Bounded Functional Encryption for Turing Machines: Adaptive Security from General Assumptions

Shweta Agrawal[1], Fuyuki Kitagawa[2], Anuja Modi[1], Ryo Nishimaki[2][0000−0002−5144−4619], Shota Yamada[3][0000−0002−7338−686X], and Takashi Yamakawa[2]

[1] IIT Madras, {shweta.a@cse.iitm.ac.in, anujamodi97@gmail.com}
[2] NTT Social Informatics Laboratories,
{fuyuki.kitagawa.yh,ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp
[3] AIST, yamada-shota@aist.go.jp

Abstract. The recent work of Agrawal et al., [Crypto '21] and Goyal et al. [Eurocrypt '22] concurrently introduced the notion of dynamic bounded collusion security for functional encryption (FE) and showed a construction satisfying the notion from identity based encryption (IBE). Agrawal et al., [Crypto '21] further extended it to FE for Turing machines in non-adaptive simulation setting from the sub-exponential learining with errors assumption (LWE). Concurrently, the work of Goyal et al. [Asiacrypt '21] constructed attribute based encryption (ABE) for Turing machines achieving adaptive indistinguishability based security against bounded (static) collusions from IBE, in the random oracle model. In this work, we significantly improve the state of art for dynamic bounded collusion FE and ABE for Turing machines by achieving *adaptive* simulation style security from a broad class of assumptions, in the standard model. In more detail, we obtain the following results:

1. We construct an adaptively secure (AD-SIM) FE for Turing machines, supporting dynamic bounded collusion, from sub-exponential LWE. This improves the result of Agrawal et al. which achieved only non-adaptive (NA-SIM) security in the dynamic bounded collusion model.
2. Towards achieving the above goal, we construct a *ciphertext policy* FE scheme (CPFE) for circuits of *unbounded* size and depth, which achieves AD-SIM security in the dynamic bounded collusion model from IBE and *laconic oblivious transfer* (LOT). Both IBE and LOT can be instantiated from a large number of mild assumptions such as the computational Diffie-Hellman assumption, the factoring assumption, and polynomial LWE. This improves the construction of Agrawal et al. which could only achieve NA-SIM security for CPFE supporting circuits of unbounded depth from IBE.
3. We construct an AD-SIM secure FE for Turing machines, supporting dynamic bounded collusions, from LOT, ABE for NC^1 (or NC) and private information retrieval (PIR) schemes which satisfy certain properties. This significantly expands the class of assumptions on which AD-SIM secure FE for Turing machines can be based. In particular, it leads to new constructions of FE for Turing machines including one based on polynomial LWE and one based on the combination of the bilinear decisional Diffie-Hellman assumption and the decisional Diffie-Hellman assumption on some specific groups. In contrast

the only prior construction by Agrawal et al. achieved only NA-SIM security and relied on *sub-exponential* LWE.
To achieve the above result, we define the notion of CPFE for read only RAM programs and succinct FE for LOT, which may be of independent interest.

4. We also construct an *ABE* scheme for Turing machines which achieves AD-IND security in the *standard model* supporting dynamic bounded collusions. Our scheme is based on IBE and LOT. Previously, the only known candidate that achieved AD-IND security from IBE by Goyal et al. relied on the random oracle model.

Keywords: Turing Machines · Functional Encryption · Attribute Based Encryption.

1 Introduction

Functional encryption (FE) [38, 15] is a powerful generalization of public key encryption, which goes beyond the traditional "all or nothing" access to encrypted data. In FE, a secret key is associated with a function f, a ciphertext is associated with an input $\mathbf{x}$ and decryption allows to recover $f(\mathbf{x})$. Security intuitively requires that the ciphertext and secret keys do not reveal anything other than the output of the computation. This can be formalized by positing the existence of a simulator which can simulate ciphertexts and secret keys given only the functions f_i and their outputs on the messages $\mathbf{x}_j$, namely $f_i(\mathbf{x}_j)$ for all secret keys sk_{f_i} and ciphertexts $\mathsf{ct}_{\mathbf{x}_j}$ seen by the adversary in the real world. This "simulation style" notion of security, commonly referred to as SIM security, is ruled out by lower bounds in a general security game [15, 2]. However, it can still be achieved in the *bounded* collusion model [30], which restricts the adversary to only request an a-priori bounded number of keys and challenge ciphertexts.

There has been intensive research in the community on FE in the last two decades, studying the feasibility for general classes of functions, from diverse assumptions, satisfying different notions of security. An exciting line of research has focused on FE for uniform models of computation supporting *unbounded* input lengths, such as Deterministic or Non-deterministic Finite Automata, Turing machines and Random Access machines [28, 11, 7, 4, 8, 34], in contrast to non-uniform models such as circuits. While circuits are expressive, they suffer from two major drawbacks in the context of FE. First, they force the input length to be fixed, a constraint that is inflexible and wasteful in most applications. Second, they necessitate the worst-case running time of the function on every input. By overcoming these limitations, FE schemes can fit demands of real world applications more seamlessly.

In this work, we study FE for Turing machines (henceforth TMFE) in the *bounded collusion* model, namely a security model which restricts the adversary to only request a bounded number of keys. Introduced by Gorbunov, Vaikuntanathan and Wee [30], this model has been popular since i) it is sufficient for multiple interesting real world scenarios, ii) it can support SIM style security, and iii) it can enable constructions from weaker assumptions or for more general functionalities. In the context of TMFE, the very recent work of Agrawal et al. [5] provided the first construction of bounded TMFE from

the (sub-exponential) Learning With Errors assumption (LWE).[4] Furthermore, this work achieved the notion of *dynamic* bounded collusion, where the collusion bound Q does not have to be declared during setup and may be chosen by the encryptor differently for each ciphertext, based on the sensitivity of the encrypted data. Thus, in their construction, the encryptor can choose an input $\mathbf{x}$ of unbounded length, a collusion bound Q and a time bound t, the key generator can choose a machine M of unbounded length and the decryptor runs M on $\mathbf{x}$ for t steps and outputs the result.

The work of Agrawal et al. [5] takes an important step forward in our understanding of bounded TMFE by providing the first feasibility result in a flexible dynamic model. However, it still leaves several important questions unanswered. For instance, the security notion achieved by TMFE is *non-adaptive* (denoted by NA-SIM) [15] where the adversary must send all the secret key requests before seeing the challenge ciphertext. Moreover, this limitation appears as a byproduct of the security notion achieved by the ingredient sub-schemes used for the construction (more on this below). Additionally, [5] relies on the heavy machinery of *succinct* single key FE for circuits [29], where succinctness means that the ciphertext size does not depend on the size of circuits supported (but may depend on output length and depth). Succinct FE is known to be constructible only from sub-exponential LWE[5] which necessitates the same assumption to underlie TMFE. This seems unnecessarily restrictive – in contrast, for the circuit model, bounded FE can be constructed from the much milder and more general assumption of public key encryption (PKE) [30, 12]. This raises the question of whether a strong primitive like succinct FE is really necessary to support the Turing machine model. As detailed below, succinct FE is a crucial tool in the construction, on whose properties the design relies heavily, and it is not clear whether this requirement can be weakened.

For the more limited primitive of *Attribute Based Encryption* (ABE), the recent work of [34] does provide a construction supporting Turing machines in the bounded collusion model (albeit without the dynamic property discussed above), assuming only the primitive of identity based encryption (IBE). Recall that ABE is a restricted class of FE in which the ciphertext is associated with both an input $\mathbf{x}$ and a message m and secret key is associated with a machine M. Decryption yields m given a secret key sk_M such that $M(\mathbf{x}) = 1$. Since IBE is a much weaker primitive than succinct FE and can be constructed from several weak assumptions such as the computational Diffie-Hellman assumption (CDH), the factoring assumption (Factoring), LWE and such others, this state of affairs is more satisfying. However, ABE is significantly weaker than FE since it does not hide the data on which the computation actually occurs, and is also an "all or nothing" primitive. Moreover, while their construction achieves strong adaptive security (denoted by AD-IND hereon), their construction relies on the random oracle model, unlike [5] which is NA-SIM in the standard model.

[4] Here, sub-exponential (resp., polynomial) LWE refers to the assumption that assumes the distinguishing advantage of the adversary for the decision version of LWE is sub-exponentially (resp., negligibly) small. The modulus to error ratio, which is another important parameter in LWE, will be referred to as approximation factor in this paper.

[5] Aside from obfustopia primitives such as compact FE [9, 13].

1.1 Our Results

In this work, we significantly improve the state of the art for dynamic bounded collusion TMFE by achieving adaptive simulation style security from a broad class of assumptions. In more detail, we obtain the following results:

1. We construct an adaptively secure (AD-SIM) TMFE, supporting dynamic bounded collusion, from sub-exponential LWE. This improves the result of [5] which achieved only NA-SIM security in the dynamic bounded collusion model.
2. Towards achieving the above goal, we construct a *ciphertext policy* FE[6] scheme (CPFE) for circuits of *unbounded* size and depth, which achieves AD-SIM security in the dynamic bounded collusion model from IBE and *laconic oblivious transfer* (LOT). Both IBE and LOT can be instantiated from a large number of mild assumptions such as CDH, Factoring or polynomial LWE. This improves the construction of [5] which could only achieve NA-SIM security for CPFE supporting circuits of unbounded depth from IBE.
3. We construct an AD-SIM secure TMFE, supporting dynamic bounded collusions, from LOT, ABE for NC^1 (or NC) and private information retrieval (PIR) schemes which satisfy certain properties. This significantly expands the class of assumptions on which AD-SIM secure TMFE can be based since ABE for NC^1 can be constructed from pairing based assumptions like the bilinear decisional Diffie-Hellman assumption (DBDH) [35] as well as *polynomial* LWE with slightly super-polynomial approximation factors[7] [31, 14], LOT can be based on CDH, Factoring and polynomial LWE [20, 21, 16], and PIR with the required properties can also be based on LWE [17], the decisional Diffie-Hellman assumption (DDH), or the quadratic residuosity assumption (QR) [22]. This leads to new constructions of TMFE as follows:
   - one based on the polynomial hardness of LWE with quasi-polynomial approximation factors,
   - one based on the combination of DBDH and DDH on some specific groups.
   - one based on the combination of DBDH and QR.

   If we instantiate PIR with LWE, we need ABE for NC and LWE with quasi-polynomial approximation factors [32] since the answer function of PIR from LWE [17, 27] is in NC. See Section 5 for the detail. In contrast the only prior construction by [5] achieved only NA-SIM security and relied on *sub-exponential* LWE. When instantiated with LWE, we observe that the above construction improves the first construction we described. However, we still present the first construction because it is much simpler.
4. We also construct an ABE scheme for Turing machines which achieves AD-IND security in the *standard model* supporting dynamic bounded collusions. Our scheme is based on IBE and LOT. Previously, the only known candidate that achieved AD-IND security from IBE relied on the random oracle model [34].

[6] A secret key and a ciphertext are associated with an input x and a function f, respectively unlike the standard FE.

[7] That is, $O(\lambda^{\omega(1)})$.

Table 1 : Comparison for bounded collusion-resistant FE for uniform models of computation

FE	Class	Security	Model	Assumption
[7]	TM	1-key NA-SIM	static	$(\mathsf{sub\text{-}exp}, \mathsf{sub\text{-}exp})$-LWE†
[6] (SKFE)	NFA	Sel-SIM	static	$(\mathsf{sub\text{-}exp}, \mathsf{sub\text{-}exp})$-LWE†
[5]	NL	AD-SIM	dynamic	$(\mathsf{sub\text{-}exp}, \mathsf{sub\text{-}exp})$-LWE†
[5]	TM	NA-SIM	dynamic	$(\mathsf{sub\text{-}exp}, \mathsf{sub\text{-}exp})$-LWE†
Ours §4	TM	AD-SIM	dynamic	$(\mathsf{sub\text{-}exp}, \mathsf{sub\text{-}exp})$-LWE†
Ours §5	TM	AD-SIM	dynamic	$(\mathsf{poly}, \mathsf{quasi\text{-}poly})$-LWE†
Ours §5	TM	AD-SIM	dynamic	DDH‡ & DBDH
Ours §5	TM	AD-SIM	dynamic	QR & DBDH

† For $\mathsf{adv} \in \{\mathsf{poly}, \mathsf{sub\text{-}exp}\}$ and $\mathsf{apprx} \in \{\mathsf{quasi\text{-}poly}, \mathsf{sub\text{-}exp}\}$, $(\mathsf{adv}, \mathsf{apprx})$-LWE means that {polynomial, sub-exponential} hardness of LWE with {quasi-polynomial, sub-exponential} approxiation factors, respectively. ‡ DDH over the multiplicative sub-group of $\mathbb{Z}_q$ where q is a prime.

