Size-Hiding Computation for Multiple Parties

Kazumasa Shinagawa[1,2], Koji Nuida[2,3], Takashi Nishide[1], Goichiro Hanaoka[2], and Eiji Okamoto[1]

[1] University of Tsukuba, Ibaraki, Japan
shinagawa@cipher.risk.tsukuba.ac.jp
[2] AIST, Tokyo, Japan
[3] Japan Science and Technology Agency (JST) PRESTO Researcher, Tokyo, Japan

Abstract. Lindell, Nissim, and Orlandi (ASIACRYPT 2013) studied feasibility and infeasibility of general two-party protocols that hide not only the contents of the inputs of parties, but also some sizes of the inputs and/or the output. In this paper, we extend their results to n-party protocols for $n \geq 2$, and prove that it is infeasible to securely compute every function while hiding two or more (input or output) sizes. Then, to circumvent the infeasibility, we naturally extend the communication model in a way that any adversary can learn neither the contents of the messages nor the numbers of bits exchanged among honest parties. We note that such "size-hiding"computation is never a trivial problem even by using our "size-hiding"channel, since size-hiding computation of some function remains infeasible as we show in the text. Then, as our main result, we give a necessary and sufficient condition for feasibility of size-hiding computation of an arbitrary function, in terms of which of the input and output sizes must be hidden from which of the n parties. In particular, it is now possible to let each input/output size be hidden from some parties, while the previous model only allows the size of at most one input to be hidden. Our results are based on a security model slightly stronger than the honest-but-curious model.

Keywords: secure multiparty computation, size-hiding

1 Introduction

Secure multiparty computation (MPC) protocols enable parties to compute a function while hiding the contents of the inputs from each other. Goldreich, Micali, and Wigderson [GMW87] first constructed a general MPC protocol in the presence of semi-honest and malicious adversaries. Here, we say that a protocol is general when it can securely compute *every* efficient function.

Most of the previous MPC protocols (implicitly) assume that the input sizes of parties may be revealed. However, the input sizes may be confidential in some settings. Let us consider the following situation: A police department has a list of suspected terrorists and each company has its customers' list. The police wants to know the intersection of the lists without revealing any information. However, if we straightforwardly utilize the standard MPC, there is no guarantee that the number of terrorists (i.e., input size) will be protected against companies, and this might cause a serious problem since the number of terrorists is often sensitive information. We may also consider the case where the police wants to hide the number of terrorists in customers' lists (i.e., output size) from companies. For resolving these issues, we require MPC that hides input and output sizes. This type of MPC is called *size-hiding computation*.

Currently, several size-hiding protocols have been proposed [MRK03, IP07, ACT11, CV12], but these protocols can compute only specific functionalities such as set intersection, homomorphic evaluation for branching programs, and database commitments. In 2013, Lindell, Nissim, and Orlandi [LNO13] exhaustively investigated feasibility and infeasibility of general size-hiding *two-party* protocols. They showed that, when the output size is not hidden, every efficient function can be securely computed while hiding one size (i.e., the input size of one party). Furthermore, they also proved that there is an efficient function that *cannot* be securely computed while hiding two sizes (i.e., either the input sizes of both parties, or the input size of one party and the output size). Recently, Chase, Ostrovsky, and Visconti [COV15] further strengthened the feasibility result of Lindell et al. by constructing a general size-hiding two-party protocol in the presence of malicious adversaries while hiding the input size of one party. However, these existing works investigated only the two-party setting, and therefore, feasibility and infeasibility of size-hiding n-party computation for $n > 2$ are still not clear.

1.1 Our Results

In this paper, we study general size-hiding $n(\geq 2)$-party protocols in the presence of static and semi-honest adversaries corrupting up to $n-1$ of the n parties. For a technical reason, our semi-honest model is slightly stronger than the standard honest-but-curious model. (See the last paragraph in this Section and Appendix.) To clarify our results, we classify size-hiding computations as *size-hiding classes* according to which of the input sizes and the output size must be hidden from which of the n parties. We note that, as in the previous work on two-party cases, we assume that every party wants to compute a *common* function. To study generalized settings is a future research plan.

Our results in the secure channel model. We extend the two-party results [LNO13] into multiparty settings in the *secure channel model*, in which an adversary cannot learn the contents of messages exchanged among honest parties, but may learn the number of bits of the messages. In the multiparty setting, the inputs and the output sizes can be hidden from a subset[4] of parties. See Table 1 (part corresponding to secure channel) for a summary. As the feasibility results, when at most one input size is hidden from some parties, every efficient

[4] An input or the output size cannot be hidden from *all* parties because an input size is known to the holding party and we assume that at least one party obtains the output value.

function can be securely computed (Lemma 3). The computation is also possible when the output size is hidden from some parties but then the input sizes are not hidden. On the other hand, when two or more sizes are hidden from some parties, there exists a function that *cannot* be securely computed (Lemmas 4 and 5).

Table 1. Our results (Sections 4 and 5)

∘ Secure channel model

	# of hidden input sizes	Output size	Feasible?
Lemma 3	≤ 1	known	yes
Lemma 4	≥ 2	known	no
(Trivial)	0	hidden	yes
Lemma 5	≥ 1	hidden	no

∘ Strong secure channel model

	Condition for hidden sizes	Output size	Feasible?
Lemma 6	(A)	known	yes
Lemma 7	(A)	known	no
Lemma 8	(B)	hidden	yes
Lemma 9	(B)	hidden	no

(A) When all parties may learn the output size; for every pair of parties P_i and P_j, there is a party P_k (possibly $P_k = P_i$ or $P_k = P_j$) who may learn both input sizes of P_i and P_j. (B) When some parties must not learn the output size; for every party P_i who must not learn the output size, P_i may learn all the input sizes, and some other party may learn the input size of P_i and the output size.

For example, if two of n parties must hide their input sizes from each other, then a general size-hiding protocol is infeasible even when the other $n-2$ parties can support the computation. Our result assumes the existence of threshold fully homomorphic encryption (threshold FHE), which is, for example, derived by combining MPC with ordinary FHE; see Appendix A of [LNO13]. The above result shows that *almost all sizes of inputs and the output must be revealed in the standard setting of MPC.*

Our results in the strong secure channel model. In order to circumvent the aforementioned infeasibility, we introduce a new communication model, a *strong secure channel model* such that an adversary cannot learn even the number of bits exchanged among honest parties. We note that this model is justified from steganographic techniques [Cachin04, HAL09], i.e., if communications are hidden from other parties using steganography, an adversary cannot learn the number of communication bits between uncorrupted parties. Moreover, secure steganography is implied by one-way functions, thus, our new model requires no additional assumption inherently. (However, it should also be noted that straightforward implementation of steganography requires large computational and communication cost.)

We show that the feasibility of size-hiding computations is dramatically improved in the strong secure channel model. See Table 1 (part corresponding to strong secure channel) for a summary of our main result. We prove that, in the strong secure channel model, a general size-hiding protocol exists if either the condition (A) holds when the output size is known to all parties (Lemma 6) or the condition (B) holds when the output size is hidden from some parties (Lemma 8). (Unlike our results in the secure channel model, these conditions depend on *what sizes a party may learn.*) We also prove the reverse direction, i.e., there is a function that cannot be securely computed if a given size-hiding class does not satisfy the conditions above (Lemmas 7 and 9). Therefore, it is a necessary and sufficient condition for a general size-hiding protocol.

Surprisingly, in contrast to the standard secure channel model, we show that each input/output size can be hidden from some parties, while the previous model only allows the size of at most one input to be hidden. For example, let us consider the case of three parties where P_1 hides $|x_1|$ from P_2 (but not P_3), P_2 hides $|x_2|$ from P_3 (but not P_1), and P_3 hides $|x_3|$ from P_1 (but not P_2), where $|x_i|$ denotes the size of the input of P_i. Now the number of hidden sizes (three) is beyond the limitation in the previous model mentioned above, but our new model allows computation of a general function even in this case. By generalizing this observation, we see that there are concrete cases where it is possible to hide *all* input and output sizes for any $n > 2$.

The honest-but-randomness-controlling model. In the two-party setting, [LNO13] classified size-hiding classes in terms of feasibility in the honest-but-curious (HBC) model. Recently, [LNO13] (uploaded on IACR ePrint Archive on 01-Apr-2016) revisited that some of their infeasibility results in fact holds in the *honest-but-deterministic* (HBD) model, proposed by Hubácek and Wichs [HW15], rather than the HBC model. In light of the revision, we have also to modify the model since some of our results are based on the results in [LNO13]. However, there is an issue that the HBD model is likely to be incomparable with the HBC model. Alternatively, we introduce a new model, the *honest-but-randomness-controlling* (HBRC) model, where an adversary can use any string as its random tape. We believe that the HBRC model would be a reasonable security model by the following reasons. First, the HBRC model is stronger than the HBC model, i.e., the security in the HBRC model implies the security in the HBC model (see Appendix). Moreover, almost all of the previous standard protocols in the HBC model are also secure in the HBRC model. In particular, all (in)feasibility results in the two-party setting [LNO13] still hold in the HBRC model by an easy observation. We left it as an open problem to give a complete feasibility characterization of both two-party and multiparty settings in the HBC model.

1.2 Our New Techniques

In this section, we clarify the most technical part of size-hiding multiparty computations, and introduce the basic idea for our main results (Section 5).

First, we recall the general *two-party* computation [LNO13]. In their protocols, at least one of the parties *always* learns *all* sizes (i.e., the input size of the other party and the output size). This party can correctly compute any function of input x_1 and x_2 by using FHE. However, in the multiparty setting, we cannot assume the existence of such a party who may learn all sizes, and otherwise, almost all sizes of inputs cannot be protected.

For circumventing the above problem, we develop new techniques which guarantee the computation of the correct output even under the situation where no party knows all of sizes. Our techniques are based on a novel way to use a threshold FHE, and we consider this is the main non-trivial part of this work. More specifically, we propose two independent techniques which handle the following two different cases: **(1)** all parties may learn the output size, and **(2)** some parties must not learn the output sizes. In the rest of this subsection, we explain them more in detail.

