Adaptively Secure and Succinct Functional Encryption: Improving Security and Efficiency, Simultaneously

Fuyuki Kitagawa [1] Ryo Nishimaki [2] Keisuke Tanaka [1] Takashi Yamakawa [2]

[1] Tokyo Institute of Technology, Japan
{kitagaw1,keisuke}@is.titech.ac.jp
[2] NTT Secure Platform Laboratories, Japan
{nishimaki.ryo,yamakawa.takashi}@lab.ntt.co.jp

Abstract

Functional encryption (FE) is advanced encryption that enables us to issue functional decryption keys where functions are hardwired. When we decrypt a ciphertext of a message m by a functional decryption key where a function f is hardwired, we can obtain $f(m)$ and nothing else. We say FE is selectively or adaptively secure when target messages are chosen at the beginning or after function queries are sent, respectively. In the setting of weakly-selective setting, function queries are also chosen at the beginning. We say FE is single-key/collusion-resistant when it is secure against adversaries that are given only-one/polynomially-many functional decryption keys, respectively. We say FE is sublinearly-succinct/succinct when the running time of an encryption algorithm is sublinear/poly-logarithmic in the function description size, respectively.

In this study, we propose adaptively secure, collusion-resistant, and succinct (we call "fully-equipped") PKFE schemes for circuits. More specifically, we propose a generic transformation from weakly-selectively secure, single-key, and sublinearly-succinct PKFE for circuits into fully-equipped PKFE for circuits. We assume only the existence of weakly-selectively secure, single-key, and sublinearly-succinct PKFE for circuits. That is, our transformation relies on *neither* concrete assumptions such as learning with errors *nor* indistinguishability obfuscation. This is the first generic construction of fully-equipped PKFE that does not rely on indistinguishability obfuscation.

As side-benefits of our results, we obtain the following primitives from weakly-selectively, single-key, and sublinearly-succinct PKFE for circuits: (1) laconic oblivious transfer (2) succinct garbling scheme for Turing machines (3) selectively secure, collusion-resistant, and succinct PKFE for Turing machines (4) low-overhead adaptively secure traitor tracing (5) key-dependent-message secure and leakage-resilient public-key encryption. We also obtain a semi-generic transformation from simulation-based adaptively secure garbling schemes into adaptively indistinguishable garbling schemes whose online complexity does not depend on the output length.

Keywords: Functional encryption, Adaptive security, Succinct encryption, Adaptive garbled circuit, Laconic oblivious transfer

Contents

1 Introduction

1.1 Background

Achieving stronger cryptographic primitives by using weaker ones is one of the central and fundamental tasks in cryptography. We would like to minimize assumptions to achieve more secure and advanced cryptography. A typical example is how to achieve IND-CCA secure public-key encryption from IND-CPA secure one [DDN00, Sah99] (there is a long series of works though we omit most references). The objective of this study is showing how to achieve more secure and efficient *functional encryption* (FE) from less secure and efficient one in a generic way.

FE [BSW11] is encryption that enables us to issue functional decryption keys sk_f where a function f is hardwired. We can decrypt a ciphertext ct_m of a message m by using sk_f. A notable feature of FE is that we obtain $f(\mathsf{m})$ and nothing else when we decrypt ct_m by sk_f. If we can encrypt messages by a public-key (resp. a master secret key), then we call public-key (resp. secret-key) FE (PKFE and SKFE for short). FE is the most advanced form of encryption and can control what information of messages can be given to owners of functional decryption keys by using various functions. Moreover, FE is a versatile tool to achieve useful cryptographic primitives such as trapdoor permutations, universal samplers, non-interactive multi-party key-exchange [GPSZ17]. The most prominent application of FE is achieving indistinguishability obfuscation (IO) [BGI+12, GGH+16] from FE [AJ15, BV15, BNPW16, KS17, KNT18a].

There are three main performance measures of FE. One is the number of issuable functional decryption keys. Another is the level of security. The other is the running time of an encryption algorithm (or the size of an encryption circuit). If an FE scheme can securely release one/polynomially-many functional decryption key/s, we call it a single-key/collusion-resistant scheme. Roughly speaking, an FE scheme is secure if adversaries cannot distinguish whether a target ciphertext is an encryption of m_0 or m_1 chosen by them. In the security game, adversaries can send functional decryption key queries and receives sk_f for queried f as long as $f(\mathsf{m}_0) = f(\mathsf{m}_1)$. If adversaries are required to commit target messages $(\mathsf{m}_0, \mathsf{m}_1)$ (resp. and queries $f_1, \ldots, f_q$) at the beginning of the game, we call it *selective* (resp. *weakly selective*) security. If adversaries can decide target messages after they send functional decryption key queries[1], then we call it *adaptive* security. The running time of an encryption algorithm must depend on the length of messages to be encrypted. Moreover, the running time might depend on the size of functions supported by the scheme as several known FE schemes do [SS10, GVW12]. The dependence on the size of functions should be as low as possible to decrease the encryption time of FE. FE is called *succinct* if the dependence is logarithmic, and is called *sublinearly-succinct* if the dependence is sublinear.

It is desirable to achieve the best properties of all performance measures simultaneously. Therefore, the following question is natural.

Can we achieve adaptively secure, collusion-resistant, and succinct PKFE for circuits by using only weakly-selectively secure, single-key, and sublinearly-succinct one?

This question has been extensively studied [ABSV15, BV15, AJS15, GS16, LM16, HJO+16, JSW17, GS18], but all previous studies gave only partial answers. In this study, we give an affirmative answer to the open question above, which was clearly stated by Garg and Srinivasan [GS16]. We sometimes call the building-block and goal-primitive in the question above *obf-minimum*[2] and *fully-equipped* PKFE, respectively in this paper.

One might wonder why we do not start with weakly-selectively secure, single-key, and *non-succinct* FE. This is because there is a huge gap between non-succinct FE and sublinearly-succinct one. We know that sub-exponentially-secure sublinearly-succinct FE implies IO for circuits [AJ15, BV15, GS16, KNT17a, KNT17b, KNT18b, KNT18a]. We also know that non-succinct PKFE (resp. SKFE) is achieved by *plain*

[1] Of course, adversaries can send queries after they decided a pair of target messages.

[2] See the subsequent paragraph for the reason of naming "obf-minimum".

public-key encryption (resp. one-way function) [SS10, GVW12]. It is unlikely that we can achieve IO from plain public-key encryption. Thus, we start with sublinearly-succinct FE. We also emphasize that we focus on transformations with *polynomial security loss* in this study. If sub-exponential security loss is allowed, we can achieve IO from obf-minimum SKFE/PKFE. We rely on neither sub-exponential security nor IO in this study.

Hereafter, we use the following notations. Relationships between different notions of PKFE and SKFE are parameterized by $(\#\mathsf{ct}, \#\mathsf{key}, \mathsf{sec}, \mathsf{eff})$. Here, $\#\mathsf{ct}, \#\mathsf{key} \in \{1, \mathsf{unb}\}$ denote the number of ciphertexts/functional-decryption-keys: unb means unbounded polynomially many, $\mathsf{sec} \in \{\mathsf{w\text{-}sel}, \mathsf{sel}, \mathsf{ada}\}$ denotes weakly-selective, selective or adaptive security, $\mathsf{eff} \in \{\mathsf{ns}, \mathsf{sls}, \mathsf{fs}\}$ denotes the efficiency: ns, sls, and fs denote non-succinct, sublinearly-succinct, and succinct, respectively. In the case of PKFE, we omit $\#\mathsf{ct}$[3].

Known transformations for better security and efficiency. There are several techniques to strengthen security and/or improve the efficiency of FE.

Ananth, Brakerski, Segev, and Vaikuntanathan [ABSV15] presented a transformation from selectively secure FE to adaptively secure FE. Unfortunately, this transformation does not preserve (sublinear-)succinctness. This is because the transformation uses a $(\mathsf{unb}, 1, \mathsf{ada}, \mathsf{ns})$-SKFE scheme[4] [GVW12] as a key building block. Due to this building block, even if we start with $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE, we obtain $(\mathsf{unb}, \mathsf{ada}, \mathsf{ns})$-PKFE.

Garg and Srinivasan [GS16], and Li and Micciancio [LM16] presented transformations from single-key and sublinearly-succinct PKFE to collusion-resistant one. More specifically, the transformation by Garg and Srinivasan [GS16] is from $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE to $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE. However, these transformations do not preserve adaptive security.

Ananth, Jain, and Sahai [AJS15] and Bitansky and Vaikuntanathan [BV15] presented a transformation from $(\mathsf{unb}, \mathsf{sel}, \mathsf{ns})$-PKFE to $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE. This transformation also does not preserve adaptive security.

Ananth and Sahai [AS16] presented a transformation (denoted by AS16 transformation) from $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE for circuits to $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$-PKFE for Turing machines (TMs) by using $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for TMs. If the building block SKFE is for *circuits*, then the transformation also works and we obtain the resulting PKFE for *circuits*. The difference from the transformation by Ananth et al. [ABSV15] is that we can start with $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE. One of the main contributions of Ananth and Sahai [AS16] is that they presented a $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE scheme for TMs by using IO. AS16 transformation is the closest to what we want, but not satisfactory since it uses IO (that is, *sub-exponentially secure* FE).

All these transformations sacrifice either adaptive security or succinctness or rely on IO. Thus, the transformation in the question above has been a famous open problem in the area of FE.

Crucial ingredient: adaptive garbling. As we saw above, if we can obtain $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for circuits from $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE, then we resolve the open question above by using the transformations of Garg and Srinivasan [GS16] and Ananth and Sahai [AS16]. In fact, $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for circuits is essentially the same as *adaptively indistinguishable garbling schemes (indistinguishability-based definition [JSW17]) whose online computational complexity is* $\mathrm{poly}(\log |C|, n, \lambda)$ where C is a circuit to be garbled, n is the input length of C, and λ is the security parameter[5]. We call garbling schemes whose

[3] In the case of PKFE, $\#\mathsf{ct}$ is trivially unb.

[4] In the setting of SKFE, only an entity that has a master secret-key can generate ciphertexts. Thus, adversaries is allowed to send messages as queries and receives ciphertexts in its security game. When adversaries can send one/polynomially-many message(s), we say one/many-ciphertext SKFE.

[5] In fact, there are subtle issues to transform a garbling scheme into a single-key and single-ciphertext SKFE (the opposite is easy). See Section 5.3 for more details.

online computational complexity is $\mathrm{poly}(\log |C|, n, \lambda)$ *circuit-succinct* garbling schemes.[6] Thus, we focus on adaptive and circuit-succinct garbling schemes.

Several adaptively secure garbling schemes have been proposed [BHR12, HJO+16, JW16, JSW17, JKK+17, GS18]. The garbling scheme of Bellare, Hoang, and Rogaway is not circuit-succinct, that is, the online computational complexity is $\mathrm{poly}(|C|, \lambda)$. The garbling scheme of Hemenway, Jafargholi, Ostrovsky, Scafuro, and Wichs [HJO+16] achieves online computational complexity $(n+m+w)\mathrm{poly}(\lambda)$ where n, m, d, and w are the input length, output length, depth, and width of a circuit to be garbled, respectively (they also presented a garbling scheme for NC^1 circuits whose complexity is $(n+m)\mathrm{poly}(\lambda)$). Jafargholi, Scafuro, and Wichs [JSW17] presented an adaptively indistinguishable garbling scheme whose online computational complexity is $(n+w)\mathrm{poly}(\lambda)$. The garbling scheme of Garg and Srinivasan [GS18] (we call GS18 scheme in this paper) achieved online computational complexity $O(n+m)+\mathrm{poly}(\log |C|, \lambda)$. Others [JW16, JKK+17] are garbling scheme for NC^1 circuits. None of these is satisfactory for our goal since the complexity depends on a polynomial of $|C|$, w, d, or m.

GS18 scheme is closest to what we want. However, there are two issues as follows.

1. GS18 scheme is based on a concrete assumption (the CDH, LWE, or factoring assumptions). More specifically, the scheme is based on updatable laconic oblivious transfer (LOT) [CDG+17], which is achieved by the CDH, LWE, or factoring assumptions [CDG+17, BLSV18, DGHM18].

2. GS18 scheme is simulation-based secure. Therefore, the online computational complexity must be at least linear in m since Applebaum, Ishai, Kushilevitz, and Waters [AIKW15] showed the lower bound of online complexity for simulation-based secure garbled circuits.

Getting rid of the dependence on output length. If we can generically transform a simulation-based adaptively secure garbling scheme whose online computational complexity is $\mathrm{poly}(n, m, g(|C|), \lambda)$ where $g(\cdot)$ is some function (such as $\log(\cdot)$) into an adaptively *indistinguishable* garbling scheme whose online computational complexity is $\mathrm{poly}'(n, g(|C|), \lambda)$, then we can solve the second issue explained above. This is because we can obtain an adaptively *indistinguishable* and circuit-succinct garbling scheme from such a transformation and GS18 scheme [GS18]. In fact, Jafargholi et al. left such a transformation as an open problem [JSW17]. We quote their sentence in a footnote.[7] This open question is related to our main question since $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE is the crucial ingredient as explained above.

1.2 Our Contributions

We solved the open problem explained in the previous section. In particular, we prove the following theorem.

Theorem 1.1 (Main theorem). *Assume that there exists weakly-selectively secure, single-key, and sublinearly-succinct PKFE for circuits, then there exists adaptively secure, collusion-resistant, and succinct PKFE for circuits.*

Note that all our constructions and transformations in this study incur only polynomial security loss. As we explained in the previous section, our crucial ingredient is a $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE scheme to prove the informal theorem above. To obtain the scheme, we will prove the (informal) theorems below, which are of independent interests, and construct an adaptively secure garbling scheme whose online computational complexity is $\mathrm{poly}(\log |C|, n, \lambda)$ by combining (a variant of) GS18 scheme.

[6] Note that this is different from succinct garbling schemes [AL18, BCG+18] since ours is for circuits while succinct garbling schemes are for TMs.

[7] Jafargholi et al. wrote *"It remains an open problem whether it is possible to show a more general transformation from garbled circuits with adaptive security (and maybe other natural properties) to garbled circuits with indistinguishability based adaptive security and online complexity independent of the output size."*[JSW17]

Theorem 1.2 (Informal, see Theorem 4.8). *Assume that there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists updatable laconic oblivious transfer.*

That is, we can generically construct updatable LOT from obf-minimum FE. This solves the first issue of GS18 scheme. This itself is interesting since this is the first construction of LOT that relies on *neither* specific number theoretic assumptions *nor* IO.[8] Therefore, we obtain an adaptively secure garbling scheme whose online computational complexity is $O(n+m)+\mathrm{poly}(\log|C|,\lambda)$ from obf-minimum PKFE (we do not rely on any concrete assumption) via GS18 scheme. Note that, to achieve this, we need some tweaks for the garbling scheme since the security level of our updatable LOT is slightly weaker than that used in GS18 scheme. In fact, we prove that such a weaker LOT is sufficient to achieve an adaptively secure garbling scheme that we need. However, for simplicity, we give only informal theorems here. See Section 2 for more details. Thus, we resolve the first issue of GS18 scheme.

We propose two solutions for the second issue of GS18 scheme. One is proposing an extension of AS16 transformation [AS16]. More specifically, we combine AS16 transformation with the following theorem.

Theorem 1.3 (Informal, see Theorem 6.1). *If there exists* $(\mathsf{unb}, \mathsf{sec}, \mathsf{eff})$*-PKFE for boolean circuits, then there exists* $(\mathsf{unb}, \mathsf{sec}, \mathsf{eff})$*-PKFE for circuits where* $\mathsf{sec} \in \{\mathsf{w\text{-}sel}, \mathsf{sel}, \mathsf{ada}\}$ *and* $\mathsf{eff} \in \{\mathsf{ns}, \mathsf{sls}, \mathsf{fs}\}$*. This transformation preserves adaptive security and succinctness.*

If we set $m = 1$ (that is, single-bit output) in adaptively secure garbling scheme whose online computational complexity is $O(n, m, \log|C|, \lambda)$, then we obtain adaptively secure circuit-succinct garbling scheme for *boolean* circuits. We plug this into AS16 transformation, and then we obtain $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$-PKFE for *boolean* circuits. Lastly, by applying the informal theorem above, we can obtain what we want (i.e., fully-equipped PKFE). See the next section for more details. Note that it is easy to transform our variant of GS18 scheme into $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for boolean circuits. See Section 5.3 for details.

The other is using the transformation in the following theorem.

Theorem 1.4 (Informal, see Theorem 7.8). *Assume that there exists a simulation-based adaptively secure garbling scheme whose online computational complexity depends on the output length of circuits, then there exists an indistinguishability-based adaptively secure garbling scheme whose online computational complexity does not depend on the output length of circuits. The overhead of the transformation is not large, that is, the online complexity affected by other parameters (*$|C|$*,* n*, and* λ*) do not change in an asymptotic sense.*

That is, we solve the open question by Jafargholi et al. [JSW17]. Note that the first solution is much simpler than the second one. However, the technique used in the transformation in Theorem 1.4 is related to other our techniques in this study, and adaptively secure circuit-succinct garbling schemes are closely related to our goal as we explained so far. Moreover, Theorem 1.4 solves the open problem presented by Jafargholi et al. [JSW17] (We think this is of an independent interest). Therefore, we also include the second solution in this paper.

More implications of our results. Ananth and Lombardi [AL18] proved that if there exists single-key and succinct PKFE for circuits and one of CDH/LWE/factoring assumptions holds, then there exists succinct garbling scheme for TMs. The concrete assumptions come from that they use LOT in their construction. We can replace their LOT with our LOT based on FE[9]. Thus, we obtain the following corollary.

[8] Ananth and Lombardi present an LOT protocol based on IO [AL18].

[9] The security level of our LOT is sufficient for their purpose.

Corollary 1.5. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists a succinct garbling scheme for TMs.*

We also obtain the following corollary by combining with the known results [AS16, GS16].

Corollary 1.6. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists* $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$*-PKFE for TMs.*

That is, we remove the concrete assumptions from the theorems of Ananth and Lombardi.[10] Agrawal and Maitra [AM18] also proved that if there exists succinct PKFE for circuits, then there exists PKFE for TMs. However, their PKFE for TMs supports only single/constant-bit output TMs. That is, our corollary above improves their result since ours supports multi-bit output TMs.[11]

Nishimaki, Wichs, and Zhandry [NWZ16] presented a traitor tracing scheme that supports an exponentially large identity space and whose ciphertext overhead is $O(\log n)$ where n is the length of identities. Their scheme is based on fully-equipped PKFE that was instantiated by IO previously. We can use our fully-equipped PKFE scheme instead of their IO-based scheme. Thus, we obtain the following corollary.

Corollary 1.7. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, there exists an adaptively secure traitor tracing scheme whose master key size is* $\mathrm{poly}(\log n)$*, secret key size is* $\mathrm{poly}(n)$*, and ciphertext size is* $|\mathsf{m}| + \mathrm{poly}(\log n)$ *where* $|\mathsf{m}|$ *is the message length.*

Brakerski, Lombardi, Segev, and Vaikuntanathan [BLSV18] presented a key-dependent-message (KDM) secure and leakage-resilient PKE scheme based on batch encryption, which is essentially the same as LOT. Thus, we obtain the following corollary (See the reference [BLSV18] for the details of parameters in the statement).

Corollary 1.8. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists a PKE scheme that satisfies (1) KDM security with respect to affine functions of the secret key and (2) leakage-resilience with leakage rate* $1 - o(1)$*.*

To the best of our knowledge, except constructions based on IO [DGL+16, MPS16], all existing generic constructions of PKE satisfying KDM security or leakage resilience of $1 - o(1)$ rate assume some algebraic property such as homomorphism to the underlying primitive. Our construction is a generic construction of PKE satisfying the above security notions based on a polynomially secure primitive without such algebraic properties.

2 Technical Overview

In this section, we give high level overviews of our techniques. We briefly summarize how to arrive at fully-equipped PKFE from obf-minimum PKFE in Figure 1.

2.1 Laconic OT from Succinct PKFE

We first show an overview of our LOT protocol based on sublinearly succinct PKFE. More precisely, we construct updatable LOT with arbitrary compression factor based on $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE.

By the transformation of Cho et al. [CDG+17] and an observation by Ananth and Lombardi [AL18], we can transform non-updatable LOT with compression factor 2 into updatable one with arbitrary

[10] Note that we cannot obtain an adaptively secure scheme in Corollary 1.6 since the succinct garbling for TMs by Ananth and Lombardi is not adaptively secure.

[11] Note that their FE for TMs satisfies a stronger security notion called distributional indistinguishability than standard indistinguishability.

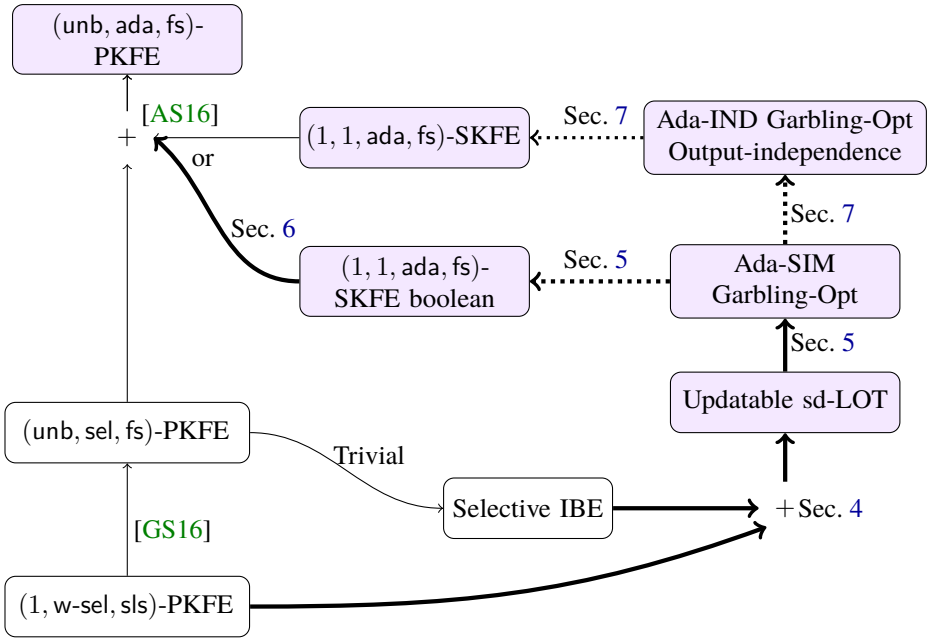

Figure 1: Illustration of the path from our starting point to the goal: In this figure, "SKFE boolean" denotes SKFE for boolean circuits. "Updatable sd-LOT" denotes selective-database updatable laconic OT. Regarding garbling scheme, "Garbling-Opt" denotes garbling schemes with nearly optimal online complexity and "Output-independence" denotes the online complexity does not depends on output-length (See Sections 5 and 7 for more details). Ada-SIM/Ada-IND denote simulation-/indistinguishability-based adaptively secure garbling schemes, respectively. Solid thin arrows denote known or trivial implications. Thick solid and dotted arrows denote implications that we prove in this study. Here, in the case of dotted lines, we assume specific properties of underlying tools. See each section for details.

---

compression factor using Merkle tree. Thus, to achieve our goal, we can focus on constructing non-updatable LOT with compression factor 2 based on sublinearly succinct PKFE. Our first observation is that we might construct such LOT based on identity-based encryption (IBE). In this overview, let the length of a database D be s, that is $D \in \{0,1\}^s$, and $D[i]$ denotes the i-th bit of D.

Laconic OT based on IBE and its problem. We first review the definition of LOT. An LOT consists of four algorithms $\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}$, and $\mathsf{Receive}$. We generate a CRS crs using Gen. Hash, given crs and a database D, outputs a short digest d and private state $\widehat{D}$. The algorithm Send, given d, a database location L, and two messages m_0 and m_1, outputs LOT's ciphertext e. By using $\mathsf{Receive}$, a receiver who has the secret state $\widehat{D}$ can decrypt e and obtain $m_{D[L]}$. For security, we require that an honest receiver cannot obtain the other message $m_{1-D[L]}$ even if he has $\widehat{D}$.

Our basic idea for how to construct LOT based on an IBE scheme IBE is as follows. When hashing a database D, we first generate a master public-key and master secret-key $(\mathsf{MPK}, \mathsf{MSK})$ of IBE and $\mathsf{sk}_{i,D[i]} \leftarrow \mathsf{KG}(\mathsf{MSK}, i\|D[i])$ for every $i \in [s]$. Then, we set MPK as a digest of D and $\{\mathsf{sk}_{i,D[i]}\}_{i\in[s]}$ as a secret state $\widehat{D}$. When generating LOT's ciphertext e for location $L \in [s]$ and two messages m_0 and m_1, we generate $e = (\mathsf{Enc}(\mathsf{MPK}, L\|0, m_0), \mathsf{Enc}(\mathsf{MPK}, L\|1, m_1))$. We see that a receiver who has $\widehat{D} = \{\mathsf{sk}_{i,D[i]}\}_{i\in[s]}$ can obtain $m_{D[L]}$. If the receiver honestly generates $\widehat{D}$ and deletes MSK, he cannot obtain $m_{D[L]}$ based on the security of IBE. Moreover, if the size of a master public-key of IBE is independent of the identity length, the size of a digest is also independent of the database size.

The above construction based on IBE seems to satisfy the syntactic and security requirement of LOT. However, the construction has a problem that the hash procedure is randomized. Though the definition of LOT by Cho et al. does not explicitly require that the hash algorithm be deterministic, we observe that the

hash algorithm needs to be deterministic for the security notion defined by Cho et al. [CDG+17] to be meaningful. In fact, the above basic construction has a crucial problem that if a receiver computes a hash value by himself, he obtains a master secret-key of IBE and can decrypt any ciphertext.

Moreover, it is not clear whether we can apply the bootstrap method proposed by Cho et al [CDG+17] if the hash function of the underlying LOT is randomized. Their bootstrapping method implicitly assumes the hash algorithm of the underlying LOT is deterministic.

Derandomization using IO. For the above reasons, we need to derandomize the hash algorithm of the above construction. We can make the hash procedure of the above construction deterministic by using IO and puncturable pseudorandom function (PRF) as follows.

In a modified construction, we generate a CRS by obfuscating a circuit that, given a database D, first generates a random coin by using D and a puncturable PRF key and then perform the hash procedure of the basic construction using the random coin. This circuit outputs a digest that is a master public-key of IBE and secret state that is secret-keys of IBE corresponding to D, but not master secret-key.

We can prove the security of the modified construction based on the punctured programming technique proposed by Sahai and Waters [SW14]. However, to complete the proof, we need to require an adversary to declare the challenge database before seeing a CRS. This is because, in the security proof, we need to generate a CRS as an obfuscated circuit that has the challenge database hardwired. This security notion for LOT is weaker than that used by Garg and Srinivasan [GS18] to construct adaptive garbling scheme.

Selective-database security. In this work, we show that we can construct an adaptive garbling scheme based on LOT whose security holds only when the challenge database is selectively determined. We call an LOT scheme satisfying such a security notion *selective-database* LOT. Note that we allow an adversary for LOT to adaptively choose the challenge location and messages. In fact, in our construction of adaptive garbling scheme, we need LOT whose security holds even if the challenge messages are adaptively chosen. In contrast, the security notion defined by Cho et al. [CDG+17] that requires an adversary to declare all challenge instances before seeing CRS is not sufficient for our adaptive garbling scheme. In Section 2.2, we explain this issue in more detail.

By weakening the required security notion to selective-database security, LOT no longer imply collision-resistant hash function while the LOT satisfying an adaptive security notion used by Garg and Srinivasan does. This weakening seems to be necessary to achieve LOT from IO due to the substantial barrier that was shown by Asharov and Segev [AS15].

Replacing IO with sublinearly succinct PKFE. We can replace IO in the above construction with sublinearly succinct PKFE by relying on the result shown by Liu and Zhandry [LZ17].

Liu and Zhandry generalized previous works [GPS16, GPSZ17, GS16], and showed we can replace IO with *decomposable obfuscation (dO)* that can be based on polynomially secure $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE if the circuit pair to be obfuscated satisfies some condition. Roughly speaking, they showed that if there is a polynomial size "witness" for the functional equivalence of a circuit pair to be obfuscated, IO can be replaced with dO. One particular situation where this condition is satisfied is that in the security proof we modify a circuit to be obfuscated so that it outputs a hardwired value for *a single input* and otherwise it runs in the same way as the original one.

Using the terminology by Liu and Zhandry, hardwiring a single output for an input into a circuit corresponds to *decompose* the circuit to the input. We explain this in more detail. Let C be a circuit of 3-bit input. For a bit string x of length less than 3, let C_x be a circuit $C(x\|\cdot)$, that is, C in which x is hardwired as the first $|x|$ bit of the input. We call such a circuit partial evaluation of C. When decomposing C to the input say 100, we represent C as the tuple of partial evaluations $(C_0, C_{11}, C_{100}, C_{101})$. When considering C as a complete binary tree, $(C_0, C_{11}, C_{100}, C_{101})$ corresponds to the cover of minimum size

that contains 100. We see that computation of C on any input can be done using $(C_0, C_{11}, C_{100}, C_{101})$. This is essentially the same as hardwiring a single output $C(100)$ on input 100 into C.

Liu and Zhandry showed if C is obfuscated by dO, we can replace it with an obfuscated circuit that is constructed from partial evaluations $(C_0, C_{11}, C_{100}, C_{101})$ without affecting the behavior of an adversary. At a high level, this change can be done by removing C and embedding $(C_0, C_{11}, C_{100}, C_{101})$ into functional keys of the underlying PKFE. Then, we can perform security proofs in a similar way as the punctured programming.

Consider a circuit of the form $C(x) = C'(x; \mathsf{F}_K(x))$, where C' is a circuit, F is a PRF, and K is a PRF key. For simplicity, let C be a circuit of 3 bit input as above. We show how to change the distribution of $C(100)$. By obfuscating C with dO, we can decompose C to 100, that is, we can replace obfuscated C with obfuscated circuit constructed from $(C_0, C_{11}, C_{100}, C_{101})$. Next, we change $\mathsf{F}_K(100)$ with a truly random string. To accomplish this step, we require that $\mathsf{F}_K(100)$ is pseudorandom even if partial evaluations of $\mathsf{F}_K(\cdot)$ for $0, 11$, and 101 are given. Liu and Zhandry call such PRF *decomposing compatible PRF* and the construction of PRF by Goldreich, Goldwasser, and Micali [GGM86] satisfies such a property. Once we can replace $\mathsf{F}_K(100)$ with a truly random string, we can change the distribution of $C(100)$. Thus, we can complete the security proof.

Instantiating our construction with sublinearly succinct PKFE. The circuit to be obfuscated in our construction is of the form $C(x) = C'(x; \mathsf{F}_K(x))$, where C' is a circuit executes a setup and key generation algorithm of IBE. In a similar manner as above, we can change the security game so that the master public-key and secret-keys related to the challenge database are generated using a truly random string. Then, we can prove the selective-database security of our LOT based on the selective security of IBE. Note that in the reduction, the challenge identity in the security game of IBE is $L^* \| 1 - D^*[L^*]$, where D^* and L^* are challenge database and position in the security game of LOT. The identity $L^* \| 1 - D^*[L^*]$ depends on the choice of L^* by an adversary for LOT. However, the reduction algorithm can guess the location with the probability at least $\frac{1}{s+1}$, which is inverse polynomial. Thus, a selectively secure IBE is sufficient for this construction.