1.2 Other Related Work

A *key policy* FE[8] (KPFE) for Turing machines supporting only a *single* key request was provided by Agrawal and Singh [7] based on sub-exponential LWE. Agrawal, Maitra and Yamada [6] provided a construction of KPFE for non-deterministic finite automata (NFA) which is secure against bounded collusions of arbitrary size. However, this construction is in the symmetric key setting. These constructions do not support the dynamic collusion setting. The first works to (concurrently) introduce and support the notion of dynamic bounded collusion are [23] and [5]. Both works obtain simulation secure KPFE schemes for circuits with dynamic collusion resistance. [5] additionally obtain succinct CPFE/KPFE schemes for circuits with dynamic collusion property, and also to support Turing machines and NL with different security trade-offs. We provide a comparison for bounded collusion-resistant FE for uniform models of computation in Table 1. All the results in the table are about FE whose encryption time depends on the running-time of computation. There are FE schemes whose encryption time does not depend the running-time of computation [28, 11, 4, 36]. However, such constructions are based on strong assumptions such as extractable witness encryption [28] and compact FE [11, 4, 36]. We also omit works based on indistinguishability obfuscation. The focus of the present work is on weak assumptions.

1.3 Our Techniques

In this section, we provide an overview of our techniques. Our final construction is obtained by going through number of steps. We refer to Figure 1 for the overview.

[8] This is the same as the standard FE. We use this term to distinguish from CPFE.

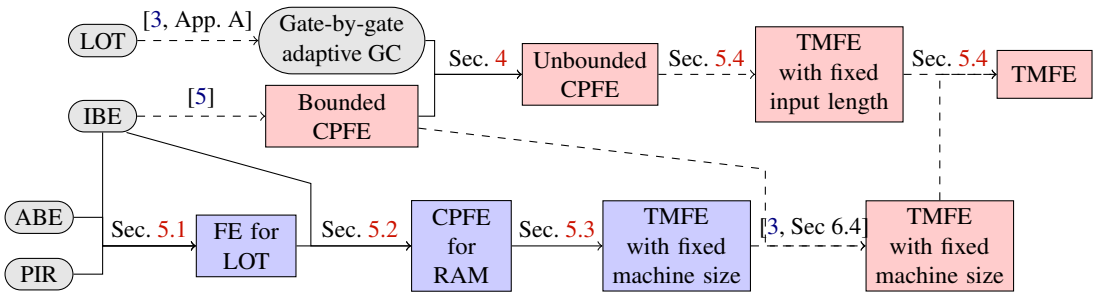

Fig. 1 Illustration of our construction path. Each rectangle represents FE and rounded rectangles represents other primitives. "Bounded CPFE" (resp., "Unbounded CPFE") means CPFE for bounded (resp., unbounded) size circuits. The red (resp., blue) rectangles represent FE with AD-SIM security under bounded dynamic collusion (resp., NA-SIM security under single key collusion). The dashed lines indicate known implications or implications that can be shown by adapting previous techniques relatively easily. The solid lines indicate implications that require new ideas and are shown by us. We do not include (selectively secure) garbled circuits, secret key encryption, and PRF in the figure in order to simplify the presentation.

Recap of TMFE by [5]: To begin, we recap some of the ideas used in the construction of TMFE provided by [5]. At a high level, their approach is to separate the cases where the length of the input x and running time bound 1^t is larger than the machine size $|M|$ and one where the opposite is true, i.e. $|(x, 1^t)| \leq |M|$ and $|(x, 1^t)| > |M|$. They observe that running these restricted schemes in parallel allows supporting either case, where the one sub-scheme is used to decrypt a ciphertext if $|(x, 1^t)| \leq |M|$ and the second is used otherwise. We note that such a compiler was first developed by [6] in the symmetric key setting and [5] uses ideas from [33] to upgrade it to the public key setting.

To construct the restricted sub-schemes, [5] uses KPFE for the case $|(x, 1^t)| \leq |M|$ and CPFE for $|(x, 1^t)| > |M|$. For concreteness, let us consider the case $|(x, 1^t)| \leq |M|$. Now, using the "delayed encryption" technique of [33] (whose details are not relevant for our purpose), one may assume that there exists an infinite sequence of KPFE instances and the i-th instance supports circuits with input length i. To encrypt a message x with respect to the time bound 1^t, they use the $|(x, 1^t)|$-th instance of KPFE. To generate the secret key for a Turing machine M, they 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 $(x, 1^t)$ of length i, and then runs the machine M for t steps on this input to generate the output. Secret keys for $C_{i,M}$ are generated using the i-th instance of KPFE for all of $i \in [|M|]$. A crucial detail here is that M can be of unbounded size, because each KPFE instance supports unbounded size circuits. Now, decryption is possible when $|(x, 1^t)| \leq |M|$ by using the $|(x, 1^t)|$-th instance. To construct the KPFE scheme, the authors enhance constructions of "succinct KPFE" from the literature [29, 1], which can be constructed from (sub-exponential) LWE. The resultant scheme satisfies AD-SIM security against (dynamic) bounded collusions.

To handle the opposite case, namely $|(x, 1^t)| > |M|$, the authors follow the "dual" of the above procedure, using a CPFE scheme in place of KPFE, which supports circuits of unbounded depth (hence size). In more detail, to encrypt a message $(x, 1^t)$, they construct 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, runs it on input x for t steps and outputs the result. They encrypt the circuit $U_{i,x,t}$ using the i-th instance of CPFE for all of $i \in [|(x, 1^t)|]$. Note that it is necessary that the CPFE scheme support circuits of unbounded *depth* and not just size, since the circuit must run Turing machine for t steps, where t may be arbitrarily large. The authors use IBE to instantiate such a CPFE. However, they can only achieve NA-SIM, where the adversary must make all its key requests before obtaining the challenge ciphertext. This limitation is inherited by the resultant TMFE scheme even though KPFE satisfies AD-SIM security as discussed above.

TMFE with AD-SIM security: As discussed above, the missing piece in constructing TMFE with AD-SIM security from LWE is an instantiation of CPFE supporting unbounded depth circuits with AD-SIM security. We now show how to design this by carefully combining (in a non black-box way) two ingredients – i) an AD-SIM secure CPFE for circuits of *bounded* depth, size and output, denoted by BCPFE, which was constructed in [5] using IBE, and ii) adaptively secure garbled circuits (GC) based on LOT [24]. Our construction makes crucial use of the structural properties of the LOT based adaptive GC constructed by Garg and Srinivasan [24]. We describe this next.

Adaptively secure garbled circuits via LOT: Garg and Srinivasan [24] provided a construction of adaptively secure GC with near optimal online rate by leveraging the power of LOT. Recall that LOT [18] is a protocol between two parties: sender and a receiver. The receiver holds a large database $D \in \{0, 1\}^N$ and sends a short digest d (of length λ) of the database to the sender. The sender has as input a location $L \in [N]$ and two messages (m_0, m_1). It computes a read-ciphertext c using its private inputs and the received digest d by running in time $\mathrm{poly}(\log N, |m_0|, |m_1|, \lambda)$ and sends c to the receiver. The receiver recovers the message $m_{D[L]}$ from the ciphertext c and the security requirement is that the message $m_{1-D[L]}$ remains hidden. Updatable LOT additionally allows updates to the database.

The main idea in [24] is to "linearize" the garbled circuit, namely to ensure that the simulation of a garbled gate g depends only on simulating one additional gate. With this linearization in place, they designed a careful sequence of hybrids based on a pebbling strategy where the number of changes required in each intermediate hybrid is $O(\log(|C|))$. In more detail, their construction views the circuit C to be garbled as a sequence of step circuits along with a database D, where the i^{th} step circuit implements the i^{th} gate in the circuit. The database D is initialized with the input x and updated to represent the state of the computation as the computation progresses. Thus, at step i, the database contains the output of every gate $g < i$ in the execution of C on x. The i^{th} step circuit reads contents from two pre-determined locations in the database, corresponding to the input wires, and writes a bit, corresponding to the output of the gate, to location i. Thus, they reduce garbling of the circuit to garbling each step circuit along with the database D.

Coming back to our goal of CPFE for unbounded depth circuits, a starting point idea is to use a sequence of bounded size schemes BCPFE to encode a sequence of low depth step circuits described above. Intuitively, we leverage the decomposability of the adaptive GC construction so that a BCPFE scheme can generate each garbled version of the step circuit, using randomness that is derived jointly from the encryptor and key generator via a pseudorandom function (PRF). Specifically, the key generator provides BCPFE keys for input x, along with a PRF input tag, and the encryptor provides BCPFE ciphertexts for the GC step circuits along with a PRF seed. Put together, BCPFE decrypts to provide the inner decomposable GC (generated using jointly computed PRF output), which may then be evaluated to recover $C(x)$. A crucial detail swept under the rug here is how the sequence of GCs interacts with the database which captures the state of the computation. In particular, as the computation proceeds, the database must be updated, and these updates must be taken into account while proceeding with the remainder of the evaluation. This detail is handled via the *updatable* property of our LOT similarly to [24]. In the interest of brevity, we do not describe it here, and refer the reader to Section 3 for details.

Assuming the sequence of BCPFE schemes can produce the garbled step circuits and garbled database, we still run into another problem in the security proof – the number of BCPFE ciphertexts is as large as the size of the circuit being encrypted while on the other hand, there is an information-theoretic barrier that the key size of an AD-SIM secure CPFE should grow with the number of challenge ciphertexts [15]. This would bring us right back to where we started as we want to handle unbounded depth circuits and the key generator cannot even know this depth, so we cannot create enough space in the key to support this embedding. Our key observation to overcome this hurdle is that we do not need to simulate all the ciphertexts simultaneously. In particular, by relying on the pebbling-based simulation strategy used in [24], we can upper bound the number of ciphertexts in "simulation mode" by a fixed polynomial in each hybrid in the proof. This allows us to embed a simulated GC into the BCPFE secret key which is of fixed size, thereby allowing the post challenge queries required for AD-SIM security. To formalize this idea, we introduce an abstraction which we call "gate-by-gate garbling" (see Section 3), which is similar to locally simulatable garbling introduced by Ananth and Lombardi [10]. For more details, please see Sections 3 and 4.

TMFE *without succinct* KPFE *for circuits:* We now describe our construction of TMFE *without* using succinct KPFE for circuits. The high level template for the final construction is the same as discussed earlier, namely, to construct two sub-schemes that handle the cases $|(x, 1^t)| \leq |M|$ and $|(x, 1^t)| > |M|$ separately. Previously, we showed how to construct CPFE with AD-SIM security, for unbounded depth circuits from IBE and LOT, and used this to handle the case $|(x, 1^t)| > |M|$. The counterpart $|(x, 1^t)| \leq |M|$ was handled using KPFE for circuits of unbounded size, which was constructed in [5] by upgrading the succinct, single key KPFE of Goldwasser et al. [29] from (sub-exponential) LWE. Our goal is to construct FE that can handle the case of $|(x, 1^t)| \leq |M|$ and satisfies AD-SIM security *without* relying on succinct KPFE.

To begin, observe that the generalized bundling technique discussed above lets us focus on the case where $|x, 1^t|$ is fixed, but $|M|$ is unbounded. Moreover, it suffices to restrict ourselves to 1-NA-SIM security, since 1-NA-SIM implies AD-SIM for FE with

bounded message length (please see Section [3, Sec 6.4]). In [5], the authors use succinct KPFE to instantiate this scheme by taking advantage of the fact that the size of the circuit associated with the secret key in the succinct KPFE is unbounded. Thus, we can embed a machine M of unbounded size into the secret key. Then we can run the Turing machine inside the circuit for $|M|$ steps, which exceeds t (since we have $t < |M|$) and thus finishes the computation.

Our starting point observation is that since $|(x, 1^t)| \leq |M|$, where $|(x, 1^t)|$ is fixed and $|M|$ is unbounded, we can think of M as a large database, x as a short input and t as bounded running time, which naturally suggest the random access machines (RAM) model of computation. Intuitively, if $|M|$ is massive but $|x|$ and t are small, then running M on x requires only a bounded number of lookups in the transition table and a bounded number of steps, regardless of the size of M. Motivated by this observation, we cast M as a large database and construct a program P which has $(x, 1^t)$ hardwired in it, and executes $M(x)$ via RAM access to M. It is important to note that even if the program does not have M as an input, RAM access to M suffices, because the transition only depends on the description of the current state and the bit that is pointed to by the header. To capture this notion in the setting of FE, we introduce a new primitive, which we call CPFE for (read only) RAM programs. Here, the encryptor encrypts the program $P_{(x,1^t)}$ above, the key generator provides a key for database M and decryption executes $P_{(x,1^t)}$ on M, which is equal to $M(x)$. Crucially, the running time of encryption is required to be independent of $|D|$.