(1) The case of public output size. Suppose that parties wish to compute a function while hiding some input sizes, but do not need to hide the output size. In addition, we assume that for every pair of parties P_i and P_j, at least one of n parties (including P_i and P_j) may learn both input sizes of P_i and P_j. In this setting, we call the party who has a longest input a *server*, and the other parties *clients*. In the protocol, all parties perform in the same way as the server since nobody (even the server itself) knows who is the server. We overview the protocol and show the idea behind it as follows.

First, all parties invoke a threshold key generation protocol of FHE. Next, each pair of parties P_i and P_j exchange ciphertexts of their inputs with the support from P_k who may learn both $|x_i|$ and $|x_j|$ as follows. Without loss of generality, we can assume $|x_i| \geq |x_j|$. First, P_i and P_j send ciphertexts $c_i = \mathsf{Enc}_{pk}(1||x_i)$ and $c_j = \mathsf{Enc}_{pk}(1||x_j)$ to the party P_k, respectively. Then, P_k computes a ciphertext $c_{(j,i)} = \mathsf{Enc}_{pk}(0^{|x_i|-|x_j|})||c_j$ and a ciphertext of zeroes $c_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|+1})$, and sends $c_{(j,i)}$ to P_i and $c_{(i,j)}$ to P_j, respectively. We call the former ciphertext a *valid ciphertext*, and the latter all-zero ciphertext a *dummy ciphertext*. Note that nobody except P_k knows whether a ciphertext is the valid one or the dummy one, due to the security of FHE. Next, parties attempt to obtain the output value using homomorphic computation. However, each party cannot estimate the output size since he/she does not know all of input sizes. Thus, a circuit which computes the output value cannot be constructed (the number of output wires is unknown). To avoid the problem, parties first obtain the output size and then compute the desired function as follows. Each party P_i constructs a circuit that takes $x'_1, \cdots, x'_n$ (x'_j is either $0^{|x_i|-|x_j|}||1||x_j$ or $0^{|x_i|+1}$) as inputs, and if one of inputs is all-zero, then outputs all-zero string, otherwise, outputs a representation of the size of the function value $|f(\vec{x})|$ of appropriate length. (For example, $(\log \kappa)^2$ bits, where κ is security parameter, can be used since it must hold $|f(\vec{x})| < 2^{(\log \kappa)^2}$ for sufficiently large κ. See also the discussion at the end of Section 4.2.) Each party P_i computes ciphertext c_i^{size} by homomorphic evaluation of the circuit from the ciphertexts $c_{(1,i)}^{\mathsf{in}}, \cdots, c_{(n,i)}^{\mathsf{in}}$. Then, each party

P_i sends c_i^{size} to all parties. Each party computes c^{size} by homomorphic evaluation of a max function from $c_1^{\mathsf{size}}, \cdots, c_n^{\mathsf{size}}$. The underlying message of c^{size} is the output size since one of the party (specifically, the server) correctly computed the encrypted output size. All parties invoke threshold decryption protocol for the ciphertext c^{size} and obtain the output size. Now we have the output size and thus can construct the circuit that computes the function. In the similar way, all parties can compute c^{out} from which the output value is decrypted. The full description appears in Protocol 2 (Section 5.3).

(2) The case of private output size. Suppose that parties wish to compute a function while hiding some input sizes, and some parties must not learn the output size. In this setting, we call parties who must not learn the output size *servers*, and the other parties *clients*. In addition, we assume that every server may learn all input sizes of parties, and each server may tell its input size to some client (we call such a client a *partner*). We overview the protocol and show the idea behind it as follows.

First, all *clients* execute a threshold key generation protocol of FHE. The reason why servers are not involved in the threshold key generation is that the clients need to be able to decrypt an output ciphertext (whose plaintext length is related to the output size) without servers. Then, every party computes secret shares of its own input (the number of shares is the number of servers), and sends a ciphertext of a share to each server. All servers securely compute ciphertexts c^{size}, whose message is the output size, and c^{out}, whose message is the output, over the FHE. Here, the plaintext for the ciphertext c^{out} is padded zeroes up to L bits, where L is an upper bound of the output size. Note that for every polynomial-time computable function f, there exists a polynomial $p(\cdot)$ such that $|f(x'_1, \cdots, x'_n)| < p(\max(|x'_1|, \cdots, |x'_n|))$ for all $x'_i \in \{0,1\}^*$. Thus, a server can compute the bound $L = p(\max(|x_1|, \cdots, |x_n|))$, since every server knows all input sizes. Next, one server attempts to send c^{size} and c^{out} to all clients, but the length of c^{out} is related to $p(\max(|x_1|, \cdots, |x_n|))$ and it may reveal the maximum input size (possibly private size) to clients. To avoid this, for each client, the server sends the ciphertext whose length only depends on sizes which the client may learn. Let σ_i be the maximum size which P_i may learn. The server sends the truncated ciphertext c^{out} of length $p(\sigma_i)$ to P_i. If a server has the longest input, then the partner learn the maximum input size. Otherwise, it is trivial that there is a client who learns the maximum input size. In any case, at least one of the clients has the ciphertext c^{out} which is not truncated. Then, all clients collaboratively decrypt c^{size} and obtain the output size ℓ. Finally, all clients decrypt ℓ-bit ciphertexts of c^{out} and obtain the output value. The full description appears in Protocol 3 (Section 5.5).

1.3 Related Works

As earlier results relevant to size-hiding computation, Micali, Rabin, and Kilian [MRK03] provided a zero-knowledge set, which is a commitment to a set S that hides also the cardinality of S, where the committer can prove $x \in S$ or

$x \notin S$ for any string x. Ishai and Paskin [IP07] constructed a public key encryption scheme that can evaluate any branching program in such a way that the size of the program is hidden. These works concentrated on efficient realization of specific functionalities, while our work aims at clarifying the (in)feasibility of general size-hiding computation depending on a given size-hiding class. On the other hand, Chase and Visconti [CV12] showed the first size-hiding protocol in the presence of malicious adversaries for a specific task, secure database commitments. Chase, Ostrovsky and Visconti [COV15] strengthened the feasibility results of [CV12, LNO13] by constructing a general size-hiding two-party protocol in the presence of malicious adversaries while hiding input size of one party. In contrast, we only consider the honest-but-randomness-controlling adversaries in this paper. We leave constructions of size-hiding MPC protocols against malicious adversaries as future work.

2 Preliminaries

We review the basic notations and the definition of threshold FHE.

2.1 Basic Notations

Throughout this paper, we use the following notations: "$\mathbb{N}$" denotes the set of natural numbers, i.e., $\mathbb{N} = \{1, 2, 3, \cdots\}$. "$\log x$" denotes the logarithm of x with the base two, i.e., $\log_2 x$. "$x||y$" denotes the concatenation of x and y. "$|x|$" denotes the bit length of x. "$\emptyset$" denotes an empty set. If S is a finite set, then "$x \overset{\mathrm{U}}{\leftarrow} S$" denotes that x is chosen uniformly at random from S. If $\vec{v}$ is a vector, "$\vec{v}[i]$" denotes the i-th element of the vector. If $m = m_1 m_2 \cdots m_\ell \in \{0,1\}^\ell$ is a plaintext and Enc_{pk} is an encryption algorithm for 1-bit message, "$c = \mathsf{Enc}_{pk}(m)$" denotes a vector of ℓ ciphertexts $(c_1, c_2, \cdots, c_\ell)$, where c_i is a ciphertext $c_i = \mathsf{Enc}_{pk}(m_i)$. If $I = \{i_1, \cdots, i_t\}$ is a subset of $\mathbb{N}$, "x_I" denotes the set $x_I = \{x_{i_1}, \cdots, x_{i_t}\}$. If $I = (i_1, \cdots, i_t)$ is an element of $\mathbb{N}^t$, "x_I" denotes the vector $x_I = (x_{i_1}, \cdots, x_{i_t})$. If $\Phi = \{\Phi(x, \kappa)\}_{x,\kappa}$ and $\Psi = \{\Psi(x, \kappa)\}_{x,\kappa}$ are probability distributions indexed by $\kappa \in \mathbb{N}$ and $x \in X_\kappa$ where X_κ is an auxiliary parameter set indexed by κ, then we say that Φ and Ψ are computationally indistinguishable, denoted by "$\Phi \overset{\mathrm{c}}{\equiv} \Psi$", if for every non-uniform probabilistic polynomial-time (PPT) algorithm $\mathcal{D}$ and every (positive) polynomial p, there exists a number $\kappa_0 \in \mathbb{N}$ with the property that $\left|\Pr[\mathcal{D}(\Phi(x,\kappa)) = 1] - \Pr[\mathcal{D}(\Psi(x,\kappa)) = 1]\right| < 1/p(\kappa)$ for any $\kappa \in \mathbb{N}$ with $\kappa > \kappa_0$ and any $x \in X_\kappa$.

2.2 Threshold Fully Homomorphic Encryption

We present a definition of threshold FHE. Asharov et al. [AJL+12] constructed an efficient threshold FHE scheme from the learning with error assumption, whose threshold key generation and threshold decryption protocols have only one round. In general, the threshold version of FHE is implied from an ordinary FHE scheme [LNO13].