Therefore, we can replace IO in our construction with dO, which can be based on $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE. Moreover, selectively secure IBE can be constructed from $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE based on the result by Garg and Srinivasan [GS16]. Their collusion-resistant PKFE based on $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE can be used as an identity-based key encapsulation mechanism the size of whose master public-key is independent of the length of identities.[12] Thus, we can construct selective-database LOT based only on $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE.

Achieving updatability. Cho et al. [CDG+17] showed we can bootstrap an LOT with the compression factor 2 into an *updatable* LOT with arbitrary compression factor using garbled circuits and a Merkle tree. Roughly speaking, an updatable LOT is an LOT that has additional two algorithms $\mathsf{SendWrite}$ and $\mathsf{ReceiveWrite}$ that enable us to transmit messages updating the database.

As observed by Ananth and Lombardi [AL18], Cho et al.'s bootstarapping method is not sufficient for LOT whose security holds only when an adversary declares the challenge database before seeing crs. Therefore, we cannot use the bootstrapping method of Cho et al. directly to make our selective-database LOT updatable. However, we can use a *minor variant* of the bootstrapping method observed by Ananth and Lombardi [AL18] to bootstrap selective-database LOT into updatable one. By applying the variant of bootstrapping method to the above construction, we obtain updatable selective-database LOT with arbitrary compression factor based on sublinearly succinct PKFE.

[12] To achieve $\frac{1}{2}$ compression in our construction, it is sufficient that the size of a master public-key is logarithmic in the length of identities. This requirement is more natural for IBE, and thus we assume only this mild condition in the actual construction.

Comparison with the construction by Ananth and Lombardi [AL18]. Ananth and Lombardi showed a construction of LOT based on IO. As they noted, it seems difficult to replace IO in their construction with polynomially secure PKFE. They first constructed a weak variant of somewhere statistically binding (SSB) hash function based on IO and one-way functions. Then, they constructed LOT based on the SSB hash function and *witness encryption* [GGSW13] by modifying the construction proposed by Cho et al. [CDG+17]. Since witness encryption can be constructed from IO [GGH+13], their LOT can be based only on IO and one-way functions.

We observe that we can replace IO in their weak variant of SSB hash function with sublinearly succinct PKFE by relying on the result by Liu and Zhandry [LZ17]. On the other hand, the result by Liu and Zhandry does not capture the construction of witness encryption based on IO, and thus we cannot construct witness encryption from sublinearly succinct PKFE using the result by Liu and Zhandry. In fact, it is believed to be hard to construct witness encryption based on some polynomially secure primitive including PKFE.

2.2 Adaptive Garbling from Selective-Database Updatable Laconic OT

The adaptive garbling scheme by Garg and Srinivasan (we write GS18 scheme for short) is based on adaptively secure updatable LOT [GS18], where adversaries can select a database after they see a CRS. However, our LOT based on sublinearly succinct PKFE achieves only selective-database updatable LOT (not adaptively secure), where adversaries must commit a database before a CRS is given. In fact, we prove that we can achieve an adaptive garbling scheme by using a *selective-database* updatable LOT.

Where is the adaptive property of LOT used in GS18 scheme? In GS18 scheme, a database of an updatable LOT is determined by an input x. More specifically, the current database is determined by x, each intermediate wire values determined by x and each gate, and output values. A CRS crs of updatable LOT is generated at the offline phase (i.e., when we generate a garbled circuit $\widetilde{C}$) and crs is hardwired in circuits to be garbled by selectively secure garbling. At this point, x might not be determined yet since we consider the *adaptive* setting. Thus, a simulator must have crs before x (and a database) is fixed. This is why Garg and Srinivasan used the adaptive security of LOT.

Overcoming the issue. The issues is that we need crs at the offline phase. Our idea is deferring using crs until we generate a garbled input (i.e., online phase). To look closer at our idea, we need to explain more on GS18 scheme. In GS18 scheme, "step circuits" are garbled by selectively secure garbling. Each step circuit has the description of each gate of the circuit C to be garbled by the adaptive garbling scheme. Roughly speaking, a step circuit takes as input a digest d of updatable LOT and does the following two procedures.

- Updating the database according to the output wire value of the gate computed from input x.
- Outputting encrypted labels of selectively secure garbling for the next gate via updatable LOT.

The important point is that crs of updatable LOT is hardwired in each step circuit to run Send and $\mathsf{SendWrite}$ algorithms, which was explained in Section 2.1. This is the problem since we do not fix crs at the offline phase. Here our idea comes in.

Instead of hardwiring crs in each step circuit, we define modified step circuits that take as input not only digest d *but also* crs. Now crs is an input for step circuits. By this change, to generate (simulated) garbled modified step circuits, we do not need crs. As a result, crs need not be determined at the offline phase. In the construction, we put crs in the state information though we generate crs at the offline phase in the construction. In the proof, a simulator can adaptively set the state information when the simulator needs it since the state information is not revealed.

The CRS crs must be fixed when a garbled input $\widetilde{x}$ is generated. However, at this point, input x and a database were already determined. Therefore, we can use the selective-database security of updatable LOT because, in the simulation, an adversary of updatable LOT can simulate garbled step circuits without crs, and when x is fixed, the adversary fixes a database based on x and can receive crs in the reduction. This is the main idea behind our adaptive garbling scheme based on selective-database updatable LOT.

Although we can generate crs at the online phase, we select that we put crs in the state information for better online complexity and compatibility with the transformation given in Section 7.

Why selective-database? As mentioned above, we modify GS18 scheme so that we do not need a CRS until generating an online encoding. Once an online input x is determined in addition to an offline input C, all internal values of the computation of $C(x)$ are fixed. From these facts, reduction algorithms attacking updatable LOT seem to be able to determine not only the challenge database but also other challenge instances before seeing CRS. However, this is not the case.

More specifically, reduction algorithms attacking updatable LOT can determine challenge database and position before seeing CRS, but it cannot determine challenge messages without seeing CRS. This is because, in some steps of the proof, reduction algorithms need to set the challenge messages as values computed by using CRS. By this reason, it seems difficult to prove the adaptive security of our garbling scheme based on selectively secure LOT defined by Cho et al. [CDG+17].

To solve the above issue, we introduce a new security notion selective-database security for LOT and prove the adaptive security of our garbling scheme based on the selective-database security of the underlying updatable LOT.

From adaptive garbling to adaptively secure 1-key 1-ciphertext SKFE. By combining two transformations explained in this section and the previous section, we obtain an adaptive garbling scheme whose online complexity is $O(n+m)+\mathrm{poly}(\log|C|,\lambda)$ based on $(1,\mathsf{w\text{-}sel},\mathsf{sls})$-PKFE. Especially, by restricting circuits supported by garbling schemes to boolean circuits (circuits of single-bit output), we obtain an adaptive garbling scheme whose online complexity is $O(n)+\mathrm{poly}(\log|C|,\lambda)$ based on the same assumption.

In the next step, we use the transformation proposed by Ananth and Sahai [AS16]. In order to use their transformation, we have to transform the constructed adaptive garbling scheme into $(1,1,\mathsf{ada},\mathsf{fs})$-SKFE. Although adaptive garbling scheme with succinct online encoding and $(1,1,\mathsf{ada},\mathsf{fs})$-SKFE are essentially the same primitives, there is a difference between them. The security game for $(1,1,\mathsf{ada},\mathsf{fs})$-SKFE allows an adversary to make an encryption query and key query in arbitrary order while that for adaptive garbling scheme requires an adversary to always make circuit query first. We can solve this issue with a simple transformation using a one-time pad. We show the transformation after the construction of adaptive garbling scheme in Section 5.

2.3 From Single-Bit to Multi-Bit Succinct FE by Leveraging Collusion-Resistance

As explained in the previous section, we obtained $(1,1,\mathsf{ada},\mathsf{fs})$-SKFE for boolean functions from $(1,\mathsf{w\text{-}sel},\mathsf{sls})$-PKFE. By applying the transformation by Ananth and Sahai [AS16] to the scheme, we obtain $(\mathsf{unb},\mathsf{ada},\mathsf{fs})$-PKFE for boolean functions. Here, we show that we can transform $(\mathsf{unb},\mathsf{ada},\mathsf{fs})$-PKFE for boolean functions to the one for functions with multi-bit outputs.

The transformation is very simple. We construct a PKFE scheme $\mathsf{MultiPKFE}$ for functions with multi-bit outputs from a PKFE scheme $\mathsf{OnePKFE}$ for boolean functions as follows. The encryption algorithm of $\mathsf{MultiPKFE}$ works completely in the same manner as that of $\mathsf{OnePKFE}$. The key generation algorithm of $\mathsf{MultiPKFE}$, given a function f with m-bit output, first decomposes the function to $\{f_i\}_{i\in[m]}$ where f_i is a function that computes the i-th bit of $f(\mathsf{m})$ on input m. Then it generates decryption keys sk_{f_i} for the function f_i for $i\in[m]$ by the key generation algorithm of $\mathsf{OnePKFE}$, and outputs

$\mathsf{sk}_f := \{\mathsf{sk}_{f_i}\}_{i\in[m]}$. The decryption algorithm of MultiPKFE, given a ciphertext CT of a message m and a decryption key $\mathsf{sk}_f = \{\mathsf{sk}_{f_i}\}_{i\in[m]}$, computes $f_i(\mathsf{m})$ for $i \in [m]$ by using the decryption algorithm of OnePKFE, and outputs $f(\mathsf{m}) = f_1(\mathsf{m})\| \cdots \| f_m(\mathsf{m})$.

In the above construction, if OnePKFE is adaptively collusion-resistant, then MultiPKFE is also adaptively collusion-resistant since a decryption key of MultiPKFE consists of a polynomial number of decryption keys of OnePKFE. Moreover, the transformation also preserves the succinctness of a ciphertext since a ciphertext of MultiPKFE consists of a ciphertext of OnePKFE.

We note that this transformation has not been explicitly pointed out before despite its simplicity. Although researchers in this filed might already observe this transformation, we explicitly write it since to the best of our knowledge, nobody explicitly claims.

By combining the transformation with the results of previous sections, we obtain fully-equipped PKFE for all polynomial-size functions from $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE.

2.4 Adaptively Indistinguishable Garbling with Near-Optimal Online Complexity

We explained how to construct fully-equipped PKFE for all polynomial-size functions from $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE through Section 2.1, 2.2, and 2.3. As mentioned in Section 1, we have another option to achieve it.

In the option, after constructing adaptive garbling scheme as explained in Section 2.2, we transform it into adaptively *indistinguishable* garbling with near-optimal online complexity. More specifically, we construct an adaptively indistinguishable garbling scheme whose online complexity only logarithmically depends on the size of a circuit being garbled, and does not depend on the output length of the circuit. Similarly to adaptive garbling scheme, adaptively indistinguishable garbling with such online complexity can be easily transformed into $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for (multi-bit output) circuits using one-time pad. Thus, by using the transformatin by Ananth and Sahai [AS16] with the resulting $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE, we obtain fully equipped PKFE for circuits.

We can generalize the transformation from adaptive garbling scheme into adaptively indistinguishable garbling that removes the dependence on the output-length of online encoding so that it captures not only our (and GS18) adaptive garbling scheme but also those proposed by Hemenway et al. [HJO+16] and Jafargholi and Wichs [JW16]. Thus, this transformation solves the open question posed by Jafargholi et al. [JSW17]. Here, we give an overview of the transformation.

Basic idea. Our starting point is the simulation-based adaptive garbling given in Section 5 (or in [GS18]), which we denote by $\mathsf{adGC}'_{\mathsf{gs}}$. Recall that the online communication complexity of $\mathsf{adGC}'_{\mathsf{gs}}$ is $n + m + \mathrm{poly}(\lambda, \log |C|)$ where C is the circuit being garbled with n-bit input and m-bit output. Especially, we remark that if we only consider circuits of 1-bit output, then the online communication complexity is $n + \mathrm{poly}(\lambda, \log |C|)$. Our first attempt is to decompose a circuit of m-bit output to circuits of 1-bit output, and garble each of them by using $\mathsf{adGC}'_{\mathsf{gs}}$. Namely, for garbling a circuit C of m-bit output, we garble C_i, which is a circuit that outputs the i-th bit of an output of C, for each $i \in [m]$. For an input x, the input garbling algorithm generates a single garbled input $\widetilde{x}$ by $\mathsf{adGC}'_{\mathsf{gs}}$.

At first glance, this idea would lead to a garbling scheme with online communication complexity $n + \mathrm{poly}(\lambda, \log |C|)$ since we only garble circuits of 1-bit input. However, this idea does not work since a garbling scheme is defined so that 1 garbled input is associated with 1 garbled circuit whereas we need a variant of garbling scheme where 1 garbled input is associated with multiple garbled circuits. Here, we notice that such a variant of garbling scheme can be seen as a single-key SKFE (with function privacy[13]) by interpreting garbled circuits and garbled inputs as ciphertexts and decryption keys of SKFE,

[13]We say that an SKFE scheme is function private if a decryption key does not reveal the associated function. As shown by Brakerski and Segev [BS15], we can generically add the function privacy to any SKFE scheme. Thus we do not care about function privacy in this overview.

respectively. By this interpretation, the online communication and computational complexity as garbling are translated into the secret key length and running time of key generation, and the size of a circuit being garbled is translated into the message length. Based on this observation, we can see that what we need to construct an adaptively indistinguishable garbling with succinct online complexity is an adaptively secure single-key SKFE scheme with succinct decryption key and key generation in the sense that they only logarithmically depend on the message-length.

Single-key SKFE with succinct decryption key and key generation. Our idea to construct such an SKFE scheme is to plug $\mathsf{adGC}'_{\mathsf{gs}}$ into the construction of adaptively secure single-key SKFE by Gorbunov, Vaikuntanathan and Wee [GVW12].[14] We first briefly review their construction. In their construction, for a message m, the encryption algorithm garbles the universal circuit $U(\mathsf{m}, \cdot)$, which is given a description of a function f as input and outputs $f(\mathsf{m})$, by Yao's garbling scheme to generate a garbled circuit $\widetilde{U}$ along with labels that are needed to evaluate the garbled circuit. Then it encrypts $\widetilde{U}$ and labels by a secret-key non-committing encryption for receiver (SK-NCER) to generate a ciphertext of the SKFE scheme.[15] Here, SK-NCER is a special type of SKE in which we can generate a "fake" ciphertext that can be opened to any message that is later chosen along with a corresponding "fake" decryption key. We note that we can construct an SK-NCER scheme whose decryption-key-length is proportional to the message-length from any SKE scheme by "double-encryption" construction similarly to some previous works [CHK05, HPW15]. A decryption key of the SKFE scheme for a function f consists of secret keys of SK-NCER that enable one to recover labels corresponding to f. By using the decryption key, one first recovers labels corresponding to f and then evaluate the garbled circuit $\widetilde{U}$ with these labels to obtain $U(\mathsf{m}, f) = f(\mathsf{m})$. Intuitively, the security of the SKFE scheme holds since an adversary who has a decryption key for f cannot obtain labels that do not correspond to f, and thus $\widetilde{U}$ does not reveal information of m beyond the value of $U(\mathsf{m}, f) = f(\mathsf{m})$ by the security of Yao's garbling. We note that it is essential to encrypt $\widetilde{U}$ by SK-NCER for achieving the adaptive security since Yao's garbling only has the selective security and thus we cannot simulate $\widetilde{U}$ before an input is determined.[16] Since the size of $\widetilde{U}$ is proportional to the message-length of the SKFE scheme and the decryption-key-length of SK-NCER depends on its message-length, the decryption-key-length of their SKFE scheme is proportional to the message-length of the SKFE scheme.

Here, we observe that if we use an adaptive garbling scheme instead of Yao's garbling, then we need not encrypt $\widetilde{U}$ since we can simulate $\widetilde{U}$ before an input is determined by the adaptive security, and we only need to encrypt labels by SK-NCER. Since the number of labels corresponds to the online communication complexity of the underlying garbling scheme, we expect that we could obtain an SKFE scheme with succinct decryption key by plugging $\mathsf{adGC}'_{\mathsf{gs}}$ into this construction. However, there is a problem that $\mathsf{adGC}'_{\mathsf{gs}}$ does not have the *decomposability*, which means that a garbled input is obtained by choosing labels according to each bit of the input whereas the above construction requires the garbling scheme to have the decomposability. Nonetheless, we observe that $\mathsf{adGC}'_{\mathsf{gs}}$ has a similar property to the decomposability called the *quasi-decomposability*, which we introduce in this paper. The quasi-decomposability roughly means that there exists a hash function H such that a garbled input for an input x is generated by choosing labels according to each bit of $\mathsf{H}(x)$ instead of x. We prove that the quasi-decomposability is sufficient to realize the above idea. See Section 7 for the details.

Now, we obtained adaptively secure single-key SKFE with succinct decryption key.[17] We can also

[14] Though Gorbunov et al. [GVW12] presented their construction in the public key setting, the same construction works in the secret key setting.

[15] Though Gorbunov et al. [GVW12] does not use an abstraction as NCER, we observe that their construction can be seen like this.

[16] Though Jafargholi and Wichs [JW16] showed that Yao's garbling scheme is adaptively secure for certain class of circuits like NC^1, we do not know how to prove its adaptive security for all circuits.

[17] Strictly speaking, the SKFE scheme achieves a security notion called key-adaptive security slightly weaker than the adaptive security, in which an adversary cannot make any encryption queries after making the key query. We note that this is sufficient for

see that the key generation algorithm of the scheme is also succinct. As discussed in the previous paragraph, such an SKFE scheme yields an adaptively indistinguishable garbling scheme with succinct online communication/computational complexity.

Other instantiations. The above construction gives a generic construction of an adaptively indistinguishable garbling scheme whose online complexity does not depend on the output length of the circuit being garbled based on any (quasi-)decomposable adaptive garbling scheme. For example, we can also instantiate the construction with adaptive garbling schemes proposed by Hemenway et al. [HJO+16] and Jafargholi and Wichs [JW16] (the latter is Yao's garbling itself) since they are decomposable. As a result, we obtain adaptively indistinguishable garbling schemes for corresponding circuit classes whose online complexity do not depend on output-length. Previously, such garbling schemes are constructed in an ad hoc manner by Jafargholi et al. [JSW17]. On the other hand, our construction is generic, and thus resolves the open question posed by Jafargholi et al. [JSW17].

3 Preliminaries

3.1 Notations

We write $x \xleftarrow{r} X$ to denote that an element x is chosen from a finite set X uniformly at random and $y \leftarrow \mathsf{A}(x; r)$ to denote that the output of an algorithm A on an input x and a randomness r is assigned to y. When there is no need to write the randomness explicitly, we omit it and simply write $y \leftarrow \mathsf{A}(x)$. For strings x and y, $x\|y$ denotes the concatenation of x and y. Throughout this paper, λ denotes a security parameter. We denote by $x[i]$ i-th bit of a string $x \in \{0,1\}^*$. poly denotes an unspecified polynomial. A function $f(\lambda)$ is a negligible function if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. PPT stands for probabilistic polynomial time. Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$.

3.2 Basic Tools

We review basic cryptographic tools.

Pseudorandom functions.

Definition 3.1 (Pseudorandom functions [GGM86]). *For sets $\mathcal{D}$ and $\mathcal{R}$, let $\{\mathsf{F}_K(\cdot) : \mathcal{D} \to \mathcal{R} | K \in \{0,1\}^\lambda\}$ be a family of polynomially computable functions. We say that F is pseudorandom if for any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{\mathsf{F}_K(\cdot)}(1^\lambda) = 1 \mid K \xleftarrow{r} \{0,1\}^\lambda] - \Pr[\mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1 \mid \mathsf{R} \xleftarrow{r} \mathcal{U}]| = \mathsf{negl}(\lambda) \ ,$$

where $\mathcal{U}$ is the set of all functions from $\mathcal{D}$ to $\mathcal{R}$.

Definition 3.2 (Puncturable pseudorandom function [BW13, KPTZ13, BGI14, SW14]). *For sets $\mathcal{D}$ and $\mathcal{R}$, a puncturable pseudorandom function PPRF consists of a tuple of algorithms $(\mathsf{F}, \mathsf{Punc})$ that satisfies the following two conditions.*

Functionality preserving under puncturing: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\mathcal{D}$, and for all $x \in \mathcal{D} \setminus \{x_i\}_{i\in[k]}$, we have $\Pr[\mathsf{F}_K(x) = \mathsf{F}_{K^*}(x) : K \leftarrow \{0,1\}^\lambda, K^* \leftarrow \mathsf{Punc}(K, \{x_i\}_{i\in[k]})] = 1$.*

constructing an adaptively indistinguishable garbling scheme since the adaptive security of a garbling scheme only considers the case where a garbled input is generated after a garbled circuit is generated.

Pseudorandomness at punctured points: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\mathcal{D}$, and any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{A}(K^*, \{\mathsf{F}_K(x_i)\}_{i\in[k]}) = 1] - \Pr[\mathcal{A}(K^*, U^k) = 1] = \mathsf{negl}(\lambda) \ ,$$

where $K \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, $K^* \leftarrow \mathsf{Punc}(K, \{x_i\}_{i\in[k]})$, and U denotes the uniform distribution over $\mathcal{R}$.

Secret key encryption.

Definition 3.3 (Secret key encryption). *A SKE scheme* SKE *is a two tuple* $(\mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms.*

- *The encryption algorithm* Enc*, given a key $K \in \{0,1\}^\lambda$ and a message $m \in \mathcal{M}$, outputs a ciphertext c, where $\mathcal{M}$ is the plaintext space of* SKE*.*
- *The decryption algorithm* Dec*, given a key K and a ciphertext c, outputs a message $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. This algorithm is deterministic.*

Correctness *We require $\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m$ for every $m \in \mathcal{M}$ and key K.*

Security *Let* SKE *be an SKE scheme whose message space is $\mathcal{M}$. We define the security game between a challenger and an adversary $\mathcal{A}$ as follows. Below, let n be a fixed polynomial of λ.*

Initialization *First the challenger selects a challenge bit $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$. Next the challenger generates n keys $K_j \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for every $j \in [n]$ and sends 1^λ to $\mathcal{A}$.*
$\mathcal{A}$ may make polynomially many encryption queries adaptively.

Encryption query *$\mathcal{A}$ sends $(j, m_0, m_1) \in [n] \times \mathcal{M} \times \mathcal{M}$ to the challenger. Then, the challenger returns $c \leftarrow \mathsf{Enc}(K_j, m_{\mathsf{coin}})$.*

Final phase *$\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},n,\mathcal{A}}(\lambda) = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]|.$$

We say that SKE *is secure if for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},n,\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.*

Definition 3.4 (SKE with pseudorandom ciphertext). *Let $\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SKE},n,\mathcal{A}}$ is the same as $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},n,\mathcal{A}}$ except that $\mathcal{M} = \{0,1\}^\ell$ and $\mathcal{A}$ sends $(j, m) \in [n] \times \mathcal{M}$ to the challenger as an encryption query and receives c where $c \leftarrow \mathsf{Enc}(K_j, m)$ if $\mathsf{coin} = 0$, otherwise $c \xleftarrow{\mathsf{r}} \{0,1\}^\ell$. We say that* SKE *is pseudorandom if for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SKE},n,\mathcal{A}}(\lambda) < \mathsf{negl}(\lambda)$.*

Theorem 3.5 ([Gol04]). *Assuming the existence of one-way functions, then there exists SKE with pseudorandom ciphertext.*

The construction above is based on pseudorandom functions. More specifically, $\mathsf{SKE.Enc}(K, \mathsf{m}) \coloneqq (r, \mathsf{PRF}_K(r) \oplus \mathsf{m})$ where $K \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, $\mathsf{PRF} : \{0,1\}^\lambda \to \{0,1\}^{\ell_\mathsf{m}}$, and $r \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$. See the textbook by Goldreich [Gol04, Construction 5.3.9] for more details.

3.3 Identity-Based Encryption

We define identity-based encryption (IBE).

Definition 3.6 (Identity-based encryption). *An IBE scheme* IBE *is a four tuple* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms. Below, let* ID *and* $\mathcal{M}$ *be the identity space and message space of* IBE*, respectively.*

- *The setup algorithm* Setup*, given a security parameter* 1^λ *and the length of identities* 1^n*, outputs a public parameter* MPK *and a master secret key* MSK*.*
- *The key generation algorithm* KG*, given a master secret key* MSK *and identity* $\mathsf{id} \in \{0,1\}^n$*, outputs a user secret key* $\mathsf{sk}_{\mathsf{id}}$*.*
- *The encryption algorithm* Enc*, given a public parameter* MPK*, identity* $\mathsf{id} \in \{0,1\}^n$*, and message* $m \in \mathcal{M}$*, outputs a ciphertext* CT*.*
- *The decryption algorithm* Dec*, given a user secret key* $\mathsf{sk}_{\mathsf{id}}$ *and ciphertext* CT*, outputs a message* $\tilde{m} \in \{\bot\} \cup \mathcal{M}$*.*

Correctness *We require* $\mathsf{Dec}(\mathsf{KG}(\mathsf{MSK}, \mathsf{id}), \mathsf{Enc}(\mathsf{MPK}, \mathsf{id}, m)) = m$ *for every* $m \in \mathcal{M}$*,* $\mathsf{id} \in \{0,1\}^n$*, and* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$*.*

We define indistinguishability against selective ID attacks for IBE.

Definition 3.7 (Selective security for IBE). *Let* n *be a polynomial of* λ *denoting the length of identities. Let* IBE *be an IBE scheme whose message space is* $\mathcal{M}$*. We define the selective security game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *At the beginning of the game,* $\mathcal{A}$ *declares the challenge identity* $\mathsf{id}^* \in \{0,1\}^n$*. The challenger chooses a challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$ *and generates* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$*. Then, the challenger sends* MPK *to* $\mathcal{A}$*.*

   Throughout the game, $\mathcal{A}$ *can make key extraction queries.*

   Extraction queries $\mathcal{A}$ *sends* $\mathsf{id} \in \{0,1\}^n$ *to the challenger. The challenger returns* $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KG}(\mathsf{MSK}, \mathsf{id})$ *to* $\mathcal{A}$ *if* $\mathsf{id} \neq \mathsf{id}^*$*. Otherwise, the challenger returns* $\bot$ *to* $\mathcal{A}$*.*

2. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ *to the challenger. We require that* $|m_0| = |m_1|$*. The challenger generates* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{id}^*, m_{\mathsf{coin}})$ *and sends* CT *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

In this game, we define the advantage of the adversary $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = |\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

We say that IBE *is selectively secure if for any PPT adversary* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{IBE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*.*

3.4 Garbling Schemes

Definition 3.8 (Garbling Scheme). *A garbling scheme* YaoGC *for circuits is a tuple* $(\mathsf{Grbl}, \mathsf{Eval})$ *of two PPT algorithms.*

- *A garbling algorithm* Grbl*, given a security parameter* 1^λ*, a circuit* C *with* n*-bit input, and input labels* $\{\mathsf{label}_{k,b}\}_{k\in[n],b\in\{0,1\}}$*. This algorithm outputs a garbled circuit* $\widetilde{C}$*. In general, labels* $\mathsf{label}_{k,b}$ *for all* $k \in [n]$ *and* $b \in \{0,1\}$ *are uniformly and randomly chosen from* $\{0,1\}^\lambda$*.*
- *The evaluation algorithm, given a garbled circuit* $\widetilde{C}$ *and* n *labels* $\{\mathsf{label}_k\}_{k\in[n]}$*, outputs* y*.*

Correctness: *We require* $\mathsf{Eval}(\widetilde{C}, \{\mathsf{label}_{k,x_k}\}_{k\in[n]}) = C(x)$ *for every* $\lambda \in \mathbb{N}$*, a circuit* C *with* n*-bit input, and* $x \in \{0,1\}^n$*, where* $\widetilde{C} \leftarrow \mathsf{Grbl}(1^\lambda, C, \{\mathsf{label}_{k,b}\}_{k\in[n],b\in\{0,1\}})$ *and* x_k *is the* k*-th bit of* x *for every* $k \in [n]$*.*

Security: *Let* Sim *be a PPT algorithm. We define the following game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$ *and sends security parameter* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends a circuit* C *with* n*-bit input and an input* $x \in \{0,1\}^n$ *to the challenger.*
   - *If* $\mathsf{coin} = 0$*, then the challenger chooses* $\mathsf{label}_{k,b} \leftarrow \{0,1\}^\lambda$ *for all* $k \in [n], b \in \{0,1\}$ *and computes* $\widetilde{C} \leftarrow \mathsf{Grbl}(1^\lambda, C, \{\mathsf{label}_{k,b}\}_{k\in[n],b\in\{0,1\}})$ *and returns* $(\widetilde{C}, \{\mathsf{label}_{k,x[k]}\}_{k\in[n]})$ *to* $\mathcal{A}$*.*
   - *If* $\mathsf{coin} = 1$*, then it chooses* $\mathsf{label}_{k,b} \leftarrow \{0,1\}^\lambda$ *for all* $k \in [n], b \in \{0,1\}$ *and computes* $\widetilde{C} \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, C(x), \{\mathsf{label}_{k,x[k]}\})$*, and returns* $(\widetilde{C}, \{\mathsf{label}_{k,x[k]}\}_{k\in[n]})$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0,1\}$*.*

In this game, we define the advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{YaoGC},\mathcal{A},\mathsf{Sim}}(\lambda) = 2|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

We say that YaoGC *is selectively secure if there exists PPT* Sim *such that for any PPT* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{YaoGC},\mathcal{A},\mathsf{Sim}}(\lambda) = \mathsf{negl}(\lambda)$*.*

Theorem 3.9 ([Yao86]). *If there exists one-way function, there exists a selectively secure garbling scheme for all poly-size circuits.*

Definition 3.10 (Adaptive garbling scheme). *A garbling scheme* AdaGC *is a three tuple* $(\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ *of PPT algorithms.*

- *The circuit garbling algorithm* GbCkt*, given a security parameter* 1^λ *and a circuit* C*, outputs a garbled circuit* $\widetilde{C}$ *and a state* st*.*
- *The input garbling algorithm* GbInp*, given a state* st *and an input* x*, and outputs a garbled input* $\widetilde{x}$*.*
- *The evaluation algorithm* GbEval*, given a garbled circuit* $\widetilde{C}$ *and a garbled input* $\widetilde{x}$*, and outputs a value* y*.*

Correctness: *We require* $\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = C(x)$ *for every* $\lambda \in \mathbb{N}$*, a circuit* C *with* n*-bit input, and* $x \in \{0,1\}^n$*, where* $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GbCkt}(1^\lambda, C)$ *and* $\widetilde{x} \leftarrow \mathsf{GbInp}(\mathsf{st}, x)$*.*

Security: *Let* $\mathsf{Sim} = (\mathsf{SimC}, \mathsf{SimIn})$ *be a tuple PPT algorithms. We define the following game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$ *and sends security parameter* 1^λ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends a circuit* C *with* n*-bit input to the challenger, and the challenger returns* $\widetilde{C}$ *generated as follows:*
   - *If* $\mathsf{coin} = 0$*, then it computes* $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GbCkt}(1^\lambda, C)$.
   - *If* $\mathsf{coin} = 1$*, then it computes* $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{SimC}(1^\lambda, 1^{|C|})$.
3. $\mathcal{A}$ *sends an input* $x \in \{0,1\}^n$ *to the challenger, and the challenger returns* $\widetilde{x}$ *generated as follows:*
   - *If* $\mathsf{coin} = 0$*, then it computes* $\widetilde{x} \leftarrow \mathsf{GbInp}(\mathsf{st}, x)$.
   - *If* $\mathsf{coin} = 1$*, then it computes* $\widetilde{x} \leftarrow \mathsf{SimIn}(\mathsf{st}, C(x))$.
4. $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0,1\}$.