To construct a CPFE for read only RAM, we build upon ideas that were developed in the context of garbled RAM constructions [37]. In these constructions, a garbled RAM program consists of t garbled copies of an augmented "step circuit" which takes as input the current CPU state, the last read bit and outputs an updated state and the next read location. Copy i of the CPU step circuit is garbled so that the labels for the output wires corresponding to the output state match the labels of the input wires corresponding to the input state in the next copy $i + 1$ of the circuit. The obvious question in this context is how to incorporate data from memory into the computation – clearly, decomposing the computation necessitates some mechanism in which the sequence of garbled circuits communicate with the outside memory[9]. To enable this, previous works have used IBE and oblivious RAM (ORAM) [37, 25]. At a very high level, IBE is used to choose the correct label of the GC as follows – the garbled memory can consist of IBE secret keys for identity (i, b) where i is the given location and b is the bit stored in it, while the garbled circuits can output IBE ciphertexts whose messages are the labels of the next circuit, under identities (i, b). On the other hand, ORAM hides the position read.

However, an immediate hurdle is that ORAM necessitates two parties (client and server) to agree on a secret key. Translated into our setting, this would require that the encryptor and key generator share some secret information – but this is not possible as we are in the public key setting. To overcome this barrier, we introduce the notion of FE for LOT, denoted by LOTFE (Section 5.1). In LOTFE, the encryptor has two messages (μ_0, μ_1) and a database location i. The key generator has a database D as input. Decryption allows to recover $\mu_{D[i]}$ and security hides both $\mu_{1-D[i]}$ as well as the

[9] The careful reader may note the similarity with the adaptive garbled circuit construction by [24] discussed above.

position i. This corresponds to hiding the "other label" as well as the position that was read, in the garbled RAM approach.

It remains to construct LOTFE. Observe that LOTFE must still satisfy the desired succinctness properties, in that a secret key should encode unbounded size data and the running time of the encryption algorithm should be independent of this size. However, by reducing the requisite functionality to something simple like LOT, we earn several benefits over using succinct KPFE. In particular, since we only need to support the table lookup functionality, we can replace the fully homomorphic encryption (FHE) which was used in the succinct KPFE construction [29] with the much weaker *private information retrieval* (PIR). Due to this, we can replace ABE for circuits that is used in the construction of [29] by the much weaker ABE for NC^1 (or NC), which in turn can be constructed by a wider variety of assumptions. We discuss this in more detail below.

Let us briefly recall the main ideas used to construct succinct KPFE. The construction of [29] carefully stitches together ABE for circuits, FHE and GC as follows. At a very high level, FHE is used to encrypt the input x, and this ciphertext $\hat{x}$ is then used as the attribute string for ABE. To encode the circuit f, we construct an ABE secret key for a closely related circuit f', which is used to restrict computation on the FHE ciphertext embedded in the ABE encodings. During decryption, we can check if $f'(\hat{x}) = 1$ and recover the message if so. Intuitively, $f'(\hat{x})$ will represent the bits of an FHE encryption of $f(x)$, denoted by $\widehat{f(x)}$ and provide a message $\mathsf{lbl}_{i,0}$ if the i^{th} bit of $\widehat{f(x)}$ is 0 and $\mathsf{lbl}_{i,1}$ if the i^{th} bit of $\widehat{f(x)}$ is 1. These labels are then used as inputs to the GC which encodes the FHE decryption circuit, so that the decryptor can recover $f(x)$ as desired. Note that the usage of GC implies that the construction can only support a single function key, since otherwise the adversary can recover labels for multiple inputs, violating GC security.

Following a similar template, we can construct 1-NA-SIM secure LOTFE using ABE for NC^1 (or NC), PIR and GC. We encrypt labels under attributes corresponding to the PIR query and provide a key for the PIR answer function to recover the labels corresponding to the PIR answer. These are subsequently fed into the garbled circuit to recover the answer in the clear. We need that the PIR answer function is in NC^1 so that it fits ABE for NC^1. Towards this, we show that PIR from QR or DDH has its answer function in NC^1 and thus can be combined with ABE for NC^1. If we instantiate PIR with an FHE-based scheme [17, 27], the answer function is in NC and we need ABE for NC, which can be instantiated with LWE with quasi-polynomial approximation factors. Our LOTFE not only allows to use various assumptions other than LWE, which was not possible before, but also allows us to remove the complexity leveraging required for the LWE based construction described before, while achieving AD-SIM secure TMFE at the end[10]. We need Sel-IND secure ABE as a building block to achieve 1-NA-SIM secure LOTFE.

The reason why Sel-IND security suffices for our case is that the reduction algorithm can guess the target attribute the adversary chooses only with polynomial guess. Although

[10] We observe that the above construction when instantiated with LWE improves the first construction we described. However, we still present the first construction because it is much simpler.

there are exponentially many possible PIR queries, the reduction algorithm only has to guess an input that is encoded inside PIR.query. This is because the randomness used for computing the query is not controlled by the adversary, but by the reduction algorithm. Since there are only polynomial number of possible inputs to PIR query function, the guessing can be done only with polynomial security loss. Please see Section 5 for the complete description.

ABE for TM from LOT *and* IBE*:* Lastly, we construct ABE for Turing machines supporting AD-SIM security with dynamic bounded collusions. Our construction relies on LOT and IBE. In contrast to the construction of [34], we do not rely on the random oracle and moreover, we support dynamic bounded collusions.

As before, we consider two cases, namely $|(x, 1^t)| \leq |M|$ and the opposite $|(x, 1^t)| > |M|$. Observe that for the case of longer input, the LOT based CPFE construction discussed above suffices since it implies TMFE where $|(x, 1^t)| > |M|$ with dynamic bounded collusions as discussed above. Therefore we focus on the case of shorter input. For this, our starting point is the *single* key, NA-IND secure (non-adaptively indistinguishable) ABE for TM constructed by the recent work of Goyal et al. [34], which relies on IBE. We upgrade this to adaptively (AD-SIM) secure ABE for TM supporting dynamic bounded collusions of arbitrary size, when $|(x, 1^t)| \leq |M|$, as follows. To begin, we observe that single key NA-IND security in fact implies single key NA-SIM security in the context of ABE (see [3, Remark 2.5] for an argument). Then, we combine the above single key NA-SIM ABE for Turing machines, denoted by 1-TMABE and AD-SIM secure BCPFE (for bounded circuits) with bounded dynamic collusion similarly to [5, Sec. 4].

In more detail, the master public key and master secret key of the final ABE scheme are those of BCPFE. To encrypt message m under attribute $(x, 1^t)$ for a collusion bound 1^Q, the encryptor first constructs a circuit $\mathsf{1\text{-}TMABE.Enc}(\cdot, x, 1^t, m)$, which is an encryption algorithm of the single-key ABE for TM scheme, that takes as input a master public key and outputs an encryption of the attribute $(x, 1^t)$ and message m under the key. The encryptor then encrypts the circuit using the BCPFE scheme with respect to the bound 1^Q. To generate a secret key for a Turing machine M, the key generator freshly generates a master key pair of 1-TMABE, namely $(\mathsf{1\text{-}TMABE.mpk}, \mathsf{1\text{-}TMABE.msk})$. It then generates a BCPFE secret key BCPFE.sk corresponding to the string 1-TMABE.mpk and an ABE secret key $\mathsf{1\text{-}TMABE.sk}_M$ for the machine M. The final secret key is $(\mathsf{BCPFE.sk}, \mathsf{1\text{-}TMABE.sk}_M)$. Decryption is done by first decrypting the BCPFE ciphertext using the BCPFE secret key to recover $\mathsf{1\text{-}TMABE.Enc}(\mathsf{1\text{-}TMABE.mpk}, x, 1^t, m)$ and then using the secret key $\mathsf{1\text{-}TMABE.sk}_M$ to perform ABE decryption and recover the message m if $M(x) = 1$ within t steps. Security follows from the individual security of the two underlying schemes and yields an AD-SIM secure ABE for TM for an a-priori bounded $|(x, 1^t)|$ with bounded dynamic collusion.

To remove the restriction on $|(x, 1^t)|$, we use the "generalized bundling" trick of [5, Sec 6.2.2], for the case of $|(x, 1^t)| < |M|$. Thus, we obtain AD-SIM ABE for TM where $|(x, 1^t)| < |M|$ with dynamic bounded collusions. Finally, we combine AD-SIM ABE for TM with $|(x, 1^t)| > |M|$ and one with $|(x, 1^t)| \leq |M|$ as described above. The

transformation yields AD-SIM secure ABE for TM with dynamic bounded collusions, which readily implies AD-IND security. Please see [3, Sec 6] for further details.

2 Preliminaries

Here, we define functional encryption (FE) with dynamic bounded collusion, which is introduced by [5, 23]. The notion is stronger than conventional bounded collusion FE [30, 12] in that the collusion bound can be determined by an encryptor dynamically, rather than being determined when the system is setup. We note that some of the definitions here are taken verbatim from [5]. We refer to [3, Sec 2] for additional preliminaries.

2.1 Functional Encryption

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 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 security parameter λ in unary 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.

$\mathsf{Encrypt}(\mathsf{mpk}, x, 1^Q) \to \mathsf{ct}$: The encryption algorithm takes as input a master public key mpk, a message $x \in \mathcal{X}_{\mathsf{prm}}$, and a bound on the collusion Q in unary. 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.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}, 1^Q) \to m$ or $\perp$: The decryption algorithm takes as input a ciphertext ct, a secret key sk, and a bound Q associated with the ciphertext. It outputs the message m or $\perp$ which represents that the ciphertext is not in a valid form.

Remark 1. We also consider single collusion FE, which is a special case where Q is always fixed to be $Q = 1$. In such a case, we drop 1^Q from the input to the algorithms for simplicity of the notation.

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

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

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

We define simulation-based security notions for FE in the following.

Definition 2 (AD-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* $\mathsf{Sim} = (\mathsf{SimEnc}, \mathsf{SimKG})$*, we consider the experiments in Figure 2.*

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathsf{Sim}}(1^\lambda)$
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 $\mathcal{V} := \{(z^{(q)} := R(x, y^{(q)}), y^{(q)}, \mathsf{sk}^{(q)})\}_{q \in [Q_1]}$.
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

Fig. 2 AD-SIM security for FE

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*. In the experiments:*

- *The oracle* $\mathcal{O}(\mathsf{msk}, \cdot) = \mathsf{KeyGen}(\mathsf{msk}, \cdot)$ *with* $1 \le Q_1 \le Q$*, 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* $Q_1 + Q_2 \le Q$

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

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

Remark 2 (Non-adaptive security). We can consider a variant of the above security definition where the adversary is not allowed to make a secret key query after the ciphertext ct is given (i.e., $Q_1 = Q$). We call the notion non-adaptive simulation security (NA-SIM). In particular, when we consider single collusion FE, the notion is called 1-NA-SIM. We refer to [3, Sec 2.4] for the formal definitions.

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

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 $R(C,x) = \bot$. 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 CPFE for circuits, even though the input length of the circuits in $\mathcal{C}_{\mathsf{inp}}$ is bounded, the size of the circuits is unbounded.

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 $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ as

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

3 Gate-by-Gate Garbling with Pebbling-based Simulation

We define a notion of *gate-by-gate garbling* and its *pebbling-based simulation*. This is an abstraction of the backbone of the adaptive garbling by Garg and Srinivasan [24].

First, we define a syntax of a standard garbling scheme.[11] A garbling scheme for a circuit class $\mathcal{C}$ cosists of PPT algorithms $\mathsf{GC} = (\mathsf{GCkt}, \mathsf{GInp}, \mathsf{GEval})$ with the following syntax:

$\mathsf{GCkt}(1^\lambda, C) \to (\widetilde{C}, \mathsf{st})$: The circuit garbling algorithm takes as input the unary representation of the security parameter λ and a circuit $C \in \mathcal{C}$ and outputs a garbled circuit $\widetilde{C}$ and state information st.

$\mathsf{GInp}(\mathsf{st}, x) \to \widetilde{x}$: The input garbling algorithm takes as input the state information st and an input x and outputs a garbled input $\widetilde{x}$.