Definition 1 (Threshold FHE) *We say that a tuple of protocols and algorithms* $(\mathsf{ThrGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{ThrDec})$ *is a* threshold FHE scheme *if* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a public-key encryption with message space* $\{0,1\}$*, that is secure under chosen-plaintext attacks, and the protocols* ThrGen *and* ThrDec *with parties* $P_1, \cdots, P_n$ *realize the following functionalities and the following conditions:*

Threshold Key Generation: *The functionality of* ThrGen *takes security parameter* 1^κ *from* $P_1, \cdots, P_n$*, computes* $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ *and chooses uniformly random values* $sk_1, \cdots, sk_{n-1} \in \{0,1\}^{|sk|}$*. Then, the functionality outputs* (pk, sk_i) *to each* P_i $(i = 1, \cdots, n)$*, where* $sk_n = sk_1 \oplus \cdots \oplus sk_{n-1} \oplus sk$*.*

Threshold Decryption: *For a subset* $I \subset \{1, \cdots, n\}$*, the functionality of* ThrDec_I *takes security parameter* 1^κ*, a ciphertext* c *and shares of secret key* $sk_1, \cdots, sk_n$ *from* $P_1, \cdots, P_n$*, computes* $m = \mathsf{Dec}_{sk_1 \oplus \cdots \oplus sk_n}(c)$*, and outputs* m *to each* P_i $(i \in I)$*. If it holds* $I = \{1, \cdots, n\}$*, we omit the index* I*.*

Correctness: *For every polynomial-size circuit* $\mathcal{C}$ *that takes* n *inputs, and every inputs of the circuit* $m_1, \cdots, m_n \in \{0,1\}$*:*

$$\Pr\left[\mathsf{Dec}_{sk}(\mathsf{Eval}_{pk}(\mathcal{C}, \mathsf{Enc}_{pk}(m_1), \cdots, \mathsf{Enc}_{pk}(m_n))) = \mathcal{C}(m_1, \cdots, m_n)\right] = 1,$$

where the probability is taken over the random coins of all the algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*.*

Security of the Threshold Key Generation: *There exists a PPT* $\mathcal{S}_{\mathsf{ThrGen}}$ *such that for every* $I \subsetneq \{1, \cdots, n\}$*, the view in a real execution of* ThrGen *with security parameter* κ *is computationally indistinguishable from the output of* $\mathcal{S}_{\mathsf{ThrGen}}$ *with inputs* $I, 1^\kappa$ *and keys obtained by* P_i $(i \in I)$*.*

Security of the Threshold Decryption: *There exists a PPT* $\mathcal{S}_{\mathsf{ThrDec}}$ *such that for every* $I \subsetneq \{1, \cdots, n\}$*, the view in a real execution of* ThrDec *with security parameter* κ *is computationally indistinguishable from the output of* $\mathcal{S}_{\mathsf{ThrDec}}$ *with inputs a subset* I*, keys, the ciphertext and the decrypted value.*

3 Size-Hiding Computation

In this section, first, we give a definition of size-hiding classes and provide their graphical representations in Section 3.1. Second, as an extension of [LNO13] to n-party settings, we give definitions of polynomial-time protocols and the security of size-hiding protocols in Section 3.2. Next, for later references, we review the previous two-party results [LNO13] using our graphical representation in Section 3.3. Finally, we introduce tools for proving lemmas in the later section, *protocol compilers*, that can derive a size-hiding protocol from another protocol in Section 3.4.

3.1 Classes of Size-Hiding

We provide a definition of a class of size-hiding that specifies *what sizes a party may learn* in an execution of a protocol. A size-hiding class can be represented by $(G, \vec{v})$, where G is a directed graph which specifies how input sizes are hidden (more precisely, which input size may be known to which party), and $\vec{v}$ is a vector which specifies how the output size may be known to each party. A directed graph G with n vertices is called an *input size graph with n vertices*, a vector $\vec{v}$ with n elements is called an *output size vector with n elements*, and a tuple $(G, \vec{v})$ is called a *size-hiding class with n parties*.

An input size graph with n vertices has a set of vertices $V(G) = \{1, 2, \cdots, n\}$ and a set of edges $E(G)$. Each vertex $i \in V(G)$ corresponds to the party P_i. If there is an edge $(j, i) \in E(G)$ directed from j to i, the party P_i *may* learn $|x_j|$, which is the input size x_j of P_j in a protocol execution. If there is no edge (j, i), the party P_i *must not* learn any partial information of $|x_j|$ except trivial information which can be computed from other information that P_i obtained legally. From now on, we assume that any input size graph with n vertices has edges $(1, 1), (2, 2), \cdots, (n, n)$ since P_i always knows its own input size $|x_i|$.

An output size vector with n elements is a member of $\{\perp, |\mathsf{f}|, \mathsf{f}\}^n$, where $\perp, |\mathsf{f}|$ and f are symbols that represent how to receive the output information. The i-th element $\vec{v}[i]$ specifies how P_i receives the output information. If $\vec{v}[i] = \perp$, the party P_i *must not* receive any partial information of the output $f(\vec{x})$ (except trivial information which can be computed efficiently). If $\vec{v}[i] = |\mathsf{f}|$, the party P_i *may* learn the output size $|f(\vec{x})|$ but *must not* receive $f(\vec{x})$ beyond the size information $|f(\vec{x})|$ (except trivial information). If $\vec{v}[i] = \mathsf{f}$, the party P_i *must* learn $f(\vec{x})$. From now on, we assume that any output size vector $\vec{v}$ contains at least one f since if there is no f in $\vec{v}$, nobody obtains the output $f(\vec{v})$ even though the protocol aims at computing the function f.

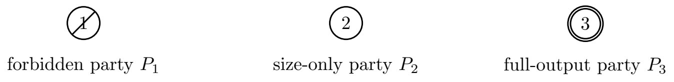

Fig. 1. Graphical representation of parties

We provide a graphical representation of a size-hiding class $(G, \vec{v})$. We use a circle to denote a vertex of G, and an arrow $i \to j$ to denote an edge $(i, j) \in E(G)$. For simplicity, we omit arrows $1 \to 1, 2 \to 2, \cdots, n \to n$ since the edges $(1, 1), (2, 2), \cdots, (n, n) \in E(G)$ always exist. We also use three types of circles to denote the output size vector $\vec{v}$ as follows. For a vertex i, we use a double circle to denote $\vec{v}[i] = \mathsf{f}$, a normal circle to denote $\vec{v}[i] = |\mathsf{f}|$, and a forbidden circle to denote $\vec{v}[i] = \perp$; see Figure 1.

Figure 2 is an example of a size-hiding class $(G, \vec{v})$ as follows: The input size graph with 3 vertices G has a set of vertices $V(G) = \{1, 2, 3\}$ and a set of edges $E(G) = \{(1, 2), (2, 3), (1, 3), (1, 1), (2, 2), (3, 3)\}$. The output size vector with 3 elements is a vector $\vec{v} = (\mathsf{f}, |\mathsf{f}|, \perp)$. The size-hiding class $(G, \vec{v})$ means the following: The party P_1 may learn $|x_1|$, must learn $f(\vec{x})$, and must not learn $|x_2|$ nor $|x_3|$. The party P_2 may learn $|x_1|, |x_2|$ and $|f(\vec{x})|$, and must not learn $|x_3|$ nor $f(\vec{x})$. The party P_3 may learn $|x_1|, |x_2|$, and must not learn $|f(\vec{x})|$.

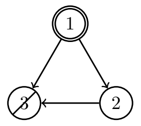

Fig. 2. Example of a size-hiding class

Throughout this paper, we use the following terminology.

Public size is a size which all parties may learn. Formally, we say that an input size $|x_i|$ is public if there are edges $(i,1),(i,2),\cdots,(i,n) \in E(G)$, and the output size is public if the output size vector $\vec{v}$ is an element of $\{|\mathsf{f}|,\mathsf{f}\}^n$.

Private size is a size which some parties must not learn. Formally, we say that an input size $|x_i|$ is private if there is a vertex $j \in V(G)$ such that $(i,j) \notin E(G)$, and the output size is private if there is an index i such that $\vec{v}[i] = \perp$.

Forbidden party is a party who must not learn the output size. "$I_\perp$" denotes all indices of the forbidden parties, i.e., $I_\perp = \{i \mid \vec{v}[i] = \perp\} \subset \{1,\cdots,n\}$.

Size-only party is a party who may learn the output size but must not learn the exact output value. "$I_{|\mathsf{f}|}$" denotes all indices of size-only parties, i.e., $I_{|\mathsf{f}|} = \{i \mid \vec{v}[i] = |\mathsf{f}|\} \subset \{1,\cdots,n\}$.

Full-output party is a party who must learn the output. "I_f" denotes all indices of full-output parties, i.e., $I_\mathsf{f} = \{i \mid \vec{v}[i] = \mathsf{f}\} \subset \{1,\cdots,n\}$.

Permitted party is a party who may learn the output size. "I_p" denotes all indices of permitted parties, i.e., $I_\mathsf{p} = I_{|\mathsf{f}|} \cup I_\mathsf{f}$. It holds $I_\perp \cup I_\mathsf{p} = I_\perp \cup I_{|\mathsf{f}|} \cup I_\mathsf{f} = \{1,\cdots,n\}$.

3.2 Basic Notions for Size-hiding Multiparty Protocols

Our definitions of notions for size-hiding n-party protocols follow the two-party version of [LNO13]. Let $(G,\vec{v})$ be a size-hiding class with n parties, and let f be an n-ary polynomial-time computable function $f : (\{0,1\}^*)^n \to \{0,1\}^*$. Let π be an n-party protocol with parties $P_1,\cdots,P_n$, and let $\kappa \in \mathbb{N}$ be a security parameter of π. Each party P_i has an input $x_i \in \{0,1\}^*$, which may be polynomially unbounded. We denote by $\mathrm{TIME}^{\pi}_{P_i}(\vec{x},\kappa)$ the running time of P_i in π for the inputs $\vec{x} = (x_1,\cdots,x_n)$. We denote by $\mathrm{OUTPUT}^{(G,\vec{v},f)}_i(\vec{x})$ the P_i's output specified by $(G,\vec{v})$, e.g., for the example of Figure 2, we have that $\mathrm{OUTPUT}^{(G,\vec{v},f)}_1(\vec{x}) = (f(\vec{x}))$, $\mathrm{OUTPUT}^{(G,\vec{v},f)}_2(\vec{x}) = (1^{|f(\vec{x})|}, 1^{|x_1|})$ and $\mathrm{OUTPUT}^{(G,\vec{v},f)}_3(\vec{x}) = (1^{|x_1|}, 1^{|x_2|})$. Now we are ready to define a polynomial-time protocol for $(G,\vec{v},f)$.

Definition 2 (Polynomial-time protocol) *Let $(G,\vec{v})$ be a size-hiding class with n parties, let f be an n-ary function, and let π be an n-party protocol. We say that π is a polynomial-time protocol for $(G,\vec{v},f)$ if there exists a polynomial $p(\cdot)$ such that for every $\kappa \in \mathbb{N}$, every $\vec{x} \in (\{0,1\}^*)^n$ and every $i \in \{1,\cdots,n\}$,*

$$\mathrm{TIME}^{\pi}_{P_i}(\vec{x},\kappa) \le p(|x_i| + |\mathrm{OUTPUT}^{(G,\vec{v},f)}_i(\vec{x})| + \kappa)$$

Next, we define the security of protocols against *honest-but-randomness-controlling* (HBRC) adversaries in the secure channel model (See Section 1.1 and Appendix for details of the HBRC model). In the HBRC model, a simulator must simulate a transcript on given random tapes which is produced by a *randomness producer*. It is a PPT algorithm that chooses corrupted parties' random tapes. Formally, we say that a PPT $\mathcal{R}$ is a *randomness producer* if $\mathcal{R}(1^\kappa, I)$ outputs a vector of strings $\vec{r}_I = (r_{i_1}, \cdots, r_{i_t}) \in (\{0,1\}^*)^{|I|}$ for all $I = \{i_1, \cdots, i_t\} \subsetneq \{1, \cdots, n\}$.