In this game, we define the advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{AdaGC},\mathcal{A},\mathsf{Sim}}(\lambda) = 2|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

We say that AdaGC *is adaptively secure if there exists PPT* Sim *such that for any PPT* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{AdaGC},\mathcal{A},\mathsf{Sim}}(\lambda) = \mathsf{negl}(\lambda)$.

We also define a weaker indistinguishability-based security for adaptive garbling.

Definition 3.11 (Adaptive indistinguishability [JSW17]). *We define the following game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$ *and sends security parameter* 1^λ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends two circuits* (C_0, C_1) *with* n*-bit input to the challenger, and the challenger computes* $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GbCkt}(1^\lambda, C_{\mathsf{coin}})$*, and returns* $\widetilde{C}$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *sends two inputs* $(x_0, x_1) \in \{0,1\}^n$ *to the challenger, and the challenger computes* $\widetilde{x} \leftarrow \mathsf{GbInp}(\mathsf{st}, x_{\mathsf{coin}})$*, and returns* $\widetilde{x}$.
4. $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0,1\}$.

In this game, we define the advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{ind\text{-}adp}}_{\mathsf{AdaGC},\mathcal{A}}(\lambda) = 2|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

We say that a PPT $\mathcal{A}$ *is valid if we have* $C_0(x_0) = C_1(x_1)$*. We say that* AdaGC *is adaptively indistinguishable if for any PPT valid* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{ind\text{-}adp}}_{\mathsf{AdaGC},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

3.5 Functional Encryption

Secret-key functional encryption. We review the definition of secret-key functional encryption (SKFE).

Definition 3.12 (Secret-key functional encryption). *Let $\mathcal{M} := \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a message domain, $\mathcal{Y} := \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ a range, and $\mathcal{F} := \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ a class of functions $f : \mathcal{M} \to \mathcal{Y}$. An* PKFE *scheme for $\mathcal{M}, \mathcal{Y}$, and $\mathcal{F}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

- *The setup algorithm* Setup*, given a security parameter 1^λ, outputs a master secret key* MSK.
- *The key generation algorithm* KG*, given a master secret key* MSK *and $f \in \mathcal{F}$, outputs a decryption key sk_f.*
- *The encryption algorithm* Enc*, given a master secret key* MSK*, and $\mathsf{m} \in \mathcal{M}$, outputs a ciphertext* CT.
- *The decryption algorithm* Dec*, given a decryption key sk_f and a ciphertext* CT*, outputs $y'z \in \mathcal{Y}$.*

Correctness: *We require* $\mathsf{Dec}(\mathsf{KG}(\mathsf{MSK}, f), \mathsf{Enc}(\mathsf{MSK}, \mathsf{m})) = f(\mathsf{m})$ *for every $f \in \mathcal{F}$, $\mathsf{m} \in \mathcal{M}$, and* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.

Definition 3.13 (Adaptive security of SKFE). *Let* SKFE *be an SKFE scheme for $\mathcal{M}, \mathcal{Y}$, and $\mathcal{F}$. Let q be a polynomial of λ. We define the adaptive security game between a challenger and an adversary $\mathcal{A}$ as follows.*

1. *The challenger generates* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$.
2. *$\mathcal{A}$ is given a security parameter 1^λ. $\mathcal{A}$ can make at most q key queries and arbitrarily many challenge queries in any order.*

   Key query. *When $\mathcal{A}$ makes a key query f, the challenger computes* $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f)$ *and returns sk_f to $\mathcal{A}$.*

   Challenge query. *When $\mathcal{A}$ makes a challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, the challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \mathsf{m}_{\mathsf{coin}})$ *and returns* CT *to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs* $\mathsf{coin}' \in \{0,1\}$.

In this game, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) = 2|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

$\mathcal{A}$ is said to be valid if for all its encryption queries $\{(\mathsf{m}_{0,i}, \mathsf{m}_{1,i})\}_{q_c}$ and key queries $\{f_j\}_{j\in[q_k]}$, we have $f_j(\mathsf{m}_{0,i}) = f_j(\mathsf{m}_{1,i})$. We say that SKFE *is q_k-key q_c-ciphertext adaptively secure if for any valid PPT $\mathcal{A}$, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

In addition, we say that SKFE *is adaptively secure and collusion-resistant if q_k and q_c are* a-priori unbounded *polynomial.*

We also consider relaxed security notions which we call key-adaptive security *and* ciphertext-adaptive security.

Key-adaptive security. *We say that $\mathcal{A}$ is key-adaptively valid if all challenge queries are made before its first key query. We say that* SKFE *is q_c-ciphertext and q_k-bounded key-adaptively secure if for any key-adaptively valid PPT $\mathcal{A}$, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$. *If we omit q_c-ciphertext, then we mean q_c is a-priori unbounded polynomial.*

Ciphertext-adaptive security. *We say that $\mathcal{A}$ is ciphertext-adaptively valid if all key queries are made before the challenge query. We say that* SKFE *is q_k-key and q-bounded ciphertext-adaptively secure if for any ciphertext-adaptively valid PPT $\mathcal{A}$, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Public-key functional encryption. We review the definition of public-key functional encryption (PKFE).

Definition 3.14 (Public-key functional encryption). *Let $\mathcal{M} := \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a message domain, $\mathcal{Y} := \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ a range, and $\mathcal{F} := \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ a class of functions $f : \mathcal{M} \to \mathcal{Y}$. An* PKFE *scheme for $\mathcal{M}, \mathcal{Y}$, and $\mathcal{F}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

- *The setup algorithm* Setup*, given a security parameter 1^λ, outputs a public parameter* MPK *and master secret key* MSK.
- *The key generation algorithm* KG*, given a master secret key* MSK *and $f \in \mathcal{F}$, outputs a functional decryption key sk_f.*
- *The encryption algorithm* Enc*, given a public parameter* MPK *and* $\mathsf{m} \in \mathcal{M}$*, outputs a ciphertext* CT.
- *The decryption algorithm* Dec*, given a functional decryption key sk_f and a ciphertext* CT*, outputs $y' \in \mathcal{Y}$.*

Correctness: *We require* $\mathsf{Dec}(\mathsf{KG}(\mathsf{MSK}, f), \mathsf{Enc}(\mathsf{MPK}, \mathsf{m})) = f(\mathsf{m})$ *for every $f \in \mathcal{F}$,* $\mathsf{m} \in \mathcal{M}$*, and* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$.

Next, we introduce adaptive security for PKFE schemes.

Definition 3.15 (q-bounded adaptive security). *Let* PKFE *be a PKFE scheme for $\mathcal{M}$, $\mathcal{Y}$, and $\mathcal{F}$. Let q be a polynomial of λ. We define the adaptive security game between a challenger and an adversary $\mathcal{A}$ as follows.*

1. *The challenger generates* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0, 1\}$. *Then, the challenger sends* MPK *to $\mathcal{A}$.*
2. *$\mathcal{A}$ is given a security parameter 1^λ. $\mathcal{A}$ can make at most q key queries and one challenge query in any order.*

   Key query. *When $\mathcal{A}$ makes a key query f, the challenger computes* $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f)$ *and returns* sk_f *to $\mathcal{A}$.*

   Challenge query. *When $\mathcal{A}$ makes a challenge query* $(\mathsf{m}_0, \mathsf{m}_1)$*, the challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{m}_{\mathsf{coin}})$ *and returns* CT *to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs* $\mathsf{coin}' \in \{0, 1\}$.

In this game, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{PKFE},\mathcal{A}}(\lambda) = 2|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}| = |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \ .$$

$\mathcal{A}$ is said to be valid if for all its key queries $\{f_i\}_{i\in[q]}$ and the challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, we have $f_i(\mathsf{m}_0) = f_i(\mathsf{m}_1)$. We say that PKFE *is q-bounded (or q-key) adaptively secure if for any valid PPT $\mathcal{A}$, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{PKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

In addition, we say that PKFE *is adaptively secure and collusion-resistant if q is an* a-priori unbounded *polynomial.*

Definition 3.16 ((Weakly) Selective Security). *In the game of adaptive security in Definition 3.15, if $\mathcal{A}$ must commit $(\mathsf{m}_0, \mathsf{m}_1)$ at the beginning of the game, that is, before MPK is give, then we say that PKFE is (q-bounded or q-key) selectively secure (and collusion-resistant).*

In addition, we say that PKFE is weakly-selectively secure if $\mathcal{A}$ must commit not only $(\mathsf{m}_0, \mathsf{m}_1)$ but also key queries $(f_1, \ldots, f_q)$ at the beginning of the game.

Next, we define the succinctness for FE.

Definition 3.17 (Succinctness). *In the following, let ℓ_f and ℓ_{m} be the bit-length needed to describe elements of $\mathcal{F}$ and $\mathcal{M}$, respectively.*

Succinct: *We say that PKFE is succinct if the size of the encryption circuit is bounded by $\mathrm{poly}(\lambda, \ell_{\mathsf{m}}, \log \ell_f)$.*

Sublinearly-succinct: *We say that PKFE is sublinearly-succinct if the size of the encryption circuit is bounded by $\ell_f^{\gamma} \cdot \mathrm{poly}(\lambda, \ell_{\mathsf{m}})$, where $\gamma < 1$ is a fixed constant.*

Known Results on Functional Encryption Ananth and Sahai proved the following theorem (See Appendix A for details).

Theorem 3.18 ([AS16]). *If there exists $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE for circuits and $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for (resp. boolean) circuits, then there exists $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$-PKFE for (resp. boolean) circuits.*

Garg and Srinivasan [GS16] proved the following theorem.

Theorem 3.19 ([GS16]). *If there exists $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE for circuits, then there exists $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE for circuits.*

By combining these theorems, we obtain the following theorem.

Theorem 3.20 ([GS16, AS16]). *If there exists $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE for circuits and $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE for (resp. boolean) circuits, then there exists $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$ for (resp. boolean) circuits.*

3.6 Indistinguishability Obfuscation

We review the definition of indistinguishability obfuscation (IO).

Definition 3.21 (Indistinguishability obfuscation [BGI+12, GGH+16]). *A PPT algorithm $i\mathcal{O}$ is an indistinguishability obfuscator (IO) for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following two conditions.*

Functionality: *for all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(1^\lambda, C)] = 1 \ .$$

Indistinguishability: *for any poly-size distinguisher $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: for all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ of the same size and such that $C_0(x) = C_1(x)$ for all inputs x, then*

$$|\Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_0)) = 1\right] - \Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1\right]| = \mathsf{negl}(\lambda) \ .$$

3.7 Somewhere Equivocal Encryption

Definition 3.22 (Somewhere Equivocal Encryption [HJO+16]). *Let s, n, and t be black-length, message length in blocks, and equivocation parameter. A somewhere equivocal encryption scheme consists of the following five algorithms.*

$\mathsf{KeyGen}(1^\lambda) \to \mathsf{sek}$**:** *This algorithm takes as input the security parameter 1^λ, and outputs a key* sek.

$\mathsf{Enc}(\mathsf{sek}, \boldsymbol{m}) \to \boldsymbol{c}$**:** *This algorithm takes as input* sek *and a vector of messages $\boldsymbol{m} = m_1 \| \cdots \| m_n$ where $m_i \in \{0,1\}^s$, and outputs a ciphertext $myvecc$.*

$\mathsf{Dec}(\mathsf{sek}, \boldsymbol{c}) \to \boldsymbol{m}$**:** *This algorithm takes as input* sek *and $\boldsymbol{c}$, and outputs $\boldsymbol{m} = m_1 \| \cdots \| m_n$.*

$\mathsf{SimEnc}(\{m_i\}_{i \notin I}, I) \to (\mathsf{st}, \widetilde{\boldsymbol{c}})$**:** *This algorithm takes as input a set of indices $I \subseteq [n]$ and a vector of messages $\{m_i\}_{i \notin I}$, and outputs a ciphertext $\widetilde{\mathbf{c}}$ and a state information* st.

$\mathsf{SimKey}(\mathsf{st}, \{m_i\}_{i \in I}) \to \widetilde{\mathsf{sek}}$**:** *This algorithm takes as input* st *and $\{m_i\}_{i \in I}$, and outputs a simulation* $\widetilde{\mathsf{sek}}$.

These algorithms satisfy the following three properties.

Correctness. *For every* $\mathsf{sek} \leftarrow \mathsf{KeyGen}(1^\lambda)$*, for every $\boldsymbol{m} \in \{0,1\}^{sn}$, it holds that*

$$\Pr[\mathsf{Dec}(\mathsf{sek}, \boldsymbol{c}) \mid \boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{sek}, \boldsymbol{m})] = 1.$$

Simulatin with No Holes. *For any $\boldsymbol{m}$, it holds that*

$$(\mathsf{sek}, \boldsymbol{c}) \overset{\mathsf{p}}{\approx} (\mathsf{sek}', \boldsymbol{c}'),$$

where $\mathsf{sek} \leftarrow \mathsf{KeyGen}(1^\lambda)$*,* $\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{sek}, \boldsymbol{m})$*,* $(\mathsf{st}, \boldsymbol{c}') \leftarrow \mathsf{SimEnc}(\boldsymbol{m}, \emptyset)$*, and* $\mathsf{sek}' \leftarrow \mathsf{SimKey}(\mathsf{st}, \emptyset)$ *and $\overset{\mathsf{p}}{\approx}$ denotes that the two distributions are indentically distributed.*

Security. *For any PPT adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathsf{Exp}^{\mathsf{sw\text{-}eqenc}}_{\mathcal{A},\Sigma}(1^\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sw\text{-}eqenc}}_{\mathcal{A},\Sigma}(1^\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda),$$

where the experiment $Pr[\mathsf{Exp}^{\mathsf{sw\text{-}eqenc}}_{\mathcal{A},\Sigma}(1^\lambda, \mathsf{coin})$ is defined as follows.

1. *$\mathcal{A}$ takes as input 1^λ and outputs a set of indices $I \subseteq [n]$ such that $|I| < t$, a vector $\{m_i\}_{i \notin I}$, and a challenge $j \in [n] \setminus I$. Let $\widehat{I} := I \cup \{j\}$.*
   - *If* $\mathsf{coin} = 0$*, then the challenger computes* $(\mathsf{st}, \boldsymbol{c}) \leftarrow \mathsf{SimEnc}(\{m_i\}_{i \notin I}, I)$.
   - *If* $\mathsf{coin} = 1$*, then the challenger computes* $(\mathsf{st}, \boldsymbol{c}) \leftarrow \mathsf{SimEnc}(\{m_i\}_{i \notin \widehat{I}}, \widehat{I})$.
2. *The challenger sends $\boldsymbol{c}$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ returns the set of remaining messages $\{m_i\}_{i \in I}$.*
   - *If* $\mathsf{coin} = 0$*, then the challenger computes* $\widetilde{\mathsf{sek}} \leftarrow \mathsf{SimKey}(\mathsf{st}, \{m_i\}_{i \in I})$.
   - *If* $\mathsf{coin} = 1$*, then the challenger computes* $\widetilde{\mathsf{sek}} \leftarrow \mathsf{SimKey}(\mathsf{st}, \{m_i\}_{i \in \widehat{I}})$.
4. *The challenger sends* $\widetilde{\mathsf{sek}}$ *to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs* $\mathsf{coin}' \in \{0,1\}$ *and this experiment outputs* coin'.

Theorem 3.23 ([HJO+16]). *If there exists one-way functions, then there exists a somewhere equivocal encryption scheme for any polynomial message-length n, block-length s, and equivocation parameter t whose key length is $t \cdot s \cdot \mathrm{poly}(\lambda)$ bits and ciphertext length is $n \cdot s \cdot \mathrm{poly}(\lambda)$ bits.*

4 Selective-Database Laconic OT from PKFE

In this section, we show how to construct (updatable) laconic OT satisfying a security notion we call selective-database security from sublinearly succinct PKFE. We first show that by using IO, we can construct selective-database laconic OT with the compression factor 2. Then, we show that we can replace IO in our construction with sublinearly succinct PKFE by relying on the result of Liu and Zhandry [LZ17]. Finally, we transform our selective-database laconic OT with compression factor 2 into updatable one based on the transformation using Merkle tree proposed by Cho et al. [CDG+17].

4.1 Definition of Selective-Database Laconic OT

We use (updatable) laconic OT proposed by Cho et al [CDG+17]. However, the security level that we need in this work is slightly different from those by Cho et al, Garg and Srinivasan [GS18], and Ananth and Lombardi [AL18].

Definition 4.1 (Selective-Database Laconic OT). *A laconic OT (LOT) A laconic OT protocol consists of four algorithms.*

$\mathsf{Gen}(1^\lambda) \to \mathsf{crs}$**:** *This algorithm takes as input the security parameter and outputs a common reference string* crs*.*

$\mathsf{Hash}(\mathsf{crs}, D) \to (d, \widehat{D})$**:** *This algorithm takes as input* crs *and a database* $D \in \{0,1\}^*$ *and outputs a digest* d *of* D *and a state* $\widehat{D}$*.*

$\mathsf{Send}(\mathsf{crs}, d, L, m_0, m_1) \to e$**:** *This algorithm takes as input* crs*,* d*, a database location* $L \in \mathbb{N}$*, and two messages* m_0 *and* m_1 *of length* $p(\lambda)$*, and outputs a ciphertext* e*.*

$\mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, e, L) \to m$**:** *This is a RAM algorithm with random read access to* $\widehat{D}$*. It takes as input* crs*,* e*, and* $L \in \mathbb{N}$*, and outputs a message* m*.*

These algorithms satisfy the following three properties.

Correctness. *For any database* D *of size at most* $M = \mathrm{poly}(\lambda)$*, any memory location* $L \in [M]$*, any pair of messages* $(m_0, m_1) \in \{0,1\}^{p(\lambda)}$*, it holds that*

$$\Pr\left[m = m_{D[L]} \;\middle|\; \begin{array}{rl} \mathsf{crs} & \leftarrow \mathsf{Gen}(1^\lambda), \\ (d, \widehat{D}) & \leftarrow \mathsf{Hash}(\mathsf{crs}, D), \\ e & \leftarrow \mathsf{Send}(\mathsf{crs}, d, L, m_0, m_1), \\ m & \leftarrow \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, e, L) \end{array} \right] = 1.$$

Selective-Database Adaptive-Message Sender Privacy against Semi-Honest Receivers. *There exists a PPT simulator* Sim *that satisfies the following.*

$$|\Pr[\mathsf{Real}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda) = 1] - \Pr[\mathsf{Sim}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Real}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$ *and* $\mathsf{Sim}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$ *are defined as follows.*

$\underline{\mathsf{Real}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)}$

1. $(D, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
2. $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$,
3. $d \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$,
4. $(L, m_0, m_1, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{crs})$,
5. $e \leftarrow \mathsf{Send}(\mathsf{crs}, d, L, m_0, m_1)$,
6. $b' \leftarrow \mathcal{A}(\mathsf{crs}, e, \mathsf{st}')$

$\underline{\mathsf{Sim}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)}$

1. $(D, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$,
2. $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$,
3. $d \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$,
4. $(L, m_0, m_1, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{crs})$,
5. $e \leftarrow \mathsf{Sim}(\mathsf{crs}, D, L, m_{D[L]})$,
6. $b' \leftarrow \mathcal{A}(\mathsf{crs}, e, \mathsf{st}')$

where $|D| = M = \mathrm{poly}(\lambda)$, $L \in [M]$, and $m_0, m_1 \in \{0,1\}^{p(\lambda)}$. We call this security selective-database sender privacy *for short in this paper.*

Efficiency. *We require that $|d|$ is bounded by a fixed polynomial in λ independent of $|D|$, the running time of* Hash *is $|D| \cdot \mathrm{poly}(\log |D|, \lambda)$, and the running time of* Send *and* Receive *are $\mathrm{poly}(\log |D|, \lambda)$.*

Definition 4.2 (Selective-Database Updatable Laconic OT). *An updatable laconic OT protocol is not only a laconic OT but also has additional two algorithms.*

$\mathsf{SendWrite}(\mathsf{crs}, d, L, b, \{m_{j,0}, m_{j,1}\}_{j=1}^{|d|}) \to e_w$**:** *This algorithm takes as input* crs*, d, L, a bit $b \in \{0,1\}$ to be written in the database, and $|d|$ pairs of messages $\{m_{j,0}.m_{j,1}\}_{j=1}^{|d|}$, where each $m_{j,c}$ is of length $p(\lambda)$, and outputs a ciphertext e_w.*

$\mathsf{ReceiveWrite}^{\widehat{D}}(\mathsf{crs}, L, b, e_w) \to \{m_j\}_{j\in[d]}$**:** *This is a RAM algorithm with random read and write access to $\widehat{D}$. It takes as input* crs*, $L \in \mathbb{N}$, $b \in \{0,1\}$, and e_w, and outputs messages $\{m_j\}_{j\in[|d|]}$.*

The additional two algorithms satisfy the following properties.

Correctness with regard to Write. *For any database D of size at most $M = \mathrm{poly}(\lambda)$, any memory location $L \in [M]$, any bit $b \in \{0,1\}$, any D^* such that D^* is identical to D except that $D^*[L] = b$, any messages $\{(m_{j,0}, m_{j,1})\}_{j=1}^{|d|}$ where $m_{j,b} \in \{0,1\}^{p(\lambda)}$, it holds that*

$$\Pr\left[\forall j\ m'_j = m_{j,d^*[j]} \;\middle|\; \begin{array}{rl} \mathsf{crs} & \leftarrow \mathsf{Gen}(1^\lambda), \\ (d, \widehat{D}) & \leftarrow \mathsf{Hash}(\mathsf{crs}, D), \\ (d^*, \widehat{D}^*) & \leftarrow \mathsf{Hash}(\mathsf{crs}, D^*), \\ e_w & \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, L, b, \{(m_{j,0}, m_{j,1})\}_{j=1}^{|d|}), \\ \{m'_j\}_{j=1}^{|d|} & \leftarrow \mathsf{ReceiveWrite}^{\widehat{D}}(\mathsf{crs}, L, e_w) \end{array}\right] = 1.$$

Selective-Database and Adaptive-Message Sender Privacy against Semi-Honest Receivers with regard to Writes. *There exists a PPT simulator* SimWrite *that satisifes the following.*

$$|\Pr[\mathsf{Real}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda) = 1] - \Pr[\mathsf{Sim}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda) = 1]| \le \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Real}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)$ and $\mathsf{Sim}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)$ are defined as follows.

$\underline{\mathsf{Real}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)}$

1. $(D, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$,
2. $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$,
3. $d \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$,
4. $(L, b, \{m_{j,0}, m_{j,1}\}_{j=1}^{|d|}, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{crs})$,
5.
6. $e \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, L, b, \{m_{j,0}, m_{j,1}\}_{j=1}^{|d|})$,
7. $b' \leftarrow \mathcal{A}(\mathsf{crs}, e, \mathsf{st}')$

$\underline{\mathsf{Sim}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)}$

1. $(D, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$,
2. $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$,
3. $d \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$,
4. $(L, b, \{m_{j,0}, m_{j,1}\}_{j=1}^{|d|}, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{crs})$,
5. $(d^*, \widehat{D}^*) \leftarrow \mathsf{Hash}(\mathsf{crs}, D^*)$ *where* D^* *is identical to* D *except that* $D^*[L] = b$,
6. $e \leftarrow \mathsf{SimWrite}(\mathsf{crs}, D, L, b, \{m_{j,d^*[j]}\}_{j\in[|d|]})$,
7. $b' \leftarrow \mathcal{A}(\mathsf{crs}, e, \mathsf{st}')$

where $|D| = M = \mathrm{poly}(\lambda)$, $L \in [M]$, *and* $m_{j,0}, m_{j,1} \in \{0,1\}^{p(\lambda)}$ *for all* $j \in [|d|]$. *We call this security* selective-database sender privacy for writes *for short in this paper.*

Efficiency. *We require that the running time of* SendWrite *and* ReceiveWrite *are* $\mathrm{poly}(\log |D|, \lambda)$.

4.2 Selective-Database Laconic OT with Compression Factor 2 from IO

We show how to construct laconic OT from IO in this subsection. Let $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.KG}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ be an IBE scheme. For simplicity, we assume that the randomness space of IBE.Setup is $\{0,1\}^\lambda$ and IBE.KG is deterministic.[18] We let the length of a master public-key of IBE be bounded by some fixed polynomial $\mathrm{poly}_{\mathsf{MPK}}(\lambda, n)$, where n is the length of identities. Then, there exists a polynomial $s = \mathrm{poly}(\lambda)$ such that $s \geq \mathrm{poly}_{\mathsf{MPK}}(\lambda, \log s + 2)$. Let $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc})$ be a puncturable PRF whose domain and range are $\{0,1\}^{2s}$ and $\{0,1\}^\lambda$, respectively. Let $i\mathcal{O}$ be an IO.

We construct an LOT protocol $\ell\mathsf{OT} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ whose hash algorithm Hash hashes a $2s$ bit database to a digest of $\mathrm{poly}_{\mathsf{MPK}}(\lambda, \log s + 2) \leq s$ bit. Thus, our construction achieves compression factor 2. In the construction, for an integer $i \in [2s]$, $\mathsf{str}(i)$ denotes the bit representation of i.

$\mathsf{Gen}(1^\lambda)$:

1. Generates $K \xleftarrow{r} \{0,1\}^\lambda$.
2. Computes $\mathsf{crs} \leftarrow i\mathcal{O}\left(1^\lambda, \mathsf{SetupKG}[K]\right)$. The circuit SetupKG is defined in Figure 2.
3. Outputs crs.

$\mathsf{Hash}(\mathsf{crs}, D)$:

1. Outputs $\left(d, \widehat{D}\right) \leftarrow \mathsf{crs}(D)$.

$\mathsf{Send}(\mathsf{crs}, d, L, \mathsf{m}_0, \mathsf{m}_1)$:

1. Parses $\mathsf{MPK} \leftarrow d$.
2. For $\alpha \in \{0,1\}$, computes $\mathsf{CT}_\alpha \leftarrow \mathsf{IBE.Enc}(\mathsf{MPK}, \mathsf{str}(L)\|\alpha, \mathsf{m}_\alpha)$.
3. Outputs $e := (\mathsf{CT}_0, \mathsf{CT}_1)$.

$\mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, e, L)$:

1. Sets $\widehat{D} := \left(D, \{\mathsf{sk}_i\}_{i\in[2s]}\right)$.
2. Parses $e \leftarrow (\mathsf{CT}_0, \mathsf{CT}_1)$.
3. Outputs $m \leftarrow \mathsf{IBE.Dec}\left(\mathsf{sk}_L, \mathsf{CT}_{D[L]}\right)$.

Setup and key generation circuit $\mathsf{SetupKG}[K]$

Hardwired: puncturable PRF key K.

Input: $D \in \{0,1\}^{2s}$.

Padding: circuit is padded to size $\mathsf{pad} := \max(|\mathsf{SetupKG}|, |\mathsf{SetupKG}^*|)$, where $\mathsf{SetupKG}^*$ is defined in the security proof.

1. Computes $r \leftarrow \mathsf{F}_K(D)$.
2. Computes $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^{\log s+2}; r)$.
3. For every $i \in [2s]$, computes $\mathsf{sk}_i \leftarrow \mathsf{IBE.KG}\left(\mathsf{MSK}, \mathsf{str}(i)\|D[i]\right)$.
4. Outputs $d := \mathsf{MPK}$ and $\widehat{D} := \left(D, \{\mathsf{sk}_i\}_{i\in[2s]}\right)$.

Figure 2: The description of $\mathsf{SetupKG}$.

Theorem 4.3. *Let* IBE *be a selectively secure IBE scheme and* PPRF *be a puncturable PRF. Let* $i\mathcal{O}$ *be IO. Then,* $\ell\mathsf{OT}$ *be a selective-database laconic OT.*

Proof. The correctness of $\ell\mathsf{OT}$ follows from that of IBE and the functionality of $i\mathcal{O}$. Below, we show the selective-database sender privacy against semi-honest receivers of $\ell\mathsf{OT}$.

Let $\mathcal{A}$ be a PPT adversary that attacks $\ell\mathsf{OT}$. We consider the following sequence of experiments. Below, let $\mathtt{T}_k$ be the event that $\mathcal{A}$ outputs 1 in Exp k.

Exp 0**:** This experiment is $\mathsf{Real}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$. The details of this experiment is as follows.

1. $(D^*, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$, where $|D| = 2s$,
2. $K \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\mathsf{crs} \leftarrow i\mathcal{O}\left(1^\lambda, \mathsf{SetupKG}[K]\right)$,
3. $\mathsf{MPK}^* = d^* \leftarrow \mathsf{crs}(D^*)$,
4. $(L^*, \mathsf{m}_0^*, \mathsf{m}_1^*, \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{crs})$,
5. $\mathsf{CT}_\alpha \leftarrow \mathsf{IBE.Enc}(\mathsf{MPK}^*, \mathsf{str}(L^*)\|\alpha, m_\alpha^*)$ for $\alpha \in \{0,1\}$,
6. $b' \leftarrow \mathcal{A}(\mathsf{crs}, e = (\mathsf{CT}_0, \mathsf{CT}_1), \mathsf{st}')$

Exp 1**:** This experiment is the same as Exp 0 except that crs is generated by obfuscating the circuit $\mathsf{SetupKG}^*$ shown in Figure 3 using $i\mathcal{O}$. $\mathsf{SetupKG}^*$ has hardwired $K\{D^*\}, \mathsf{MPK}^*$, and $\{\mathsf{sk}_i^*\}_{i\in[2s]}$. They are generated by $K\{D^*\} \leftarrow \mathsf{Punc}\,(K, D^*)$, $(\mathsf{MPK}^*, \mathsf{MSK}^*) \leftarrow \mathsf{IBE.Setup}(1^\lambda; r^*)$, and $\mathsf{sk}_i^* \leftarrow \mathsf{IBE.KG}\,(\mathsf{MSK}^*, \mathsf{str}(i)\|D^*[i])$ for every $i \in [2s]$, where $r^* \leftarrow \mathsf{F}_K\,(D^*)$.

We see that $\mathsf{SetupKG}$ and $\mathsf{SetupKG}^*$ are functionally equivalent. Thus, from the indistinguishability property of $i\mathcal{O}$, $|\Pr[\mathtt{T}_0] - \Pr[\mathtt{T}_1]| = \mathsf{negl}(\lambda)$.

Exp 2**:** This experiment is the same as Exp 1 except that r^* is generated as a truly random string.

From the pseudorandomness at punctured points of PPRF, $|\Pr[\mathtt{T}_1] - \Pr[\mathtt{T}_2]| = \mathsf{negl}(\lambda)$.

Exp 3**:** This experiment is the same as Exp 2 except that $\mathsf{CT}_{1-D^*[L^*]}$ is generated by $\mathsf{CT}_{1-D^*[L^*]} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{MPK}^*, \mathsf{str}(L^*)\|1 - D^*[L^*], 0^\lambda\right)$.