$\mathsf{GEval}(\widetilde{C}, \widetilde{x}) \to y$: The evaluation algorithm takes as input the garbled circuit $\widetilde{C}$ and garbled input $\widetilde{x}$ and outputs an output y.

Definition 3 (Correctness). *A garbling scheme* $\mathsf{GC} = (\mathsf{GCkt}, \mathsf{GInp}, \mathsf{GEval})$ *is correct if for all circuits* $C \in \mathcal{C}$ *and its input* x*,*

$$\Pr[(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GCkt}(1^\lambda, C), \widetilde{x} \leftarrow \mathsf{GInp}(\mathsf{st}, x) : \mathsf{GEval}(\widetilde{C}, \widetilde{x}) = C(x)] = 1.$$

In addition to the security notion for a standard garbling scheme in [3, Definition 2.8], we introduce a new security notion specific to gate-by-gate garbling, which we call *pebbling-based security*. For defining gate-by-gate garbling, we prepare some notations about circuits.

[11] Note that the syntax defined here is more general than that of Yao's garbling defined in [3, Definition 2.8].

Notations. For a circuit C, we denote by Gates the set of all gates of C. We use inp and out to mean the input-length and output-length of C, respectively. A unique index from 1 to N is assigned to each bit of input and gate where N is the sum of the input-length and the number of gates of C. In particular, each bit of input is assigned by indices from 1 to inp, each intermediate gate is assigned by indices from $\mathsf{inp}+1$ to $N-\mathsf{out}$, and each output gate is assigned by indices from $N-\mathsf{out}+1$ to N. We assume that a circuit has fan-in 2 and unbounded fan-out without loss of generality. A gate $\mathsf{G} \in \mathsf{Gates}$ is represented as $(g_{\mathsf{G}}, i_{\mathsf{G}}, A_{\mathsf{G}}, B_{\mathsf{G}})$ where $g_{\mathsf{G}} : \{0,1\} \times \{0,1\} \to \{0,1\}$ is a function corresponding to G, i_{G} is the index of G, and A_{G} and B_{G} are indices of the input bits or gates whose values are passed to G as input. We assume $A_{\mathsf{G}} < i_{\mathsf{G}}$ and $B_{\mathsf{G}} < i_{\mathsf{G}}$ without loss of generality. We call $(i_{\mathsf{G}}, A_{\mathsf{G}}, B_{\mathsf{G}})$ the *topology* of G and denote it by $\mathsf{top}(\mathsf{G})$. We call the set of topology of all gates the topology of C and denote it by $\mathsf{top}(C)$.

Intuitively, gate-by-gate garbling is a garbling scheme whose circuit garbling algorithm can be decomposed into gate garbling algorithms for each gate, whose efficiency is independent of the size of the circuit. The formal definition is given below:

Definition 4 (Gate-by-Gate Garbling:). *A garbling scheme* $\mathsf{GC} = (\mathsf{GCkt}, \mathsf{GInp}, \mathsf{GEval})$ *for a circuit class* $\mathcal{C}$ *is said to be gate-by-gate garbling with* $N_{\mathsf{rand}} = N_{\mathsf{rand}}(\mathsf{top}(C))$ *randomness slots and randomness length* ℓ *if* GCkt *can be decomposed into PPT sub-algorithms* $(\mathsf{GSetup}, \mathsf{GGate})$ *as follows:*

$\mathsf{GCkt}(1^\lambda, C)$*: The circuit garbling algorithm proceeds as follows:*

1. *Run* $\mathsf{GSetup}(1^\lambda, \mathsf{top}(C))$ *to generate a public parameter* pp.
2. *For* $i \in [N_{\mathsf{rand}}]$*, generate a randomness* $\mathsf{R}_i \leftarrow \{0,1\}^\ell$ *for* $\ell = \mathrm{poly}(\lambda)$ *that does not depend on* C.
3. *For* $\mathsf{G} \in \mathsf{Gates}$*, run* $\mathsf{GGate}(\mathsf{pp}, \mathsf{G}, \{\mathsf{R}_i\}_{i \in S(\mathsf{G})})$ *to generate a garbled gate* $\widetilde{\mathsf{G}}$*. Here,* $S(\mathsf{G}) \subseteq [N_{\mathsf{rand}}]$ *is a subset of size* $O(1)$ *that is efficiently computable from* G *and* $\mathsf{top}(C)$.
4. *Output a garbled circuit* $\widetilde{C} := \left(\mathsf{pp}, \{\widetilde{\mathsf{G}}\}_{\mathsf{G} \in \mathsf{Gates}}\right)$ *and the state information* $\mathsf{st} := (\mathsf{pp}, \{\mathsf{R}_i\}_{i \in S_{\mathsf{st}}})$ *where* $S_{\mathsf{st}} \subseteq [N_{\mathsf{rand}}]$ *is a subset of size* $O(\mathsf{inp} + \mathsf{out})$ *that is efficiently computable from* $\mathsf{top}(C)$.

We require GC *to satisfy the following requirements.*

1. $\mathsf{GGate}(\mathsf{pp}, \mathsf{G}, \{\mathsf{R}_i\}_{i \in S(\mathsf{G})})$ *runs in time* $\mathrm{poly}(\lambda)$ *independently of the size of* C *where* $\mathsf{pp} \leftarrow \mathsf{GSetup}(1^\lambda, \mathsf{top}(C))$ *and* $\mathsf{R}_i \leftarrow \{0,1\}^\ell$ *for* $i \in [N_{\mathsf{rand}}]$.
2. $\mathsf{GInp}(\mathsf{st}, x)$ *is deterministic and runs in time* $\mathrm{poly}(\lambda, \mathsf{inp}, \mathsf{out})$ *independently of the size of* C *where* $\mathsf{pp} \leftarrow \mathsf{GSetup}(1^\lambda, \mathsf{top}(C))$*,* $\mathsf{R}_i \leftarrow \{0,1\}^\ell$ *for* $i \in [N_{\mathsf{rand}}]$*, and* $\mathsf{st} := (\mathsf{pp}, \{\mathsf{R}_i\}_{i \in S_{\mathsf{st}}})$.

We require gate-by-gate garbling to satisfy (M, T)*-pebbling-based security* for some parameters M and T, which intuitively requires the following: There are three modes of gate garbling algorithms, the white mode, black mode, and gray mode. The white mode gate garbling algorithm is identical to the real gate garbling algorithm. The black mode gate garbling algorithm is a "simulation" algorithm that simulates a garbled gate without knowing the functionality of the gate. The gray mode garbling algorithm is an

"input-dependent simulation" algorithm that simulates a garbled gate by using both the circuit C and its input x that are being garbled. We call a sequence of modes of each gate of C a configuration of C. There is a configuration-based input-garbling algorithm that simulates a garbled input by using C and a configuration as additional inputs. When the configuration is all-white, it corresponds to the real input garbling algorithm, and when the configuration is all-black, it corresponds to a legitimate "simulation" algorithm that only uses $C(x)$ instead of C and x. The security requires that there is a sequence of configurations of length T that starts from the all-white configuration, which corresponds to the real garbling algorithm, to the all-black configuration, which corresponds to the simulation algorithm, such that:

1. the number of gates in the gray mode in any intermediate configuration is at most M, and
2. garbled circuits and garbled inputs generated in neighboring configurations are computationally indistinguishable even if the distinguisher can specify all the randomness needed for generating garbled gates whose modes are black or white in both of those configurations.

We give the formal definition of pebbling-based security in [3, Sec 3.1].

Instantiation. Our definition of gate-by-gate garbling with pebbling-based simulation security captures the backbone of the proof technique of adaptive garbling in [24]. The following lemma is implicit in their work. The lemma is proven in [3, Appendix A] for completeness.

Lemma 1 (Implicit in [24]). *If there exists LOT, which exists assuming either of* CDH, Factoring, *or* LWE, *there exists a gate-by-gate garbling for all polynomial-size circuits that satisfies (M, T)-pebbling-based simulation security for $M = O(\log \mathsf{size})$ and $T = \mathrm{poly}(\mathsf{size})$ where* size *is the size of a circuit being garbled.*

4 AD-SIM CPFE with Dynamic Bounded Collusion

In this section, we construct an AD-SIM secure CPFE scheme CPFE for unbounded polynomial-size circuits with dynamic bounded collusion. CPFE supports the function class $\mathcal{C}_{\mathsf{inp},\mathsf{out}}$ for any polynomials $\mathsf{inp} = \mathsf{inp}(\lambda)$ and $\mathsf{out} = \mathsf{out}(\lambda)$ where $\mathcal{C}_{\mathsf{inp},\mathsf{out}}$ is the class of circuits with input-length inp and output-length out.

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

1. A gate-by-gate garbling scheme $\mathsf{GC} = (\mathsf{GCkt}, \mathsf{GInp}, \mathsf{GEval})$ for $\mathcal{C}_{\mathsf{inp},\mathsf{out}}$ with N_{rand} randomness slots and randomness length ℓ that satisfies (M, T)-pebbling-based simulation security for $M = \mathrm{poly}(\lambda)$ and $T = \mathrm{poly}(\lambda)$. By Lemma 1, such a scheme exists under the existence of laconic OT, which in turn exists under either of CDH, Factoring, or LWE. We denote by $\mathcal{R}$ the randomness space of GGate.
2. A PRF $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ from $\{0,1\}^{\mathsf{inp}}$ to $\{0,1\}^{\ell}$.
3. A PRF $\mathsf{PRF}' = (\mathsf{PRF}'.\mathsf{Setup}, \mathsf{PRF}'.\mathsf{Eval})$ from $\{0,1\}^{\mathsf{inp}}$ to $\mathcal{R}$.

Circuit $\mathsf{GarbleCirc}$

Input: A string $x \in \{0,1\}^{\mathsf{inp}}$ and a key-tag $r \in \{0,1\}^{\lambda}$
Hardwired value: a public parameter pp, gate G, set $S(\mathsf{G})$, tuple of PRF keys $\{\mathsf{K}_i\}_{i\in S(\mathsf{G})}$, and PRF key K'_{G}.

1. Compute $\mathsf{R}_i \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_i, r)$ for $i \in S(\mathsf{G})$ and $\mathsf{R}'_{\mathsf{G}} \leftarrow \mathsf{PRF'.Eval}(\mathsf{K}'_{\mathsf{G}}, r)$.
2. Output $\widetilde{\mathsf{G}} := \mathsf{GGate}(\mathsf{pp}, \mathsf{G}, \{\mathsf{R}_i\}_{i\in S(\mathsf{G})}; \mathsf{R}'_{\mathsf{G}})$.

Fig. 3 The description of GarbleCirc

4. An M-CT AD-SIM secure CPFE scheme with dynamic bounded collusion denoted by $\mathsf{BCPFE} = (\mathsf{BCPFE.Setup}, \mathsf{BCPFE.Enc}, \mathsf{BCPFE.KeyGen}, \mathsf{BCPFE.Dec})$ for bounded polynomial-size circuits. Here, M-CT AD-SIM security roughly means security against adversaries that see M challenge ciphertexts. Due to a technical reason, however, our notion of M-CT AD-SIM security is slightly different from the standard one e.g., [30, Appendix A]. Our definition is given in [3, Sec 4.1.2]. We construct a CPFE scheme for bounded polynomial-size circuits that satisfies this security notion from IBE in [3, Appendix B.1].
   We require BCPFE to support a circuit class $\mathcal{C}_{\mathsf{BCPFE.inp},\mathsf{BCPFE.out},\mathsf{BCPFE.size}}$ consisting of circuits with input length $\mathsf{BCPFE.inp}$, output length $\mathsf{BCPFE.out}$, and size at most $\mathsf{BCPFE.size}$, where $\mathsf{BCPFE.inp} := \mathsf{inp} + \lambda$ and $\mathsf{BCPFE.out}$ and $\mathsf{BCPFE.size}$ are output-length and the maximum size of the circuit GarbleCirc defined in Figure 3, respectively. By the efficiency requirements of GC (Definition 4), $\mathsf{BCPFE.out} = \mathrm{poly}(\lambda, \mathsf{inp})$ and $\mathsf{BCPFE.size} = \mathrm{poly}(\lambda, \mathsf{inp})$ independently of the size of the circuit C being encrypted.
5. A (single-ciphertext) AD-SIM-secure CPFE scheme with dynamic bounded collusion denoted by $\mathsf{BCPFE}' = (\mathsf{BCPFE}'.\mathsf{Setup}, \mathsf{BCPFE}'.\mathsf{Enc}, \mathsf{BCPFE}'.\mathsf{KeyGen}, \mathsf{BCPFE}'.\mathsf{Dec})$ for bounded polynomial-size circuits. We require BCPFE' to support a circuit class $\mathcal{C}_{\mathsf{BCPFE}'.\mathsf{inp},\mathsf{BCPFE}'.\mathsf{out},\mathsf{BCPFE}'.\mathsf{size}}$ consisting of circuits with input length $\mathsf{BCPFE}'.\mathsf{inp}$, output length $\mathsf{BCPFE}'.\mathsf{out}$, and size at most $\mathsf{BCPFE}'.\mathsf{size}$, where $\mathsf{BCPFE}'.\mathsf{inp} := \mathsf{inp} + \lambda$ and $\mathsf{BCPFE}'.\mathsf{out}$ and $\mathsf{BCPFE}'.\mathsf{size}$ are the output length and the maximum size of the circuit GarbleInp defined in Figure 4, respectively. By the efficiency requirements of GC (Definition 4), $\mathsf{BCPFE}'.\mathsf{out} = \mathrm{poly}(\lambda, \mathsf{inp})$ and $\mathsf{BCPFE}'.\mathsf{size} = \mathrm{poly}(\lambda, \mathsf{inp})$ independently of the size of the circuit C being encrypted.