We denote by $\mathrm{MSIZE}^\pi(\vec{x})$ the numbers of all bits exchanged among $P_1, \cdots, P_n$ in an execution of π with inputs $\vec{x}$, expressed by unary expression such as $1^{|m|}$. The view of the party P_i during an execution of π with inputs $\vec{x}$ is defined as $\mathrm{view}_i^\pi(\vec{x}) = (x_i, r_i, m_{i_1}, \cdots, m_{i_t})$, where r_i is his internal coin tosses and m_{i_j} is the j-th message that was received by P_i in the protocol execution. We also use $\mathrm{view}_i^\pi(\vec{x})|_{r_i} = (x_i, m_{i_1}, \cdots, m_{i_t})$ to denote $\mathrm{view}_i^\pi(\vec{x})$ on given randomness r_i. Here, if the length of r_i is shorter than the length of its internal randomness, its internal randomness is $r_i||0^k$ for appropriate $k \in \mathbb{N}$.

Definition 3 (Security in the secure channel model) *Let $(G, \vec{v})$ be a size-hiding class with n parties, let f be an n-ary function, and let π be a polynomial-time protocol for $(G, \vec{v}, f)$. We say that π correctly computes $(G, \vec{v}, f)$ if for every $\kappa \in \mathbb{N}$, and every $\vec{x} \in (\{0,1\}^*)^n$, all full-output parties output $f(\vec{x})$ at the end of the execution of π with the input $\vec{x}$ and security parameter κ. We say that π realizes $(G, \vec{v}, f)$ in the* secure channel model *if π correctly computes $(G, \vec{v}, f)$ and for every randomness producer $\mathcal{R}$, there exists a PPT $\mathcal{S}$ such that for every $I \subsetneq \{1, \cdots, n\}$, every polynomials $q_1, q_2, \cdots, q_n$,*

$$\left\{\mathcal{S}(1^\kappa, I, \vec{x}_I, \mathrm{OUTPUT}_I^{(G,\vec{v},f)}(\vec{x}), \vec{r}_I \leftarrow \mathcal{R}(1^\kappa, I))\right\}_{\kappa,\vec{x}} \overset{\mathrm{c}}{\equiv} \left\{(\mathrm{view}_I^\pi(\vec{x})|_{\vec{r}_I}, \mathrm{MSIZE}^\pi(\vec{x}))\right\}_{\kappa,\vec{x}}$$

where $x_1 \in \{0,1\}^{q_1(\kappa)}, \cdots, x_n \in \{0,1\}^{q_n(\kappa)}$.

In this paper, we focus on which size-hiding class has a general protocol. For a size-hiding class $(G, \vec{v})$, we say that $(G, \vec{v})$ is *feasible* if for every polynomial-time computable function f, there exists a protocol π that realizes $(G, \vec{v}, f)$ in the secure channel model. On the other hand, we say that $(G, \vec{v})$ is infeasible if it is not feasible.

3.3 Overview of the Two-Party Results

We overview the results in the two-party setting shown by Lindell, Nissim and Orlandi [LNO13] using our graphical representation. Later, we use them in order to prove infeasibility results in multiparty settings. We note that their original paper shows their feasibility and infeasibility against honest-but-curious adversaries. However, very recently, they (implicitly) revised the infeasibility of class 1.d is in fact holds against honest-but-randomness-controlling (HBRC) adversaries[5] rather than honest-but-curious adversaries. Since all of their protocol

[5] Their original revision states that it holds against HBD adversaries. But it also holds in the HBRC model.

can be easily modified to the HBRC setting, the following results are based on the HBRC model.

They defined three classes of size-hiding: (class 0) the input sizes of both parties are revealed, (class 1) the input size of one party is revealed and the other is hidden, (class 2) the input sizes of both parties are hidden. In addition, they define five subclasses of class 1, and three subclasses of class 2.

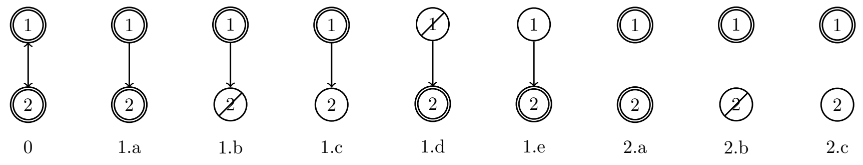

Fig. 3. Graphical representations of subclasses in the two-party setting

Let $(G_0, \vec{v}_0), (G_{1.a}, \vec{v}_{1.a}), \cdots, (G_{2.c}, \vec{v}_{2.c})$ be size-hiding classes $0, 1.a, \cdots, 2.c$ in Figure 3, respectively. They (implicitly) showed that size-hiding classes $(G_0, \vec{v}_0)$, $(G_{1.a}, \vec{v}_{1.a}), (G_{1.c}, \vec{v}_{1.c})$ and $(G_{1.e}, \vec{v}_{1.e})$ are feasible while the other classes are infeasible in the HBRC model. Later in this paper, we use the following results.

– There is a two-ary function f such that the functionality $(G_{1.b}, \vec{v}_{1.b}, f)$ cannot be realized. An example of the f is the oblivious transfer; see Section 4.3.
– There is a two-ary function f such that the functionality $(G_{1.d}, \vec{v}_{1.d}, f)$ cannot be realized. An example of the f is an oblivious multi-input pseudorandom function evaluation omprf introduced in [LNO13].
– There is a two-ary function f with constant output length such that the functionality $(G_{2.a}, \vec{v}_{2.a}, f)$ cannot be realized. An example of the f is the binary inner product $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}$; see [LNO13].

3.4 Tools for infeasibility – Protocol Compilers

Here we introduce auxiliary algorithms used in the proofs of our infeasibility results, which we call *protocol compilers*. Namely, to give a proof by contradiction, we start with an n-party protocol for a given size-hiding class whose existence is assumed, and convert it by the protocol compilers into a two-party size-hiding protocol for some size-hiding class, where the existence of the latter protocol has been denied by the result of [LNO13]. Below we give two kinds of protocol compilers, which we call a *reduction compiler* and a *wrapping compiler*.

Reduction Compiler. A *reduction compiler* takes as inputs an MPC protocol with $P_1, \cdots, P_n$ and two subsets I_1, I_2 that is a partition of $\{1, \cdots, n\}$, and outputs a two-party protocol with P'_1 and P'_2, where P'_i $(i \in \{1,2\})$ behaves in the same way as $\{P_j\}_{j \in I_i}$. More concretely, if P_i computes/sends/receives messages in π, then P'_j $(i \in I_j)$ behaves in the same way as P_i. At the end of the compiled protocol, if P_i outputs $f(\vec{x})$ in π, then P'_j $(i \in I_j)$ outputs $f(\vec{x})$ in π'. The reason why we call it a "reduction" compiler is that it *reduces* the number of parties, and we use it in a *reduction* to prove infeasibility results.

Lemma 1. *Let f' be a two-ary function, let f be an n-ary function such that $f(x_1, x_2, \cdots, x_n) = f'(x_1, x_2)$, let $(G, \vec{v}, f)$ be a functionality for n parties, and let π be a protocol that realizes $(G, \vec{v}, f)$. Let I_1 and I_2 be non-empty subsets of $\{1, \cdots, n\}$ such that $1 \in I_1$, $2 \in I_2$, $I_1 \cap I_2 = \emptyset$ and $I_1 \cup I_2 = \{1, 2, \cdots, n\}$. There exists a protocol π' that realizes a functionality $(G', \vec{v}', f')$ as follows:*

- *The party P'_1 has the same input x_1 of P_1, and the party P'_2 has the same input x_2 of P_2.*
- *The input size graph with two parties G' has a set of edges $E(G')$ as follows. The edge $(1, 2)$ exists in $E(G')$ if and only if an edge $(1, i)$ exists in $E(G)$ such that $i \in I_2$. Similarly, the edge $(2, 1)$ exists in $E(G')$ if and only if an edge $(2, i)$ exists in $E(G)$ such that $i \in I_1$.*
- *$\vec{v}'$ is an output size vector with two elements such that $\vec{v}'[i] = \max_{j \in I_i}(\vec{v}[j])$, for an order $\perp < |\mathsf{f}| < \mathsf{f}$.*

Proof. Based on a simulator $\mathcal{S}$ of the protocol π, we construct a simulator $\mathcal{S}'$ of the protocol π'. By the symmetry, it suffices to show the simulator when P'_1 is corrupted. Given 1^κ, $I' = \{1\}$, the input x_1, the output $f(x_1, x_2)$ if there is an full-output party P_i $(i \in I_1)$, and a random tape r_1 produced by a randomness producer, $\mathcal{S}'$ invokes $\mathcal{S}$ on the same inputs except I_1 instead of I'. Since the simulator $\mathcal{S}'$ works correctly, the protocol π' securely computes $(G', \vec{v}', f')$. $\square$

Wrapping Compiler. For a subset $I \subset \{1, \cdots, n\}$, we say that a protocol π is I-independent[6] if there exists a polynomial p such that for every $\kappa \in \mathbb{N}$ and every $\vec{x} \in (\{0,1\}^*)^n$, the output size and the number of bits, exchanged among all parties in an execution of π with κ and $\vec{x}$, are upper bounded by $p(\kappa, |x_{j_1}|, \cdots, |x_{j_t}|)$ except negligible probability, where $\{j_1, \cdots, j_t\} \cap I = \emptyset$. A *wrapping compiler* takes an I-independent protocol π (it is not necessary for π to be secure) that computes $f(\vec{x})$, and outputs a size-hiding protocol π' that computes $f(\vec{x})$ while hiding the inputs $|x_i|$ $(i \in I)$ from all parties. It is used in the proof of Lemma 4. The following lemma is the security of a protocol that is compiled by the wrapping compiler.