[18] We can always modify any IBE scheme so that it satisfies these two conditions by using PRF.

Setup and key generation circuit $\mathsf{SetupKG}^*\left[D^*, K\{D^*\}, \mathsf{MPK}^*, \{\mathsf{sk}_i^*\}_{i\in[2s]}\right]$.

Hardwired: database D^*, punctured PRF key $K\{D^*\}$, master public-key MPK^*, and secret keys $\{\mathsf{sk}_i^*\}_{i\in[2s]}$.
Input: $D \in \{0,1\}^{2s}$.
Padding: circuit is padded to size $\mathsf{pad} := \max(|\mathsf{SetupKG}|, |\mathsf{SetupKG}^*|)$.

1. If $D = D^*$, then return $d^* := \mathsf{MPK}^*$ and $\widehat{D}^* := \left(D^*, \{\mathsf{sk}_i^*\}_{i\in[2s]}\right)$. Otherwise, compute as follows.
2. Compute $r \leftarrow \mathsf{F}_K(D)$.
3. Compute $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^{\log s+2}; r)$.
4. For every $i \in [2s]$, compute $\mathsf{sk}_i \leftarrow \mathsf{IBE.KG}\left(\mathsf{MSK}, \mathsf{str}(i)\|D[i]\right)$.
5. Return $d := \mathsf{MPK}$ and $\widehat{D} := \left(D, \{\mathsf{sk}_i\}_{i\in[2s]}\right)$.

Figure 3: The description of $\mathsf{SetupKG}^*$.

By the previous change, MPK^* is generated by using truly randomness instead of $\mathsf{F}_K(D^*)$. Moreover, $\widehat{D}^*$ does not include $\overline{\mathsf{sk}}_{L^*}^* \leftarrow \mathsf{IBE.KG}(\mathsf{MSK}^*, \mathsf{str}(L^*)\|1 - D^*[L^*])$ while $\widehat{D}^*$ includes $\mathsf{sk}_{L^*}^* \leftarrow \mathsf{IBE.KG}(\mathsf{MSK}^*, \mathsf{str}(L^*)\|D^*[L^*])$. Then, based on the selective security of IBE, we can prove $|\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]| = \mathsf{negl}(\lambda)$. More specifically, using $\mathcal{A}$, we can construct the following adversary $\mathcal{B}$ that attacks the selective security of IBE.

1. $\mathcal{B}$ first invokes $\mathcal{A}(1^\lambda)$ and obtains (D^*, st). $\mathcal{B}$ next randomly picks $L^* \xleftarrow{\mathsf{r}} [2s]$. Then, $\mathcal{B}$ sends $\mathsf{str}(L^*)\|1 - D^*[L^*]$ to the challenger as its challenge identity and obtains MPK^*.
2. $\mathcal{B}$ generates a CRS crs as follows. $\mathcal{B}$ first generates $K \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and computes $K\{D^*\} \leftarrow \mathsf{Punc}\left(K, D^*\right)$. Next, $\mathcal{B}$ makes a key extraction query $\mathsf{str}(i)\|D^*[i]$ and obtains sk_i^* for every $i \in [2s]$. Then, $\mathcal{B}$ generates $\mathsf{crs} \leftarrow i\mathcal{O}\left(1^\lambda, \mathsf{SetupKG}^*[D^*, K\{D^*\}, \mathsf{MPK}^*, \{\mathsf{sk}_i^*\}_{i\in[2s]}]\right)$.
3. $\mathcal{B}$ invokes $\mathcal{A}(\mathsf{st}, \mathsf{crs})$ and obtains $(L', \mathsf{m}_0^*, \mathsf{m}_1^*, \mathsf{st}')$. If, $L^* \neq L'$, $\mathcal{B}$ outputs $\mathsf{coin}' \xleftarrow{\mathsf{r}} \{0,1\}$ and terminates. Otherwise, $\mathcal{B}$ runs as follows.
4. $\mathcal{B}$ sends $(\mathsf{m}_{1-D[L^*]}, 0^\lambda)$ as its challenge messages and obtains the answer $\mathsf{CT}_{1-D[L^*]}$. $\mathcal{B}$ also generates $\mathsf{CT}_{D[L^*]} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{MPK}^*, \mathsf{str}(L^*)\|D^*[L^*], \mathsf{m}_{D^*[L^*]}\right)$ and sets $e := (\mathsf{CT}_0, \mathsf{CT}_1)$.
5. $\mathcal{B}$ invokes $b' \leftarrow \mathcal{A}(\mathsf{crs}, e, \mathsf{st}')$, outputs $\mathsf{coin}' := b'$, and terminates.

Let coin be the challenge bit in the selective security game for IBE. We can estimate the advantage of $\mathcal{B}$ as

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sel}}_{\mathsf{IBE},\mathcal{B}}(\lambda) &= |\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]| \\ &= \frac{1}{2s}|\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0 \wedge L^* = L'] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1 \wedge L^* = L']| \ .\end{aligned}$$

If $\mathsf{coin} = 0$ and $L^* = L'$ hold, $\mathcal{B}$ perfectly simulates Exp 2 for $\mathcal{A}$. If $\mathsf{coin} = 1$ and $L^* = L'$ hods, $\mathcal{B}$ perfectly simulates Exp 3 for $\mathcal{A}$. Moreover, when $L^* = L'$, $\mathcal{B}$ outputs if and only if $\mathcal{A}$ outputs 1. Thus, we have $|\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0 \wedge L^* = L'] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1 \wedge L^* = L']| = |\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]|$. Therefore, we obtain $|\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]| = 2s \cdot \mathsf{Adv}^{\mathsf{sel}}_{\mathsf{IBE},\mathcal{B}}(\lambda)$. Since IBE is selectively secure, we also obtain $|\Pr[\mathtt{T}_2] - \Pr[\mathtt{T}_3]| = \mathsf{negl}(\lambda)$.

Exp 4**:** We undo the change from Exp 1 to Exp 2. Concretely, this experiment is the same as Exp 3 except that r^* is generated by $r^* \leftarrow \mathsf{F}_K(D^*)$.

From the pseudorandomness at punctured points of PPRF, $|\Pr[\mathtt{T}_3] - \Pr[\mathtt{T}_4]| = \mathsf{negl}(\lambda)$.

Exp 5**:** We undo the change from Exp 0 to Exp 1. Concretely, this experiment is the same as Exp 4 except that crs is generated by obfuscating SetupKG using $i\mathcal{O}$.

By using the indistinguishability property of $i\mathcal{O}$ again, we can prove $|\Pr[\mathtt{T}_4] - \Pr[\mathtt{T}_5]| = \mathsf{negl}(\lambda)$.

In Exp 5, crs is generated in exactly the same way as Gen. Moreover, there exists a PPT simulator Sim that given crs, D^*, L^*, and $\mathsf{m}^*_{D[L]}$, generates $e = (\mathsf{CT}_0, \mathsf{CT}_1)$ exactly in the same way as Exp 5. The description of Sim is as follows.

$\mathsf{Sim}\left(\mathsf{crs}, D^*, L^*, \mathsf{m}^*_{D^*[L^*]}\right)$:

- Compute $\left(\mathsf{MPK}^*, \widehat{D}^*\right) \leftarrow \mathsf{crs}(D^*)$.
- Compute $\mathsf{CT}_{D^*[L^*]} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{MPK}^*, \mathsf{str}(L^*) \| D^*[L^*], \mathsf{m}^*_{D^*[L^*]}\right)$
- Compute $\mathsf{CT}_{1-D^*[L^*]} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{MPK}^*, \mathsf{str}(L^*) \| 1 - D^*[L^*], 0^\lambda\right)$.
- Return $e := (\mathsf{CT}_0, \mathsf{CT}_1)$.

Therefore, Exp 5 corresponds to $\mathsf{Sim}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$ in which the above Sim is used. From these, we see that $\ell\mathsf{OT}$ satisfies selective-database sender privacy against semi-honest receivers. ■

4.3 Replacing IO with sublinearly Succinct PKFE

IO in our construction can be replaced with sublinearly succinct PKFE by relying on the result of Liu and Zhandry [LZ17]. Liu and Zhandry showed we can replace IO with *decomposable obfuscation (dO)* that can be based on sublinearly succinct PKFE if the circuit pair to be obfuscated satisfies some condition by generalizing previous works [GPS16, GPSZ17, GS16]. Roughly speaking, they showed that if there is a polynomial size "witness" for the functional equivalence of a circuit pair to be obfuscated, IO can be replaced with dO. One particular situation where this condition is satisfied is that in the security proof we modify a circuit to be obfuscated so that it outputs a hard-wired value for a single input and otherwise it runs in the same way as the original one. More formally, we obtain the following theorem as a special case of the result by Liu and Zhandry.

Theorem 4.4 ([LZ17]). *Let $C'(x, r)$ be a circuit. Let $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc})$ be a punctured PRF and $K \in \{0,1\}^\lambda$. Let Punc be deterministic. We define a circuit C_K as $C_K(x) = C'(x, \mathsf{F}_K(x))$. Moreover, we define a circuit C^* as*

$$C^*_{x^*,K^*,y^*}(x) = \begin{cases} y^* & (x = x^*) \\ C'\left(x, \mathsf{F}_{K^*}(x)\right) & (otherwise) \end{cases},$$

where x^*, $K^* \leftarrow \mathsf{Punc}(K, x^*)$, and $y^* = C(x^*)$ are hardwired into C^*. C_K and $C^*_{x^*,K^*,y^*}$ are parameterized by K and x^*, and they are functionally equivalent for all K and x^*.

Then, assuming $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$-PKFE, there exists a special type of punctured PRF and decomposable obfuscation whose indistinguishability property holds for each pair of circuits $\{(C_K, C^*_{x^*,K^*,y^*})\}_{K,x^*}$ by implementing them using the PRF.

In the above theorem, "a special type of punctured PRF" is a primitive called decomposing compatible PRF by Liu and Zhandry. Decomposing compatible PRF can be constructed from one-way functions via the construction proposed by Goldreich et al. [GGM86], and thus its existence is implied by that of PKFE. See Section 2.1 or the paper by Liu and Zhandry [LZ17] for details.

In the construction of selective-database laconic OT based on IO in Section 4.2, we apply IO for a pair of circuits SetupKG and SetupKG*. We see that when we apply IO to these circuits, they have exactly the same functional relationship as C and C^* in Theorem 4.4. That is, we obtain the following.

Lemma 4.5. *Cricuits* $\mathsf{SetupKG}[K]$ *and* $\mathsf{SetupKG}^*[D^*, K\{D^*\}, \mathsf{MPK}^*, \{\mathsf{sk}_i^*\}_{i\in[2s]}]$ *in Section 4.2 fall into the circuit class* C_K *and* $C^*_{x^*,K^*,y^*}$ *defined in Theorem 4.4.*

Therefore, from Theorem 4.4 and Lemma 4.5, IO that is needed in our construction of selective-database laconic OT in Section 4.2 can be instantiated based on sublinearly succinct PKFE.

Moreover, selectively secure IBE can be constructed from sublinearly succinct PKFE [GS16], and puncturable PRF can be based on one-way functions. Thus, we obtain the following theorem.

Theorem 4.6. *Assume that there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits. Then, there exists selective-database laconic OT with compression factor* 2*.*

4.4 From Non-Updatable to Updatable

Cho et al. [CDG+17] showed we could bootstrap a laconic OT with the compression factor 2 into an updatable laconic OT with arbitrary compression factor using a garbling scheme and Merkle hash tree. Their bootstrapping method considers laconic OT that satisfies a weak security notion where in addition to the challenge database, the challenge location and messages are also fixed at the beginning of the security game. As Ananth and Lombardi [AL18] pointed out, if we use selective-database laconic OT as a building block for the bootstrapping method, then we have to use a minor variant of the method to obtain selective-database updatable laconic OT (the original bootstrapping method is not sufficient for us). More specifically, we have to sample fresh crs_j for each depth j of the Merkle hash tree in the bootstrapping method. We use this variant since our laconic OT is selective-database secure. That is, we have the following theorem.

Theorem 4.7 ([CDG+17, AL18]). *Assume that there exists selective-database laconic OT with the compression factor* 2*. Then, there exists selective-database updatable laconic OT with arbitrary compression factor.*

By combining Theorems 4.6 and 4.7, we obtain the following theorem.

Theorem 4.8. *Assume that there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE. Then, there exists selective-database updatable laconic OT with arbitrary compression factor.*

5 Adaptive Garbling from Selective-Database Laconic OT

In this section, we present an adaptive garbling scheme with nearly optimal online communication/computational complexity based on selective-database updatable LOT. Garg and Srinivasan presented such an adaptive garbling scheme based on *adaptively secure* updatable LOT [GS18], which is instantiated by concrete assumptions such as CDH [CDG+17, DGHM18, BLSV18]. However, we cannot directly use their adaptive garbling scheme due to the following two reasons.

1. Our goal in this section is achieving adaptive garbling scheme from succinct PKFE (i.e., we do not rely on any specific assumption such as the CDH assumption).

2. The updatable LOT protocol presented in Section 4 is *selective-database* updatable LOT.

We will show that we can achieve an adaptive garbling scheme with nearly optimal online communication/computational complexity from *selective-database* updatable LOT in the rest of this section.

5.1 Description of Our Adaptive Garbling Scheme

In this section, we present our adaptive garbling scheme and properties that it satisfies.

Theorem 5.1. *If there exist selective-database updatable LOT, somewhere equivocal encryption, and selectively secure garbled circuits, then there exists an adaptively secure garbling scheme for circuits with online communication complexity* $n + m + \mathrm{poly}(\lambda, \log |C|)$ *and online computational complexity* $O(n + m) + \mathrm{poly}(\lambda, \log |C|)$.

From this theorem and Theorems 3.9, 3.23 and 4.8, we obtain the following theorem.

Theorem 5.2. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE, then there exists an adaptively secure garbling scheme for circuits with online communication complexity* $n + m + \mathrm{poly}(\lambda, \log |C|)$ *and online computational complexity* $O(n + m) + \mathrm{poly}(\lambda, \log |C|)$.

Conventions. Without loss of generality, we assume that circuits consist of only NAND gates. Let n, m, and $N - n$ be the input length, output length, and the number of NAND gates of the circuit. An index is assigned to each input and gate. That is, from 1 to n are input wires, from $n + 1$ to $N - m$ are intermediate NAND gates, and $N - m + 1$ to N are output gates of the circuit. Note that a gate whose inputs come from gate i and j has an index greater than i and j. Each gate $g \in [n + 1, N]$ is represented by a pair $(i, j) \in [g - 1] \times [g - 1]$. That is, the inputs of g is outputs of gates i and j. In this section, we use r_i, x_i, and y_i instead of $r[i]$, $x[i]$, and $y[i]$ to mean the i-th bit of r, x, and y, respectively for notational simplicity.

A variant of GS18 garbling scheme. We prove Theorem 5.1 in the rest of this section. First, we describe our adaptive garbling scheme. We put red underlines at different points from the adaptive garbling scheme by Garg and Srinivasan [GS18]. Let $\Sigma := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SimEnc}, \mathsf{SimKey})$, $\mathsf{GC} := (\mathsf{GC.Grbl}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$, and $\Pi := (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive}, \mathsf{SendWrite}, \mathsf{ReceiveWrite})$ be a somewhere equivocal encryption scheme, a (selectively secure) garbling scheme, and an updatable LOT protocol, respectively. Our adaptive garbling scheme $\mathsf{adGC}'_{\mathsf{gs}} := (\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ is as follows.

$\mathsf{GbCkt}(1^\lambda, C)$**:** This algorithm garbles a circuit $C : \{0,1\}^n \to \{0,1\}^m$ as follows.

1. Generates $\mathsf{sek} \leftarrow \mathsf{KeyGen}(1^\lambda)$, and chooses $r \leftarrow \{0,1\}^N$.
2. Generates $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.
3. Chooses $\mathsf{label}^g_{k,b} \leftarrow \{0,1\}^\lambda$ and $\underline{\mathsf{label}^{g,\mathsf{crs}}_{k,b} \leftarrow \{0,1\}^\lambda}$ for $g \in [n + 1, N + 1]$, $k \in [\lambda]$, and $b \in \{0,1\}$.
4. From $g = N$ to $g = n + 1$ (decrement g), does the following.
   (a) Interprets gate g as (i, j).
   (b) Computes $\widetilde{\mathsf{SC}}_g \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{SC}[(r_i, r_j, r_g), (i, j), \{(\mathsf{label}^{g+1}_{k,b}, \underline{\mathsf{label}^{g+1,\mathsf{crs}}_{k,b}})\}_{k\in[\lambda],b\in\{0,1\}}, 0], (\{\mathsf{label}^g_{k,b}\}_{k\in[\lambda],b\in\{0,1\}}, \underline{\{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\}_{k\in[\lambda],b\in\{0,1\}}}))$.
5. Generates $\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{sek}, \{\widetilde{\mathsf{SC}}_g\}_{g\in[n+1,N]})$.
6. Outputs $\widetilde{C} := \boldsymbol{c}$ and $\mathsf{st} := (r, \mathsf{sek}, \{(\mathsf{label}^{n+1}_{k,b}, \underline{\mathsf{label}^{n+1,\mathsf{crs}}_{k,b}})\}_{k\in[\lambda],b\in\{0,1\}}, \underline{\mathsf{crs}})$.

$\mathsf{GbInp}(\mathsf{st}, x)$**:** This algorithm garbles an input $x \in \{0,1\}^n$ as follows.

1. Parses $\mathsf{st} := (r, \mathsf{sek}, \{(\mathsf{label}^{n+1}_{k,b}, \underline{\mathsf{label}^{n+1,\mathsf{crs}}_{k,b}})\}_{k\in[\lambda],b\in\{0,1\}}, \underline{\mathsf{crs}})$.

2. Sets $D := r_1 \oplus x_1 \| \cdots \| r_n \oplus x_n \| 0^{N-n}$.
3. Computes $(d, \widehat{D}) := \mathsf{Hash}(\mathsf{crs}, D)$.
4. Outputs $\widetilde{x} := (\{(\mathsf{label}^{n+1}_{k,d[k]}, \mathsf{label}^{n+1,\mathsf{crs}}_{k,\mathsf{crs}[k]})\}_{k\in[\lambda]}, \mathsf{crs}, r_1 \oplus x_1, \ldots, r_n \oplus x_n, \mathsf{sek}, r_{N-m+1}, \ldots, r_N)$.

$\mathsf{GbEval}(\widetilde{C}, \widetilde{x})$**:** This evaluation algorithm does the following.

1. Parses $\widetilde{C} = \boldsymbol{c}$ and $\widetilde{x} := (\{(\mathsf{label}_{k,d[k]}, \mathsf{label}^{\mathsf{crs}}_{k,\mathsf{crs}[k]})\}_{k\in[\lambda]}, \mathsf{crs}, r_1 \oplus x_1, \ldots, r_n \oplus x_n, \mathsf{sek}, r_{N-m+1}, \ldots, r_N)$.
2. Sets $D := r_1 \oplus x_1 \| \cdots \| r_n \oplus x_n \| 0^{N-n}$.
3. Computes $(d, \widehat{D}) := \mathsf{Hash}(\mathsf{crs}, D)$.
4. Computes $\{\widetilde{\mathsf{SC}}_g\}_{g\in[n+1,N]} \leftarrow \mathsf{Dec}(\mathsf{sek}, \boldsymbol{c})$.
5. Set $\overline{\mathsf{label}} := \{\mathsf{label}_{k,d[k]}\}_{k\in[\lambda]}$ and $\overline{\mathsf{label}}^{\mathsf{crs}} := \{\mathsf{label}^{\mathsf{crs}}_{k,\mathsf{crs}[k]}\}_{k\in[\lambda]}$.
6. For $g = n+1, \ldots, N$
   (a) Interprets g as (i, j).
   (b) Computes $(\mathsf{gout}_1, \mathsf{gout}_2) := \mathsf{GC.Eval}(\widetilde{\mathsf{SC}}_g, (\overline{\mathsf{label}}, \overline{\mathsf{label}}^{\mathsf{crs}}))$.
   (c) Computes $(\gamma, e) := \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{gout}_1, i), j)$.
   (d) Sets $\overline{\mathsf{label}} := \mathsf{ReceiveWrite}^{\widehat{D}}(\mathsf{crs}, g, \gamma, e)$ and $\overline{\mathsf{label}}^{\mathsf{crs}} := \mathsf{gout}_2$.
7. Reads D from $\widehat{D}$.
8. Outputs $D_{N-m+1} \oplus r_{N-m+1} \| \cdots \| D_N \oplus r_N$.

Remark 5.3. We assume that the length of crs is λ for ease of notation instead of writing $\{\mathsf{label}^{g,\mathsf{crs}}_{k',b}\}_{k'\in[\mathrm{poly}(\lambda)],b\in\{0,1\}}$. We often omit the region where indices (k, b) run if it is clear from the context. That is, we often write $\{\mathsf{label}^{g}_{k,b}\}$ and $\{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\}$ to denote $\{\mathsf{label}^{g}_{k,b}\}_{k\in[\lambda],b\in\{0,1\}}$ and $\{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\}_{k\in[\lambda],b\in\{0,1\}}$.

Modified Step Circuit SC

Input: A digest d and the CRS crs.

Hardwired value: (r_i, r_j, r_g), (i, j), $\{\mathsf{label}_{k,b}\}$, $\{\mathsf{label}^{\mathsf{crs}}_{k,b}\}$, and $\mathsf{flag} \in \{0, 1\}$.

1. Generates $e_b \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, g, b, \{\mathsf{label}_{k,0}, \mathsf{label}_{k,1}\}_{k\in[\lambda]})$ for $b \in \{0, 1\}$.
2. If $\mathsf{flag} = 0$, then $\gamma(\alpha, \beta) := \mathsf{NAND}(\alpha \oplus r_i, \beta \oplus r_j) \oplus r_g$ for all $\alpha, \beta \in \{0, 1\}$.
3. If $\mathsf{flag} = 1$, then $\gamma(\alpha, \beta) := r_g$ for all $\alpha, \beta \in \{0, 1\}$.
4. Generates

$$f_0 \leftarrow \mathsf{Send}(\mathsf{crs}, d, j, (\gamma(0,0), e_{\gamma(0,0)}), (\gamma(0,1), e_{\gamma(0,1)}))$$
$$f_1 \leftarrow \mathsf{Send}(\mathsf{crs}, d, j, (\gamma(1,0), e_{\gamma(1,0)}), (\gamma(1,1), e_{\gamma(1,1)}))$$

5. Outputs $\mathsf{Send}(\mathsf{crs}, d, i, f_0, f_1)$ and $\{\mathsf{label}^{\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$.

Figure 4: The description of modified step circuit

Claim 5.4. $\mathsf{adGC}'_{\mathsf{gs}}$ satisfies the correctness.

Proof. Let $D^{g^\star}$ and $D^{g^\star}_g$ be the all entries and g-th entry of the database at the end of Step 6.(c) of GbEval in the $g^\star$-th iteration of the "For" loop where $g^\star \in [n+1, N]$ $g \in [N]$. We will show that $D^{g^\star}_g$ is a masked value of the output of gate g for any $g \in [1, g^\star]$ by an inductive argument. That is,

$D_g^{g^\star} = \mathsf{NAND}(D_i^{g^\star-1} \oplus r_i, D_j^{g^\star-1} \oplus r_j) \oplus r_{g^\star}$ holds for $n < g \leq g^\star$ where the inputs of g are the outputs of i and j and $D_g^{g^\star} = x_g \oplus r_g$ for $1 \leq g \leq n$.
Base case (Before into the loop): It is easy to see that the equation holds for $g^\star = n$ since the initial database is $D^n = r_1 \| \cdots \| r_n \oplus x \| 0^{N-n}$.
Inductive step: We will show that γ in Step 6.(b) of GbEval is equal to $\mathsf{NAND}(D_i^{g^\star-1} \oplus r_i, D_j^{g^\star-1} \oplus r_j) \oplus r_{g^\star}$. This value is written in $D_g^{g^\star}$ by the $\mathsf{ReceiveWrite}(\mathsf{crs}, g, \gamma, e)$ in Step 6.(c).

First, by the definition of the modified step circuit and the correctness of GC, it holds that

$$(\mathsf{gout}_1, \mathsf{gout}_2) = \mathsf{GC.Eval}(\widetilde{\mathsf{SC}}_g, (\overline{\mathsf{label}}, \overline{\mathsf{label}}^{\mathsf{crs}})) = \left(\mathsf{Send}(\mathsf{crs}, d, i, f_0, f_1), \{\mathsf{label}_{k,\mathsf{crs}[k]}^{g+1,\mathsf{crs}}\}\right).$$

Next, it holds that

$$f_{D_i^{g^\star-1}} = \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, d, i, f_0, f_1), i)$$

since the output of i-th gate is $D_i^{g^\star-1}$. By the definition of f_b for $b \in \{0,1\}$, it holds that

$$f_{D_i^{g^\star-1}} = \mathsf{Send}(\mathsf{crs}, d, j, (\gamma(D_i^{g^\star-1}, 0), e_{\gamma(D_i^{g^\star-1},0)}), (\gamma(D_i^{g^\star-1}, 1), e_{\gamma(D_i^{l-1},1)})).$$

Therefore, it holds that

$$\begin{aligned}
(\gamma, e) &\coloneqq \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{gout}_1, i), j) \\
&= (\gamma(D_i^{g^\star-1}, D_j^{g^\star-1}), e_{\gamma(D_i^{g^\star-1}, D_j^{g^\star-1})}) \\
&= (\mathsf{NAND}(D_i^{g^\star-1} \oplus r_i, D_j^{g^\star-1} \oplus r_j) \oplus r_{g^\star}, e_{\mathsf{NAND}(D_i^{g^\star-1} \oplus r_i, D_j^{g^\star-1} \oplus r_j) \oplus r_{g^\star}}).
\end{aligned}$$

Therefore, the inductive step is completed.

Finally, the correctness follows by setting $g^\star \coloneqq N$ since it is easy to see that we obtain $D_g^N \oplus r_g = C(x)[g]$ where $g \in [N-m+1, N]$ at Step 8 of GbEval. ■

Online Complexity of GbInp. We confirm that our garbling scheme satisfies the complexity described in Theorem 5.2.

Online communication complexity: It is easy to see that $|\widetilde{x}| = \lambda^2 + \lambda + |\mathsf{crs}| + n + m + |\mathsf{sek}|$. By the efficiency of updatable LOT, $|\mathsf{crs}| = \lambda$ holds[19]. Recall that $|\mathsf{sek}| = t \cdot s \cdot \mathrm{poly}(\lambda)$ where s is the block-length and t is the equivocation parameter. In our setting, we set $s \coloneqq |\widetilde{\mathsf{SC}}|$ and $t \coloneqq \log N$. Moreover, by the efficiency of updatable LOT, $|\widetilde{\mathsf{SC}}| = \mathrm{poly}(\log N, \lambda)$. Therefore, $|\mathsf{sek}| = \mathrm{poly}(\log N, \lambda)$. Thus, $|\widetilde{x}| = n + m + \mathrm{poly}(\log |C|, \lambda)$ (note that $|C| = N$).

Online computational complexity: The running time of our GbInp depends on N since it computes $\mathsf{Hash}(\mathsf{crs}, D)$. However, we can reduce the computational complexity using a specific structure of the updatable LOT by Cho et al. [CDG+17] (recall that our updatable LOT in Section 4 also uses this structure) by using the same technique as GS18 scheme. We briefly review it.

The construction uses Merkle hash tree technique. Therefore, we can efficiently *update* a hash value. Let y and y' consist of ℓ blocks of λ-bits strings. Assume that y is different from y' only in the first k blocks. Given the Merkle hash on y and a set of $\log |y|$ hash values, there exists an efficient algorithm that computes the Merkle hash on y' and whose running time is $O(\lambda(k + \log |y|))$.

By using this efficient update algorithm, we can reduce the computational complexity as follows. At offline phase, we compute a hash value on 0^N. We set each block length to be 1. That is, when

[19] In fact, in our LOT protocol in Section 4, $|\mathsf{crs}| = \mathrm{poly}(\lambda)$. However, it does not matter here since it is absorbed in $\mathrm{poly}(\log |C|, \lambda)$ part.

$x \in \{0,1\}^n$ is given, we update the first $\lceil n \rceil$ blocks. For updating the hash value on 0^N to the hash value on $(r \oplus x \| 0^{N-n})$, it takes $O(1 \cdot (n + \log N))$ time. That is, the running time of GbInp is $O(n+m) + \mathrm{poly}(\log |C|, \lambda)$ since GbInp computes the hash value and outputs $\mathrm{poly}(\lambda) + n + m$ values. Note that GbInp need not output $(r_{n+1}, \ldots, r_{N-m})$.

5.2 Security Proof of Adaptive Garbling from Selective-Database Laconic OT

In this section, we prove the adaptive security of $\mathsf{adGC}'_{\mathsf{gs}}$.

Simulation and Hybrids.

To prove the security, we must present simulation algorithms SimC and SimIn and prove that simulated garbled circuit $\widetilde{C}$ and garbled input $\widetilde{x}$ are indistinguishable from the real ones. We can define the simulation algorithm of $\mathsf{adGC}'_{\mathsf{gs}}$ and hybrid simulations via the notion of circuit configurations introduced by Hemenway et al. [HJO+16].

Definition 5.5 (Circuit configuration). *A circuit configuration* conf *consists of a set* $I \subseteq [n+1, N]$ *and a set of tuples* (g, mode_g) *where* $\mathsf{mode}_g \in \{\mathsf{White}, \mathsf{Gray}, \mathsf{Black}\}$ *indicates the mode of simulation strategies for each gate* $g \in [n+1, N]$*. That is,* $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g \in [n+1,N]})$.

Definition 5.6 (Valid configuration). *We say that a configuration* $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g \in [n+1,N]})$ *is valid if and only if*

- *If* $\mathsf{mode}_g = \mathsf{Black}$*, then for every* $k > g$*,* $\mathsf{mode}_k = \mathsf{Black}$.
- *If* $\mathsf{mode}_g = \mathsf{Gray}$*, then* $g \in I$.

We define $\mathsf{Hyb}_{\mathsf{conf}}$ to be a hybrid distribution of $\widetilde{C}$ and $\widetilde{x}$ for every valid circuit configuration $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g \in [n+1,N]})$ in Figure 5. We can observe that

- The real experiment of adaptive garbling scheme, that is, $\mathsf{coin} = 0$ case in Definition 3.10 corresponds to $\mathsf{r.conf} := \mathsf{conf} = (\emptyset, \{(g, \mathsf{White})\}_{g \in [n+1,N]})$.
- The ideal experiment of adaptive garbling scheme, that is, $\mathsf{coin} = 1$ case in Definition 3.10 corresponds to $\mathsf{s.conf} := \mathsf{conf} = (\emptyset, \{(g, \mathsf{Black})\}_{g \in [n+1,N]})$.

Note that we abuse the notation. We use Hyb to denote not only a hybrid distribution in a hybrid game but also the description of the hybrid game.

Hybrid Simulation for configuration $\mathsf{conf} = (I, \{(h, \mathsf{mode}_h)\}_{h\in[n+1,N]})$

$\mathsf{SimC}(1^\lambda, C)$**:** Note that SimC uses the circuit $C : \{0,1\}^n \to \{0,1\}^m$ only in *hybrid simulations*.