Construction. In the construction, for a circuit C, we define the universal circuit U_C such that $U_C(x)$=$C(x)$. We define U_C in such a way that the topology of U_C does not reveal anything beyond the size of C.[12] The description of CPFE is given below.

$\mathsf{Setup}(1^\lambda, \mathsf{prm})$: On input the security parameter λ and the parameter prm, do the following:
1. Run $(\mathsf{BCPFE.mpk}, \mathsf{BCPFE.msk}) \leftarrow \mathsf{BCPFE.Setup}(1^\lambda, \mathsf{BCPFE.prm})$.
2. Run $(\mathsf{BCPFE}'.\mathsf{mpk}, \mathsf{BCPFE}'.\mathsf{msk}) \leftarrow \mathsf{BCPFE}'.\mathsf{Setup}(1^\lambda, \mathsf{BCPFE}'.\mathsf{prm})$.

[12] We explain how to construct such U_C in [3, Sec 2.1].

Circuit $\mathsf{GarbleInp}$

Input: A string $x \in \{0,1\}^{\mathsf{inp}}$ and a key-tag $r \in \{0,1\}^{\lambda}$
Hardwired value: a public parameter pp and a tuple of PRF keys $\{\mathsf{K}_i\}_{i \in S_{\mathsf{st}}}$.

1. Compute $\mathsf{R}_i \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_i, r)$ for $i \in S_{\mathsf{st}}$.
2. Set $\mathsf{st} := (\mathsf{pp}, \{\mathsf{R}_i\}_{i \in S_{\mathsf{st}}})$.
3. Output $\widetilde{x} := \mathsf{GInp}(\mathsf{st}, x)$.

Fig. 4 The description of $\mathsf{GarbleInp}$

3. Output $(\mathsf{mpk}, \mathsf{msk}) := ((\mathsf{BCPFE.mpk}, \mathsf{BCPFE}'.\mathsf{mpk}), (\mathsf{BCPFE.msk}, \mathsf{BCPFE}'.\mathsf{msk}))$.

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

1. Compute the universal circuit U_C. In the following, we write Gates to mean the set of gates of U_C rather than C.
2. Run $\mathsf{pp} \leftarrow \mathsf{GSetup}(1^\lambda, \mathsf{top}(U_C))$.
3. For $i \in [N_{\mathsf{rand}}]$, generate $\mathsf{K}_i \leftarrow \mathsf{PRF.Setup}(1^\lambda)$
4. For $\mathsf{G} \in \mathsf{Gates}$ generate $\mathsf{K}'_{\mathsf{G}} \leftarrow \mathsf{PRF}'.\mathsf{Setup}(1^\lambda)$.
5. For $\mathsf{G} \in \mathsf{Gates}$, run

$$\mathsf{BCPFE.ct}_{\mathsf{G}} \leftarrow \mathsf{BCPFE.Enc}(\mathsf{BCPFE.mpk}, \mathsf{GarbleCirc}[\mathsf{pp}, \mathsf{G}, S(\mathsf{G}), \{\mathsf{K}_i\}_{i \in S(\mathsf{G})}, \mathsf{K}'_{\mathsf{G}}], 1^Q)$$

where $\mathsf{GarbleCirc}[\mathsf{pp}, \mathsf{G}, S(\mathsf{G}), \{\mathsf{K}_i\}_{i \in S(\mathsf{G})}, \mathsf{K}'_{\mathsf{G}}]$ is the circuit as defined in Figure 3.

6. Run $\mathsf{BCPFE}'.\mathsf{ct} \leftarrow \mathsf{BCPFE}'.\mathsf{Enc}(\mathsf{BCPFE}'.\mathsf{mpk}, \mathsf{GarbleInp}[\mathsf{pp}, \{\mathsf{K}_i\}_{i \in S_{\mathsf{st}}}])$ where $\mathsf{GarbleInp}[\mathsf{pp}, \{\mathsf{K}_i\}_{i \in S_{\mathsf{st}}}]$ is the circuit as defined in Figure 4.
7. Output $\mathsf{ct} := (\{\mathsf{BCPFE.ct}_{\mathsf{G}}\}_{\mathsf{G} \in \mathsf{Gates}}, \mathsf{BCPFE}'.\mathsf{ct})$.

$\mathsf{KeyGen}(\mathsf{msk}, x)$: On input the master secret key $\mathsf{msk} = (\mathsf{BCPFE.msk}, \mathsf{BCPFE}'.\mathsf{msk})$ and an input $x \in \{0,1\}^{\mathsf{inp}}$, do the following:

1. Generate $r \leftarrow \{0,1\}^\lambda$.
2. Run $\mathsf{BCPFE.sk} \leftarrow \mathsf{BCPFE.KeyGen}(\mathsf{BCPFE.msk}, (x, r))$.
3. Run $\mathsf{BCPFE}'.\mathsf{sk} \leftarrow \mathsf{BCPFE}'.\mathsf{KeyGen}(\mathsf{BCPFE}'.\mathsf{msk}, (x, r))$.
4. Output $\mathsf{sk} := (r, \mathsf{BCPFE.sk}, \mathsf{BCPFE}'.\mathsf{sk})$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}, 1^Q)$: On input a ciphertext $\mathsf{ct} = (\{\mathsf{BCPFE.ct}_{\mathsf{G}}\}_{\mathsf{G} \in \mathsf{Gates}}, \mathsf{BCPFE}'.\mathsf{ct})$ and a secret key $\mathsf{sk} = (r, \mathsf{BCPFE.sk}, \mathsf{BCPFE}'.\mathsf{sk})$, do the following:

1. For $\mathsf{G} \in \mathsf{Gates}$, run $\widetilde{\mathsf{G}} \leftarrow \mathsf{BCPFE.Dec}(\mathsf{BCPFE.ct}_{\mathsf{G}}, \mathsf{BCPFE.sk}, 1^Q)$
2. Run $\widetilde{x} \leftarrow \mathsf{BCPFE}'.\mathsf{Dec}(\mathsf{BCPFE}'.\mathsf{ct}, \mathsf{BCPFE}'.\mathsf{sk}, 1^Q)$.
3. Set $\widetilde{U_C} := (\mathsf{pp}, \{\widetilde{\mathsf{G}}\}_{\mathsf{G} \in \mathsf{Gates}})$.
4. Compute and output $\mathsf{GEval}(\widetilde{U_C}, \widetilde{x})$.

Correctness Let $\mathsf{ct} = (\{\mathsf{BCPFE.ct}_{\mathsf{G}}\}_{\mathsf{G} \in \mathsf{Gates}}, \mathsf{BCPFE}'.\mathsf{ct})$ be an honestly generated ciphertext for a circuit C and $\mathsf{sk} = (r, \mathsf{BCPFE.sk}, \mathsf{BCPFE}'.\mathsf{sk})$ be an honestly generated secret key for an input x. By the correctness of BCPFE, for each $\mathsf{G} \in \mathsf{Gates}$, if we generate $\widetilde{\mathsf{G}} \leftarrow \mathsf{BCPFE.Dec}(\mathsf{BCPFE.ct}_{\mathsf{G}}, \mathsf{BCPFE.sk}, 1^Q)$, then we have

$\widetilde{\mathsf{G}} = \mathsf{GGate}(\mathsf{pp}, \mathsf{G}, \{\mathsf{R}_i\}_{i\in S(\mathsf{G})}; \mathsf{R}'_{\mathsf{G}})$ where $\mathsf{R}_i \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_i, r)$ for $i \in S(\mathsf{G})$ and $\mathsf{R}'_{\mathsf{G}} \leftarrow \mathsf{PRF}'.\mathsf{Eval}(\mathsf{K}'_{\mathsf{G}}, r)$. Similarly, by the correctness of BCPFE', if we generate $\widetilde{x} \leftarrow \mathsf{BCPFE}'.\mathsf{Dec}(\mathsf{BCPFE}'.\mathsf{ct}, \mathsf{BCPFE}'.\mathsf{sk}, 1^Q)$, then we have $\widetilde{x} = \mathsf{GInp}(\mathsf{st}, x)$ where $\mathsf{R}_i \leftarrow \mathsf{PRF.Eval}(\mathsf{K}_i, r)$ for $i \in S_{\mathsf{st}}$ and $\mathsf{st} := (\mathsf{pp}, \{\mathsf{R}_i\}_{i\in S_{\mathsf{st}}})$. Then, by the (perfect) correctnes of GC, $\mathsf{GEval}(\widetilde{U_C}, \widetilde{x}) = U_C(x) = C(x)$ where $\widetilde{U_C} := (\mathsf{pp}, \{\widetilde{\mathsf{G}}\}_{\mathsf{G}\in\mathsf{Gates}})$.

Security The following theorem asserts the security of CPFE. The proof appears in [3, Sec 4.2].

Theorem 1. *If* GC *satisfies* (M, T)*-pebbling-based simulation security for* $M = \mathrm{poly}(\lambda)$ *and* $T = \mathrm{poly}(\lambda)$*,* BCPFE *is* M*-CT* $\mathsf{AD\text{-}SIM}$*-secure against dynamic bounded collusion,* BCPFE' *is* $\mathsf{AD\text{-}SIM}$*-secure against dynamic bounded collusion, and* PRF *and* PRF' *are secure pseudorandom functions, then* CPFE *is* $\mathsf{AD\text{-}SIM}$*-secure against dynamic bounded collusion.*

FE for Turing machines. Agrawal et al. [5] (implicitly) showed that one can construct FE for TM with $\mathsf{AD\text{-}SIM}$ security against dynamic bounded collusion based on CPFE for unbounded polynomial-size circuits with $\mathsf{AD\text{-}SIM}$ security against dynamic bounded collusion additionally assuming sub-exponential LWE. Since (even polynomial) LWE implies LOT and IBE, by combining their result and Theorem 1, we obtain the following theorem:

Theorem 2. *Assuming sub-exponential* LWE*, we have FE for TM with* $\mathsf{AD\text{-}SIM}$ *security against dynamic bounded collusion.*

This improves one of the main results of [5] that constructed a similar scheme with $\mathsf{NA\text{-}SIM}$ security based on the same assumption. Since this is further improved in regard to assumptions in Section 5, we omit the details.

5 TMFE without Succinct FE

In this section, we propose an alternative route to construct FE for Turing machine that does not use succinct FE. We refer to Section 1 for the overview.

5.1 FE for Laconic OT Functionality

Here, we define FE for LOT functionality by specifying the relation $R_{\mathsf{LOTFE}} : \mathcal{X}_{\mathsf{LOTFE}} \times \mathcal{Y}_{\mathsf{LOTFE}} \to \{0,1\}^*$.

FE for Laconic OT Functionality. To define FE for LOT functionality, we set $\mathsf{prm} = \bot$, $\mathcal{X}_{\mathsf{LOTFE}} = \mathbb{N} \times \mathbb{N} \times \{0,1\}^* \times \{0,1\}^*$, and $\mathcal{Y}_{\mathsf{LOTFE}} = \{0,1\}^*$. An element in $\mathcal{X}_{\mathsf{LOTFE}}$ is represented by (N, i, μ_0, μ_1) with $N \in \mathbb{N}$, $i \in [N]$, and $\mu_0, \mu_1 \in \{0,1\}^*$ with $|\mu_0| = |\mu_1|$. We assume that both i and N are represented in binary form. We then define

$$R_{\mathsf{LOTFE}}((N, i, \mu_0, \mu_1), D) = \begin{cases} (N, \mu_{D[i]}) & \text{if } |D| = N \\ (N, 1^{|\mu_0|}) & \text{otherwise} \end{cases},$$

where $|D|$ is the length of D as a binary string and $D[i]$ is the i-th bit of D.