Lemma 2. *Let I be a non-empty subset of $\{1, \cdots, n\}$, and let π be an I-independent protocol computing $f(\vec{x})$ with $P_1, \cdots, P_n$. Assume that threshold FHE exists. There exists a protocol π' that realizes a functionality $(G', \vec{v}', f)$ as follows:*

- *The party P'_i $(i \in \{1, \cdots, n\})$ has the same input x_i of P_i.*
- *G' is an input size graph with n parties, where $\{|x_i|\}_{i \in I}$ are private sizes and the others are public sizes.*

[6] It is an generalization of *size independent protocol*; see Section 4.3 in [LNO13].

– *$\vec{v}'$ is any element of $\{|\mathsf{f}|, \mathsf{f}\}^n$.*

Proof. Given an I-independent protocol π that computes $f(\vec{x})$, the compiled protocol π' with $P'_1, \cdots, P'_n$ proceeds as follows. First, parties execute a threshold key generation protocol of threshold FHE, and each party encrypts own input under the public key. Second, every party P'_j ($j \notin I$) sends $|x_j|$ to all parties. On receiving $|x_{j_1}|, \cdots, |x_{j_t}|$ (they are not independent sizes), parties compute the upper bound $B = p(|x_{j_1}|, \cdots, |x_{j_t}|)$. Since the communication complexity is bounded by B, each party P_i can construct a circuit that can produce the next messages of P_i. More concretely, for each k round (k is also bounded by B), the circuit takes as inputs previous messages that are received by P_i at $1, 2, \cdots, k-1$ rounds and P_i's input x_i, and outputs the next messages for each party. Using these circuits, all parties homomorphically evaluate the protocol π. Finally, parties obtain an output ciphertext (whose message is of length B), invoke a threshold decryption protocol, and obtain the output value. (It is easy to obtain a protocol for *any* $\vec{v}' \in \{|\mathsf{f}|, \mathsf{f}\}^n$ by specifying parties who can obtain the output appropriately.)

Now we show the above protocol π' realizes $(G', \vec{v}', f)$ in the secure channel model. In order to prove the security of π', we construct a simulator $\mathcal{S}$ that can generate views of corrupted parties. Given 1^κ, $I \subsetneq \{1, \cdots, n\}$, the inputs $\vec{x}_I$, the output $f(\vec{x})$ (or the output size $|f(\vec{x})|$), all input sizes which are not independent sizes $\{1^{|x_{j_1}|}, \cdots, 1^{|x_{j_t}|}\}$, and random tapes $\vec{r}_I$ produced by a randomness producer, the simulator $\mathcal{S}$ first computes the upper bound $B = p(\kappa, |x_{j_1}|, \cdots, |x_{j_t}|)$. Second, $\mathcal{S}$ simulates a threshold key generation protocol, and computes ciphertexts of x_i for all $i \in I$. Next, $\mathcal{S}$ simulates messages sent by P_i to P_j ($i \in \{1, \cdots, n\}$ and $j \in I$) as follows. If P_i is corrupted, $\mathcal{S}$ does the same as P_i. Otherwise, $\mathcal{S}$ computes a ciphertext for zero string of appropriate length. At the end of the protocol π', $\mathcal{S}$ simulates a threshold decryption protocol. Finally, $\mathcal{S}$ computes message sizes, and outputs views of corrupted parties and message sizes generated as above. The views generated by $\mathcal{S}$ are indistinguishable from the views in a real execution of the protocol due to the IND-CPA security of FHE and the security of the threshold protocols. Thus, the protocol π' securely computes $(G', \vec{v}', f)$ in the secure channel model. □

4 Results in the Secure Channel Model

In this section, we show that every function can be realized while hiding one (input or output) size in the secure channel model. On the other hand, we also prove that there exists a function that cannot be realized while hiding two or more (input or output) sizes in the secure channel model. Our result shows that, in the secure channel model, a general size-hiding protocol exists only in the case where parties wish to hide at most one of $n + 1$ (n inputs and the output) sizes. In Section 4.1, we give a formal statement of our result in Theorem 1, and show examples of (feasible or infeasible) classes. Then, we show the feasibility part of the theorem in Section 4.2, and the infeasibility part of the theorem in Section 4.3.

4.1 Our Result

Our result in the secure channel model is as follows.

Theorem 1 *Let $(G, \vec{v})$ be a size-hiding class with n parties. Assume that threshold FHE exists. The class $(G, \vec{v})$ is feasible in the secure channel model if and only if the number of private sizes of $(G, \vec{v})$ is at most 1.*

Examples. Examples of feasible size-hiding classes are shown in Figure 4. The number of private sizes of them is just one. On the other hand, classes shown in Figure 5 are infeasible. The number of private sizes of the left and the center graphs is two, and of the right graph is three.

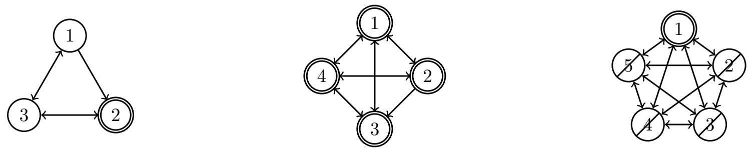

Fig. 4. Examples of feasible classes

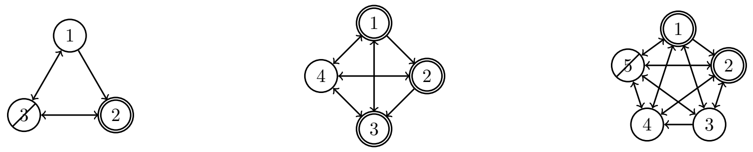

Fig. 5. Examples of infeasible classes

4.2 Protocol Hiding One Size

We construct a general size-hiding MPC protocol that can hide one (input or output) size, in order to show the feasibility part of Theorem 1. The case where only the output size is private is an easy application of ordinary MPC. Indeed, since now the input sizes are public, the output size also has a public and efficient upper bound derived from the complexity of the function f, therefore the output size can be hidden from the forbidden parties by a naive padding technique.

From now on, we consider the case where the output size is public. Let "server" denote the unique party who wants to hide its own input size, and let "clients" denote the other parties. The outline of the protocol construction, which is a natural extension of the two-party results for classes 1.a, 1.c, and

1.e in [LNO13], is explained as follows. Each client sends an FHE ciphertext of its own input to the server, which can be freely performed since their input sizes are public. Given these ciphertexts, the server seems to be able to compute the encrypted output of the function using homomorphic evaluation, which is then decrypted for the full-output parties by the threshold decryption. However, the ciphertext of the output value may have a length longer than the actual output size since the precise inputs are not known at the homomorphic evaluation, and the difference from the actual output size may reveal some non-trivial information on the server's input size. To avoid the problem, the server first homomorphically computes the ciphertext of the *actual output length* ℓ, and the parties know ℓ via the threshold decryption. Then the server generates the ciphertext of the output value where the length is exactly set to ℓ, which prevents the leak of the server's input size mentioned above.

The full description of the protocol appears in Protocol 1. In the following argument, we assume by symmetry that P_1 is the server.

Protocol 1 *Suppose that parties $P_1, P_2, \cdots, P_n$ have inputs $x_1, x_2, \cdots, x_n$, respectively, and all sizes are public except the input size $|x_1|$ of the party P_1. The protocol proceeds as follows.*

1. *All parties invoke a* ThrGen *protocol with inputs 1^κ, and each party P_i obtains a public key pk and a share of the secret key sk_i.*
2. *Each party P_i computes $c_i^{\mathsf{in}} = \mathsf{Enc}_{pk}(x_i)$, and sends c_i^{in} to P_1.*
3. *P_1 constructs a circuit C_{size}, which takes $\vec{x}$ as inputs and outputs $|f(\vec{x})|$ padded with zeroes up to $(\log \kappa)^2$ bits. Then, P_1 computes $c^{\mathsf{size}} \leftarrow \mathsf{Eval}_{pk}(C_{\mathsf{size}}, c_1^{\mathsf{in}}, \cdots, c_n^{\mathsf{in}})$ and sends c^{size} to all parties.*
4. *All parties invoke a* ThrDec *protocol with the ciphertext c^{size}, and obtain the decrypted value ℓ.*
5. *P_1 computes $c^{\mathsf{out}} \leftarrow \mathsf{Eval}_{pk}(C_{\mathsf{out}}, c_1^{\mathsf{in}}, \cdots, c_n^{\mathsf{in}})$, where the circuit C_{out} computes $f(x_1, \cdots, x_n)$ of length ℓ, and sends c^{out} to all parties.*
6. *All parties invoke a $\mathsf{ThrDec}_{I_{\mathsf{f}}}$ protocol with the ciphertext c^{out} as only full-output parties obtain the decrypted value $z \in \{0,1\}^\ell$. Then, all full-output parties output z, and the other parties output nothing. The protocol terminates.*

Lemma 3 (Security of Protocol 1). *Let $(G, \vec{v}, f)$ be a functionality with n parties, where $|x_1|$ is private and the other sizes are public. Assume that threshold FHE exists. Then, Protocol 1 realizes the functionality $(G, \vec{v}, f)$ in the secure channel model.*

Proof. In order to prove the security, we construct a simulator $\mathcal{S}$ that, given inputs, outputs, and random tapes of corrupted parties, generates their view in the protocol. We note that it suffices to only consider the most difficult case that $|x_1|$ is hidden from *all* other parties. Given 1^κ, $I = \{i_1, \cdots, i_t\}$, the inputs $\vec{x}_I$, public sizes $(1^{|x_2|}, \cdots, 1^{|x_n|}, 1^{|f(\vec{x})|})$, the output $f(\vec{x})$ if $I \cap I_{\mathsf{f}} \neq \emptyset$, and random tapes $\vec{r}_I = (r_{i_1}, \cdots, r_{i_t})$ produced by a randomness producer, the simulator $\mathcal{S}$ works as follows. (In the following probabilistic computation, $\mathcal{S}$ uses a string $r_i || 000 \cdots$ as P_i's random tape.) First, $\mathcal{S}$ computes $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, chooses $sk_i \overset{\mathrm{U}}{\leftarrow} \{0,1\}^{|sk|}$ for all $i \in I$, and simulates a threshold key generation protocol under the keys. If P_1 is corrupted, $\mathcal{S}$ computes $c_i^{\mathsf{in}} = \mathsf{Enc}_{pk}(x_i)$ $(i \in I)$, $c_i^{\mathsf{in}} = \mathsf{Enc}_{pk}(0^{|x_i|})$ $(i \notin I)$, and evaluates c^{size} and c^{out} from these ciphertexts. Otherwise, $\mathcal{S}$ computes $c^{\mathsf{size}} = \mathsf{Enc}_{pk}(0^{(\log\kappa)^2})$ and $c^{\mathsf{out}} = \mathsf{Enc}_{pk}(0^{|f(\vec{x})|})$. Next, $\mathcal{S}$ simulates threshold decryption protocols for c^{size} and c^{out}. Then, $\mathcal{S}$ computes message sizes $\mathrm{MSIZE}^{\pi}(\vec{x})$. ($\mathcal{S}$ can compute them since all sizes of messages are only dependent on the public sizes $|x_2|, \cdots, |x_n|$ and $|f(\vec{x})|$.) Finally, $\mathcal{S}$ outputs views of corrupted parties and message sizes generated as above.