1. Chooses $r \leftarrow \{0,1\}^N$.
2. Chooses $\mathsf{label}^g_{k,b} \leftarrow \{0,1\}^\lambda$ and $\mathsf{label}^{g,\mathsf{crs}}_{k,b} \leftarrow \{0,1\}^\lambda$ for $g \in [n+1, N+1]$, $k \in [\lambda]$, and $b \in \{0,1\}$.
3. From $g = N$ to $g = n+1$ (decrement g) such that $g \notin I$, does the following.
   - Interprets gate g as (i, j).
   - If $\mathsf{mode}_g = \mathsf{White}$, then computes
   $$\widetilde{\mathsf{SC}}_g \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{SC}[(r_i, r_j, r_g), (i, j), \overline{\mathsf{labels}}^{g+1}, 0], \overline{\mathsf{labels}}^g)$$
   where SC is described in Figure 4 and $\overline{\mathsf{labels}}^g = (\{\mathsf{label}^g_{k,b}\}, \{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\})$.
   - If $\mathsf{mode}_g = \mathsf{Black}$, then computes
   $$\widetilde{\mathsf{SC}}_g \leftarrow \mathsf{GC.Grbl}(1^\lambda, \mathsf{SC}[(0, 0, r_g), (i, j), \overline{\mathsf{labels}}^{g+1}, 1], \overline{\mathsf{labels}}^g)$$
   where $\overline{\mathsf{labels}}^g = (\{\mathsf{label}^g_{k,b}\}, \{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\})$.
4. Generates $(\mathsf{st}_1, \boldsymbol{c}) \leftarrow \mathsf{SimEnc}(I, \{\widetilde{\mathsf{SC}}_g\}_{g\notin I})$.
5. Outputs $\widetilde{C} := \boldsymbol{c}$ and $\mathsf{st} := (r, \mathsf{st}_1, \{\mathsf{label}^{n+1}_{k,b}\}, \{\mathsf{label}^{n+1,\mathsf{crs}}_{k,b}\})$

In the ideal experiment, we do not need any gate description since all $g \in [n+1, N]$ are Black mode and $\mathsf{flag} = 1$ in SC.

Notation for simulated input garbling: For $g \in [n+1, N]$, D_g is defined as follows.

$$D_g[w] := \begin{cases} x_w \oplus r_w & w \le n \\ E_w \oplus r_w & n+1 \le w < g \\ 0 & \text{otherwise,} \end{cases}$$

where E_w is the bit assigned to wire w of C when the input is x. We also use d_g for the digest of D_g, that is, $(d_g, \widehat{D}_g) := \mathsf{Hash}(\mathsf{crs}, D_g)$. We let $d_g[k]$ be the k-th bit of d_g.

$\mathsf{SimIn}(\mathsf{st}, x, y)$**:** Note that SimIn uses the input $x \in \{0,1\}^n$ only in the *hybrid simulations*.

1. Parses $\mathsf{st} = (r, \mathsf{st}_1, \{\mathsf{label}^{n+1}_{k,b}\}, \{\mathsf{label}^{n+1,\mathsf{crs}}_{k,b}\})$.
2. Generates $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.
3. From $g = N$ to $g = n+1$ (decrement g) such that $g \in I$, does the following.
   (a) Computes $e \leftarrow \mathsf{SimWrite}(\mathsf{crs}, D_g, g, D_{g+1}, \{\mathsf{label}^{g+1}_{k,d_{g+1}[k]}\})$.
   (b) Computes $\mathsf{gout}_1 \leftarrow \mathsf{Sim}(\mathsf{crs}, D_g, i, \mathsf{Sim}(\mathsf{crs}, D_g, j, e))$ and sets $\mathsf{gout}_2 := \{\mathsf{label}^{g+1,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$.
   (c) Computes $\widetilde{\mathsf{SC}}_g \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|\mathsf{SC}|}, (\mathsf{gout}_1, \mathsf{gout}_2), (\{\mathsf{label}^g_{k,d_g[k]}\}, \{\mathsf{label}^{g,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}))$.
4. Computes $\widetilde{\mathsf{sek}} \leftarrow \mathsf{SimKey}(\mathsf{st}_1, \{\widetilde{\mathsf{SC}}_g\}_{g\in I})$.
5. From $g \in [N-m+1, N]$ (increment g), does the following
   (a) If $\mathsf{mode}_g = \mathsf{Black}$, then sets $r'_g := r_g \oplus y[w-N+m]$.
   (b) Else, sets $r'_g := r_g$.

   Outputs $\widetilde{x} := (\{\mathsf{label}^{n+1}_{k,d_{n+1}[k]}\}_{k\in[\lambda]}, \{\mathsf{label}^{n+1,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}_{k\in[\lambda]}, r_1 \oplus x_1 \| \ldots \| r_n \oplus x_n, \widetilde{\mathsf{sek}}, r'_{N-m+1}, \ldots, r'_N)$.

In the ideal experiment, we consider $x := 0^n$ and it holds $r_1 \oplus x_1 \| \cdots \| r_n \oplus x_n = r_1 \| \cdots \| r_n$. Thus, we do not need x.

Figure 5: Simulation for hybrid configurations.

Rules of Indistinguishability

Our goal is proving that $\mathsf{Hyb}_{\mathsf{r.conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{s.conf}}$. To prove this, we use the same indistinguishability rules introduced by Garg and Srinivasan [GS18]. We can use those rules as they are though the proofs of indistinguishability are slightly different from theirs since our scheme is based on selective-database updatable LOT.

Definition 5.7 (Rule A [GS18]). *For any valid configuration* conf*, we can change the mode of a gate* g^* *such that* $\mathsf{mode}_{g^*} = \mathsf{White}$ *into* $\mathsf{mode}_{g^*} = \mathsf{Gray}$ *if* g^* *is the first gate or if its predecessor is also in* Gray *mode.*

Let $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g\in[n+1,N]})$ *and* $\mathsf{conf}' = (I', \{(g, \mathsf{mode}'_g)\}_{g\in[n+1,N]})$ *be two valid circuit configurations and* $g^* \in [n+1, N]$ *be a gate such that*

- *For all* $g \in [n+1, N] \setminus g^*$*, it holds that* $\mathsf{mode}_g = \mathsf{mode}'_g$.
- *It holds that* $g^* \notin I$*,* $I' = I \cup \{g^*\}$*, and* $|I'| \leq t$ *where* t *is the equivocation parameter of somewhere equivocal encryption.*
- *It holds either* $g^* = n+1$ *or* $(g^* - 1, \mathsf{Gray}) \in \mathsf{conf}$.
- *It holds that* $(g^*, \mathsf{White}) \in \mathsf{conf}$ *and* $(g^*, \mathsf{Gray}) \in \mathsf{conf}'$.

In Lemma 5.10, we will prove that it holds $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{conf}'}$.

Definition 5.8 (Rule B [GS18]). *For any valid configuration* conf*, we can change the mode of a gate* g^* *such that* $\mathsf{mode}_{g^*} = \mathsf{Gray}$ *into* $\mathsf{mode}_{g^*} = \mathsf{Black}$ *if all gates subsequent to* g^* *are in* Black *mode and its predecessor is in* Gray *mode.*

Let $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g\in[n+1,N]})$ *and* $\mathsf{conf}' = (I', \{(g, \mathsf{mode}'_g)\}_{g\in[n+1,N]})$ *be two valid circuit configurations and* $g^* \in [n+1, N]$ *be a gate such that*

- *For all* $g \in [n+1, N] \setminus g^*$*, it holds that* $\mathsf{mode}_g = \mathsf{mode}'_g$.
- *It holds that* $g^* \in I$*,* $I' = I \setminus \{g^*\}$*, and* $|I| \leq t$ *where* t *is the equivocation parameter of somewhere equivocal encryption.*
- *For all* $g \in [g^*+1, N]$*, it holds that* $(g, \mathsf{Black}) \in \mathsf{conf}$.
- *It holds either* $g^* = n+1$ *or* $(g^* - 1, \mathsf{Gray}) \in \mathsf{conf}$.
- *It holds that* $(g^*, \mathsf{Gray}) \in \mathsf{conf}$ *and* $(g^*, \mathsf{Black}) \in \mathsf{conf}'$.

In Lemma 5.18, we will prove that it holds $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{conf}'}$.

Pebbling Game and Optimal Strategy

We can prove $\mathsf{Hyb}_{\mathsf{r.conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{s.conf}}$ by using rules A and B many times. However, to achieve the communication/computational complexity described in Theorem 5.1, we cannot apply these rules in a naive way. This is because we can use *at most* $\log N$ Gray *mode gates* (i.e., $|I| \leq t = \log N$) in each configuration. Garg and Srinivasan proposed the optimal pebbling strategy, which enables us to set $t = \log N$.

Lemma 5.9 (Optimal pebbling strategy [GS18]). *For any* $N \in \mathbb{N}$*, there is a procedure that changes* $\mathsf{r.conf} = (\emptyset, \{g, \mathsf{White}\}_{g\in[n+1,N]})$ *into* $\mathsf{s.conf} = (\emptyset, \{g, \mathsf{Black}\}_{g\in[n+1,N]})$ *by applying Rule A and Rule B at most* $\mathrm{poly}(N)$ *times, and using at most* $\log N$ Gray *modes during the transition. That is, during the transition from all* White *mode to all* Black *mode,* $\mathsf{conf} = (I, \{g, \mathsf{mode}_g\}_{g\in[n+1,N]})$ *always satisfies* $|I| \leq \log N$.

For proving Theorem 5.1, what remains is to prove that if conf is changed to conf' by applying Rule A or Rule B, then we have $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{conf}'}$.

Proof of Indistinguishability for Rule A and B

Now, all we have to do is proving the indistinguishability of the two rules.

Lemma 5.10 (Rule A). *Let* conf *and* conf' *be two valid configurations that satisfy the conditions of rule A in Definition 5.7. If there exists selectively secure garbling scheme for circuits, somewhere equivocal encryption, and selective-database updatable laconic OT in Definitions 3.8, 3.22 and 4.1, then it holds that* $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{conf}'}$.

Proof of Lemma 5.10. We define the sequence of hybrid games.

$\mathsf{Hyb}_{\mathsf{conf}}$**:** We start from the configuration $\mathsf{conf} = (I, \{(g, \mathsf{mode}_g)\}_{g\in[n+1,N]})$.

Hyb_1**:** This hybrid is the same as $\mathsf{Hyb}_{\mathsf{conf}}$ except that we change the configuration from $(I, \{(g, \mathsf{mode}_g)\}_{g\in[n+1,N]})$ to $(I', \{(g, \mathsf{mode}_g)\}_{g\in[n+1,N]})$. Note that we do not change how to generate $\widetilde{\mathsf{SC}}_{g^*}$ at this point. We will prove $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_1$ by using the security of somewhere equivocal encryption in Lemma 5.11.

Hyb_2**:** This hybrid is the same as Hyb_1 except that we defer choosing $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ and computing $\widetilde{\mathsf{SC}}_{g^*}$ to SimIn phase (not SimC phase). This is purely conceptual change. We will prove $\mathsf{Hyb}_1 = \mathsf{Hyb}_2$ (i.e., two distributions are identical) in Lemma 5.12.

Hyb_3**:** This hybrid is the same as Hyb_2 except that we change how to choose $(\{\mathsf{label}^{g^*}_{k,b}\}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\})$ and compute $\widetilde{\mathsf{SC}}_{g^*}$ as follows.

- We do not choose $\{\mathsf{label}^{g^*}_{k,1-d_{g^*}[k]}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,1-\mathsf{crs}[k]}\}$ anymore. That is, we use only $\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$. These are sufficient for our simulation of $\widetilde{x}$ and $\widetilde{\mathsf{SC}}_{g^*-1}$ since $g^* - 1$ is in Gray mode in this setting.
- We compute $(\mathsf{gout}_1, \mathsf{gout}_2) \leftarrow \mathsf{SC}[\mathsf{crs}, (r_i, r_j, r_g), (i, j), \{\mathsf{label}^{g^*+1}_{k,b}\}, \{\mathsf{label}^{g^*+1,\mathsf{crs}}_{k,b}\}, 0](d_{g^*}, \mathsf{crs})$ and

$$\widetilde{\mathsf{SC}}_{g^*} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|\mathsf{SC}|}, (\mathsf{gout}_1, \mathsf{gout}_2), (\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}6))$$

  by using the simulator of the selectively secure garbling scheme. Note that $\widetilde{\mathsf{SC}}_{g^*}$ is generated at SimIn and crs is already generated at this point. We will prove $\mathsf{Hyb}_2 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_3$ by using the selective security of garbling scheme $(\mathsf{GC.Grbl}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ in Lemma 5.13.

Hyb_4**:** This hybrid is the same as Hyb_3 except that we change how to compute gout_1 hardwired in $\widetilde{\mathsf{SC}}_{g^*}$ as follows. Instead of using both f_0 and f_1 as in Hyb_3, we compute

$$\mathsf{gout}_1 \leftarrow \mathsf{Sim}(\mathsf{crs}, D_{g^*}, i, f_{D_{g^*}[i]}).$$

That is, we use the simulator of updatable laconic OT Π by only using $f_{D_{g^*}[i]}$. We will prove $\mathsf{Hyb}_3 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_4$ by using the selective-database sender privacy in Lemma 5.14. Recall that i is the index of the gate whose output is an input of the gate g^*.

Hyb_5**:** This hybrid is the same as Hyb_4 except that we change how to compute $f_{D_{g^*}[i]}$ as follows. Instead of using $f_{D_{g^*}[i]} \leftarrow \mathsf{Send}(\mathsf{crs}, d, j, (\gamma(D_{g^*}[i], 0), e_{\gamma(D_{g^*}[i],0)}), (\gamma(D_{g^*}[i], 1), e_{\gamma(D_{g^*}[i],1)}))$, we compute

$$\mathsf{Sim}(\mathsf{crs}, D_{g^*}, j, (D_{g^*+1}[g^*], e_{D_{g^*+1}[g^*]})),$$

where $e_{D_{g^*+1}[g^*]} \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, g^*, D_{g^*+1}[g^*], \{\mathsf{label}_{k,0}^{g^*+1}, \mathsf{label}_{k,1}^{g^*+1}\})$. Note that, by the definition, $D_{g^*+1}[g^*] = \gamma(D_{g^*}[i], D_{g^*}[j])$. We will prove $\mathsf{Hyb}_4 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_5$ by using the selective-database sender privacy in Lemma 5.15.

Hyb_6**:** This hybrid is the same as Hyb_5 except that we change how to compute $e_{D_{g^*+1}[g^*]}$ as follows. Instead of using $\mathsf{SendWrite}(\mathsf{crs}, d, g^*, D_{g^*+1}[g^*], \{\mathsf{label}_{k,0}^{g^*+1}, \mathsf{label}_{k,1}^{g^*+1}\})$, we compute

$$\mathsf{SimWrite}(\mathsf{crs}, D_{g^*}, g^*, D_{g^*+1}[g^*], \{\mathsf{label}_{k,d_{g^*+1}[k]}^{g^*+1}\}).$$

That is, we use the simulator for writes of updatable laconic OT Π by using $\{\mathsf{label}_{k,d_{g^*+1}[k]}^{g^*+1}\}$. We will prove $\mathsf{Hyb}_5 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_6$ by using the selective-database sender privacy for writes in Lemma 5.16.

Hyb_7**:** This hybrid is the same as Hyb_6 except that we change how to choose $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$. We choose $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$ for all $k \in [\lambda]$ and $b \in \{0,1\}$ at SimC again. This change is purely conceptual. It is easy to see that this game is the same as $\mathsf{Hyb}_{\mathsf{conf}'}$.

Lemma 5.11. *If Σ is somewhere equivocal encryption scheme, then it holds that $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_1$.*

Proof. We construct an algorithm $\mathcal{B}$ that breaks the security of somewhere equivocal encryption by using a distinguisher $\mathcal{A}$ for hybrid games Hyb_0 and Hyb_1. When $\mathcal{B}$ is given 1^λ, it uses I in conf as an index set and sets $\widehat{I} := I \cup \{g^*\}$, that is, g^* is the challenge index.

Simulation of the garbled circuit. $\mathcal{B}$ does the following to generate $\widetilde{C}$.

1. Does the 1st to 4th steps of SimC described in Figure 5.
2. Sends $\{\widetilde{\mathsf{SC}}_g\}_{g \notin I}$ generated above to the challenger of somewhere equivocal encryption as the challenge messages with the challenge set I and index g^*. $\mathcal{B}$ receives $\boldsymbol{c}$ from the challenger.
3. Outputs $\widetilde{C} := \boldsymbol{c}$ and $\mathsf{st} := (r, \mathsf{st}_1, \{\mathsf{label}_{k,b}^{n+1}\}, \{\mathsf{label}_{k,b}^{n+1,\mathsf{crs}}\})$.

Simulation of the garbled input. $\mathcal{B}$ does the following to generate $\widetilde{x}$.

1. Does the 1st to 3rd steps of SimIn described in Figure 5.
2. Sends $\{\widetilde{\mathsf{SC}}_g\}_{g \in I}$ generated above to the challenger of somewhere equivocal encryption as the remaining challenge messages and receives $\widetilde{\mathsf{sek}}$.
3. Does the 5th step of SimIn described in Figure 5 and outputs $\widetilde{x}$ by using $\widetilde{\mathsf{sek}}$ from the challenger.

It is easy to see that if $(\boldsymbol{c}, \widetilde{\mathsf{sek}})$ comes from $\mathsf{coin} = 0$ case of somewhere equivocal encryption, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{\mathsf{conf}}$ and if $(\boldsymbol{c}, \widetilde{\mathsf{sek}})$ comes from $\mathsf{coin} = 1$ case of somewhere equivocal encryption, $\mathcal{B}$ perfectly simulates Hyb_1. This completes the proof. ■

Lemma 5.12. *It holds that $\mathsf{Hyb}_1 = \mathsf{Hyb}_2$ (i.e., two distributions are identical).*

Proof. We can choose $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$ at SimIn because $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$ are not used in SimC but in SimIn to compute $\widetilde{x}$, $\widetilde{\mathsf{SC}}_{g^*-1}$, and $\widetilde{\mathsf{SC}}_{g^*}$. We note that it holds that $\mathsf{mode}_{g^*-1} = \mathsf{Gray}$ by the condition of Rule A and $g^* - 1 \in I'$. Thus, we do not need $\{\mathsf{label}_{k,b}^{g^*}\}$ to generate $\widetilde{\mathsf{SC}}_{g^*}$. Therefore, deferring choosing $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$ and computing $\widetilde{\mathsf{SC}}_{g^*}$ does not change anything. ■

Lemma 5.13. *If GC is a selectively secure garbling scheme, then it holds that $\mathsf{Hyb}_2 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_3$.*

Proof. We construct an algorithm $\mathcal{B}$ that breaks the selective security of garbling scheme by using a distinguisher $\mathcal{A}$ for hybrid games Hyb_2 and Hyb_3. Note that we do not need any value from the selective security of GC for simulating the garbled circuit $\widetilde{C}$ since we change how to generate $\widetilde{\mathsf{SC}}_{g^*}$ and $g^* \in I$ at this point.

Simulation of the garbled circuit. Note that $\mathcal{B}$ do not need $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$. This is because they need not to be committed by the assumption that $g^* - 1$ is in Gray mode or $g^* = n + 1$. Moreover, $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ is needed only for $\widetilde{x}$.

1. $\mathcal{B}$ chooses $\mathsf{label}^{g}_{k,b}$ and $\mathsf{label}^{g,\mathsf{crs}}_{k,b}$ for all $g \in [n+1, N] \setminus \{g^*\}$, $k \in [\lambda]$, and $b \in \{0,1\}$.
2. $\mathcal{B}$ generates $\widetilde{C}$ as in Hyb_2.

Simulation of the garbled input. $\mathcal{B}$ does the following.

1. Prepares $\mathsf{SC}[(r_i, r_j, r_{g^*}), (i,j), \{\mathsf{label}^{g^*+1}_{k,b}\}, \{\mathsf{label}^{g^*+1,\mathsf{crs}}_{k,b}\}, 0]$ and (d_{g^*}, crs) by using $\mathsf{Hash}(\mathsf{crs}, D_{g^*})$ and sends them as the challenge circuit and input for selective security of GC.
2. Receives $\widetilde{\mathsf{SC}}_{g^*}$ and $(\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}_{k\in[\lambda]}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}_{k\in[\lambda]})$ from the challenger of GC.
3. Generates $\widetilde{\mathsf{SC}}_g$ as in Figure 5 for all $g \in I' \setminus \{g^*\}$. Note that we need only $\{\mathsf{label}^{g,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$ for $g \in I'$. Moreover, we need only $\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}_{k\in[\lambda]}$ to generate $\widetilde{\mathsf{SC}}_{g^*-1}$.
4. Does the same things for the rest of the steps in Figure 5 and outputs $\widetilde{x}$.

It is easy to see that if $\widetilde{\mathsf{SC}}_{g^*}$ comes from $\mathsf{coin} = 0$ of the selective security of GC, then $\mathcal{B}$ perfectly simulates Hyb_2 and if $\widetilde{\mathsf{SC}}_{g^*}$ comes from $\mathsf{coin} = 1$ case then $\mathcal{B}$ perfectly simulates Hyb_3. This completes the proof. ■

Lemma 5.14. *If Π is selective-database sender private updatable laconic OT, then it holds that $\mathsf{Hyb}_3 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_4$.*

Proof of Lemma 5.14. We construct an adversary $\mathcal{B}$ for selective-database sender private updatable laconic OT that uses a distinguisher $\mathcal{A}$ for hybrid games Hyb_3 and Hyb_4. First, $\mathcal{B}$ must decide a database before $\mathcal{B}$ is given crs.

Simulation of the garbled circuit. $\mathcal{B}$ simulate $\widetilde{C} = \boldsymbol{c}$ as in Hyb_3. Note that $\mathcal{B}$ do not need crs for this simulation since crs is not hardwired in $\widetilde{\mathsf{SC}}$ for $g \notin I$.

Simulation of the garbled input. When $\mathcal{B}$ generates $\widetilde{x}$, $\mathcal{B}$ is given the input x. Thus, D_{g^*} is determined at this point according to the definition of $D_g[w]$ in Figure 5. Note that (i,j) are input wires of the gate g^*. Therefore, $\mathcal{B}$ sets a database to D_{g^*}, sends D_{g^*} to the challenger of selective-database sender privacy, and receives crs.

Now, $\mathcal{B}$ can compute $\widetilde{\mathsf{SC}}_{g^*}$ as follows.

1. Computes e_0, e_1, f_0, f_1 as in Figure 4 since $\mathcal{B}$ has crs at this point.
2. Sends the location i and the messages (f_0, f_1) to the challenger of selective-database sender privacy. Then, $\mathcal{B}$ obtains gout_1 as the target ciphertext.
3. Computes $\widetilde{\mathsf{SC}}_{g^*} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|\mathsf{SC}|}, (\mathsf{gout}_1, \mathsf{gout}_2), (\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}))$ where $\mathsf{gout}_2 = \{\mathsf{label}^{g^*+1,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$.

Next, $\mathcal{B}$ generates $\widetilde{\mathsf{SC}}_g$ for all $g \in I' \setminus \{g^*\}$ as in Hyb_3. For other steps of SimIn in Figure 5, $\mathcal{B}$ does the same things and generates $\widetilde{x}$.

It is easy to see that if gout_1 comes from $\mathsf{Real}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$, then $\mathcal{B}$ perfectly simulates Hyb_3 and if gout_1 comes from $\mathsf{Sim}^{\mathsf{sel\text{-}db}}_{\ell\mathsf{OT}}(\lambda)$ case then $\mathcal{B}$ perfectly simulates Hyb_4. This completes the proof. ■

Lemma 5.15. *If Π is selective-database sender private updatable laconic OT, then it holds that $\mathsf{Hyb}_4 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_5$.*

Proof. We can prove this lemma by the similar argument to that in the proof of Lemma 5.14 since when $\mathcal{B}$ generates $\widetilde{x}$, $\mathcal{B}$ is given the input x and D_{g^*} is determined at this point according to the definition of $D_g[w]$ in Figure 5. Moreover, $\mathcal{B}$ does not need crs for the simulation of $\widetilde{C}$. Thus, $\mathcal{B}$ can prepare the challenge messages $e_b \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, g^*, b, \{\mathsf{label}_{k,0}, \mathsf{label}_{k,1}\})$ and the challenge location j. For all $g \in I' \setminus \{g^*\}$, $\mathcal{B}$ can simulate $\widetilde{\mathsf{SC}}_g$ since crs is already given at this point. See the proof of Lemma 5.14 for the detail. ■

Lemma 5.16. *If Π is selective-database sender private updatable laconic OT, then it holds that $\mathsf{Hyb}_5 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_6$.*

Proof. We construct an algorithm $\mathcal{B}$ that breaks the selective-database sender privacy for writes by using a distinguisher $\mathcal{A}$ for hybrid games Hyb_5 and Hyb_6. First, $\mathcal{B}$ must decide a database before $\mathcal{B}$ is given crs.

Simulation of the garbled circuit. $\mathcal{B}$ simulate $\widetilde{C} = \boldsymbol{c}$ as in Hyb_5. Note that $\mathcal{B}$ do not need crs for this simulation since crs is not hardwired in $\widetilde{\mathsf{SC}}$ for $g \notin I$.

Simulation of the garbled input. When $\mathcal{B}$ generates $\widetilde{x}$, $\mathcal{B}$ is given the input x. Thus, D_{g^*} is determined at this point according to the definition of $D_g[w]$ in Figure 5. Note that (i, j) are input wires of the gate g^*. Therefore, $\mathcal{B}$ sets a database to D_{g^*}, sends D_{g^*} to the challenger of selective-database sender privacy for writes, and receives crs.

Now, $\mathcal{B}$ can compute $\widetilde{\mathsf{SC}}_{g^*}$ as follows.

1. Sends g^*, $D_{g^*+1}[g^*]$, and $\{\mathsf{label}^{g^*+1}_{k,b}\}$ as the challenge location, bit, and messages, respectively.
2. Receives $e_{D_{g^*+1}[g^*]}$ from the challenger.
3. Computes $f_{D_{g^*+1}[g^*]}$, gout_1, and gout_2 as in Hyb_5.
4. Computes $\widetilde{\mathsf{SC}}_{g^*} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|\mathsf{SC}|}, (\mathsf{gout}_1, \mathsf{gout}_2), (\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}))$.

Next, $\mathcal{B}$ generates $\widetilde{\mathsf{SC}}_g$ for all $g \in I' \setminus \{g^*\}$ as in Hyb_5. For other steps of SimIn in Figure 5, $\mathcal{B}$ does the same things and generates $\widetilde{x}$.

It is easy to see that if $e_{D_{g^*+1}[g^*]}$ comes from $\mathsf{Real}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)$, then $\mathcal{B}$ perfectly simulates Hyb_5 and if $e_{D_{g^*+1}[g^*]}$ comes from $\mathsf{Sim}^{\mathsf{sel\text{-}db\text{-}wr}}_{\ell\mathsf{OT}}(\lambda)$ then $\mathcal{B}$ perfectly simulates Hyb_6. This completes the proof. ■

Lemma 5.17. *It holds that $\mathsf{Hyb}_6 = \mathsf{Hyb}_7$.*

Proof. The simulator can choose $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ for all $k \in [\lambda], b \in \{0,1\}$ at this point since we had already used the selective security of GC. Moreover, the simulator can choose those at SimC. Thus, the distributions of Hyb_6 and Hyb_7 are completely the same. ■

By Lemmata 5.11 to 5.17, we complete the proof of Lemma 5.10. ■

Lemma 5.18 (Rule B). *Let* conf *and* conf' *be two valid configurations that satisfy the conditions of rule B in Definition 5.8. If there exists selectively secure garbling scheme for circuits, somewhere equivocal encryption, and selective-database secure updatable laconic OT in Definitions 3.8, 3.22 and 4.1, then it holds that* $\mathsf{Hyb}_{\mathsf{conf}} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_{\mathsf{conf}'}$.

Proof of Lemma 5.18. We define the sequence of hybrid games. The lemma is proved with hybrid games in reverse order similarly to GS18 scheme.

$\mathsf{Hyb}_{\mathsf{conf}'}$**:** We start from the configuration $\mathsf{conf}' = (I', \{(g, \mathsf{mode}'_g)\}_{g\in[n+1,N]})$.

Hyb_1**:** This hybrid is the same as $\mathsf{Hyb}_{\mathsf{conf}'}$ except that we change the configuration from $(I', \{(g, \mathsf{mode}'_g)\}_{g\in[n+1,N]})$ to $(I, \{(g, \mathsf{mode}'_g)\}_{g\in[n+1,N]})$. If Σ is somewhere equivocal encryption, it holds that $\mathsf{Hyb}_{\mathsf{conf}'} \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_1$. We can prove this in a similar way to Lemma 5.11.

Hyb_2**:** This hybrid is the same as Hyb_2 except that we defer choosing $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ and computing $\widetilde{\mathsf{SC}}_{g^*}$ to SimIn phase (not SimC phase). We can do this because $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ are not used in SimC but in SimIn to compute $\widetilde{\mathsf{SC}}_{g^*-1}$, $\widetilde{\mathsf{SC}}_{g^*}$, and $\widetilde{x}$. We note that it holds that $\mathsf{mode}_{g^*-1} = \mathsf{Gray}$ by the condition of Rule B and $g^* - 1 \in I'$. It holds that $\mathsf{Hyb}_1 = \mathsf{Hyb}_2$ (i.e., two distributions are identical). We can prove this in a similar way to Lemma 5.12.

Hyb_3**:** This hybrid is the same as Hyb_2 except that we change how to choose $\{\mathsf{label}^{g^*}_{k,b}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,b}\}$ and computing $\widetilde{\mathsf{SC}}_{g^*}$ as follows.

- We do not choose $\{\mathsf{label}^{g^*}_{k,1-d_{g^*}[k]}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,1-\mathsf{crs}[k]}\}$ anymore. That is, we use only $\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}$ and $\{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$. These are sufficient for our simulation of $\widetilde{\mathsf{SC}}_{g^*-1}$ and $\widetilde{x}$ since $g^* - 1$ is in Gray mode in this setting and $\{\mathsf{label}^{g,\mathsf{crs}}_{k,\mathsf{crs}[k]}\}$ is used only in $\widetilde{x}$.
- We compute $(\mathsf{gout}_1, \mathsf{gout}_2) \leftarrow \mathsf{SC}[\mathsf{crs}, (0,0,r_g), (i,j), \{\mathsf{label}^{g^*+1}_{k,b}\}, \{\mathsf{label}^{g^*+1,\mathsf{crs}}_{k,b}\}, 1](d_{g^*}, \mathsf{crs})$ and
$$\widetilde{\mathsf{SC}}_{g^*} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|\mathsf{SC}|}, (\mathsf{gout}_1, \mathsf{gout}_2), (\{\mathsf{label}^{g^*}_{k,d_{g^*}[k]}\}, \{\mathsf{label}^{g^*,\mathsf{crs}}_{k,\mathsf{crs}[k]}\})).$$
Note that $\widetilde{\mathsf{SC}}_{g^*}$ is generated at SimIn and crs is already generated at this point.

If GC is selectively secure garbled circuit, it holds that $\mathsf{Hyb}_2 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_3$. We can prove this in a similar way to Lemma 5.13.