Remark 4 (Succinctness). We note that the encryption algorithm should run in fixed polynomial time in λ that is independent from N, since N is input to the encryption algorithm in binary form. This in particular implies that the running time of the encryption algorithm is independent from the size of the database D supported by the scheme. This property can be seen as an analogue of the efficiency requirements for the succinctness of FE [29] or laconic OT [18].

Remark 5. We also note that the above FE has similar functinality to that of LOT [18]. However, the important difference is that we intend to hide the index i while they do not. This security requirement is captured by the definition of R_{LOTFE} above, where i is not part of the output.

Ingredients. We now describe the underlying building blocks used for our construction of FE for LOT functionality. We need the following ingredients.

1. PIR scheme $\mathsf{PIR} = (\mathsf{PIR.Query}, \mathsf{PIR.Answer}, \mathsf{PIR.Reconstruct})$ that satisfies the efficiency requirements in [3, Definition 2.9]. In particular, we require that PIR.Query and PIR.Reconstruct run in fixed polynomial time for any $N \leq 2^{\log^2 \lambda}$ (even for super-polynomial N). This implies that the lengths of PIR.query, PIR.answer, and PIR.st are bounded by a fixed polynomial in the security parameter that is independent of N. We use the uniform upper bound $\ell_{\mathsf{PIR}} = \mathrm{poly}(\lambda)$ for them and assume that they are represented by binary strings of length ℓ_{PIR}. Additionally, we require that the function PIR.Answer has shallow circuit implementations. As we show in [3, Appendix C], we have the following instantiations:
   (a) PIR constructions from (the polynomial hardness of) LWE [17, 27] has implementation of the answer function in NC.
   (b) For PIR constructions from DDH/QR [22], we have implementations of the answer function in NC^1. For DDH based construction, we have to use the multiplicative sub-group of $\mathbb{Z}_q$ for prime q.
2. 1-Sel-IND secure ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for circuits that can evaluate the answer function of PIR in the above. We consider two types of instantiations:
   (a) We can use general ABE for circuit [31, 14]. If the answer function of PIR is implemented in NC, we can base the security of the scheme on polynomial hardness of LWE with quasi-polynomial approximation factors (i.e., $O(\lambda^{\mathrm{poly}(\log \lambda)})$).
   (b) We can use ABE for NC^1 circuits. In more details, we need the scheme to support circuit with fixed input length and any depth d, where we allow the key generation algorithm and the decryption algorithms to run in time $\mathrm{poly}(\lambda, 2^d)$. This effectively limits the class of the circuits to be NC^1. We can instantiate such an ABE from (the polynomial hardness of) LWE with super-polynomial approximation factor [32] (i.e., $O(\lambda^{\omega(1)})$) or various assumptions on pairing groups including DBDH or CBDH (the computational bilinear Diffie-Hellman assumption) [35].
3. Selectively secure garbled circuit $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Sim})$. We can instantiate it from any one-way function [39].

Circuit $C[N, \mathsf{PIR.st}, \mu_0, \mu_1]$

Hardwired constants: The integer N, The PIR secret state $\mathsf{PIR.st}$, the messages μ_0, μ_1.
Input: String $X \in \{0,1\}^{\ell_{\mathsf{PIR}}}$

1. Parse $X \to \mathsf{PIR.answer}$.
2. Run $\mathsf{PIR.Reconstruct}(\mathsf{PIR.st}, \mathsf{PIR.answer}, N) \to b$.
3. Output μ_b.

Fig. 5 Circuit $C[N, \mathsf{PIR.st}, \mu_0, \mu_1]$

4. IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ with AD-IND security. Since IBE with AD-IND security is implied by IBE with Sel-IND [19] and the latter is trivially implied by the ABE for NC^1 circuit, this does not add a new assumption.

We assume that all the ingredients above have perfect correctness for simplicity. We consider correctness error only for PIR, because DDH based instantiation does not have perfect correctness.

Construction. Here, we describe our scheme $\mathsf{LOTFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The construction is similar to that of succinct FE by [29] where FHE is replaced by PIR. In the construction, we assume that $N, |D| \leq 2^{\log^2 \lambda}$. This is sufficient for dealing with unbounded size D, because $2^{\log^2 \lambda} = \lambda^{\log \lambda}$ is super-polynomial.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, do the following:
1. Run $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Run $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, \mathsf{prm})$, where $\mathsf{prm} := 1^{\ell_{\mathsf{PIR}}+2\lambda}$. This means that the ABE supports circuit with input length $\ell_{\mathsf{PIR}} + 2\lambda$.
3. Output $\mathsf{mpk} := (\mathsf{ABE.mpk}, \mathsf{IBE.mpk})$ and $\mathsf{msk} := (\mathsf{ABE.msk}, \mathsf{IBE.msk})$.

$\mathsf{Enc}(\mathsf{mpk}, (N, i, \mu_0, \mu_1))$: On input the master public key $\mathsf{mpk} = (\mathsf{ABE.mpk}, \mathsf{IBE.mpk})$ and the message (N, i, μ_0, μ_1), do the following:
1. Run $(\mathsf{PIR.query}, \mathsf{PIR.st}) \leftarrow \mathsf{PIR.Query}(1^\lambda, i, N)$.
2. Pick $\mathsf{lab}_{k,b} \leftarrow \{0,1\}^\lambda$ for $k \in [\ell_{\mathsf{PIR}}]$, $b \in \{0,1\}$.
3. For $k \in [\ell_{\mathsf{PIR}}]$, $b \in \{0,1\}$, compute

$$\mathsf{ABE.ct}_{k,b} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, (\mathsf{PIR.query}, k, b), \mathsf{lab}_{k,b}).$$

4. Construct circuit $C[N, \mathsf{PIR.st}, \mu_0, \mu_1]$ as Figure 5.
5. Run $\widetilde{C} \leftarrow \mathsf{GC.Garble}\left(1^\lambda, C[N, \mathsf{PIR.st}, \mu_0, \mu_1]\right)$.
6. Set $\mathsf{msg} := \left(\widetilde{C}, \mathsf{PIR.query}, \{\mathsf{ABE.ct}_{k,b}\}_{k \in [\ell_{\mathsf{PIR}}], b \in \{0,1\}}\right)$.
7. Run $\mathsf{IBE.ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, N, \mathsf{msg})$.
8. Output $\mathsf{ct} := (N, 1^{|\mu_0|}, \mathsf{IBE.ct})$.

$\mathsf{KeyGen}(\mathsf{msk}, D)$: On input the master secret key $\mathsf{msk} = (\mathsf{ABE.msk}, \mathsf{IBE.msk})$, an input $D \in \{0,1\}^*$ with $|D| \leq 2^{\log^2 \lambda}$, do the following:
1. Construct the circuit $F[D]$ as Figure 6.

Circuit $F[D]$

Hardwired constants: The data $D \in \{0,1\}^{|D|}$.
Input: String $X \in \{0,1\}^{\ell_{\mathsf{PIR}}+\lambda+1}$

1. Parse the input as $X \to (\mathsf{PIR.query}, k, b)$ where $\mathsf{PIR.query} \in \{0,1\}^{\ell_{\mathsf{PIR}}}$, $k \in \{0,1\}^{\lambda}$, and $b \in \{0,1\}$.
2. If $k \notin [\ell_{\mathsf{PIR}}]$, output 0, where k is interpreted as an integer.
3. Run $\mathsf{PIR.Answer}(\mathsf{PIR.query}, D, 1^{|D|}) \to \mathsf{PIR.answer}$.
4. Compute $b' = \mathsf{PIR.answer}_k \oplus b \oplus 1$, where $\mathsf{PIR.answer}_k$ is the k-th bit of $\mathsf{PIR.answer}$.
5. Output b'.

Fig. 6 Circuit $F[D]$

2. Run
$$\mathsf{ABE.sk} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}, F[D]).$$
3. Run $\mathsf{IBE.sk} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, |D|)$.
4. Output $\mathsf{sk} := (D, \mathsf{ABE.sk}, \mathsf{IBE.sk})$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$: On input a ciphertext $\mathsf{ct} = (N, 1^{|\mu_0|}, \mathsf{IBE.ct})$, a secret key $\mathsf{sk} = (D, \mathsf{ABE.sk}, \mathsf{IBE.sk})$, do the following:

1. If $|D| \neq N$, output $(N, 1^{|\mu_0|})$.
2. Run $\mathsf{msg} \leftarrow \mathsf{IBE.Dec}(\mathsf{IBE.sk}, \mathsf{IBE.ct})$.
3. Parse $\mathsf{msg} \to (\widetilde{C}, \mathsf{PIR.query}, \{\mathsf{ABE.ct}_{k,b}\}_{k\in[n],b\in\{0,1\}})$.
4. Run $\mathsf{PIR.answer} \leftarrow \mathsf{PIR.Answer}(\mathsf{PIR.query}, D, 1^N)$.
5. Run $\mathsf{lab}_k \leftarrow \mathsf{ABE.Dec}(\mathsf{ABE.sk}, \mathsf{ABE.ct}_{k,\mathsf{PIR.answer}_k})$ for $k \in [\ell_{\mathsf{PIR}}]$.
6. Compute $\mu := \mathsf{GC.Eval}(\widetilde{C}, \{\mathsf{lab}_k\}_k)$.
7. Output (N, μ).

Efficiency. We discuss the efficiency of the scheme. It is not hard to see that Setup and Enc run in polynomial time in its input length. We note that Enc runs in polynomial time in λ and $|\mu|$ even for super-polynomial N as large as $2^{\log^2 \lambda}$ by the efficiency property of PIR.Query and PIR.Reconstruct. For evaluating the efficiency of KeyGen, we consider two settings based on how we instantiate ABE and PIR. The first case is the combination of ABE for circuits and any PIR, whereas the second case is ABE for NC^1 circuits and PIR with answer function in NC^1. We focus on the latter case since the former case is much simpler. Evaluating the efficiency of KeyGen in this case is a bit subtle, because ABE.KeyGen used inside the algorithm runs in exponential time in the depth of the input circuit. In order to bound the running time of the algorithm, we evaluate the depth of $F[D]$ by going over all the computation steps inside the circuit. We observe that only the second and the third steps out of the five steps are non-trivial. The second step can be implemented by a circuit of depth $O(\log \ell_{\mathsf{PIR}}) = O(\log \lambda)$ by checking $k \stackrel{?}{=} i$ for all $i \in [\ell_{\mathsf{PIR}}]$ in parallel and taking OR of all the outcomes. The third step can be implemented by a circuit of depth $O(\log |D|)$ by our assumption on PIR. Overall, the

depth of the circuit $F[D]$ can be upper bounded by $O(\log \lambda + \log |D|)$ and thus the key generation algorithm runs in time

$$2^{O(\log \lambda + \log |D|)} = \mathrm{poly}(\lambda, |D|)$$

as desired. We finally observe that the decryption algorithm runs in time polynomial in the length of ct and sk, which are bounded by $\mathrm{poly}(\lambda, |D|)$ by the efficiency of the encryption and key generation algorithms. Therefore, the decryption algorithm runs in time $\mathrm{poly}(\lambda, |D|)$ as well.

Correctness. We focus on the case of $|D| = N$, since otherwise it is trivial. By the correctness of IBE, msg is correctly recovered in the first step of the decryption. Furthermore, since $F[D](\mathsf{PIR.query}, k, \mathsf{PIR.answer}_k) = \mathsf{PIR.answer}_k \oplus \mathsf{PIR.answer}_k \oplus 1 = 1$, we observe that lab_k recovered in the 5-th step of the decryption equals to $\mathsf{lab}_{k,\mathsf{PIR.answer}_k}$ by the correctness of ABE. Finally, by the correctness of GC and PIR, μ recovered in the 6-th step of the decryption equals to $C[N, \mathsf{PIR.st}, \mu_0, \mu_1](\mathsf{PIR.answer}) = \mu_{D[i]}$ with overwhelming probability as desired.