Let us observe the difference between the view generated in a real execution and the view generated by $\mathcal{S}$. The views of threshold key generation and threshold decryption protocols generated by $\mathcal{S}$ are indistinguishable from them in a real execution due to the security of these protocols. The ciphertexts generated by $\mathcal{S}$ are indistinguishable from them in a real execution due to the IND-CPA security of the underlying FHE scheme. Therefore, the above protocol realizes the functionality in the secure channel model. □

Fully avoiding upper bounding of input sizes. In the same way as the protocols in [LNO13], Protocol 1 above assumes that all input sizes are bounded by $2^{(\log\kappa)^2} = \kappa^{\log\kappa}$. From the viewpoint of security, this restriction causes no problems, since now the input sizes are polynomially bounded and thus the bound above indeed holds asymptotically. However, it may cause a problem from the viewpoint of correctness, since now the polynomial bounds for input sizes do not exist and the correctness should be satisfied at every parameter κ rather than just asymptotically. To resolve the issue, we show the assumption $|x_i| < 2^{(\log\kappa)^2}$ can be avoided by using a *flag technique*. A flag function $\mathsf{flag}_\ell : \{0,1\}^\ell \to \{0,1\}^\ell$ takes $x = x_\ell \cdots x_2 x_1 \in \{0,1\}^\ell$ as an input, and outputs $z = 0^{\ell-i}||1^i$, where i is an index such that $i = \max(j-1 \text{ s.t. } x_j = 1)$. For example, a flag function flag_{10} with an input $x = 0010000001$ outputs $z = 0001111111$. Next we explain how to use the flag function in Protocol 1. Let p be a polynomial such that $|f(x'_1, \cdots, x'_n)| < p(|x'_1|, \cdots, |x'_n|)$ for all $x'_i \in \{0,1\}^*$. In step 3, the party P_1 first computes $B = \log_2 p(|x_1|, \cdots, |x_n|)$, and then constructs a circuit C_{size}, which takes $\vec{x}$ as inputs and outputs $|f(\vec{x})|$ padded with zeroes up to $B = \log_2 p(|x_1|, \cdots, |x_n|)$, and a circuit C_{flag}, which takes $x \in \{0,1\}^B$ as an input and outputs a string $\mathsf{flag}_B(x)$. Then, P_1 computes $c^{\mathsf{size}} \leftarrow \mathsf{Eval}_{pk}(C_{\mathsf{size}}, c_1^{\mathsf{in}}, \cdots, c_n^{\mathsf{in}})$ and $c^{\mathsf{flag}} \leftarrow \mathsf{Eval}_{pk}(C_{\mathsf{flag}}, c^{\mathsf{size}})$. For $i = 1, 2, 3, \cdots$, P_1 sends $c^{\mathsf{flag}}[i]$ to all parties, and parties decrypt it. If the decrypted value equals zero, then P_1 sends $(c^{\mathsf{size}}[j])_{1\le j\le i}$ to all parties, otherwise, continue the loop. Now $(c^{\mathsf{size}}[j])_{1\le j\le i}$ indeed involves the whole information of $|f(\vec{x})|$ by the definition of the flag function, and thus we can avoid the upper bound of input sizes. The flag technique can also be applied to all of our protocols and previous two-party protocols [LNO13].

4.3 Infeasibility for Hiding Two Sizes

Unfortunately, in the secure channel model, there is no general size-hiding MPC protocol that can hide two or more (input or output) sizes. The rest of this subsection is devoted to proving the infeasibility part of Theorem 1. In particular, we prove the infeasibility when two input sizes are hidden (Lemma 4), and the infeasibility when one input and the output sizes are hidden (Lemma 5).

We first prove the infeasibility when two input sizes are hidden. In this case, the infeasibility of n-party protocol can be reduced to the infeasibility of two-party protocol when both input sizes are hidden (class 2). First, assume by contradiction, there exists a protocol π that realizes an n-ary function. Then, using a reduction compiler and a wrapping compiler, we compile the protocol π into a two-party protocol π' that realizes an impossible functionality. By the contradiction, we conclude that there exists a function while hiding two input sizes. The formal statement and the proof are as follows.

Lemma 4 (Hiding two input sizes). *Let $(G, \vec{v})$ be a size-hiding class for n parties, such that two input sizes are private, and the others are public. Assuming the existence of threshold FHE, there exists a function f such that the functionality $(G, \vec{v}, f)$* cannot *be realized in the secure channel model.*

Proof. Without loss of generality, we can assume the private input sizes are $|x_1|$ and $|x_2|$. Let f' be a two-ary function such that its range is a constant size, and $(G_{2a}, \vec{v}_{2a}, f')$ cannot be realized in the secure channel model. (The existence of such a function is shown by [LNO13].) Let f be an n-ary function such that $f(x_1, \cdots, x_n) = f'(x_1, x_2)$. Assume by contradiction that there exists an n-party protocol π with $P_1, \cdots, P_n$ that realizes $(G, \vec{v}, f)$ in the secure channel model.

Let $T(\kappa, \vec{x})$ be a random variable representing the number of bits exchanged among all parties when running π with inputs $\vec{x}$ and security parameter κ. In this case, by the argument similar to [LNO13], there exists a polynomial p such that $T(\kappa, \vec{x}) < p(\kappa)$ for all large enough κ. Let us consider the simulator $\mathcal{S}$ for the protocol π corrupting $P_2, \cdots, P_n$. For a fixed output value α, let x_2^* be the smallest string for which there exists x_1 such that $f'(x_1, x_2^*) = \alpha$. At this time, there exists a polynomial p_α such that the running time of the simulator $\mathcal{S}$ is bounded by $p_\alpha(|x_2^*|, |\alpha|, \kappa)$, and there exists a polynomial p'_α such that $p'_\alpha(\kappa) = p_\alpha(|x_2^*|, |\alpha|, \kappa)$ since $|x_2^*|$ and $|\alpha|$ are constant sizes. We claim that, for every (x_1, x_2) such that $f'(x_1, x_2) = \alpha$, the length of the transcript with input (x_1, x_2) is upper bounded by $p'_\alpha(\kappa)$ except negligible probability. Otherwise, it contradicts the security of π. (For example, the simulator $\mathcal{S}$, corrupting P_3 only, cannot compute the message size since $\mathcal{S}$ does not know P_2's input is x_2^* or not.) Since the number of possible output value is constant, there exists a polynomial p such that $T(\kappa, \vec{x}) < p(\kappa)$ for every $\vec{x}$ except negligible probability. Therefore, the protocol π is I-independent for any I (especially, $I = \{1, 2\}$).

Now we are ready to derive the contradiction. We first construct a two-party π' with $P'_1 = \{P_1\}$ and $P'_2 = \{P_2, \cdots, P_n\}$ that is compiled by a reduction compiler from the protocol π. Note that the protocol π' is also I-independent ($I = \{1, 2\}$) since the communication complexity and the computation complexity are the same as π. Then, we construct a protocol π'' that is compiled by a wrapping compiler from the protocol π'. From Lemma 2, the protocol π'' realizes $(G_{2.a}, \vec{v}_{2.a}, f')$, in contradiction to the infeasibility of f'. Now we have that the functionality $(G, \vec{v}, f)$ cannot be realized in the secure channel model. □

Next, we prove the infeasibility when one input and the output sizes are hidden. In order to prove this, we introduce a new function, a *truncated oblivious multi-input pseudorandom function* tomprf defined as follows. Let F be a pseudorandom function $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$. A *truncated oblivious multi-input pseudorandom function* tomprf_n is an n-party functionality (but ignoring inputs $x_4, \cdots, x_n$) that takes as inputs a vector of arbitrary length $x_1 = (a_1, \cdots, a_m) \in (\{0,1\}^\kappa)^m$ from P_1, a κ-bit string $x_2 \in \{0,1\}^\kappa$ from P_2, and a key for the pseudorandom function $x_3 \in \{0,1\}^\kappa$ from P_3. The functionality outputs to P_1 $(F_{x_3}(a_1), \cdots, F_{x_3}(a_\ell))$, where $\ell = \min(x_2, m)$. Now we are ready to prove the following lemma.

Lemma 5 (Hiding an input and the output sizes). *Let $(G, \vec{v})$ be a size-hiding class for n parties, such that an input and the output sizes are private, and the others are public. Assume that one-way functions exist. There exists a function f such that $(G, \vec{v}, f)$* cannot *be realized in the secure channel model.*

Proof. Without loss of generality, we can assume the private input size is $|x_1|$. Essentially, there are three settings regarding who must not learn $|x_1|$ and $|f(\vec{x})|$:

1. The party P_2 must not learn both of $|x_1|$ and $|f(\vec{x})|$.
2. The party P_2 must not learn $|x_1|$, and the party P_1 must not learn $|f(\vec{x})|$.
3. The party P_2 must not learn $|x_1|$, and the party P_3 must not learn $|f(\vec{x})|$.