Hyb_4**:** This hybrid is the same as Hyb_3 except that we change how to compute gout_1 hardwired in $\widetilde{\mathsf{SC}}_{g^*}$ as follows. Instead of using both f_0 and f_1, we compute
$$\mathsf{gout}_1 \leftarrow \mathsf{Sim}(\mathsf{crs}, D_{g^*}, i, f_{D_{g^*}[i]}).$$
That is, we use the simulator of updatable laconic OT Π by only using $f_{D_{g^*}[i]}$. If Π is selective-database sender private, it holds that $\mathsf{Hyb}_3 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_4$. We can prove this in a similar way to Lemma 5.14.

Hyb_5**:** This hybrid is the same as Hyb_4 except that we change how to compute $f_{D_{g^*}[i]}$ as follows. Instead of using $f_{D_{g^*}[i]} \leftarrow \mathsf{Send}(\mathsf{crs}, d, j, e_{r_{g^*}}, e_{r_{g^*}})$, we compute
$$\mathsf{Sim}(\mathsf{crs}, D_{g^*}, j, e_{r_{g^*}}),$$
where $e_{r_{g^*}} \leftarrow \mathsf{SendWrite}(\mathsf{crs}, d, g^*, r_{g^*}, \{\mathsf{label}^{g^*+1}_{k,0}, \mathsf{label}^{g^*+1}_{k,1}\})$. If Π is selective-database sender private, it holds that $\mathsf{Hyb}_4 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_5$. We can prove this in a similar way to Lemma 5.15.

Hyb_6**:** This hybrid is the same as Hyb_5 except that we change how to compute $e_{r_{g^*}}$ as follows. Instead of using $\mathsf{SendWrite}(\mathsf{crs}, d, g^*, r_{g^*}, \{\mathsf{label}_{k,0}^{g^*+1}, \mathsf{label}_{k,1}^{g^*+1}\})$, we compute

$$\mathsf{SimWrite}(\mathsf{crs}, D_{g^*}, g^*, r_{g^*}, \{\mathsf{label}_{k,d_{g^*+1}[k]}^{g^*+1}\}).$$

If Π is selective-database sender private for writes, it holds that $\mathsf{Hyb}_5 \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_6$. We can prove this in a similar way to Lemma 5.16.

Hyb_7**:** This hybrid is the same as Hyb_6 except that we change how to choose $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$. We choose $\{\mathsf{label}_{k,b}^{g^*}\}$ and $\{\mathsf{label}_{k,b}^{g^*,\mathsf{crs}}\}$ for all $k \in [\lambda]$ and $b \in \{0,1\}$ at SimC again. It holds that $\mathsf{Hyb}_6 = \mathsf{Hyb}_7$ (i.e., two distributions are identical). We can prove this in a similar way to Lemma 5.12.

Hyb_8**:** This hybrid is the same as Hyb_7 except that how to generate $D_{g^*+1}[g^*]$ is different as follows. In Hyb_8, $D_{g^*+1}[g^*] := r_{g^*} \oplus \mathsf{NAND}(D_{g^*}[i] \oplus r_i, D_{g^*}[j] \oplus r_j)$. Note that, in Hyb_7, $D_{g^*+1}[g^*] := r_{g^*}$. In fact, the distributions of Hyb_7 and Hyb_8 are the same. There are two cases as follows.

Case $g^* \leq N - m$**:** In this case, r_{g^*} does not appear in $\widetilde{x}$. Moreover, r_{g^*} is not hardwired in $\widetilde{\mathsf{SC}}$ since $\widetilde{\mathsf{SC}}$ is simulated by the simulation of GC. Thus, both r_{g^*} and $r_{g^*} \oplus \mathsf{NAND}(D_{g^*}[i] \oplus r_i, D_{g^*}[j] \oplus r_j)$ are uniformly and identically distributed.

Case $g^* > N - m$**:** In this case, it holds that $r'_{g^*} = y_{g^*-N+m} \oplus r_{g^*}$ in Hyb_7 (this corresponds to Black mode by the definition in Figure 5). This is because $D_{g^*}[i] = r_i \oplus E_i$ (resp. $D_{g^*}[j] = r_j \oplus E_j$) and (E_i, E_j) are real wire values connected to gate g^* when the input is x. That is,

$$r_{g^*} \oplus \mathsf{NAND}(D_{g^*}[i] \oplus r_i, D_{g^*}[j] \oplus r_j) = r_{g^*} \oplus \mathsf{NAND}(E_i, E_j) = r_{g^*} \oplus y_{g^*-N+m}.$$

The distribution of $r'_{g^*} = y_{g^*-N+m} \oplus r_{g^*}$ is identical to that of $r'_{g^*} = r_{g^*}$ in $\mathsf{Hyb}_{\mathsf{conf}}$ since r_{g^*} is uniformly random.

Finally, it is easy to see that Hyb_8 is exactly the same as $\mathsf{Hyb}_{\mathsf{conf}}$.

■

5.3 Secret-Key FE from Our Adaptive Garbling

We slightly modify the garbling scheme in Section 5.1. The modification is as follows. Instead of running procedure $\mathsf{GbCkt}(1^\lambda, C)$ (described in Section 5.1), which would sample fresh values for the state information, it runs a slightly modified procedure $\mathsf{GbCkt}^*(1^\lambda, C, \mathsf{MSK})$ which takes the state information as an external input. That is, we set $\mathsf{MSK} := (r, \mathsf{sek}, \{\mathsf{label}_{k,b}^{n+1}\}, \{\mathsf{label}_{k,b}^{n+1,\mathsf{crs}}\}, \mathsf{crs})$. Note that it is not a problem that crs is included in MSK because a master secret key of secret-key FE is never public in the setting of secret-key FE and a simulator adaptively set a master secret-key in the security game.

1-key and 1-bounded ciphertext-adaptively secure and succinct SKFE. First, we present 1-key and 1-bounded ciphertext-adaptively secure and succinct SKFE scheme $\mathsf{ctadFE}_{\mathsf{gs}}^{\text{1-1}} = (\mathsf{Setup}', \mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$, which is based on the garbling scheme $\mathsf{adGC}'_{\mathsf{gs}}$ in Section 5.1.

$\mathsf{Setup}'(1^\lambda)$**:** This algorithm outputs a master secret-key MSK as follows.

1. Generates $\mathsf{sek} \leftarrow \mathsf{KeyGen}(1^\lambda)$, and chooses $r \leftarrow \{0,1\}^N$.
2. Generates $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.

3. Chooses $\mathsf{label}^{g}_{k,b} \leftarrow \{0,1\}^{\lambda}$ and $\mathsf{label}^{g,\mathsf{crs}}_{k,b} \leftarrow \{0,1\}^{\lambda}$ for $g \in [n+1, N+1]$, $k \in [\lambda]$, and $b \in \{0,1\}$.
4. Outputs $\mathsf{MSK} := (r, \mathsf{sek}, \{(\mathsf{label}^{g}_{k,b}, \mathsf{label}^{g,\mathsf{crs}}_{k,b})\}_{k\in[\lambda],b\in\{0,1\},g\in[n+1,N+1]}, \mathsf{crs})$.

$\mathsf{KG}'(\mathsf{MSK}, f)$**:** This algorithm outputs a functional decryption key sk_f as follows.

1. Parses $\mathsf{MSK} = (r, \mathsf{sek}, \{(\mathsf{label}^{g}_{k,b}, \mathsf{label}^{g,\mathsf{crs}}_{k,b})\}_{k\in[\lambda],b\in\{0,1\},g\in[n+1,N+1]}, \mathsf{crs})$.
2. From $g = N$ to $g = n+1$ (decrement g) of f, does the following.
   - Interprets gate g as (i, j).
   - Computes $\widetilde{\mathsf{SC}}_g \leftarrow \mathsf{GC.Grbl}(1^{\lambda}, \mathsf{SC}[(r_i, r_j, r_g), (i, j), \{\mathsf{label}^{g+1}_{k,b}\}, \{\mathsf{label}^{g+1,\mathsf{crs}}_{k,b}\}, 0], (\{\mathsf{label}^{g}_{k,b}\}, \{\mathsf{label}^{g,\mathsf{crs}}_{k,b}\}))$.
3. Generates $\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{sek}, \{\widetilde{\mathsf{SC}}_g\}_{g\in[n+1,N]})$.
4. Outputs $\mathsf{sk}_f := \boldsymbol{c}$.

$\mathsf{Enc}'(\mathsf{MSK}, \mathsf{m})$**:** This algorithm outputs a ciphertext CT_{m} as follows.

1. Parses $\mathsf{MSK} = (r, \mathsf{sek}, \{(\mathsf{label}^{g}_{k,b}, \mathsf{label}^{g,\mathsf{crs}}_{k,b})\}_{k\in[\lambda],b\in\{0,1\},g\in[n+1,N+1]}, \mathsf{crs})$.
2. Sets $D := r_1 \oplus \mathsf{m}_1 \| \cdots \| r_n \oplus \mathsf{m}_n \| 0^{N-n}$.
3. Computes $(d, \widehat{D}) := \mathsf{Hash}(\mathsf{crs}, D)$.
4. Outputs $\mathsf{CT}_{\mathsf{m}} := (\{\mathsf{label}^{n+1}_{k,d_k}\}_{k\in[\lambda]}, \{\mathsf{label}^{n+1,\mathsf{crs}}_{k,\mathsf{crs}_k}\}_{k\in[\lambda]}, \mathsf{crs}, r_1 \oplus \mathsf{m}_1, \ldots, r_n \oplus \mathsf{m}_n, \mathsf{sek}, r_{N-m+1}, \ldots, r_N)$.

$\mathsf{Dec}'(\mathsf{sk}_f, \mathsf{CT}_{\mathsf{m}})$**:** This algorithm runs as follows.

1. Parses $\mathsf{sk}_f = \boldsymbol{c}$ and $\mathsf{CT}_{\mathsf{m}} = (\{\mathsf{label}_k\}_{k\in[\lambda]}, \{\mathsf{label}^{\mathsf{crs}}_k\}_{k\in[\lambda]}, r_1 \oplus \mathsf{m}_1, \ldots, r_n \oplus \mathsf{m}_n, \mathsf{sek}, r_{N-m+1}, \ldots, r_N)$.
2. Sets $D := r_1 \oplus \mathsf{m}_1 \| \cdots \| r_n \oplus \mathsf{m}_n \| 0^{N-n}$.
3. Computes $(d, \widehat{D}) := \mathsf{Hash}(\mathsf{crs}, D)$.
4. Computes $\{\widetilde{\mathsf{SC}}_g\}_{g\in[n+1,N]} \leftarrow \mathsf{Dec}(\mathsf{sek}, \boldsymbol{c})$.
5. Set $\overline{\mathsf{label}} := \{\mathsf{label}_k\}_{k\in[\lambda]}$ and $\overline{\mathsf{label}}^{\mathsf{crs}} := \{\mathsf{label}^{\mathsf{crs}}_k\}_{k\in[\lambda]}$.
6. For $g = n+1, \ldots, N$
   - Interprets g as (i, j).
   - Computes $(\mathsf{gout}_1, \mathsf{gout}_2) := \mathsf{GC.Eval}(\widetilde{\mathsf{SC}}_g, (\overline{\mathsf{label}}, \overline{\mathsf{label}}^{\mathsf{crs}}))$.
   - Computes $(\gamma, e) := \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{gout}_1, i), j)$.
   - Sets $\overline{\mathsf{label}} := \mathsf{ReceiveWrite}^{\widehat{D}}(\mathsf{crs}, g, \gamma, e)$ and $\overline{\mathsf{label}}^{\mathsf{crs}} := \mathsf{gout}_2$.
7. Reads D from $\widehat{D}$.
8. Outputs $D_{N-m+1} \oplus r_{N-m+1} \| \cdots \| D_N \oplus r_N$.

Theorem 5.19. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE, then* $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}} = (\mathsf{Setup}', \mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$ *is* 1*-key and* 1*-bounded ciphertext-adaptively secure SKFE scheme for circuits whose encryption algorithm runs in time* $O(n+m) + \mathrm{poly}(\log |C|, \lambda)$*. Moreover, if we set* $m = 1$*, then* $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}}$ *is* 1*-key and* 1*-bounded ciphertext-adaptively secure and succinct SKFE for boolean circuits.*

Note that we use the hash update trick explained in Section 5.1 for the efficiency.

Proof. This immediately follows from Theorem 5.2. ■

Adaptively secure, single-key, single-ciphertext, and succinct SKFE. Our adaptively secure scheme $\mathsf{adFE}^{\text{1-1}}_{\mathsf{gs}} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is based on $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}} = (\mathsf{Setup}', \mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$. Let $\ell(m)$ be the upper bound of the length of a ciphertext of $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}}$ for a message of length m.

$\mathsf{Setup}(1^\lambda)$**:** This algorithm outputs a master secret-key MSK as follows.

1. Generates $\mathsf{MSK}' \leftarrow \mathsf{Setup}'(1^\lambda)$, and chooses a random mask $\rho \leftarrow \{0,1\}^{\ell(m)}$.
2. Outputs $\mathsf{MSK} := (\mathsf{MSK}', \rho)$.

$\mathsf{KG}(\mathsf{MSK}, f)$**:** This algorithm outputs a functional decryption key sk_f as follows.

1. Parses $\mathsf{MSK} = (\mathsf{MSK}', \rho)$.
2. Generates $\mathsf{sk}'_f \leftarrow \mathsf{KG}'(\mathsf{MSK}', f)$.
3. Outputs $\mathsf{sk}_f := (\rho, \mathsf{sk}'_f)$.

$\mathsf{Enc}(\mathsf{MSK}, \mathsf{m})$**:** This algorithm outputs a ciphertext CT_m as follows.

1. Parses $\mathsf{MSK} = (\mathsf{MSK}', \rho)$.
2. Computes $\mathsf{CT}'_\mathsf{m} \leftarrow \mathsf{Enc}'(\mathsf{MSK}', \mathsf{m})$.
3. Outputs $\mathsf{CT}_\mathsf{m} := \mathsf{CT}'_\mathsf{m} \oplus \rho$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}_\mathsf{m})$**:** This algorithm decrypts as follows.

1. Parses $\mathsf{sk}_f = (\rho, \mathsf{sk}'_f)$
2. Compute $\mathsf{CT}'_\mathsf{m} = \mathsf{CT}_\mathsf{m} \oplus \rho$.
3. Computes and outputs $y' \leftarrow \mathsf{Dec}'(\mathsf{sk}'_f, \mathsf{CT}'_\mathsf{m})$.

Theorem 5.20. *If* $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}}$ *is* 1*-key and* 1*-bounded ciphertext-adaptively secure succinct SKFE for (boolean) circuits, then* $\mathsf{adFE}^{\text{1-1}}_{\mathsf{gs}}$ *is* $(1, 1, \mathsf{ada}, \mathsf{fs})$*-SKFE for (boolean) circuits.*

It is easy to see that the correctness holds. We give the proof of Theorem 5.20 in Appendix B.1 since it is a standard proof.

Combining Theorems 5.19 and 5.20 we obtain the following theorem.

Theorem 5.21. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists* $(1, 1, \mathsf{ada}, \mathsf{fs})$*-SKFE for boolean circuits.*

By combining Theorems 3.20 and 5.21, we obtain the following theorem.

Theorem 5.22. *If there exists* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits, then there exists* $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$*-PKFE for boolean circuits.*

6 Adaptively Secure, Collusion-Resistant, and Succint FE

In this section, we show a conversion from collusion-resistant PKFE for boolean circuits to one for all circuits without sacrificing succinctness. Combined with Theorem 5.22, this gives our main theorem, Theorem 1.1.

6.1 From Single-bit to Multi-bit Succinct FE by Leveraging Collusion-Resistance

Let $\mathsf{OnePKFE} = (\mathsf{OnePKFE.Setup}, \mathsf{OnePKFE.KG}, \mathsf{OnePKFE.Enc}, \mathsf{OnePKFE.Dec})$ be an PKFE scheme for $\mathcal{M}, \mathcal{Y}' := \{0,1\}$, and boolean circuits. We construct an PKFE scheme $\mathsf{MultiPKFE} = (\mathsf{MultiPKFE.Setup}, \mathsf{MultiPKFE.KG}, \mathsf{MultiPKFE.Enc}, \mathsf{MultiPKFE.Dec})$ for $\mathcal{M}$, $\mathcal{Y} := \{0,1\}^\ell$, and circutis as follows.

$\mathsf{MultiPKFE.Setup}(1^\lambda)$:

1. Computes $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{OnePKFE.Setup}(1^\lambda)$.
2. Outputs $(\mathsf{MPK}, \mathsf{MSK})$.

$\mathsf{MultiPKFE.KG}(\mathsf{MSK}, f)$:

1. Computes $\mathsf{sk}_i \leftarrow \mathsf{OnePKFE.KG}(\mathsf{MSK}, f_i)$ for every $i \in [\ell]$ where $f_i(\mathsf{m})$ outputs the i-th bit of $f(\mathsf{m})$.
2. Outputs $\mathsf{sk}_f := \{\mathsf{sk}_{f_i}\}_{i\in[\ell]}$.

$\mathsf{MultiPKFE.Enc}(\mathsf{MSK}, \mathsf{m})$:

1. Computes $\mathsf{CT}_\mathsf{m} \leftarrow \mathsf{OnePKFE.Enc}(\mathsf{MSK}, \mathsf{m})$.
2. Outputs $\mathsf{CT} := \mathsf{CT}_\mathsf{m}$.

$\mathsf{PKFE.Dec}(\mathsf{sk}_f, \mathsf{CT}_\mathsf{m})$:

1. Parses $\{\mathsf{sk}_{f_i}\}_{i\in[\ell]} \leftarrow \mathsf{sk}_f$.
2. Computes $y_i \leftarrow \mathsf{OnePKFE.Dec}(\mathsf{sk}_{f_i}, \mathsf{CT}_\mathsf{m})$ for every $i \in [\ell]$.
3. Outputs $y := y_1 \| \dots \| y_\ell$.

Correctness. Correctness of MultiPKFE easily follows from correctness of OnePKFE.

Security. The security of MultiPKFE can be stated as follows.

Theorem 6.1. *If* $\mathsf{OnePKFE}$ *is* $(\mathsf{unb}, \mathsf{sec}, \mathsf{eff})$*-PKFE for boolean circuits, then* $\mathsf{MultiPKFE}$ *is* $(\mathsf{unb}, \mathsf{sec}, \mathsf{eff})$*-PKFE for circuits where* $\mathsf{sec} \in \{\mathsf{w\text{-}sel}, \mathsf{sel}, \mathsf{ada}\}$ *and* $\mathsf{eff} \in \{\mathsf{ns}, \mathsf{sls}, \mathsf{fs}\}$.

This can be proven by a standard hybrid argument.

The running time of encryption algorithm. Since the encryption algorithm of MultiPKFE only runs the encryption algorithm of OnePKFE, the running time of MultiPKFE.Enc is the same as that of OnePKFE.Enc. OnePKFE.Enc is succinct, so MultiPKFE.Enc is.

6.2 Fully-Equipped PKFE

By combining Theorems 5.22 and 6.1, we obtain the main theorem in this study, that is, Theorem 1.1. We obtain adaptively secure, collusion-resistant, and succinct public-key FE for circuits from weakly-selecitvely secure, single-key, and sublinearly-succinct puclic-key FE for circuits.

7 Adaptively Indistinguishable Garbling with Near-Optimal Online Complexity

In this section, we give a construction of an adaptively indistinguishable garbling scheme for all circuits whose online complexity does not depend on output-length of the circuit to garble. Namely, the length of online part in our construction is $2n + \mathrm{poly}(\log |C|, \lambda)$ where n and $|C|$ denote the input-length and circuit size, respectively. Our construction is generic, and can be instantiated based on any adaptive garbling scheme with a special structure called *quasi-decomposability* and a weaker security called *input-privacy-free security*. Our adaptively indistinguishable garbling scheme for all circuits can be obtained by plugging the input-privacy-free variant of the adaptive garbling scheme given in Section 5 (or GS18 scheme [GS18]) into the above generic construction. We also note that we can instantiate the generic construction based on input-privacy-free variants of known adaptive garbling schemes based on one-way functions [HJO+16, JW16]. This gives an alternative modular construction of adaptively indistinguishable garbling schemes for certain function classes, which were originally constructed in an ad hoc manner by Jafargholi et al. [JSW17].

7.1 Roadmap

Here, we describe a roadmap of this section. In Section 7.2, we introduce a cryptographic primitive called secret key non-committing encryption for reveiver (SK-NCER), which is used in our construction of SKFE from a garbling scheme in Section 7.4, and show that an SK-NCER scheme can be constructed based on any SKE scheme. In Section 7.3, we define a property called the quasi-decomposability for garbling schemes, and observe that a slight variant of the adaptive garbling scheme given in Section 5 (and the one given by Garg and Srinivasan [GS18]) have the quasi-decomposability. Then we convert a quasi-decomposable adaptive garbling scheme with online communication complexity $n + m + \mathrm{poly}(\log |C|, \lambda)$ and online computational complexity $O(n + m) + \mathrm{poly}(\log |C|, \lambda)$ to an adaptively indistinguishable garbling scheme with online communication complexity $2n + \mathrm{poly}(\log |C|, \lambda)$ and online computational complexity $O(n) + \mathrm{poly}(\log |C|, \lambda)$ where $C : \{0,1\}^n \to \{0,1\}^m$ is the circuit being garbled. The conversion consists of the following three steps:

- In Section 7.4, we give a construction of SKFE scheme for 1-bit functions with succinct key generation based on quasi-decomposable with succinct online-complexity. Here, succinct key generation means that the computational complexity of the key generation algorithm (and thus the decryption-key-length) only logarithmically depend on the message-length. This part is based on a similar construction to the one by Gorbunov, Vaikuntanathan and Wee [GVW12].
- In Section 7.5, we convert SKFE scheme for 1-bit functions to one for multi-bit functions without losing efficiecy of key generation. This part is based on a very simple idea to decompose a function in a output-bit-wise manner similarly to the conversion in Section 6.1.
- In Section 7.6, we give a construction of adaptively indistinguishable garbling scheme (for multi-bit functions) with succinct online complexity based on an SKFE scheme (for multi-bit functions) with succinct key generation. This part is based on the work by Brakerski and Segev [BS15], which showed how to add function privacy to any SKFE scheme.

By applying the above transformations to a slight variant of the adaptive garbling scheme given in Section 5 (or GS18 scheme [GS18]), we obtain an adaptively indistinguishable garbling scheme with succinct online complexity as desired. In Section 7.7, we also discuss instantiations of the above conversions with adaptive garbling schemes based on one-way functions [HJO+16, JW16].

7.2 Secret-Key Non-Committing Encryption for Receiver

Here, we define a secret-key non-committing encryption for receiver (SK-NCER), which is the secret-key variant of NCER [JL00, CHK05]. Roughly speaking, SK-NCER is SKE which allows one to generate a "fake" ciphertext so that it can be later revealed to any message along with a "fake" decryption key. The formal definition is given below.

Definition 7.1 (Secret-key non-committing encryption for receiver). *A secret key non-committing encryption for reveiver (SK-NCER) scheme* NCER *is a five tuple* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal})$ *of PPT algorithms.*

- *The key generation algorithm* KG*, given a security parameter* 1^λ *and message length* 1^{ℓ_m}*, outputs an encryption key* ek *and decryption key* dk*.* [20]
- *The encryption algorithm* Enc*, given an encryption key* ek *and message* $m \in \{0,1\}^{\ell_m}$*, outputs a ciphertext* CT*.*
- *The decryption algorithm* Dec*, given a decryption key* dk *and ciphertext* CT*, outputs a message* $m \in \{0,1\}^{\ell_m}$*.*
- *The fake algorithm* Fake*, given an encryption key* ek*, outputs a fake ciphertext* CT *and state* st*.*
- *The reveal algorithm* Reveal*, given a state* st *and message* $m^* \in \{0,1\}^{\ell_m}$*, outputs a fake decryption key* dk^**.*

Correctness *We require* $\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m)) = m$ *for every* $m \in \{0,1\}^{\ell_m}$ *and* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda, 1^{\ell_m})$*.*

Security *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a PPT adversary. We consider the following game between* $\mathcal{A}$ *and a challenger for any* $\lambda \in \mathbb{N}$ *and* $\ell_m \in \mathbb{N}$*.*

1. *The challenger chooses the challenge bit* $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$*, generates* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KG}(1^\lambda, 1^{\ell_m})$ *and sends security parameter* 1^λ *and message length* 1^{ℓ_m} *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *makes arbitrarily many encryption queries and single challenge query in any order.*

   Encryption query. *When* $\mathcal{A}$ *makes an encryption query* $m \in \{0,1\}^{\ell_m}$*, the challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ *and returns* CT *to* $\mathcal{A}$*.*

   Challenge query. *When* $\mathcal{A}$ *makes a challenge query* $m^* \in \{0,1\}^{\ell_m}$*, the challenger returns* $(\mathsf{CT}^*, \mathsf{dk}^*)$ *generated as follows:*
   - *If* $\mathsf{coin} = 0$*, then it computes* $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m^*)$ *and* $\mathsf{dk}^* := \mathsf{dk}$*.*
   - *If* $\mathsf{coin} = 1$*, then it computes* $(\mathsf{CT}^*, \mathsf{st}) \leftarrow \mathsf{Fake}(\mathsf{ek})$ *and* $\mathsf{dk}^* \leftarrow \mathsf{Reveal}(\mathsf{st}, m^*)$*.*
3. $\mathcal{A}$ *outputs* $\mathsf{coin}' \in \{0,1\}$*.*

In this game, we define the advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{ncer}}_{\mathsf{NCER},\mathcal{A}}(\lambda, \ell_m) = 2\left|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}\right| = \left|\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]\right| .$$

We say that NCER *is secure if for any PPT* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{ncer}}_{\mathsf{NCER},\mathcal{A}}(\lambda, \ell_m) = \mathsf{negl}(\lambda)$ *fot any* $\ell_m = \mathrm{poly}(\lambda)$*.*

[20] One may wonder why encryption and decryption keys are separately defined though we consider a secret-key primitive. The reason is that the security of SK-NCER involves an adversary that obtains a decryption key, and we cannot include all secret information in a decryption key.

Construction. Here, we give a construction of an SK-NCER scheme based on any CPA-secure SKE scheme, which in turn can be constructed based on any one-way function. Let $\mathsf{SKE} = (\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a CPA-secure SKE scheme on the message space $\{0,1\}$. We construct an SK-NCER scheme $\mathsf{NCER} = (\mathsf{NCER.KG}, \mathsf{NCER.Enc}, \mathsf{NCER.Dec}, \mathsf{NCER.Fake}, \mathsf{NCER.Reveal})$ as follows.

$\mathsf{NCER.KG}(1^\lambda, 1^{\ell_m})$:

1. Generates $K_{i,b} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for every $i \in [\ell_m]$ and $b \in \{0,1\}$.
2. Generates $x \xleftarrow{\mathsf{r}} \{0,1\}^{\ell_m}$.
3. Outputs $\mathsf{ek} := \{K_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}$ and $\mathsf{dk} := \left(x, \{K_{i,x[i]}\}_{i\in[\ell_m]}\right)$.

$\mathsf{NCER.Enc}(\mathsf{ek}, m)$:

1. Parses $\{K_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}} \leftarrow \mathsf{ek}$.
2. Generates $\mathsf{CT}_{i,b} \leftarrow \mathsf{SKE.Enc}(K_{i,b}, m[i])$ for every $i \in [\ell_m]$ and $b \in \{0,1\}$.
3. Outputs $\mathsf{CT} := \{\mathsf{CT}_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}$.

$\mathsf{NCER.Dec}(\mathsf{dk}, \mathsf{CT})$:

1. Parses $\left(x, \{K_i\}_{i\in[\ell_m]}\right) \leftarrow \mathsf{dk}$ and $\{\mathsf{CT}_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}} \leftarrow \mathsf{CT}$.
2. Compute $m[i] \leftarrow \mathsf{SKE.Dec}(K_i, \mathsf{CT}_{i,x[i]})$ for every $i \in [\ell_m]$.
3. Outputs $m := m[1]\| \ldots \|m[\ell_m]$.

$\mathsf{NCER.Fake}(\mathsf{ek})$:

1. Parses $\{K_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}} \leftarrow \mathsf{ek}$.
2. Generates $x^* \xleftarrow{\mathsf{r}} \{0,1\}^{\ell_m}$.
3. Compute $\mathsf{CT}_{i,x^*[i]} \leftarrow \mathsf{SKE.Enc}(K_{i,x^*[i]}, 0)$ and $\mathsf{CT}_{i,1-x^*[i]} \leftarrow \mathsf{SKE.Enc}(K_{i,1-x^*[i]}, 1)$ for every $i \in [\ell_m]$.
4. Outputs $\mathsf{CT} := \{\mathsf{CT}_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}$ and $\mathsf{st} := \left(x^*, \{K_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}\right)$.

$\mathsf{NCER.Reveal}(\mathsf{st}, m^*)$:

1. Parses $\left(x^*, \{K_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}\right) \leftarrow \mathsf{st}$.
2. Outputs $\mathsf{dk}^* := \left(x^* \oplus m^*, \{K_{i,x^*[i]\oplus m^*[i]}\}_{i\in[\ell_m]}\right)$.

Correctness of NCER easily follows from correctness of SKE.

The security of NCER is stated as follows.

Theorem 7.2. *If* SKE *is CPA-secure, then* NCER *is a secure SK-NCER scheme.*

Proof. Let $\mathcal{A}$ be a PPT adversary that attacks the security of NCER. We consider the following sequence of experiments.

Exp 0**:** This is the experiment defining the security of SK-NCER in which coin is fixed to 0. Specifically, this is described as follows.

1. The challenger generates $K_{i,b} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for every $i \in [\ell_m]$ and $b \in \{0,1\}$ and $x \xleftarrow{\mathsf{r}} \{0,1\}^{\ell_m}$, and sends security parameter 1^λ and message length 1^{ℓ_m} to $\mathcal{A}$.
2. $\mathcal{A}$ makes arbitrarily many encryption queries and single challenge query in any order.

Encryption query. When $\mathcal{A}$ makes an encryption query $m \in \{0,1\}^{\ell_m}$, the challenger computes $\mathsf{CT}_{i,b} \leftarrow \mathsf{SKE.Enc}(K_{i,b}, m[i])$ for every $i \in [\ell_m]$ and $b \in \{0,1\}$, and returns $\mathsf{CT} := \{\mathsf{CT}_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}$ to $\mathcal{A}$.

Challenge query. When $\mathcal{A}$ makes a challenge query $m^* \in \{0,1\}^{\ell_m}$, the challenger computes $\mathsf{CT}^*_{i,b} \leftarrow \mathsf{SKE.Enc}(K_{i,b}, m^*[i])$ for every $i \in [\ell_m]$ and $b \in \{0,1\}$, sets $\mathsf{CT}^* := \{\mathsf{CT}^*_{i,b}\}_{i\in[\ell_m],b\in\{0,1\}}$, and returns $(\mathsf{CT}^*, \mathsf{dk})$.

3. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Exp 1**:** This experiment is identical to the previous experiment except that $\mathsf{CT}^*_{i,1-x[i]}$ is generated by $\mathsf{SKE.Enc}(K_{i,1-x[i]}, 1-m[i])$ for every $i \in [\ell_m]$. Since $\{K_{i,1-x[i]}\}_{i\in[\ell_m]}$ is not given to $\mathcal{A}$, $\mathcal{A}$'s advantage to distinguish these two experiments is negligible by the CPA-security of SKE.

