outputs 0 if the decryption result is $\emptyset$. Since the original scheme in Section 3.2 is NA-SIM secure, so is the resulting scheme by the security of our bundling construction in Section 5.1.

5.2.2 The Case of $|(x, 1^t)| \leq |M|$ We next show that by applying the conversion in Section 5.1 to the KPFE scheme in Section 4, we can obtain an FE scheme for Turing machines for the case where $|(x, 1^t)| \leq |M|$. Formally, we construct an FE for $R^{\leq} : \mathcal{A} \times \mathcal{B} \to \{0,1\}$, where $\mathcal{A} = \{0,1\}^*$, $\mathcal{B}$ is the set of all Turing machines, and

$$R^{\leq}((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \left(|(x, 1^t)| \leq |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we observe that the scheme in Section 4 can be used as an FE for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, depth $i \cdot \lambda$, and output length 1 and $\mathcal{Y}_i = \{0,1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1, 2, \ldots, i\}, \quad f(x, 1^t) = (x, 1^t), \quad g(M) = \{U_{i,M}(\cdot)\}_{i \in [M]},$$

where $U_{i,M}(\cdot)$ is defined as Figure 3. Here, we check that $U_{i,M}(\cdot)$ is in $\mathcal{Y}_i$. Recall that even though $\mathcal{Y}_i$ supports circuits with unbounded size, it has a bound on the depth of the circuit. We therefore argue that the depth of $U_{i,M}(\cdot)$ does not exceed $i\lambda$, even for unbounded size $|M|$. We evaluate the depth of Step 2 of the circuit, since this is the only non-trivial step. In the full version [6], we prove that this step can be implemented by a circuit with depth

$$t \cdot \mathrm{poly}(\log |x|, \log t, \log |M|) \leq i \cdot \mathrm{poly}(\log \lambda) \leq i \cdot \lambda$$

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M.
On input $y \in \{0,1\}^i$, proceed as follows:

1. Parse the input $y \in \{0,1\}^i$ as $(x, 1^t)$.
2. Otherwise, run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Fig. 3 : Circuit $U_{i,M}$.

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Equation (5.1) is equivalent to $R^{\leq}$ except for the case of $|(x, 1^t)| > |M|$. In this case, the decryption result in FE for R^{bndl} is an empty set $\emptyset$, whereas it should be 0 in FE for $R^{>}$. However, the former can be converted into the latter by adding the extra step to the decryption algorithm where it outputs 0 if the decryption result is $\emptyset$. This gives us the construction of FE for $R^{\leq}$. Since the original scheme is AD-SIM secure, so is the resulting scheme by the security of our bundling construction in Section 5.1.

5.2.3 Putting the Pieces Together Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathcal{A} = \{0,1\}^*$ and $\mathcal{B}$ to be the set of all Turing machines. We also set $R_1 = R^{>}$, $R_2 = R^{\leq}$, $\mathcal{X}_i = \mathcal{A}$, $\mathcal{Y}_i = \mathcal{B}$ for $i = 1, 2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1,2\}, \quad \mathcal{T}(i) = \{1,2\}, \quad f(x, 1^t) = \{(x, 1^t), (x, 1^t)\}, \quad g(M) = \{M, M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathcal{A}, \mathcal{B}$ defined as above, R^{bndl} defined as Equation (5.1) is

$$R^{\mathsf{bndl}}((x, 1^t), M) = \begin{cases} (1,0) & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \big(|(x, 1^t)| > |M|\big) \\ (0,1) & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \big(|(x, 1^t)| \leq |M|\big) \\ (0,0) & \text{otherwise.} \end{cases}.$$

An FE for the above relation is not exactly the same as the FE for Turing machines we defined in Section 2.1.3. However, the former readily implies the latter. To do so, we add the extra step to the decryption algorithm of the former where it checks whether there is 1 in the left or right slot of the decryption result and outputs 1 if there is. Otherwise, it outputs 0.

It is obvious that the obtained scheme satisfies correctness and efficiency requirements if so does the underlying schemes. As for the security, we can see by the security of our bundling construction in Section 5.1 that the resulting scheme is NA-SIM secure if we combine an AD-SIM and an NA-SIM secure scheme for $R^{\leq}$ and $R^{>}$ respectively.

Remark 7. One might think that FE for R^{bndl} constructed above leaks more information on $(x, 1^t)$ than FE for Turing machines and so is our final scheme, because the decryption result can be $\emptyset$, in addition to 0 or 1 and the decryptor can know whether $|(x, 1^t)| > |M|$ or not. However, it is not the case since whether the decryption result is $\emptyset$ or not can be checked only from the length of the string $|(x, 1^t)|$, which is not meant to be hidden in our definition (and other standard definitions) of FE.

To sum up, we have the following theorem:

Theorem 6. *We have FE for Turing machines with delayed collusion property that satisfies* NA-SIM *security from the sub-exponential LWE assumption.*

Due to space constraints, we provide our AD-SIM secure FE for NL in the full version [6].

Acknowledgement The fourth author was supported by JSPS KAKENHI Grant Number 19H01109 and JST CREST JPMJCR19F6.

References

1. S. Agrawal. Stronger security for reusable garbled circuits, new definitions and attacks. In *Crypto*, 2017.
2. S. Agrawal. Indistinguishability obfuscation minus multilinear maps: New methods for bootstrapping and instantiation. In *Eurocrypt*, 2019.
3. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.
4. S. Agrawal, S. Gurbanov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013.
5. S. Agrawal and M. Maitra. Fe and io for turing machines from minimal assumptions. In *TCC*, 2018.
6. S. Agrawal, M. Maitra, N. S. Vempati, and S. Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. Cryptology ePrint Archive, Report 2021/848, 2021. https://eprint.iacr.org/2021/848.
7. S. Agrawal, M. Maitra, and S. Yamada. Attribute based encryption (and more) for nondeterministic finite automata from lwe. In *CRYPTO*, 2019.
8. S. Agrawal and A. Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, 2017.
9. S. Agrawal and I. P. Singh. Reusable garbled deterministic finite automata from learning with errors. In *ICALP*, 2017.
10. P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. In *Crypto*, 2019.
11. P. Ananth and A. Sahai. Functional encryption for turing machines. In *TCC*, 2016.
12. P. Ananth and V. Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *TCC*, 2019.
13. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
14. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

15. N. Döttling and S. Garg. Identity-based encryption from the diffie-hellman assumption. In *CRYPTO*, 2017.
16. R. Garg, R. Goyal, G. Lu, and B. Waters. Dynamic collusion bounded functional encryption from identity-based encryption. Personal Communication, 2021.
17. R. Gay, A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *STOC*, 2021.
18. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions (extended abstract). In *FOCS*, 1984.
19. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, 2013.
20. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.
21. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012.
22. R. Goyal, V. Koppula, and B. Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *TCC*, 2016.
23. A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials over r to build io. In *EUROCRYPT*, 2019.
24. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. *Cryptology ePrint Archive, Report 2020/1003*, 2020.
25. F. Kitagawa and K. Tanaka. Key dependent message security and receiver selective opening security for identity-based encryption. In *PKC*, 2018.
26. H. Lin and J. Luo. Compact adaptively secure abe from k-lin: Beyond nc1 and towards nl. In *EUROCRYPT*, 2020.
27. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.
28. A. Sahai and H. Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *CCS*, 2010.
29. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.
30. A. C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, 1982.