First, let us consider the case where P_2 must not learn both of $|x_1|$ and $|f(\vec{x})|$. Let f be an n-ary function ignoring $x_3, \cdots, x_n$ such that $f(\vec{x}) = f'(x_1, x_2)$, where the functionality $(G_{1.d}, \vec{v}_{1.d}, f')$ cannot be realized in the secure channel model. Assume by contradiction that there exists an n-party protocol π with $P_1, \cdots, P_n$ that securely computes $(G, \vec{v}, f)$ in the secure channel model. We can construct a two-party protocol π' with $P'_1 = \{P_2\}$ and $P'_2 = \{P_1, P_3, \cdots, P_n\}$ that is compiled by a reduction compiler from π. From Lemma 1, the protocol π' realizes $(G_{1.d}, \vec{v}_{1.d}, f')$, in contradiction to the infeasibility of f'. Now in this case we obtain a function f such that $(G, \vec{v}, f)$ *cannot* be realized in the secure channel model.

Second, let us consider the case where P_2 must not learn $|x_1|$, and P_1 must not learn $|f(\vec{x})|$. An oblivious transfer OT is a two-party function that takes $x_1 = (s_0, s_1)$ from P_1, where s_0 and s_1 are strings of arbitrary length, and $x_2 \in \{0,1\}$ from P_2 as inputs, and outputs a string s_{x_2} to only the party P_2. Let f be an n-ary function such that $f(x_1, \cdots, x_n) = \mathsf{OT}(x_1, x_2)$. Now we show that the function f cannot be realized in the secure channel model by the technique similar to [LNO13]. Assume by contradiction that there exists an n-party protocol π with $P_1, \cdots, P_n$ that realizes $(G, \vec{v}, f)$ in the secure channel model. We denote the inputs $\vec{x}$ by $\vec{x} = ((s_0, s_1), x_2)$ since inputs $x_3, \cdots, x_n$ are ignored. Let $T(\kappa, \vec{x})$ be a random variable representing the number of bits exchanged among $P_1, \cdots, P_n$ when running π with inputs $\vec{x}$ and security parameter κ. For inputs $\vec{x}^* = ((0,0),0)$, there exists a polynomial p such that $T(\kappa, \vec{x}^*) < p(\kappa)$ for all large enough κ since π is a polynomial-time protocol. Let s' be a random string whose length is $\omega(p(\kappa))$, and let $\vec{x}'_0 = ((0, s'), 0)$ and $\vec{x}'_1 = ((0, s'), 1)$. It must hold that $T(\kappa, \vec{x}'_0) < p(\kappa)$, otherwise P_2 can distinguish the other input of P_1 is 0 or s'. And it must hold $T(\kappa, \vec{x}'_1) < p(\kappa)$, otherwise P_1 can distinguish that P_2 obtains 0 or s'. However, in the case of $\vec{x}'_1 = ((0, s'), 1)$, the party P_2 must compute s', the random string of length $\omega(p(\kappa))$, from a transcript of length less than $p(\kappa)$. This contradicts to the incompressibility of a random string. Thus, in this case, there is a function f such that $(G, \vec{v}, f)$ *cannot* be realized in the secure channel model.

Finally, let us consider the case where P_2 must not learn $|x_1|$, and P_3 must not learn $|f(\vec{x})|$. Let f be a truncated oblivious multi-input pseudorandom function tomprf_n with a pseudorandom function $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$. Assume by contradiction that there exists an n-party protocol π with $P_1, \cdots, P_n$ that realizes $(G, \vec{v}, f)$ in the secure channel model. Let $T(\kappa, \vec{x})$ be a random variable representing the number of bits exchanged among $P_1, \cdots, P_n$ when running π with inputs $\vec{x}$ and security parameter κ. There exists a polynomial p such that $T(\kappa, (\emptyset, 0, x_3)) < p(\kappa)$ for all large enough κ since π is a polynomial-time protocol. For any x_1^* of the cardinality $\omega(p(\kappa))$, it must hold $T(\kappa, (x_1^*, 0, x_3)) < p(\kappa)$ for all large enough κ, otherwise P_2 can distinguish that P_1 has $\emptyset$ or x_1^*, although P_2 must not learn $|x_1|$. (Note that since $\mathsf{tomprf}_n(x_1^*, 0, x_3) = \emptyset$, the party P_2, who may learn the output, must not learn any partial information of the size of P_1.) It must also hold $T(\kappa, (x_1^*, 2^\kappa - 1, x_3)) < p(\kappa)$ for all large enough κ, otherwise P_3 can distinguish that the output size is 0 or $\omega(p(\kappa))$. Now we construct an algorithm $\mathcal{D}$ that distinguishes between outputs of the pseudorandom function $F_{x_3}(a_1), \cdots, F_{x_3}(a_m) \in \{0,1\}^\kappa$, and truly random values $r_1, \cdots, r_m \in \{0,1\}^\kappa$, using a simulator $\mathcal{S}$ for a randomness producer $\mathcal{R}(1^\kappa) = 0$. The distinguisher $\mathcal{D}$ invokes $\mathcal{S}$ with inputs $(1^\kappa, x_i, \vec{z}, 0)$ where x_i is the input of P_i, $\vec{z}$ is either $(F_{x_3}(a_1), \cdots, F_{x_3}(a_m))$ or $(r_1, \cdots, r_m)$ (here, we omit a set of indices I and the input sizes). If $\vec{z}$ is the pseudorandom values, the simulator $\mathcal{S}$ outputs a transcript of length less than $p(\kappa)$, otherwise $\mathcal{S}$ cannot output consistent transcript due to the incompressibility of a random string. The distinguisher $\mathcal{D}$ should output 1 if $\mathcal{S}$ outputs consistent transcript. $\mathcal{D}$ distinguishes pseudorandom values and random values[7], in contradiction to the pseudorandomness of F. Thus, in this case, assuming the existence of one-way functions, there is a function f such that $(G, \vec{v}, f)$ *cannot* be realized in the secure channel model. $\square$

Theorem 1 is proven by Lemmas 3, 4 and 5.

5 Results in the Strong Secure Channel Model

In previous section, we show that, in the secure channel model, a general size-hiding protocol cannot hide two or more (input or output) size information. In order to circumvent the infeasibility, we introduce a new communication model, a *strong secure channel model* such that an adversary cannot learn the number of

[7] The above strategy works even for any randomness producer whose output size is bounded. However, in the HBC model, the proof does not work since a simulator can generate a transcript with a *long random tape*.

bits exchanged among honest parties. We show that, in the strong secure channel model, a general size-hiding protocol exists even hiding all sizes of inputs and output from some parties, while the secure channel model only allows the size of at most one input to be hidden. Furthermore, we also prove that some functions still remain infeasible even in the strong secure channel model. More specifically, we give a sufficient and necessary condition under which a general size-hiding protocol can exist. Because the condition depends on whether the output size is public or private, our result is stated in Theorem 2 (when the output size is public) and Theorem 3 (when the output size is private).

In Section 5.1, we introduce the strong secure channel model. In Section 5.2, we give our main results, Theorem 2 and Theorem 3, and some examples of (feasible or infeasible) classes. We show the feasibility part of Theorem 2 in Section 5.3, and the infeasibility part of the theorem in Section 5.4. We show the feasibility part of Theorem 3 in Section 5.5, and the infeasibility part of the theorem in Section 5.6.

5.1 Strong Secure Channel Model

One of the standard communication model is the secure channel model, which is an abstraction of secure communication. In the secure channel model, an adversary cannot learn messages exchanged among honest parties, but can learn the number of bits of them. The model is very powerful and used in various works, however, in the context of size-hiding computations, there are strong infeasibility results. In order to circumvent the infeasibility, we introduce a new communication model, a strong secure channel model such that an adversary can learn neither messages nor the number of bits exchanged among honest parties, At first glance, the existence of such a communication channel seems to be suspicious, but we emphasize that the strong secure channel model can be instantiated by using steganographic techniques.

We provide a security definition of the strong secure channel model. The only difference from the secure channel model is that a simulator does not have to create message sizes in the strong secure channel model. Thus, the security in the secure channel model implies the security in the strong secure channel model. The security of protocols in the model is formally defined as follows.

Definition 4 (Security in the strong secure channel model) *Let $(G, \vec{v}, f)$ be a functionality for n parties and let π be a protocol that correctly computes $(G, \vec{v}, f)$. We say that π realizes $(G, \vec{v}, f)$ in the* strong secure channel model *if for every randomness producer $\mathcal{R}$, there exists a PPT $\mathcal{S}$ such that for every $I \subsetneq \{1, \cdots, n\}$, every polynomials $q_1, \cdots, q_n$,*

$$\left\{\mathcal{S}(1^\kappa, I, \vec{x}_I, \mathrm{OUTPUT}_I^{(G,\vec{v},f)}(\vec{x}), \vec{r}_I \leftarrow \mathcal{R}(1^\kappa, I))\right\}_{\kappa,\vec{x}} \overset{\mathrm{c}}{\equiv} \left\{\mathrm{view}_I^\pi(\vec{x})_{\vec{r}_I}\right\}_{\kappa,\vec{x}}$$

where $x_1 \in \{0,1\}^{q_1(\kappa)}, \cdots, x_n \in \{0,1\}^{q_n(\kappa)}$.

5.2 Our Main Results

In the strong secure channel model, it is possible to realize any functionality while hiding two or more sizes. The condition under which any functionality can exist is different depending on the case where the output size is public and the case where the output size is private.

The main theorem for the case where the output size is public is as follows.

Theorem 2 (Public output size) *Let $(G, \vec{v})$ be a size-hiding class with n parties, where the output size is public. Assume that threshold FHE exists. A class of size-hiding $(G, \vec{v})$ is feasible in the strong secure channel model if and only if for every two distinct vertices $i, j \in V(G)$, there exists a vertex k such that $(i, k) \in E(G)$ and $(j, k) \in E(G)$.*

Examples. Suppose parties $P_1, \cdots, P_5$ wish to compute a function while hiding their input sizes, but each party thinks it is permitted to leak its own size information to the neighboring parties; see the right most graph in Figure 6. In this case, the parties can securely compute every function, since every two distinct parties have a party who may learn both input sizes of them. (For example, a pair of parties P_1 and P_4 has the party P_5 who may learn input sizes of P_1 and P_4.) Thus, in such a pentagon case, a general size-hiding protocol exists in the strong secure channel model. Similarly, the triangle and the square cases also have a general size-hiding protocol. On the other hand, there is no general size-hiding protocol in the hexagon case; see the right most graph in Figure 7. This is due to the fact that the pair of parties P_1 and P_4 do not have a party who may learn both input sizes of them. Other feasible and infeasible classes are shown in Figure 6 and 7.