Exp 2**:** This experiment is identical to the previous experiment except that the challenger generates $x^* \xleftarrow{\mathsf{r}} \{0,1\}^{\ell_m}$, sets $\mathsf{CT}^*_{i,x[i]^*} \leftarrow \mathsf{NCER.Enc}(K_{i,x^*[i]}, 0)$ and $\mathsf{CT}^*_{i,1-x[i]^*} \leftarrow \mathsf{NCER.Enc}(K_{i,1-x^*[i]}, 1)$, and $\mathsf{dk} := \left(x^* \oplus m^*, \{K_{i,x[i]^*\oplus m^*[i]}\}_{i\in[\ell_m]}\right)$. $\mathcal{A}$'s advantage to distinguish these two experiments is 0 since we can see that these two experiments are identical if we set $x := x^* \oplus m^*$.

Here, we notice that Exp 2 is exactly identical to the experiment defining the security of SK-NCER in which coin is fixed to 1. The above sequence of experimants demonstrates that $\mathcal{A}$ cannot distinguish the two cases of $\mathsf{coin} = 0$ and $\mathsf{coin} = 1$, which concludes the security proof of NCER. ■

7.3 Additional Definitions for Adaptive Garbling.

Here, we define additional properties of adaptive garbling scheme that is needed in this section. Let $\mathsf{AdaGC} = (\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ be a garbling scheme. We define input-privacy-free adaptive security and quasi-decomposability for AdaGC as follows.

Input-privacy-free adaptive security: Here, we define a weaker security notion for adaptive garbling which we call *input-privacy-free security*. Roughly speaking, this security notion allows an online encoding $\widetilde{x}$ to reveal x itself, whereas a garbled circuit $\widetilde{C}$ still hides C. This is captured by giving an input x to a simulator. The formal definition is given below.

Let $\mathsf{Sim} = (\mathsf{SimC}, \mathsf{SimIn})$ be a tuple PPT algorithm. We define the following game between a challenger and an adversary $\mathcal{A}$ as follows.

1. The challenger chooses the challenge bit $\mathsf{coin} \xleftarrow{\mathsf{r}} \{0,1\}$ and sends security parameter 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ sends a circuit C with n-bit input to the challenger, and the challenger returns $\widetilde{C}$ generated as follows:
   - If $\mathsf{coin} = 0$, then it computes $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{GbCkt}(1^\lambda, C)$.
   - If $\mathsf{coin} = 1$, then it computes $(\widetilde{C}, \mathsf{st}) \leftarrow \mathsf{SimC}(1^\lambda, 1^{|C|})$.
3. $\mathcal{A}$ sends an input $x \in \{0,1\}^n$ to the challenger, and the challenger returns $\widetilde{x}$ generated as follows:
   - If $\mathsf{coin} = 0$, then it computes $\widetilde{x} \leftarrow \mathsf{GbInp}(\mathsf{st}, x)$.
   - If $\mathsf{coin} = 1$, then it computes $\widetilde{x} \leftarrow \mathsf{SimIn}(\mathsf{st}, x)$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

In this game, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{ipf\text{-}adp}}_{\mathsf{AdaGC},\mathcal{A},\mathsf{Sim}}(\lambda) = 2\left|\Pr[\mathsf{coin} = \mathsf{coin}'] - \frac{1}{2}\right| = \left|\Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 0] - \Pr[\mathsf{coin}' = 1 \mid \mathsf{coin} = 1]\right| .$$

We say that AdaGC is adaptively input-privacy-free-secure if there exists PPT Sim such that for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{ipf\text{-}adp}}_{\mathsf{AdaGC},\mathcal{A},\mathsf{Sim}}(\lambda) = \mathsf{negl}(\lambda)$.

It is easy to see that the adaptive security in Definition 3.10 implies the input-privacy-free adaptive security.

Quasi-decomposability. We say that $\mathsf{AdaGC} = (\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ for a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a quasi-decomposable garbling scheme associated with PPT algorithms $(\mathsf{HashGen}, \mathsf{AuxGen}, \mathsf{GbCkt}')$ if GbCkt and GbInp are of the form given below:

- $\mathsf{GbCkt}(1^\lambda, C)$ takes as inputs security parameter 1^λ and a circuit $C \in \mathcal{C}_\lambda$ with n-bit input, generates a polynomial-time-computable hash function $\mathsf{H} \leftarrow \mathsf{HashGen}(1^\lambda, 1^{|C|})$ with n-bit input and ℓ_α-bit output whose description size is ℓ_{H}, auxiliary information $\mathsf{aux} \leftarrow \mathsf{AuxGen}(1^\lambda, 1^{|C|})$ whose length is ℓ_{aux}, and labels $\mathsf{label}_{k,b} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, computes $\widetilde{C} \leftarrow \mathsf{GbCkt}'\left(C, \mathsf{aux}, \{\mathsf{label}_{k,b}\}_{k\in[\ell],b\in\{0,1\}}\right)$, and outputs a garbled circuit $\widetilde{C}$ and a state information $\mathsf{st} = \left(\mathsf{H}, \mathsf{aux}, \{\mathsf{label}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$.
- $\mathsf{GbInp}(\mathsf{st}, x)$ parses $\left(\mathsf{H}, \mathsf{aux}, \{\mathsf{label}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right) \leftarrow \mathsf{st}$, computes $\alpha = \mathsf{H}(x)$, and outputs a garbled input $\widetilde{x} = \left(x, \mathsf{H}, \mathsf{aux}, \{\mathsf{label}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\right)$.

We denote the description size of H by ℓ_{H}, the length of α by ℓ_α, and the length of aux by ℓ_{aux}. We call them hash-description-size, hash-output-length, auxiliary-information-length, redpectivaly. We note that these parameters depend on the circuit class $\mathcal{C}$. We omit to explicitly state them as a function of $\mathcal{C}$ for notational simplicity.

Remark 7.3. Since $\widetilde{x}$ contains x itself, a quasi-decomposable garbling scheme cannot satisfy a usual security definition of garbling schemes. On the other hand, quasi-decomposable garbling scheme may satisfy the input-privacy-free security as discussed below.

Examples. Here, we give some examples of quasi-decomposable garbling scheme that satisfies the input-privacy-free adaptive security.

First, we consider an "input-privacy-free variant" of the scheme given in Section 5 (or GS18 scheme [GS18]). Namely, we set "input masks" $(r_1, ..., r_n)$ to be 0^n instead of choosing them randomly. We can see that this variant has the quasi-decomposability by defining $\mathsf{H}(x) := (\mathsf{Hash}(\mathsf{crs}, (x_1\|\cdots\|x_n\|0^{N-n}), \mathsf{crs}),\ \mathsf{aux} := (\mathsf{sek}, r_{N-m+1}, \ldots, r_N)$. In this construction, we have $\ell_{\mathsf{H}} = \mathrm{poly}(\lambda)$, $\ell_\alpha = \mathrm{poly}(\lambda)$, $\ell_{\mathsf{aux}} = \mathrm{poly}(\log|C|, \lambda) + m$, and computational complexity of H is $O(n) + \mathrm{poly}(\log|C|, \lambda)$. We can see that this variant satisfies the input-privacy-free adaptive security in almost the same manner as the security proof of the adaptive security given in Section 5.

Second, we remark that "input-privacy-free variants" of adaptive garbling schemes based on one-way functions [HJO+16, JW16] also satisfy the quasi-decomposability and input-privacy-free adaptive security. An online encoding for an input x in the scheme in [HJO+16] is of the form $\widetilde{x} = \left(\mathsf{sek}, \{\mathsf{label}_{k,x_k}\}_{k\in[n],b\in\{0,1\}}\right)$ where sek is a key of somewhere equivocal encryption. If we consider a variant of this scheme in which x itself is included in $\widetilde{x}$, then this variant satisfies the quasi-decomposaibility by setting H to be the identity function and $\mathsf{aux} := \mathsf{sek}$. Moreover, it is easy to reduce the input-privacy-free adaptive security of the variant to the adaptive security of the original scheme. In this construction, we have $\ell_{\mathsf{H}} = 0$, $\ell_\alpha = n$, and $\ell_{\mathsf{aux}} = (n + m + t)\mathrm{poly}(\lambda)$ where m is the output-length and t is a parameter called the pebble complexity of the circuit being garbled. In particular, we can set $t = w$ where w is the width of the circuit, or $t = d$ where d is the depth of the circuit if the circuit is in NC^1, i.e., $d = O(\log n)$. See [HJO+16] for the details of the pebble complexity. We can also see that input-privacy-free variant of the scheme in [JW16] (which is essentially the same as the Yao's garbled circuit) satisfies the similar property.

7.4 SKFE from Quasi-Decomposable Garbling

Here, we construct a 1-bounded key-adaptively secure SKFE scheme based on quasi-decomposable input-privacy-free adaptive garbling. (Recall that the key-adaptive security means the security against adversaries that does not make any challenge query after making its first key query.) A good feature of the scheme is that the complexity of key generation algorithm only logarithmically depends on the message length if the online computational complexity only logarithmically depends on the size of circuit being garbled. To state this property more precisely, we first define a slight variant of SKFE called *SKFE for tagged functions*.

SKFE for tagged functions. Let $\mathcal{F} = \{f_{\tau,\lambda} : \{0,1\}^{\ell_{\mathsf{m}}} \to \{0,1\}^*\}_{\tau\in\{0,1\}^{\ell_\tau},\lambda\in\mathbb{N}}$ be a class of polynomial-time-computable functions f_τ indexed by a tag $\tau \in \{0,1\}^{\ell_\tau}$. An SKFE scheme for a class $\mathcal{F}$ of tagged functions is similarly defined to an SKFE scheme as defined in Definition 3.12 except that the key generation and decryption algorithm take a tag τ as input instead of the description of the function f_τ. As an efficiency requirement, we require that the running time and decryption key size are $\mathrm{poly}(\lambda, \ell_\tau, \log \ell_{\mathsf{m}})$. Especially, they only logarithmically depend on the input-length ℓ_{m} of a function. We note that this efficiency requirement is meaningful when ℓ_τ is much smaller than ℓ_{m}. Looking ahead, this is needed in the construction of a garbling scheme from an SKFE scheme where a message and decryption key correspond to a garbled circuit and garbled input, respectively. Since we want to construct a garbling scheme whose online complexity only logarithmically depends on the circuit size, we have to assume that the decryption-key-length of the underlying SKFE scheme only logarithmically depends on the message-length as defined above. In the following, we omit λ from subscripts, and simply denote a family of tagged functions as $\mathcal{F} = \{f_\tau : \{0,1\}^{\ell_{\mathsf{m}}} \to \{0,1\}^*\}_{\tau\in\{0,1\}^{\ell_\tau}}$ for notational simplicity.

Construction. Let $\mathcal{F} = \{f_\tau : \{0,1\}^{\ell_{\mathsf{m}}} \to \{0,1\}^*\}_{\tau\in\{0,1\}^{\ell_\tau}}$ be a class of polynomial-time-computable tagged functions. Let $\mathsf{NCER} = (\mathsf{NCER.KG}, \mathsf{NCER.Enc}, \mathsf{NCER.Dec}, \mathsf{NCER.Fake}, \mathsf{NCER.Reveal})$ be an SK-NCER scheme. For $\mathsf{m} \in \{0,1\}^{\ell_{\mathsf{m}}}$, U_{m} denotes a circuit such that $U_{\mathsf{m}}(\tau) = f_\tau(\mathsf{m})$ for any $\tau \in \{0,1\}^{\ell_\tau}$. We assume that for all $\ell_{\mathsf{m}} \in \mathbb{N}$, U_{m} has the same size M for all $\mathsf{m} \in \{0,1\}^{\ell_{\mathsf{m}}}$ by an appropriate padding. Let $\mathsf{AdaGC} = (\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ be a quasi-decomposable garbling scheme with hash-description-size ℓ_{H}, hash-output-length ℓ_α, and auxiliary-information-length ℓ_{aux} associated with $(\mathsf{HashGen}, \mathsf{AuxGen}, \mathsf{GbCkt}')$. We construct an adaptively secure SKFE scheme $\mathsf{SKFE} = (\mathsf{SKFE.Setup}, \mathsf{SKFE.KG}, \mathsf{SKFE.Enc}, \mathsf{SKFE.Dec})$ for the class $\mathcal{F}$ of tagged functions as follows.

$\mathsf{SKFE.Setup}(1^\lambda)$:

1. Generates $\mathsf{H} \leftarrow \mathsf{HashGen}(1^\lambda, 1^M)$.
2. Generates $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^{\ell_{\mathsf{aux}}})$.
3. Generates $(\mathsf{ek}_{k,b}, \mathsf{dk}_{k,b}) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^\lambda)$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$.
4. Outputs $\mathsf{MSK} := \Big(\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\Big)$.

$\mathsf{SKFE.KG}(\mathsf{MSK}, \tau)$:

1. Parses $(\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}) \leftarrow \mathsf{MSK}$.
2. Computes $\alpha := \mathsf{H}(\tau)$.
3. Outputs $\mathsf{sk}_\tau := \Big(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\Big)$.

$\mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m})$:

1. Parses $(\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}) \leftarrow \mathsf{MSK}$.

2. Generates $\mathsf{aux} \leftarrow \mathsf{AuxGen}(1^\lambda, 1^M)$.
3. Generates $\mathsf{label}_{k,b} \xleftarrow{r} \{0,1\}^\lambda$ for $k \in [\ell_\alpha]$ and $b \in \{0,1\}$.
4. Computes $\widetilde{U} \leftarrow \mathsf{GbCkt}'(U_{\mathsf{m}}, \mathsf{aux}, \{\mathsf{label}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}})$.
5. Computes $\mathsf{CT}_0 \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_0, \mathsf{aux})$.
6. Computes $\mathsf{CT}_{k,b} \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_{k,b}, \mathsf{label}_{k,b})$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$.
7. Outputs $\mathsf{SKFE.CT} := \left(\widetilde{U}, \mathsf{CT}_0, \{\mathsf{CT}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$.

$\mathsf{SKFE.Dec}(\tau, \mathsf{sk}_\tau, \mathsf{SKFE.CT})$:

1. Parses $\left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_k\}_{k\in[\ell_\alpha]}\right) \leftarrow \mathsf{sk}_\tau$.
2. Parses $\left(\widetilde{U}, \mathsf{CT}_0, \{\mathsf{CT}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right) \leftarrow \mathsf{SKFE.CT}$.
3. Computes $\mathsf{aux} \leftarrow \mathsf{NCER.Dec}(\mathsf{dk}_0, \mathsf{CT}_0)$.
4. Computes $\alpha := \mathsf{H}(\tau)$.
5. Computes $\mathsf{label}_k \leftarrow \mathsf{NCER.Dec}\left(\mathsf{dk}_k, \mathsf{CT}_{k,\alpha[k]}\right)$ for every $k \in [\ell_\alpha]$.
6. Sets $\widetilde{\tau} := \left(\tau, \mathsf{H}, \mathsf{aux}, \{\mathsf{label}_k\}_{k\in[\ell_\alpha]}\right)$.
7. Outputs $z \leftarrow \mathsf{GbEval}(\widetilde{U}, \widetilde{\tau})$.

Correctness. Let $\mathsf{sk}_\tau = \left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\right)$ and $\mathsf{SKFE.CT} = \left(\widetilde{U}, \mathsf{CT}_0, \{\mathsf{CT}_{k,b}\}_{j\in[\lambda],b\in\{0,1\}}\right)$ be a secret key and ciphertext generated by $\mathsf{SKFE.KG}(\mathsf{MSK}, \tau)$ and $\mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m})$, respectively. By correctness of NCER, we have $\mathsf{aux} = \mathsf{NCER.Dec}(\mathsf{dk}_0, \mathsf{CT}_0)$ and $\mathsf{label}_{k,\alpha[k]} = \mathsf{NCERDec}(\mathsf{dk}_k, \mathsf{CT}_{k,\alpha[k]})$ for every $k \in [\ell_\alpha]$ where aux and $\{\mathsf{label}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}$ are the ones generated in the execution of $\mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m})$ and $\alpha := \mathsf{H}(\tau)$. Then by correctness of AdaGC, we have $\mathsf{GbEval}(\widetilde{U}, \widetilde{\tau}) = U_{\mathsf{m}}(\tau) = f_\tau(\mathsf{m})$ where $\widetilde{\tau} = (\tau, \mathsf{H}, \mathsf{aux}, \{\mathsf{label}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]})$ and $\alpha = \mathsf{H}(\tau)$.

Security. The security of SKFE can be stated as follows.

Theorem 7.4. *If* AdaGC *satisfies the input-privacy-free adaptive security and* NCER *is secure, then* SKFE *is 1-bounded key-adaptively secure.*

Proof. Let $\mathcal{A}$ be a key-adaptively valid PPT adversary that attacks the 1-bounded key-adaptive security of SKFE. For $i \in \{0, ..., q\}$ where q denotes the number of $\mathcal{A}$'s challenge queries and $a \in \{0, 1, 2\}$, we consider experiments Exp $i.a$ as follows.

Exp $i.0$**:** In this experiment, the challenger encrypts m_1 for the first i challenge queries, and m_0 for the rest.

1. The challenger generates $\mathsf{H} \leftarrow \mathsf{HashGen}(1^\lambda, 1^M)$, $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^{\ell_{\mathsf{aux}}})$, and $(\mathsf{ek}_{k,b}, \mathsf{dk}_{k,b}) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^\lambda)$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and sets $\mathsf{MSK} := (\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}})$.
2. $\mathcal{A}$ is given security parameter 1^λ, and $\mathcal{A}$ makes arbitrarily many challenge queries.

   Challenge query. When $\mathcal{A}$ makes its j-th challenge query $(\mathsf{m}_0^{(j)}, \mathsf{m}_1^{(j)})$, the challenger works as follows:

- If $j \leq i$, then the challenger returns $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_1^{(j)})$ to $\mathcal{A}$. Namely, the challenger generates $\mathsf{aux}^{(j)} \leftarrow \mathsf{AuxGen}(1^\lambda, 1^M)$ and $\mathsf{label}_{k,b}^{(j)}$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, computes $\widetilde{U}^{(j)} \leftarrow \mathsf{GbCkt}'(U_{\mathsf{m}_1^{(j)}}, \mathsf{aux}^{(j)}, \{\mathsf{label}_{k,b}^{(j)}\}_{k\in[\ell_\alpha],b\in\{0,1\}})$, $\mathsf{CT}_0^{(j)} \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_0, \mathsf{aux}^{(j)})$, and $\mathsf{CT}_{k,b}^{(j)} \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_{k,b}, \mathsf{label}_{k,b}^{(j)})$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and returns $\mathsf{SKFE.CT}^{(j)} := \left(\widetilde{U}^{(j)}, \mathsf{CT}_0^{(j)}, \{\mathsf{CT}_{k,b}^{(j)}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$ to $\mathcal{A}$.
- If $j > i$, then the challenger returns $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_0^{(j)})$ to $\mathcal{A}$. Namely, the challenger generates $\mathsf{aux}^{(j)} \leftarrow \mathsf{AuxGen}(1^\lambda, 1^M)$ and $\mathsf{label}_{k,b}^{(j)}$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, computes $\widetilde{U}^{(j)} \leftarrow \mathsf{GbCkt}'(U_{\mathsf{m}_0^{(j)}}, \mathsf{aux}^{(j)}, \{\mathsf{label}_{k,b}^{(j)}\}_{k\in[\ell_\alpha],b\in\{0,1\}})$, $\mathsf{CT}_0^{(j)} \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_0, \mathsf{aux}^{(j)})$, and $\mathsf{CT}_{k,b}^{(j)} \leftarrow \mathsf{NCER.Enc}(\mathsf{ek}_{k,b}, \mathsf{label}_{k,b}^{(j)})$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and returns $\mathsf{SKFE.CT}^{(j)} := \left(\widetilde{U}^{(j)}, \mathsf{CT}_0^{(j)}, \{\mathsf{CT}_{k,b}^{(j)}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$ to $\mathcal{A}$.

3. $\mathcal{A}$ makes a single key query τ.

   Key query. When $\mathcal{A}$ makes a key query τ, the challenger computes $\alpha := \mathsf{H}(\tau)$ and returns $\mathsf{sk}_\tau := \left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\right)$ to $\mathcal{A}$.

4. $\mathcal{A}$ outputs coin'.

Exp $i.1$**:** This experiment is identical to the previous game except that the challenger generates $\mathsf{SKFE.CT}$ for $\mathcal{A}$'s $(i+1)$-th challenge query by using the fake encryption algorithm and sk_τ by using the reveal algorithm of NCER. Specifically, the experiment is described as follows.

1. The challenger generates $\mathsf{H} \leftarrow \mathsf{HashGen}(1^\lambda, 1^M)$, $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^{\ell_{\mathsf{aux}}})$, and $(\mathsf{ek}_{k,b}, \mathsf{dk}_{k,b}) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^\lambda)$for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and sets $\mathsf{MSK} := (\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}})$.
2. $\mathcal{A}$ is given security parameter 1^λ, and $\mathcal{A}$ makes arbitrarily many challenge queries.

   Challenge query. When $\mathcal{A}$ makes its j-th challenge query $(\mathsf{m}_0^{(j)}, \mathsf{m}_1^{(j)})$, the challenger works as follows:
   - If $j \leq i$, then the challenger returns $\mathsf{SKFE.CT}^{(j)} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_1^{(j)})$ to $\mathcal{A}$.
   - If $j = i+1$, then the challenger generates $\mathsf{aux}^{(i+1)} \leftarrow \mathsf{AuxGen}(1^\lambda, 1^M)$ and $\mathsf{label}_{k,b}^{(i+1)}$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, computes $\widetilde{U}^{(i+1)} \leftarrow \mathsf{GbCkt}'(U_{\mathsf{m}_0^{(j)}}, \mathsf{aux}^{(i+1)}, \{\mathsf{label}_{k,b}^{(i+1)}\}$ and $(\mathsf{CT}_0^{(i+1)}, \mathsf{st}_0) \leftarrow \mathsf{NCER.Fake}(\mathsf{ek}_0)$, and $(\mathsf{CT}_{k,b}^{(i+1)}, \mathsf{st}_{k,b}) \leftarrow \mathsf{NCER.Fake}(\mathsf{ek}_{k,b})$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and returns
   $$\mathsf{SKFE.CT} := \left(\widetilde{U}^{(i+1)}, \mathsf{CT}_0^{(i+1)}, \{\mathsf{CT}_{k,b}^{(i+1)}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$$
   to $\mathcal{A}$.
   - If $j > i+1$, then the challenger returns $\mathsf{SKFE.CT}^{(j)} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_0^{(j)})$ to $\mathcal{A}$.
3. $\mathcal{A}$ makes a single key query τ.

   Key query. When $\mathcal{A}$ makes a key query τ, the challenger computes $\alpha := \mathsf{H}(\tau)$, $\mathsf{dk}_0 \leftarrow \mathsf{NCER.Reveal}(\mathsf{st}_0, \mathsf{aux}^{(j)})$, and $\mathsf{dk}_{k,\alpha[k]} \leftarrow \mathsf{NCER.Reveal}(\mathsf{st}_{k,\alpha[k]}, \mathsf{label}_{k,\alpha[k]}^{(j)})$ for every $k \in [\ell_\alpha]$ and returns $\mathsf{sk}_\tau := \left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\right)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs coin'.

It is straightforward to reduce the indistinguishability from the previous experiment to the security of NCER by a hybrid argument.

Exp $i.2$**:** This experiment is identical to the previous one except that H, $\widetilde{U}^{(i+1)}$, $\mathsf{aux}^{(i+1)}$, and $\{\mathsf{label}^{(i+1)}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}$ (where $\alpha := \mathsf{H}(\tau)$) are generated by the simulators of AdaGC, and $\{\mathsf{label}^{(i+1)}_{k,1-\alpha[k]}\}_{k\in[\ell_\alpha]}$ is not generated. We remark that this modification makes sense since H is not used in the challenge phase, and $\{\mathsf{label}^{(i+1)}_{k,1-\alpha[k]}\}_{k\in[\ell_\alpha]}$ is not used throughout the experiment. Specifically, the experiment is described as follows.

1. The challenger generates $(\mathsf{ek}_0, \mathsf{dk}_0) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^{\ell_{\mathsf{aux}}})$, and $(\mathsf{ek}_{k,b}, \mathsf{dk}_{k,b}) \leftarrow \mathsf{NCER.KG}(1^\lambda, 1^\lambda)$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and sets $\mathsf{MSK} := \left(\mathsf{H}, \mathsf{ek}_0, \{\mathsf{ek}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}, \mathsf{dk}_0, \{\mathsf{dk}_{k,b}\}_{k\in[\ell_\alpha],b\in\{0,1\}}\right)$.
2. $\mathcal{A}$ is given security parameter 1^λ, and $\mathcal{A}$ makes arbitrarily many challenge queries.

   Challenge query. When $\mathcal{A}$ makes its j-th challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, the challenger works as follows:
   - If $j \leq i$, then the challenger returns $\mathsf{SKFE.CT}^{(j)} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_1^{(j)})$ to $\mathcal{A}$.
   - If $j = i+1$, then the challenger computes $(\widetilde{U}^{(i+1)}, \mathsf{st}_{\mathsf{Sim}}) \leftarrow \mathsf{SimC}(1^\lambda, 1^M)$, $(\mathsf{CT}_0^{(i+1)}, \mathsf{st}_0) \leftarrow \mathsf{NCER.Fake}(\mathsf{ek}_0)$, and $(\mathsf{CT}_{k,b}^{(i+1)}, \mathsf{st}_{k,b}) \leftarrow \mathsf{NCER.Fake}(\mathsf{ek}_{k,b})$ for every $k \in [\ell_\alpha]$ and $b \in \{0,1\}$, and returns $\mathsf{SKFE.CT}^{(i+1)} := \left(\widetilde{U}^{(i+1)}, \mathsf{CT}_0^{(i+1)}, \{\mathsf{CT}_{k,b}^{(i+1)}\}_{k\in[\lambda],b\in\{0,1\}}\right.$ to $\mathcal{A}$.
   - If $j > i+1$, then the challenger returns $\mathsf{SKFE.CT}^{(j)} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, \mathsf{m}_0^{(j)})$ to $\mathcal{A}$.
3. $\mathcal{A}$ makes a single key query τ.

   Key query. When $\mathcal{A}$ makes a key query τ, the challenger computes $(\tau, \mathsf{H}, \mathsf{aux}, \{\mathsf{label}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}) \leftarrow \mathsf{SimIn}(\mathsf{st}_{\mathsf{Sim}}, \tau)$, $\mathsf{dk}_0 \leftarrow \mathsf{NCER.Reveal}(\mathsf{st}_0, \mathsf{aux})$, and $\mathsf{dk}_{k,\alpha[k]} \leftarrow \mathsf{NCER.Reveal}(\mathsf{st}_{k,\alpha[k]}, \mathsf{label}_{k,\alpha[k]})$ for every $k \in [\ell_\alpha]$ and returns $\mathsf{sk}_\tau := \left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha[k]}\}_{k\in[\ell_\alpha]}\right)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs coin'.

It is straightforward to reduce the indistinguishability from the previous experiment to the input-privacy-free adaptive security of AdaGC.

By considering game hops from Exp $i.0$ to Exp $i.2$ in the reverse order, we can prove that $\mathcal{A}$ cannot distinguish Exp $i.2$ and Exp $(i+1).0$ with non-negligible advantages. Then by a standard hybrid argument shows that $\mathcal{A}$ cannot distinguish Exp 0.0 and Exp $q.0$ with non-negligible probability. This means that SKFE is 1-bounded key-adaptively secure. ■

Efficiency of key generation. We note that the key generation algorithm of the above SKFE scheme is efficient if the underlying garbling scheme has small online complexity. Especially, if we instantiate the scheme based on the input-privacy-free variant of the scheme given in Section 5, the key generation algorithm works in time $\mathrm{poly}(\lambda, \log M, m)$ and the size of decryption key of SKFE is $m + \mathrm{poly}(\lambda, \log M)$ where m is the output length of functions in the class. Especially, if $\mathcal{F}$ is a class of 1-bit output functions, (i.e., if $m = 1$), then the key generation algorithm works in time $\mathrm{poly}(\lambda, \log M)$ and the decryption key size is $\mathrm{poly}(\lambda, \log M)$.

7.5 From 1-Bit to Multi-Bit SKFE

Here, we give a transformation from SKFE scheme for 1-bit output functions to a one for multi-bit output functions without increasing the running time of the key generation algorithm and decryption

key length. Let $\mathcal{F} = \{f_\tau : \{0,1\}^{\ell_{\mathsf{m}}} \to \{0,1\}^m\}_{\tau\in\{0,1\}^{\ell_\tau}}$ be a family of polynomial-time computable tagged functions. Let $\mathcal{F}' = \{f'_\tau : [m] \times \{0,1\}^{\ell_{\mathsf{m}}} \to \{0,1\}\}_{\tau\in\{0,1\}^{\ell_\tau}}$ be a family of tagged functions that consists of functions f'_τ that is given $i \in [m]$ and $\mathsf{m} \in \{0,1\}^{\ell_{\mathsf{m}}}$ as input, and outputs the i-th bit of $f_\tau(\mathsf{m})$. Let $\mathsf{OneSKFE} = (\mathsf{OneSKFE.Setup}, \mathsf{OneSKFE.KG}, \mathsf{OneSKFE.Enc}, \mathsf{OneSKFE.Dec})$ be an SKFE scheme for $\mathcal{F}'$. We construct an SKFE scheme $\mathsf{MultiSKFE} = (\mathsf{MultiSKFE.Setup}, \mathsf{MultiSKFE.KG}, \mathsf{MultiSKFE.Enc}, \mathsf{MultiSKFE.Dec})$ for $\mathcal{F}$ as follows.

$\mathsf{MultiSKFE.Setup}(1^\lambda)$:

1. Computes $\mathsf{MSK} \leftarrow \mathsf{OneSKFE.Setup}(1^\lambda)$.
2. Outputs MSK.

$\mathsf{MultiSKFE.KG}(\mathsf{MSK}, \tau)$:

1. Computes $\mathsf{sk}_\tau \leftarrow \mathsf{OneSKFE.KG}(\mathsf{MSK}, \tau)$
2. Outputs sk_τ.

$\mathsf{MultiSKFE.Enc}(\mathsf{MSK}, \mathsf{m})$:

1. Computes $\mathsf{CT}_i \leftarrow \mathsf{OneSKFE.Enc}(\mathsf{MSK}, (i, \mathsf{m}))$ for every $i \in [m]$.
2. Outputs $\mathsf{CT} \coloneqq \{\mathsf{CT}_i\}_{i\in[m]}$.

$\mathsf{SKFE.Dec}(\tau, \mathsf{sk}_\tau, \mathsf{CT})$:

1. Parses $\{\mathsf{CT}_i\}_{i\in[m]} \leftarrow \mathsf{CT}$.
2. Computes $z_i \leftarrow \mathsf{OneSKFE.Dec}(\tau, \mathsf{sk}_\tau, \mathsf{CT}_i)$ for every $i \in [m]$.
3. Outputs $z \coloneqq z_1 \| \dots \| z_m$.

Correctness. Correctness of MultiSKFE easily follows from correctness of OneSKFE.

Security. The security of MultiSKFE can be stated as follows.