Security. The following theorem addresses the security of LOTFE, whose proof is similar to that of succinct FE by [29]. However, we need somewhat more careful analysis in order to base the security of the scheme on $\mathsf{Sel\text{-}IND}$ security of the underlying ABE rather than on $\mathsf{AD\text{-}IND}$ security. The reason why $\mathsf{Sel\text{-}IND}$ security suffices for our case is that the reduction algorithm can guess the target attribute the adversary chooses only with polynomial security loss. In more detail, we change ABE ciphertexts encrypting $\mathsf{lab}_{k,1-\mathsf{PIR.answer}_k}$ for attribute $(\mathsf{PIR.query}, k, 1 - \mathsf{PIR.answer}_k)$ to be that encrypting $\mathsf{lab}_{k,\mathsf{PIR.answer}_k}$ in the security proof. A naive way of guessing the attribute $(\mathsf{PIR.query}, k, 1 - \mathsf{PIR.answer}_k)$ ends up with exponential security loss, since there are exponentially many possible $\mathsf{PIR.query}$. However, the reduction algorithm only has to guess (N, i) that is encoded inside $\mathsf{PIR.query}$, because the randomness used for computing the query is not controlled by the adversary, but by the reduction algorithm. Since there are only polynomial number of possible $(N, i, k, 1 - \mathsf{PIR.answer}_k)$, the guessing can be done only with polynomial security loss.

Theorem 3. *If* IBE *is* $\mathsf{AD\text{-}IND}$ *secure,* ABE *is* $\mathsf{Sel\text{-}IND}$ *secure,* GC *is selectively secure, and* PIR *is private, then the above FE is* $1\text{-}\mathsf{NA\text{-}SIM}$ *secure.*

The proof of Theorem 3 appears in [3, Sec 5.1].

5.2 CPFE for read only RAM

Here, we define CPFE for read only RAM by specifying the relation $R_{\mathsf{CPRAMFE}} : \mathcal{X}_{\mathsf{CPRAMFE}} \times \mathcal{Y}_{\mathsf{CPRAMFE}} \to \{0,1\}^*$.

CPFE for read only RAM computation. To define CPFE for read only RAM, we set $\mathcal{Y}_{\mathsf{CPRAMFE}} = \{0,1\}^*$ and $\mathcal{X}_{\mathsf{CPRAMFE}}$ to be a set of read only RAM programs of the form $P = \{P^\tau\}_{\tau \in [t]}$. We define $R_{\mathsf{CPRAMFE}}(P, D) \in \{0,1\}^*$ to be the output obtained by executing P with the RAM access to the data D. In our case, we consider RAM programs with some specific structure. To define this, we introduce the parameter $\mathsf{prm} = 1^{\ell_{\mathsf{st}}}$ and

the set of RAM programs $\mathcal{P}_{T,\mathsf{size}}$. We then constrain the domain as $\mathcal{X}_{\mathsf{prm}} = \mathcal{P}_{T,\mathsf{size}}$ and $\mathcal{Y}_{\mathsf{prm}} = \{0,1\}^*$. A RAM program in $\mathcal{P}_{T,\mathsf{size}}$ is of the form $P = \{P^\tau\}_\tau$ where the step circuit P^τ is of the form $P^\tau : \{0,1\}^{\ell_{\mathsf{st}}} \to \{0,1\}^{\ell_{\mathsf{st}}-1} \times \{0,1\}^{\log^2 \lambda}$. For $D \in \{0,1\}^N$ and $P = \{P^\tau\}_{\tau \in [t]}$, we define $(\mathsf{st}^\tau, L^\tau) \in \{0,1\}^{\ell_{\mathsf{st}}-1} \times \{0,1\}^{\log^2 \lambda}$ for $\tau \in [2,t]$ by induction as

$$(\mathsf{st}^\tau, L^\tau) := P^{\tau-1}(\mathsf{st}^{\tau-1}, D[L^{\tau-1}]) \quad \text{where} \quad (\mathsf{st}^1, D[L^1]) := (0^{\ell_{\mathsf{st}}-1}, 0). \tag{5.1}$$

Here, st^τ and $L^\tau \in \{0,1\}^{\log^2 \lambda}$ output by $P^{\tau-1}$ represent the state information and the position L^τ to be read from the data D respectively. The state st^τ and the read bit $D[L^\tau]$, which is the L^τ-th bit of D, is then input to the next step circuit P^τ. In the above computation, the initial state st^1 and the initial read bit $D[L^1]$ are defined to be zero strings. Note that the position to be read is assumed to be represented by a binary string of $\{0,1\}^{\log^2 \lambda}$, which is interpreted as an integer in $[0, 2^{\log^2 \lambda}]$. Since $2^{\log^2 \lambda} = \lambda^{\log \lambda}$ is super-polynomial, this is sufficient for pointing a position in any unbounded size data. The output obtained by running P with the RAM access to the data D is denoted by P^D and is defined to be $P^D := \mathsf{st}^{t+1}$.

Remark 6 (Succinctness). Similarly to the case of LOTFE, the running time of the encryption algorithm is independent from the size of the database D supported by the scheme. This property can be seen as an analogue of the efficiency requirements for the succinctness of FE [29] or laconic OT [18].

Remark 7 (Efficiency). We note that we do not require the decryption time to be independent of the size of the database D, while the encryption time is required to be so. This is in contrast to ABE/FE for RAM efficiency in the literature [26, 8], where the decryption time is also required to be sublinear in the size of the database. However, our weaker definition suffices for our purpose of constructing FE for TM.

Ingredients. We now describe the underlying building blocks used for our construction of CPFE for read only RAM.

1. FE with laconic OT functionality $\mathsf{LOTFE} = (\mathsf{LOTFE.Setup}, \mathsf{LOTFE.KeyGen}, \mathsf{LOTFE.Enc}, \mathsf{LOTFE.Dec})$ with 1-NA-SIM security. This can be instantiated by the scheme in Section 5.1. For simplicity, we assume that the encryption algorithm of LOTFE only requires randomness of λ bits. This can be achieved by using the randomness as a PRF key to derive longer pseudorandom string for example.
2. IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ with AD-IND security. Since the construction of LOTFE in Section 5.1 already uses IBE, this does not add new assumption.
3. Selectively secure garbled circuit $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Sim})$. We can instantiate it from any one-way function [39].

Construction. Here, we describe our scheme $\mathsf{CPRAMFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The construction is inspired by the garbled RAM construction by [25], which in turn is based on [37], where a sequence of garbled circuits read the memory stored outside of the circuits via RAM access. Whereas they use the combination of IBE and ORAM to

Circuit $\mathsf{SC}[\tau, P^\tau, \mathsf{LOTFE.mpk}, \mathsf{R}^\tau, \{\mathsf{lab}_{k,b}^{\tau+1}\}_{k,b}]$

Hardwired constants: The step number τ, the step circuit P^τ, the master public key $\mathsf{LOTFE.mpk}$, randomness R^τ, and the set of labels $\{\mathsf{lab}_{k,b}^{\tau+1}\}_{k\in[n],b\in\{0,1\}}$.
Input: String $X \in \{0,1\}^n$.

1. Parse the input as $X \to (N, \mathsf{st}, \mathsf{rData})$, where $N \in \{0,1\}^{\log^2 \lambda}$, $\mathsf{st} \in \{0,1\}^{\ell_{\mathsf{st}}-1}$, and $\mathsf{rData} \in \{0,1\}$.
2. Run $P^\tau(\mathsf{st}, \mathsf{rData}) = (\mathsf{st}', L)$.
3. Run $\mathsf{LOTFE.Enc}\left(\mathsf{LOTFE.mpk}, (N, L, \mathsf{lab}_{n,0}^{\tau+1}, \mathsf{lab}_{n,1}^{\tau+1}); \mathsf{R}^\tau\right) \to \mathsf{LOTFE.ct}$.
4. Set $Y{:=}(N, \mathsf{st}')$.
5. Output $\begin{cases} \left(\left\{\mathsf{lab}_{k,Y_k}^{\tau+1}\right\}_{k\in[n-1]}, \mathsf{LOTFE.ct}\right) & \text{if } \tau \neq t \\ \mathsf{st}' & \text{if } \tau = t \end{cases}$

Fig. 7 Circuit $\mathsf{SC}^\tau = \mathsf{SC}[\tau, P^\tau, \mathsf{LOTFE.mpk}, \mathsf{R}^\tau, \{\mathsf{lab}_{k,b}^{\tau+1}\}_{k,b}]$

enable the oblivious access to the memory, we use LOTFE for this purpose instead. This change is crucial for us because ORAM is a secret key primitive and is not compatible with our setting of public key FE.

$\mathsf{Setup}(1^\lambda, \mathsf{prm})$: On input the security parameter λ, the parameter $\mathsf{prm} = 1^{\ell_{\mathsf{st}}}$, do the following:
1. Run $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Run $(\mathsf{LOTFE.mpk}, \mathsf{LOTFE.msk}) \leftarrow \mathsf{LOTFE.Setup}(1^\lambda)$.
3. Output $\mathsf{mpk}{:=}(\mathsf{LOTFE.mpk}, \mathsf{IBE.mpk})$ and $\mathsf{msk}{:=}(\mathsf{LOTFE.msk}, \mathsf{IBE.msk})$.

$\mathsf{Enc}(\mathsf{mpk}, P)$: On input the master public key $\mathsf{mpk} = (\mathsf{LOTFE.mpk}, \mathsf{IBE.mpk})$, a read only RAM program $P = \{P_\tau\}_{\tau\in[t]} \in \mathcal{P}_{\ell_{\mathsf{st}}}$, do the following:
1. Set $n{:=}\log^2 \lambda + \ell_{\mathsf{st}}$.
2. Pick $\mathsf{lab}_{k,b}^\tau \leftarrow \{0,1\}^\lambda$ for $\tau \in [t]$, $k \in [n]$, and $b \in \{0,1\}$.
3. Pick $\mathsf{R}^\tau \leftarrow \{0,1\}^\lambda$ for $\tau \in [t]$.
4. Construct circuit $\mathsf{SC}^\tau := \mathsf{SC}[\tau, P^\tau, \mathsf{LOTFE.mpk}, \mathsf{R}^\tau, \{\mathsf{lab}_{k,b}^{\tau+1}\}_{k,b}]$ for $\tau \in [t]$ as Figure 7, where we define $\mathsf{lab}_{k,b}^{t+1} = \bot$ for $k \in [n]$, $b \in \{0,1\}$.
5. For all $\tau \in [t]$, run
$$\widetilde{\mathsf{SC}}^\tau \leftarrow \mathsf{GC.Garble}\left(1^\lambda, \mathsf{SC}^\tau, \{\mathsf{lab}_{k,b}^\tau\}_{k,b}\right).$$
6. For all $k \in [n]$, $b \in \{0,1\}$, run
$$\mathsf{IBE.ct}_{k,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (k,b), \mathsf{lab}_{k,b}^1).$$
7. Output $\mathsf{ct}{:=}\left(\{\widetilde{\mathsf{SC}}^\tau\}_{\tau\in[t]}, \{\mathsf{IBE.ct}_{k,b}\}_{k\in[n],b\in\{0,1\}}\right)$.

$\mathsf{KeyGen}(\mathsf{msk}, D)$: On input the master secret key $\mathsf{msk} = \mathsf{LOTFE.msk}$, an input $D \in \{0,1\}^N$, where $N \leq 2^{\log^2 \lambda}$, do the following:
1. Run
$$\mathsf{LOTFE.sk} \leftarrow \mathsf{LOTFE.KeyGen}(\mathsf{LOTFE.msk}, D).$$

2. Set $X = N \| 0^{n-\log^2 \lambda}$, where N is represented as a string in $\{0,1\}^{\log^2 \lambda}$.
3. For all $k \in [n]$, run

$$\mathsf{IBE.sk}_{k,X_k} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (k, X_k)).$$

4. Output $\mathsf{sk} := \big(D, \mathsf{LOTFE.sk}, \{\mathsf{IBE.sk}_{k,X_k}\}_{k\in[n]}\big)$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$: On input a ciphertext $\mathsf{ct} = (\{\widetilde{\mathsf{SC}}^\tau\}_\tau, \{\mathsf{IBE.ct}_{k,b}\}_{k,b})$ and a secret key $\mathsf{sk} = (D, \mathsf{LOTFE.sk}, \{\mathsf{IBE.sk}_{k,X_k}\}_k)$, do the following:

1. Run Set $X := N \| 0^{n-\log^2 \lambda}$.
2. Run $\mathsf{lab}_k^1 := \mathsf{IBE.Dec}(\mathsf{IBE.sk}_{k,X_k}, \mathsf{IBE.ct}_{k,X_k})$ for $k \in [n]$.
3. Set $\overline{\mathsf{label}} := \{\mathsf{lab}_k^1\}_{k\in[n]}$.
4. For $\tau = 1, \ldots, t$
   (a) Compute $\mathsf{gout} := \mathsf{GC.Eval}(\widetilde{\mathsf{SC}}^\tau, \overline{\mathsf{label}})$.
   (b) If $\tau = t$, set $y := \mathsf{gout}$ and break out of the loop.
   (c) Parse $\mathsf{gout} \to \big(\{\mathsf{lab}_k\}_{k\in[n-1]}, \mathsf{LOTFE.ct}\big)$.
   (d) Compute $(N, \mathsf{lab}_n) := \mathsf{LOTFE.Dec}(\mathsf{LOTFE.sk}, \mathsf{LOTFE.ct})$.
   (e) Set $\overline{\mathsf{label}} := \{\mathsf{lab}_k\}_{k\in[n]}$.
5. Output y.