Theorem 7.5. *If* OneSKFE *is* 1*-bounded key-adaptively secure, then* MultiSKFE *is* 1*-bounded key-adaptively secure.*

This can be proven by a standard hybrid argument.

Efficiency of key generation. Since the key generation algorithm of MultiSKFE just runs that of OneSKFE, the computational complexity of the key generation algorithm and decryption key length of MultiSKFE are the same as those of OneSKFE.

7.6 Adaptively Indistinguishable Garbling from 1-Bounded Key-Adaptive SKFE

Here, we describe a construction of adaptively indistinguishable garbling scheme based on any 1-bounded key-adaptively secure SKFE scheme. The construction is similar to the construction of function-private SKFE scheme based on any SKFE scheme by Brakerski and Segev [BS18]. Let $\mathcal{C}$ be a class of circuits of n-bit inputs. For $(c, c') \in \{0,1\}^n \times \{0,1\}^n$, we define a function $f_{c,c'} : \mathcal{C} \times \mathcal{C} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^*$ by

$$f_{c,c'}(C, C', r, r') \coloneqq \begin{cases} C(c \oplus r) & \text{if } C \neq \bot \\ C'(c' \oplus r') & \text{otherwise} \end{cases}.$$

Let $\mathsf{SKFE} = (\mathsf{SKFE.Setup}, \mathsf{SKFE.KG}, \mathsf{SKFE.Enc}, \mathsf{SKFE.Dec})$ be an SKFE scheme for the class $\mathcal{F} := \{f_{c,c'}\}_{(c,c')\in\{0,1\}^n\times\{0,1\}^n}$ of tagged functions. Then we construct a garbling scheme $\mathsf{AdaGC} = (\mathsf{GbCkt}, \mathsf{GbInp}, \mathsf{GbEval})$ for circuit class $\mathcal{C}$ as follows.

$\mathsf{GbCkt}(1^\lambda, C)$:

1. Generates $\mathsf{MSK} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$.
2. Generates $r, r' \xleftarrow{\mathsf{r}} \{0,1\}^n$.
3. Computes $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, (C, \bot, r, \bot))$.
4. Outputs $\widetilde{C} := \mathsf{SKFE.CT}$ and $\mathsf{st} := (\mathsf{MSK}, r, r')$.

$\mathsf{GbInp}(\mathsf{st}, x)$:

1. Parses $(\mathsf{MSK}, r, r') \leftarrow \mathsf{st}$.
2. Computes $c = x \oplus r$ and $c' = x \oplus r'$.
3. Computes $\mathsf{sk}_{c,c'} \leftarrow \mathsf{SKFE.KG}(\mathsf{MSK}, (c, c'))$.
4. Outputs $\widetilde{x} := (c, c', \mathsf{sk}_{c,c'})$.

$\mathsf{GbEval}(\widetilde{C}, \widetilde{x})$:

1. Parses $\mathsf{SKFE.CT} \leftarrow \widetilde{C}$ and $(c, c', \mathsf{sk}_{c,c'}) \leftarrow \widetilde{x}$.
2. Computes $z \leftarrow \mathsf{SKFE.Dec}((c, c'), \mathsf{sk}_{c,c'}, \mathsf{SKFE.CT})$.
3. Outputs z.

Correctness. Let $\widetilde{C} = \mathsf{SKFE.CT}$ and $\widetilde{x} = (c, c', \mathsf{sk}_{c,c'})$ be an honestly generated garbled circuit and garbled input, i.e., $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, (C, \bot, r, \bot))$, $c = x \oplus r$, $c' = x \oplus r'$ and $\mathsf{sk}_{c,c'} \leftarrow \mathsf{SKFE.KG}(\mathsf{MSK}, (c, c'))$ where $\mathsf{MSK} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and $r, r' \xleftarrow{\mathsf{r}} \{0,1\}^n$. By correctness of SKFE, we have $\mathsf{SKFE.Dec}((c, c'), \mathsf{sk}_{c,c'}, \mathsf{SKFE.CT}) = f_{c,c'}(C, \bot, r, \bot)) = C(c \oplus r) = C(x)$.

Security. The security of AdaGC can be stated as follows.

Theorem 7.6. *If* SKFE *is 1-bounded key-adaptively secure, then* AdaGC *is adaptively indistinguishable.*

Proof. Let $\mathcal{A}$ be a PPT valid adversary that attacks adaptive indistinguishability of AdaGC. We consider the following sequence of experiments.

Exp 0**:** This is the experiment defining adaptive indistinguishability in which coin is fixed to 0. Specifically, the experiment is described as follows.

1. $\mathcal{A}$ is given security parameter 1^λ.
2. $\mathcal{A}$ sends two circuits (C_0, C_1) to the challenger, and the challenger generates $r, r' \xleftarrow{\mathsf{r}} \{0,1\}^n$, computes $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, (C_0, \bot, r, \bot))$, and returns $\widetilde{C} = \mathsf{SKFE.CT}$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends two inputs $(x_0, x_1) \in \{0,1\}^n$ to the challenger, and the challenger computes $c = x_0 \oplus r$, $c' = x_0 \oplus r'$, and $\mathsf{sk}_{c,c'} \leftarrow \mathsf{SKFE.KG}(\mathsf{MSK}, (c, c'))$, and returns $\widetilde{x} := (c, c', \mathsf{sk}_{c,c'})$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

We note that we have $C_0(x_0) = C_1(x_1)$ since $\mathcal{A}$ is valid.

Exp 1**:** This experiment is identical to the previous experiment except that c' is generated as $c' = x_1 \oplus r'$. $\mathcal{A}$ cannot distinguish Exp 1 and 2 because r' is only used for generating c' in both experiments, and thus c' is a uniformly and independently random string from $\mathcal{A}$'s view in both experiments.

Exp 2**:** This experiment is identical to the previous experiment except that $\mathsf{SKFE.CT}$ is generated as $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, (\bot, C_1, \bot, r'))$. $\mathcal{A}$'s advantage to distinguish Exp 1 and 2 is negligible if SKFE is 1-bounded key-adaptively secure since all ciphertexts are generated before generating a decryption key in these experiments, and we have $f_{x_0\oplus r, x_1\oplus r'}(C_0, \bot, r, \bot) = C_0(x_0) = C_1(x_1) = f_{x_0\oplus r, x_1\oplus r'}(\bot, C_1, \bot, r')$.

Exp 3**:** This experiment is identical to the previous experiment except that c is generated as $c = x_1 \oplus r$. $\mathcal{A}$ cannot distinguish Exp 2 and 3 because r is only used for generating c in both experiments, and thus c is a uniformly and independently random string from $\mathcal{A}$'s view in both experiments.

Exp 4**:** This experiment is identical to the previous experiment except that $\mathsf{SKFE.CT}$ is generated as $\mathsf{SKFE.CT} \leftarrow \mathsf{SKFE.Enc}(\mathsf{MSK}, (C_1, \bot, r, \bot))$. $\mathcal{A}$ cannot distinguish Exp 3 and 4 under the 1-bounded key-adaptive security of SKFE similarly to the game hop from Exp 1 and 2.

It is clear that Exp 4 is the experiment defining adaptive indistinguishability in which coin is fixed to 1, and $\mathcal{A}$ cannot distinguish Exp 0 and 4 with non-negligible advantage. ■

7.7 Adaptively Indistinguishable Garbling with Near-Optimal Online Complexity

Here, we discuss how to obtain adaptively indistinguishable garbling with near-optimal online complexity (especially, whose online encoding length does not depend on the output length of a circuit) based on any selective-database LOT (which in turn can be constructed based on CDH, factoring, LWE, or selectively secure succinct PKFE) combining the constructions given in Section 7.4, Section 7.5, and Section 7.6. As discussed in Section 7.3, a slight modification to the scheme given in Section 5 (or GS18 scheme [GS18]) gives a quasi-decomposable input-privacy-free adaptive garbling from any selective-database LOT. Especially, if we only consider garbling for 1-bit output circuits, then we have $\ell_{\mathsf{H}} = \mathrm{poly}(\lambda)$, $\ell_\alpha = \mathrm{poly}(\lambda)$, and auxiliary-information-length is $\ell_{\mathsf{aux}} = \mathrm{poly}(\log |C|, \lambda)$, and H can be computed in time $O(n) + \mathrm{poly}(\log |C|, \lambda)$.

By applying the conversion in Section 7.4 to the above scheme, we obtain 1-bounded key-adaptive SKFE scheme for 1-bit-output tagged functions whose decryption key is of the form $\left(\mathsf{H}, \mathsf{dk}_0, \{\mathsf{dk}_{k,\alpha_k}\}_{k\in[\ell_\alpha]}\right)$ where dk_0 and dk_{k,α_k} are decryption keys of an SK-NCER scheme of message-length ℓ_{aux} and λ, respectively. Especially, its length is $\ell_{\mathsf{H}} + \mathrm{poly}(\lambda, \ell_{\mathsf{aux}}) + \ell_{\mathsf{aux}} \cdot \mathrm{poly}(\lambda) = \mathrm{poly}(\log M, \lambda)$ and the runnning time of the key generation algorithm is $O(\ell_\tau) + \mathrm{poly}(\log M, \lambda)$ where ℓ_τ and M are as defined in Section 7.4. Next, by applying the conversion in Section 7.5, we can transform any SKFE scheme for 1-bit-output functions to a scheme for multi-bit-output functions neither increasing decryption key size nor computational complexity of key-generation algorithm. Finally, by applying the conversion in Section 7.6, we obtain an adaptively indistinguishable garbling for all circuits with succinct online complexity. Namely for obtaining a garbling scheme for a circuit class $\mathcal{C}$ of n-bit inputs, we rely on SKFE scheme for $f_{c,c'} : \mathcal{C} \times \mathcal{C} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^*$ which is defined in Section 7.6. For this function class, we have $M = \mathrm{poly}(|C|, \lambda)$ and $\ell_\tau = 2n$ where M is the maximum size of a universal circuit that is used in the construction in Section 7.4. Since a garbled input consists of $(c, c') \in \{0,1\}^n \times \{0,1\}^n$ and a decryption key of the underlying SKFE, the online communication complexity is $2n + \mathrm{poly}(\log |C|, \lambda)$ and online computational complexity is $O(n) + \mathrm{poly}(\log |C|, \lambda)$. In summary, we obtain the following theorem.

Theorem 7.7. *If one of the {CDH,Factoring, LWE} assumptions holds or* $(1, \mathsf{w\text{-}sel}, \mathsf{sls})$*-PKFE for circuits exists, then there exists an adaptively indistinguishable garbling scheme whose online communication*

complexity is $2n + \mathrm{poly}(\log |C|, \lambda)$ *and online computational complexity is* $O(n) + \mathrm{poly}(\log |C|, \lambda)$ *where* C *is the circuit being garbled of* n*-bit input.*

More generally, the above conversion works based on any quasi-decomposable input-privacy-free adaptive garbling schemes (including the input-privacy-free varitants of [HJO+16, JW16]). To state this formally, we introduce a notation for circuit classes. For a circuit class $\mathcal{C}$ of circuits of n-bit input and m-bit output, we consider another circuit class $\mathcal{U}[\mathcal{C}] := \{U_{i,C,C',r,r'} : i \in [m], C, C' \in \mathcal{C}, r, r' \in \{0,1\}^n\}$ where $U_{i,C,C',r,r'}$ is a circuit with $2n$-bit input defined by

$$U_{i,C,C',r,r'}(c, c') := \begin{cases} C_i(c \oplus r) & \text{if } C \neq \bot \\ C'_i(c' \oplus r') & \text{otherwise} \end{cases}.$$

where C_i and C'_i denote circuits that computes i-th bit of outputs of C and C', respectively.

Then our result can be stated as follows.

Theorem 7.8. *If there exists quasi-decomposable input-privacy-free adaptive garbling scheme for a circuit class* $\mathcal{U}[\mathcal{C}]$ *with hash-description-size* ℓ_{H}*, hash-output-length* ℓ_α*, and auxiliary-information-length* ℓ_{aux}*, there exists an adaptively indistinguishable garbling scheme for a circuit class* $\mathcal{C}$ *with online communication complexity is* $2n + \ell_{\mathsf{H}} + \ell_{\mathsf{aux}} \cdot (\lambda + 1) + \ell_\alpha \cdot \lambda(\lambda + 1)$ *and online computational complexity is the hash-computation-complexity of the underlying quasi-decomposable garbling plus* $(\ell_\alpha + \ell_{\mathsf{aux}}) \cdot \mathrm{poly}(\lambda)$.

As discussed in Section 7.3, "input-privacy-free variants" of known adaptive garbling schemes based on one-way functions [HJO+16, JW16] satisfy the quasi-decomposability and input-privacy-free adaptive security with $\ell_{\mathsf{H}} = 0$, $\ell_\alpha = n$, and $\ell_{\mathsf{aux}} = (n + m + t)\mathrm{poly}(\lambda)$ where m is the output-length and t is a parameter called the pebble complexity of the circuit being garbled. In particular, we can set $t = w$ where w is the width of the circuit, or $t = d$ where m is the output-length and d is the depth of the circuit if the circuit is in NC^1, i.e., $d = O(\log n)$.

It is easy to see that if the maximum width of a circuit in $\mathcal{C}$ is at most w, then the maximum width of a circuit in $\mathcal{U}[\mathcal{C}]$ is at most $\max\{w, 2n\}$, and if $\mathcal{C} \in \mathsf{NC}^1$, then we also have $\mathcal{U}[\mathcal{C}] \in \mathsf{NC}^1$. Then by applying Theorem 7.8 to the input-privacy-free variant of adaptive garbling schemes of [HJO+16, JW16], we obtain the following corollary.

Corollary 7.9 (Also proven in [JSW17]). *The following hold:*

1. *If a one-way function exists, then there exists an adaptively indistinguishable garbling scheme for* NC^1 *whose online communication/computational complexity are* $n \cdot \mathrm{poly}(\lambda)$ *where* n *is the input-length of the circuit being garbled.*

2. *If a one-way function exists, then there exists an adaptively indistinguishable garbling scheme for all circuits whose online communication/computational complexity are* $(n + w) \cdot \mathrm{poly}(\lambda)$ *where* n *is the input-length and* w *is the width of the circuit being garbled.*

Though Jafargholi et al. [JSW17] already proved the same statement, their construction is obtained by modifying (simulation-based) adaptive garbling scheme by Hemenway et al. [HJO+16] in an ad hoc and complicated manner. On ther other hand, our construction is generic, and gives a modular construction. We note that we can generalize the above corollary by parametrizing circuits by the pebble complexity similarly to [JSW17] . Namely, we can prove that if a one-way function exists, then there exists an adaptively indistinguishable garbling scheme whose online communication/computational complexity are $(n + t) \cdot \mathrm{poly}(\lambda)$ for circuits with pebble complexity t. This can be seen by observing that the pebble complexity of $U_{i,C,C',r,r'}$ is almost the same as that of C and C'. We only gave special cases above for avoiding complicated discussions on pebble complexity.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015. (Cited on page 1, 2.)

[AIKW15] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM J. Comput.*, 44(2):433–466, 2015. (Cited on page 3.)

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. (Cited on page 1.)

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. http://eprint.iacr.org/2015/730. (Cited on page 1, 2.)

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. *IACR Cryptology ePrint Archive*, 2018:759, 2018. TCC 2018 (to appear). (Cited on page 3, 4, 5, 8, 9, 22, 28.)

[AM18] Shweta Agrawal and Monosij Maitra. Fe and io for turing machines from minimal assumptions. *IACR Cryptology ePrint Archive*, 2018:908, 2018. TCC 2018 (to appear). (Cited on page 5.)

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015. (Cited on page 7.)

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016. (Cited on page 2, 4, 5, 6, 10, 11, 20, 61, 62.)

[BCG+18] Nir Bitansky, Ran Canetti, Sanjam Garg, Justin Holmgren, Abhishek Jain, Huijia Lin, Rafael Pass, Sidharth Telang, and Vinod Vaikuntanathan. Indistinguishability obfuscation for RAM programs and succinct randomized encodings. *SIAM J. Comput.*, 47(3):1123–1210, 2018. (Cited on page 3.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 1, 20.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 13.)

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012. (Cited on page 3.)

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018. (Cited on page 3, 5, 28.)

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016. (Cited on page 1.)

[BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 306–324. Springer, Heidelberg, March 2015. (Cited on page 11, 44, 61.)

[BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31(1):202–225, January 2018. (Cited on page 53.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. (Cited on page 1.)

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015. (Cited on page 1, 2.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 13.)

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017. (Cited on page 3, 5, 7, 8, 9, 10, 22, 28, 31.)

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168. Springer, Heidelberg, February 2005. (Cited on page 12, 45.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Cited on page 1.)

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Heidelberg, March 2018. (Cited on page 3, 28.)

[DGL+16] Dana Dachman-Soled, S. Dov Gordon, Feng-Hao Liu, Adam O'Neill, and Hong-Sheng Zhou. Leakage-resilient public-key encryption from obfuscation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 101–128. Springer, Heidelberg, March 2016. (Cited on page 5.)

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Cited on page 9.)

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. (Cited on page 1, 20.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. (Cited on page 8, 13, 27.)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. (Cited on page 9.)

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. (Cited on page 14.)

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 579–604. Springer, Heidelberg, August 2016. (Cited on page 7, 27.)

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 156–181. Springer, Heidelberg, April / May 2017. (Cited on page 1, 7, 27.)

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Heidelberg, October / November 2016. (Cited on page 1, 2, 5, 6, 7, 8, 20, 27, 28.)

[GS18] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Heidelberg, April / May 2018. (Cited on page 1, 3, 7, 9, 11, 22, 28, 29, 34, 44, 48, 55.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 1, 2, 12, 44.)

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, August 2016. (Cited on page 1, 3, 11, 13, 21, 32, 44, 48, 56.)

[HPW15] Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 443–469. Springer, Heidelberg, November / December 2015. (Cited on page 12.)

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017. (Cited on page 3.)

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Heidelberg, May 2000. (Cited on page 45.)

[JSW17] Zahra Jafargholi, Alessandra Scafuro, and Daniel Wichs. Adaptively indistinguishable garbled circuits. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 40–71. Springer, Heidelberg, November 2017. (Cited on page 1, 2, 3, 4, 11, 13, 17, 44, 56.)

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of Yao's garbled circuits. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 433–458. Springer, Heidelberg, October / November 2016. (Cited on page 3, 11, 12, 13, 44, 48, 56.)

[KNT17a] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. From single-key to collusion-resistant secret-key functional encryption by leveraging succinctness. Cryptology ePrint Archive, Report 2017/638, 2017. http://eprint.iacr.org/2017/638. (Cited on page 1.)

[KNT17b] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. Cryptology ePrint Archive, Report 2017/361, 2017. http://eprint.iacr.org/2017/361. (Cited on page 1.)

[KNT18a] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 603–648. Springer, Heidelberg, April / May 2018. (Cited on page 1.)

[KNT18b] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 187–217. Springer, Heidelberg, March 2018. (Cited on page 1.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013. (Cited on page 13.)

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 122–151. Springer, Heidelberg, April / May 2017. (Cited on page 1.)

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Heidelberg, October / November 2016. (Cited on page 1, 2.)

[LZ17] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In Yael Kalai and Leonid Reyzin,

editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 138–169. Springer, Heidelberg, November 2017. (Cited on page 7, 9, 22, 27.)

[MPS16] Antonio Marcedone, Rafael Pass, and Abhi Shelat. Bounded KDM security from iO and OWF. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 571–586. Springer, Heidelberg, August / September 2016. (Cited on page 5.)

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016. (Cited on page 5.)

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. (Cited on page 1.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 463–472. ACM Press, October 2010. (Cited on page 1, 2.)

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. (Cited on page 7, 13.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 16.)

A Transformation by Ananth and Sahai

We review the transformation by Ananth and Sahai [AS16].

Building Blocks.

- Selectively secure succinct PKFE for circuits: $\mathsf{ssFE}^{\mathsf{pk}} = (\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Setup}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{KG}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Enc}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Dec})$
- Selective-message function private[21] single-key succinct SKFE for circuits: $\mathsf{1keyFE}^{\mathsf{sk}} = (\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{Setup}, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{KG}, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{Enc}, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{Dec})$
- Adaptively secure single-key and single-ciphertext succinct SKFE for circuits: $\mathsf{1ctkeyFE}^{\mathsf{ad}} = (\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Setup}, \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{KG}, \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Enc}, \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Dec})$
- PRF: PRF
- SKE with pseudorandom ciphertext: $\mathsf{SKE} = (\mathsf{SKE}.\mathsf{Enc}, \mathsf{SKE}.\mathsf{Dec})$

[21] Informally, in the security game, adversaries send not only challenge messages $(\mathsf{m}_0, \mathsf{m}_1)$ but also challenge functions (f_0, f_1) such that $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$. We can obtain function privacy for free. See the paper by Brakerski and Segev [BS15] for details.

AS16 transformation. fully-equipped scheme $\mathsf{FullFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is as follows.

$\mathsf{Setup}(1^\lambda)$**:** Generates $(\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MPK}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MSK}) \leftarrow \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Setup}(1^\lambda)$ and outputs $(\mathsf{MPK}, \mathsf{MSK}) := (\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MPK}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MSK})$.

$\mathsf{KG}(\mathsf{MSK}, f)$**:**

1. Samples $\mathsf{ske.CT} \leftarrow \{0,1\}^{\ell_{\mathsf{ske}}}$.
2. Samples $\tau := \tau_0 \| \tau_1 \| \tau_2 \| \tau_3 \leftarrow \{0,1\}^{4\lambda}$.
3. Generates $\mathsf{sk}_{g[f,\mathsf{ske.CT},\tau]} \leftarrow \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{KG}(\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MSK}, g[f, \mathsf{ske.CT}, \tau])$ where $g[f, \mathsf{ske.CT}, \tau]$ is defined in Figure 6.
4. Outputs $\mathsf{sk}_f := \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{sk}_{g[f,\mathsf{ske.CT},\tau]}$.

$\mathsf{Enc}(\mathsf{MPK}, \mathsf{m})$**:**

1. Chooses a PRF key $\mathsf{K} \leftarrow \{0,1\}^\lambda$.
2. Generates $\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{MSK} \leftarrow \mathsf{1keyFE}^{\mathsf{sk}}(1^\lambda)$.
3. Generates $\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{sk}_{h[\mathsf{m}]} \leftarrow \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{KG}(\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{MSK}, h[\mathsf{m}])$, where $h[\mathsf{m}]$ is defined in Figure 7.
4. Generates $\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{CT} \leftarrow \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Enc}(\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{MPK}, (\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{MSK}, \mathsf{K}, \bot, 0))$
5. Outputs $\mathsf{CT}_{\mathsf{m}} := (\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{sk}_{h[\mathsf{m}]}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{CT})$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}_{\mathsf{m}})$**:**

1. Computes $(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{sk}_f, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{CT}) \leftarrow \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{Dec}(\mathsf{ssFE}^{\mathsf{pk}}.\mathsf{sk}_{g[f,\mathsf{ske.CT},\tau]}, \mathsf{ssFE}^{\mathsf{pk}}.\mathsf{CT})$.
2. Computes $\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{CT} \leftarrow \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{Dec}(\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{sk}_{h[\mathsf{m}]}, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{CT})$.
3. Computes and outputs $\mathsf{m}' \leftarrow \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Dec}(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{sk}_f, \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{CT})$.

$g[f, \mathsf{ske.CT}, \tau](\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{MSK}, \mathsf{K}, \mathsf{ske.k}, \mathsf{flag}_1)$

1. Parses $\tau = \tau_0 \| \tau_1 \| \tau_2 \| \tau_3$.
2. If $\mathsf{flag}_1 = 0$, then computes

$$\begin{aligned} \rho_i &\leftarrow \mathsf{PRF}_{\mathsf{K}}(\tau_i) \\ \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{MSK} &\leftarrow \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Setup}(1^\lambda; \rho_0) \\ \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{sk}_f &\leftarrow \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{KG}(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{MSK}, f; \rho_1) \\ \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{CT} &\leftarrow \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{Enc}(\mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{MSK}, (\rho_2, 0); \rho_3) \end{aligned}$$

   and outputs $(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{sk}_f, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{CT})$.
3. Else computes and outputs $(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{sk}_f, \mathsf{1keyFE}^{\mathsf{sk}}.\mathsf{CT}) \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.k}, \mathsf{ske.CT})$.

Figure 6: This circuit g is an input of the key generation algorithm of $\mathsf{ssFE}^{\mathsf{pk}}$ (selectively secure and succinct public-key FE). This circuit outputs a functional decryption key of f by the single-key and single-ciphertext scheme and an encryption of randomness that will be used in the circuit h under the master-key of the single-key scheme.

As we see the transformation, if $\mathsf{1ctkeyFE}^{\mathsf{ad}}$ is for boolean circuits, the transformation gives FE for boolean circuits.

Ananth and Sahai [AS16] proved the following.

Theorem A.1 ([AS16]). *If all building blocks are secure, then* FullFE *is* $(\mathsf{unb}, \mathsf{ada}, \mathsf{fs})$*-PKFE.*

Since all building blocks except $(1, 1, \mathsf{ada}, \mathsf{fs})$-SKFE can be constructed from $(\mathsf{unb}, \mathsf{sel}, \mathsf{fs})$-PKFE, we obtain Theorem 3.18.

$h[\mathsf{m}](\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{MSK}, \rho, \mathsf{flag}_2)$

1. If $\mathsf{flag}_2 = 0$, then computes and outputs $\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{CT}_{\mathsf{m}} \leftarrow \mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{Enc}(\mathsf{1ctkeyFE}^{\mathsf{ad}}.\mathsf{MSK}, \mathsf{m}; \rho)$.
2. Else outputs $\bot$.

Figure 7: This circuit h is an input of the key generation algorithm of $\mathsf{1keyFE}^{\mathsf{sk}}$ (selectively secure, single-key, and succinct secret-key FE). This circuit outputs an encryption of m under the master secret key of $\mathsf{1ctkeyFE}^{\mathsf{ad}}$.

B Omitted Proofs

B.1 Proof of Theorem 5.20

Proof. The security proof is as follows.

Exp 0**:** This is the experiment defining the security of 1-key 1-ciphertext adaptive security in which coin is fixed to 0. Specifically, this is described as follows.

1. The challenger generates $\mathsf{MSK}' \leftarrow \mathsf{Setup}'(1^\lambda)$ and chooses a random mask $\rho \xleftarrow{\mathsf{r}} \{0,1\}^{\ell(\mathsf{m})}$, and sends security parameter 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ makes a single key query and single challenge query in any order.

   Key query. When $\mathcal{A}$ makes a key query f, the challenger computes $\mathsf{sk}'_f \leftarrow \mathsf{KG}'(\mathsf{MSK}', f)$, and returns $\mathsf{sk}_f := (\rho, \mathsf{sk}'_f)$ to $\mathcal{A}$.

   Challenge query. When $\mathcal{A}$ makes a challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, the challenger computes $\mathsf{CT}'_{\mathsf{m}_0} \leftarrow \mathsf{Enc}'(\mathsf{MSK}', \mathsf{m}_0)$, sets $\mathsf{CT}^* := \mathsf{CT}'_{\mathsf{m}_0} \oplus \rho$, and returns CT^* to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$.

Exp 1**:** This experiment is identical to the previous experiment except that when $\mathcal{A}$ makes a challenge query $(\mathsf{m}_0, \mathsf{m}_1)$ *before* it makes a key query f, the challenger does the following:

Challenge query: $\mathcal{A}$ chooses a random string $\rho' \xleftarrow{\mathsf{r}} \{0,1\}^{\ell(\mathsf{m}_0)}$ and sends $\mathsf{CT}^* := \rho'$ as a target ciphertext to $\mathcal{A}$.

Key query: When the challenger receives a key query f after the challenger query, the challenger generates $\mathsf{sk}'_f \leftarrow \mathsf{KG}'(\mathsf{MSK}', f)$ and $\mathsf{CT}'_{\mathsf{m}_0} \leftarrow \mathsf{Enc}'(\mathsf{MSK}', \mathsf{m}_0)$, sets $\widetilde{\rho} := \rho' \oplus \mathsf{CT}'_{\mathsf{m}_0}$ and sends $\mathsf{sk}_f(\widetilde{\rho}, \mathsf{sk}'_f)$ to $\mathcal{A}$.

That is, we defer generating $\mathsf{CT}'_{\mathsf{m}_0}$ until the key query phase. The distribution of CT^* and sk_f are the same as those in the original experiment for $\mathsf{coin} = 0$. This is because (1) It holds $\widetilde{\rho} \oplus \rho' = \mathsf{CT}'_{\mathsf{m}_0}$ in this experiment and $\mathcal{A}$ obtains the same decryption result as in the original experiment. (2) This is the 1-key and 1-ciphertext setting and ρ and ρ' are uniformly random elements. Thus, it holds that $(\rho, \mathsf{sk}'_f, \mathsf{CT}'_{\mathsf{m}_0} \oplus \rho) \overset{\mathsf{p}}{\approx} (\rho' \oplus \mathsf{CT}'_{\mathsf{m}_0}, \mathsf{sk}'_f, \rho')$.

Exp 2**:** This experiment is identical to the previous experiment except that when the challenger generates $\mathsf{CT}'_{\mathsf{m}_1} \leftarrow \mathsf{Enc}'(\mathsf{MSK}', \mathsf{m}_1)$ instead of $\mathsf{CT}'_{\mathsf{m}_0} \leftarrow \mathsf{Enc}'(\mathsf{MSK}', \mathsf{m}_0)$.

If $\mathcal{A}$ can distinguish the difference between the first and second experiments, then we can break 1-key and 1-bounded ciphertext-adaptive security of $\mathsf{ctadFE}^{\text{1-1}}_{\mathsf{gs}}$. There are two cases:

1. $\mathcal{A}$ first sends a key query f, and then $\mathcal{A}$ sends a challenge query $(\mathsf{m}_0, \mathsf{m}_1)$. This is the exactly the same setting as that in 1-key and 1-bounded ciphertext-adaptive security. Therefore, we can use the distinguisher of the two experiments as it is.

2. $\mathcal{A}$ first sends a challenge query $(\mathsf{m}_0, \mathsf{m}_1)$, then $\mathcal{A}$ sends a key query f. In this case, an adversary $\mathcal{B}$ of 1-key and 1-bounded ciphertext-adaptive security does the same thing as in the first experiment. That is, when $\mathcal{A}$ send the challenge query, $\mathcal{B}$ sends a fake ciphertext $\mathsf{CT}^* \coloneqq \rho'$ to $\mathcal{A}$. Then, when $\mathcal{A}$ sends the key query f, $\mathcal{B}$ passes f to its own challenger in 1-key and 1-bounded ciphertext-adaptive security and receives sk'_f. Now, $\mathcal{B}$ can passes the challenge query $(\mathsf{m}_0, \mathsf{m}_1)$ to its own challenger and receives $\mathsf{CT}'_{\mathsf{m}_{\mathsf{coin}}}$. Finally, $\mathcal{B}$ sets $\mathsf{sk}_f \coloneqq (\mathsf{CT}'_{\mathsf{m}_{\mathsf{coin}}} \oplus \rho', \mathsf{sk}'_f)$ to $\mathcal{A}$. This completes the reduction.

In both cases, we can reduce distinguishing two experiments to break 1-key and 1-bounded ciphertext-adaptive security.

We can arrive at the experiment defining the security of 1-key 1-ciphertext adaptive security in which coin is fixed to 1 by applying the changes in reverse order. This completes the proof. ■