Correctness and Security. The correctness of CPRAMFE is shown in [3, Sec 5.2]. We prove that CPRAMFE is 1-NA-SIM secure in [3, Sec 5.2].

Efficiency. By the efficiency of LOTFE and IBE, it is easy to see that Setup and KeyGen run in time $\mathrm{poly}(\lambda)$ and $\mathrm{poly}(\lambda, |D|)$, respectively. We can also see that $|\widetilde{\mathsf{SC}}^\tau| = \mathrm{poly}(\lambda, |P^\tau|)$ and thus the running time of Enc can be bounded by $\mathrm{poly}(\lambda, |P|)$. Finally, Dec runs in polynomial time in its input length by the efficiency of the underlying primitives and thus run in time $\mathrm{poly}(\lambda, |P|, |D|)$.

5.3 FE for Turing Machines with Fixed Input Length

Here, we show that CPRAMFE we constructed in Section 5.2 can easily be converted into FE for TM. The resulting construction can handle TM of unbounded size, but it is only 1-NA-SIM secure and can only handle the case where the length of $(x, 1^t)$ is bounded. These limitations will be removed in the next subsection.

RAM Programs reading multi-bit at once. To simplify the description, we assume that each step of RAM computation reads a block consisting of $B(\lambda) = \mathrm{poly}(\lambda)$ bits at once instead of reading a single bit. Correspondingly, we assume that the database contains B bits of data at a single location. This is without loss of generality because a RAM program that reads single bit at once can be converted into that reads B bits at once by making the length of step circuits B times longer and increasing the size of each step circuit so that it can keep B bits inside it.

Representing Turing machine computation as RAM computation. In order to represent the computation executed by a Turing machine as a computation by RAM program, we introduce the following mappings:

$P^{\tau}_{(x,1^t)}$

Hardwired constants: The step number τ, the input x to the TM, and the running time t.
Input: $(\mathsf{st}, \mathsf{rData}) \in \{0,1\}^{t+2\lambda} \times \{0,1\}^{B}$.

1. Parse the input as $\mathsf{st} \to (i, W, q)$, where $i, q \in \{0,1\}^{\lambda}$ and $W \in \{0,1\}^{t}$ and $\mathsf{rData} \to ((q'_0, b'_0, \Delta i_0), (q'_1, b'_1, \Delta i_1), \mathsf{pre}_0, \mathsf{pre}_1)$.
2. If $\tau = 1$, replace W with $W = x \| 0^{t-n}$.
3. Set $i' := i + \Delta i_{W[i]}$, $q' = q'_{W[i]}$, $\mathsf{pre}' = \mathsf{pre}_{W[i]}$, and $W'[j] = \begin{cases} W[j] & \text{if } j \neq i \\ b'_{W[i]} & \text{if } j = i \end{cases}$ for $j \in [t]$.
4. Set $(\mathsf{st}', L) = ((i', W', q'), q')$.
5. Output $\begin{cases} (\mathsf{st}', L) & \text{if } \tau \neq t \\ \mathsf{pre}' & \text{if } \tau = t \end{cases}$

Fig. 8 Circuit $P^{\tau}_{(x,1^t)}$

f : This mapping takes as input $(x \in \{0,1\}^n, 1^t)$ and then convert it into a read only RAM program $P_{(x,1^t)} = \{P^{\tau}_{(x,1^t)}\}_{\tau \in [t]}$ defined as in Figure 8. Here, we set $B(\lambda) = 3\lambda$. In the circuit, q, q'_0, and q'_1 are represented by strings in $\{0,1\}^{\lambda}$. In particular, this means that the circuit can handle any size of Turing machines because we can assume $q \leq Q < 2^{\lambda}$ without loss of generality.

g : This mapping takes as input description of a Turing machine $M = (Q, \delta, F)$ and outputs a database D_M that contains Q blocks each consisting of B bits. At its q-th block, D_M contains

$$D_M[q] := (\delta(q,0), \delta(q,1), \mathsf{pre}_0, \mathsf{pre}_1),$$

where $\mathsf{pre}_c \in \{0,1\}$ for $c \in \{0,1\}$ indicates whether q'_c defined by $\delta(q,c) = (q'_c, b'_c, \Delta i_c)$ is in the set of accepting states F or not. Since $\delta(q,b) \in [Q] \times \{0,1\} \times \{0, \pm 1\}$ and $Q < 2^{\lambda}$, each block can be represented by a binary string of length at most $B(\lambda) = 3\lambda$.

We observe that the output of $P^{D_M}_{(x,1^t)}$ is the same as that obtained by running the Turing machine M on input x for t steps. This is because each step circuit $P^{\tau}_{(x,1^t)}$ of $P_{(x,1^t)}$ is designed to emulate τ-th step of the computation done by the machine. This means that by applying the above mappings, we can convert CPRAMFE into an FE scheme for Turing machine with fixed input length. It is easy to see that the security and correctness of the scheme are preserved. In particular, the resulting scheme inherits the 1-NA-SIM security. We also observe that the size of the program $P = \{P^{\tau}_{(x,1^t)}\}_{\tau}$ is bounded by a fixed polynomial in $|(x, 1^t)|$.

5.4 Getting the Full-Fledged Construction

Here, we remove the restrictions from the consturction in Section 5.3 and obtain full-fledged FE scheme for TM.

Removing the non-adaptive and single key restriction. In the first step, we apply the conversion in [3, Sec 6.4], which is essentially the same as the conversion given by Agrawal et. al [5, Section 4], to the scheme to upgrade the security. The resulting scheme is AD-SIM secure against bounded dynamic collusion. We refer to [3, Sec 6.4] for the details.

Removing the fixed input length restriction. Our goal in the second step is constructing an FE scheme for $R^{\leq} : \mathcal{A} \times \mathcal{B} \to \mathcal{M} \cup \{\bot\}$, where $\mathcal{A} = \{0,1\}^*$, $\mathcal{B}$ is the set of all Turing machines, and

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

This step is done in essentially the same manner as [5, Section 6.2.2] using [3, Theorem 2.1]. We observe that the FE scheme that we obtained above can be seen as an FE scheme for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i = \{0,1\}^i$ and $\mathcal{Y}_i$ is the set of all Turing machines, and

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

That is, $|(x,1^t)|$ is a-priori bounded by i. We set $\mathcal{S}$, $\mathcal{T}$, f, and g as

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

Here, we crucially rely on $|(x,1^t)| \leq |M|$.

Recall that

$$R^{\mathsf{bndl}}(x,y) = \{R_i(f(x)_i, g(y)_i)\}_{i\in\mathcal{S}(|x|)\cap\mathcal{T}(|y|)}, \tag{5.2}$$

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.

It is easy to see that R^{bndl} is equivalent to $R^{\leq}$ except for the case $|(x,1^t)| > |M|$. In this case, the decryption outputs an empty set $\emptyset$. However, the output should be 0 in FE for $R^{>}$. This issue can be easily fixed as observed by Agrawal et al. [5, Section 6.2.2]. Namely, we modify the decryption algorithm so that it outputs 0 if the decryption result is $\emptyset$. We note that the resulting scheme inherits AD-SIM security, which is guaranteed by [3, Theorem 2.1].

Removing the shorter input length restriction. In the above construction, there is a restriction that the decryption is possible only when $|(x,1^t)| \leq |M|$. To remove the restriction, we first construct an AD-SIM secure FE scheme for TM such that the decryption is possible only when $|(x,1^t)| > |M|$. Such a scheme can be obtained by applying the conversion by Agrawal et al. [5, Section 6.1 and 6.2.1] to AD-SIM secure CPFE with dynamic bounded collusion for $\mathcal{C}_{\mathsf{inp},\mathsf{out}}$ obtained in Section 4. We then combine these two schemes to obtain the full-fledged scheme without the restriction by applying the conversion by Agrawal et al. [5, Section 6.2.3]. Then, we obtain AD-SIM secure FE for TM. Based on the discussion above, we obtain the following theorem:

Theorem 4. *Assuming IBE with* AD-IND *security, ABE for circuits with circuit class $\mathcal{C}$ with* Sel-IND *security, updatable LOT (as per [3, Definition A.1]), and PIR (as per [3, Definition 2.9] whose answer function is in $\mathcal{C}$, we have FE for TM with* AD-SIM *security against dynamic bounded collusion.*

Acknowledgements. This work of Shweta Agrawal is partly supported by the DST "Swarnajayanti" fellowship, the Indian National Blockchain Project, and the CCD Center of Excellence. Shota Yamada is partially supported by JST AIP Acceleration Research JPMJCR22U5 and JSPS KAKENHI Grant Number 19H01109.

References

1. Agrawal, S.: Stronger security for reusable garbled circuits, new definitions and attacks. In: CRYPTO (2017)
2. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: New perspectives and lower bounds. In: CRYPTO (2013)
3. Agrawal, S., Kitagawa, F., Modi, A., Nishimaki, R., Yamada, S., Yamakawa, T.: Bounded functional encryption for turing machines: Adaptive security from general assumptions. Cryptology ePrint Archive, Paper 2022/316 (2022)
4. Agrawal, S., Maitra, M.: FE and iO for Turing machines from minimal assumptions. In: TCC (2018)
5. Agrawal, S., Maitra, M., Vempati, N.S., Yamada, S.: Functional encryption for Turing machines with dynamic bounded collusion from LWE. In: CRYPTO (2021)
6. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: CRYPTO (2019)
7. Agrawal, S., Singh, I.P.: Reusable garbled deterministic finite automata from learning with errors. In: ICALP (2017)
8. Ananth, P., Fan, X., Shi, E.: Towards attribute-based encryption for RAMs from LWE: Sub-linear decryption, and more. In: ASIACRYPT (2019)
9. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: CRYPTO (2015)
10. Ananth, P., Lombardi, A.: Succinct garbling schemes from functional encryption through a local simulation paradigm. In: TCC (2018)
11. Ananth, P., Sahai, A.: Functional encryption for Turing machines. In: TCC (2016)
12. Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: TCC (2019)
13. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. J. ACM **65**(6), 39:1–39:37 (2018)
14. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: EUROCRYPT (2014)
15. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC (2011)
16. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: EUROCRYPT (2018)
17. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS (2011)

18. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: CRYPTO (2017)
19. Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: TCC (2017)
20. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: CRYPTO (2017)
21. Döttling, N., Garg, S., Hajiabadi, M., Masny, D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: PKC (2018)
22. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: CRYPTO (2019)
23. Garg, R., Goyal, R., Lu, G., Waters, B.: Dynamic collusion bounded functional encryption from identity-based encryption. In Eprint 2021/847 (2021), to appear in Eurocrypt'22
24. Garg, S., Srinivasan, A.: Adaptively secure garbling with near optimal online complexity. In: EUROCRYPT (2018)
25. Gentry, C., Halevi, S., Raykova, M., Wichs, D.: Garbled RAM revisited, part I. In: EUROCRYPT (2014)
26. Gentry, C., Halevi, S., Raykova, M., Wichs, D.: Outsourcing private RAM computation. In: FOCS (2014)
27. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO (2013)
28. Goldwasser, S., Tauman Kalai, Y., Popa, R., Vaikuntanathan, V., Zeldovich, N.: How to run Turing machines on encrypted data. In: CRYPTO (2013)
29. Goldwasser, S., Tauman Kalai, Y., Popa, R., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC (2013)
30. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions from multiparty computation. In: CRYPTO (2012)
31. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC (2013)
32. Gorbunov, S., Vinayagamurthy, D.: Riding on asymmetry: Efficient ABE for branching programs. In: ASIACRYPT (2015)
33. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: TCC (2016)
34. Goyal, R., Syed, R., Waters, B.: Bounded collusion ABE for TMs from IBE. In: ASIACRYPT (2021)
35. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS (2006)
36. Kitagawa, F., Nishimaki, R., Tanaka, K., Yamakawa, T.: Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In: CRYPTO (2019)
37. Lu, S., Ostrovsky, R.: How to garble RAM programs. In: EUROCRYPT (2014)
38. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT (2005)
39. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: FOCS (1986)