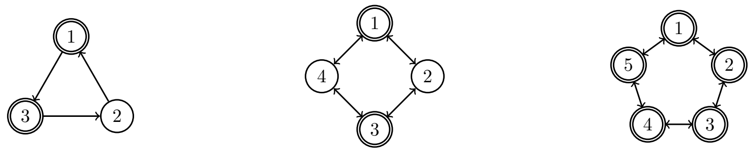

Fig. 6. Feasible classes with public output size

The main theorem for that case where the output size is private is as follows.

Theorem 3 (Private output size) *Let $(G, \vec{v})$ be a size-hiding class with n parties, where the output size is private. Assume that threshold FHE exists. A class of size-hiding $(G, \vec{v})$ is feasible in the strong secure channel model if and only if any vertex $i \in V(G)$ such that $\vec{v}[i] = \perp$ satisfies the both conditions:*

1. *For all vertices $j \in V(G)$, there exists an edge $(j, i) \in E(G)$.*
2. *There exists an edge $(i, j) \in E(G)$ such that $\vec{v}[j] \neq \perp$.*

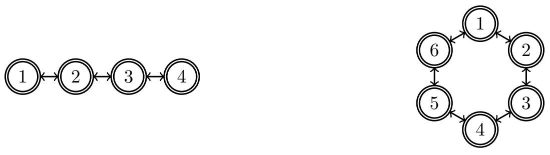

Fig. 7. Ineasible classes with public output size

Examples. See the center graph in Figure 8. Suppose two clients (P_3 and P_4) wish to compute a function while hiding their input sizes, with the help of servers (P_1 and P_2, they also have input data). Furthermore, suppose clients want to hide the output size from servers. In this case, if servers may learn all input sizes of clients and each server has a client who may learn the server's input size, every function can be realized while meeting the demand. Every feasible class with private output size is interpreted as such a client-server situation. On the other hand, there is no general size-hiding protocol in classes of Figure 9.

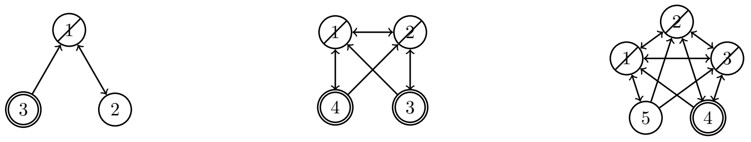

Fig. 8. Feasible classes with private output size

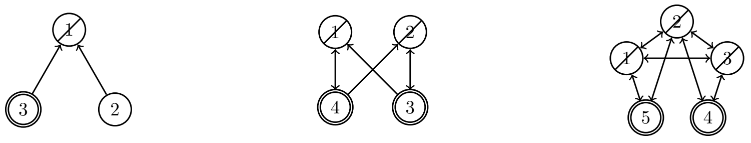

Fig. 9. Infeasible classes with private output size

5.3 Protocol with Public Output Size

We show that, in the strong secure channel model, the feasibility of size-hiding computations is dramatically improved compared to the secure channel model. In this subsection, we construct a size-hiding protocol where all parties may learn the output size. (The case of private output size is described in Section 5.5.) In particular, we construct a general size-hiding protocol if every pair of parties has a party who may learn input sizes of them. The condition includes the case where each input size is hidden from some parties, i.e., the number of hidden sizes is the number of parties. The protocol idea is explained in Introduction; see Section 1.2. The full description of the protocol appears in Protocol 2.

Building blocks – circuits for homomorphic evaluation. Let f be a function $f : (\{0,1\}^*)^n \to \{0,1\}^*$ and let ℓ be an integer. We can construct the following circuit, denoted by $C^{\mathsf{Sac}}_{f,\ell}$. On receiving a string $x'_1, x'_2, \cdots, x'_n \in \{0,1\}^*$ as inputs, if there is an all-zero string in inputs, then it outputs 0^ℓ, otherwise, parse as $x'_i = 00\cdots0||1||x_i$ $(i = 1, \cdots, n)$, and outputs $f(x_1, \cdots, x_n)$ padded with zeroes up to ℓ.

Protocol 2 *Suppose that parties $P_1, P_2, \cdots, P_n$ have inputs $x_1, x_2, \cdots, x_n$, respectively, and each party is either a full-output party or a size-inly party. The protocol proceeds as follows.*

1. *All parties invoke a* ThrGen *protocol with inputs 1^κ, and each party P_i obtains a public key pk and a share of the secret key sk_i.*
2. *For all edges $(i,j) \in E(G)$, P_i sends the size information $|x_i|$ to P_j.*
3. *For all two vertices $i, j \in V(G)$, the party P_i sends ciphertexts $c^{\mathsf{in}}_{(i,j)}$ to P_j as follows. Let P_k be a party who may learn both of $|x_i|$ and $|x_j|$. The party P_i computes $c_i = \mathsf{Enc}_{pk}(1||x_i)$, and sends to P_k. (If P_i and P_k are the same party, then P_i computes only.) If it holds $|x_j| \geq |x_i|$, then P_k computes a ciphertext $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|-|x_i|})||c_i$, and sends it to P_j. Otherwise, P_k computes a ciphertext of zeroes $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|+1})$, and sends it to P_j.*
4. *Each party P_i constructs the circuit $C^{|f|}_{(\log\kappa)^2}$ (described as above), where $|f| = |f(x_1, \cdots, x_n)|$, computes $c^{\mathsf{size}}_i \leftarrow \mathsf{Eval}_{pk}(C^{|f|}_{(\log\kappa)^2}, c^{\mathsf{in}}_{(1,i)}, \cdots, c^{\mathsf{in}}_{(n,i)})$, and sends c^{size}_i to the party P_1 (or other designated party).*
5. *The party P_1 computes c^{size} by homomorphic evaluation of a max function from $c^{\mathsf{size}}_1, \cdots, c^{\mathsf{size}}_n$, and sends it to all parties. Then, all parties invoke a* ThrDec *protocol with the ciphertext c^{size}, and obtain the decrypted value ℓ.*
6. *Each party P_i computes $c^{\mathsf{out}}_i \leftarrow \mathsf{Eval}_{pk}(C^f_\ell, c^{\mathsf{in}}_{(1,i)}, \cdots, c^{\mathsf{in}}_{(n,i)})$, where C^f_ℓ is a circuit described as above for the function f and the integer ℓ, and then sends c^{out}_i to the party P_1.*
7. *The party P_1 computes c^{out} by homomorphic evaluation of a max function from $c^{\mathsf{out}}_1, \cdots, c^{\mathsf{out}}_n$, and sends it to all parties. Then, all parties invoke a* $\mathsf{ThrDec}_{I_{\mathsf{f}}}$ *protocol with the ciphertext c^{out}, and all full-output parties obtain the decrypted value z. All full-output parties output z, and the other parties output nothing. The protocol terminates.*

Lemma 6 (Security of Protocol 2). *Let $(G, \vec{v})$ be a size-hiding class with n parties, which holds the conditions stated in Theorem 2. Let $f(x_1, \cdots, x_n)$ be any n-ary polynomial-time computable function. Assuming the existence of threshold FHE, Protocol 2 realizes $(G, \vec{v}, f)$ in the strong secure channel model.*

Proof. Given 1^κ, I, inputs $\vec{x}_I$, input sizes $\{1^{|x_j|} | (j,i) \in E(G)\}_{i \in I}$, the output $f(\vec{x})$ if $I \cap I_{\mathsf{f}} \neq \emptyset$, and random tapes $\vec{r}_I$ produced by a randomness producer, the simulator $\mathcal{S}$ works as follows. First, $\mathcal{S}$ computes $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, chooses $sk_i \overset{\mathrm{U}}{\leftarrow} \{0,1\}^{|sk|}$ for every $i \in I$, and simulates ThrGen protocol with keys $(pk, sk_i)_{i \in I}$. Next, for every $i \in \{1, \cdots, n\}$ and every $j \in I$, $\mathcal{S}$ computes $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|-|x_i|}||1||x_i)$ if it holds $i \in I$ and $|x_j| \geq |x_i|$, otherwise, $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|+1})$. Then, $\mathcal{S}$ computes c^{size}_i and c^{out}_i for every $i \in I$, and evaluates c^{size} and c^{out}. $\mathcal{S}$ simulates threshold decryption protocols for c^{size} and c^{out}. Finally, $\mathcal{S}$ outputs views of corrupted parties generated as above. The views generated by $\mathcal{S}$ are indistinguishable from the views in a real execution of the protocol due to the IND-CPA security of threshold FHE and the security of threshold key generation and decryption protocols. □

5.4 Infeasibility Result with Public Output Size

We show that, when all parties may learn the output size, the condition "every pair has a party who may learn input sizes of the pair" is a necessary and sufficient condition under which a general size-hiding protocol can exist. In this subsection, in order to prove this, we show that the condition is not satisfied, a general size-hiding protocol does not exist.

Lemma 7. *Let $(G, \vec{v})$ be a size-hiding class with n parties, where the output size is public. The size-hiding class $(G, \vec{v})$ is infeasible in the strong secure channel model if there exists two distinct vertices $i^*, j^* \in V(G)$ such that there is no vertex $k \in V(G)$ such that $(i^*, k) \in E(G)$ and $(j^*, k) \in E(G)$.*

Proof. Let $(G, \vec{v})$ be a size-hiding class that satisfies the conditions as above. Without loss of generality, we can assume $i^* = 1$ and $j^* = 2$. Let $P'_1 = \{P_i | (2, i) \notin E(G)\}$ and let $P'_2 = \{P_1, \cdots, P_n\} \setminus P'_1$. The parties P'_1 must not learn $|x_2|$ from the definition, and the parties P'_2 must not learn $|x_1|$, otherwise, it contradicts to the condition. (Note that $P_1 \in P'_1$ and $P_2 \in P'_2$.) Let f' be a two-ary function such that $(G_{2.a}, \vec{v}_{2.a}, f')$ cannot be realized in the (strong) secure channel model[8], and let f be a function such that $f(x_1, x_2, \cdots, x_n) = f'(x_1, x_2)$. Assume by contradiction that there exists an n-party protocol π that realizes $(G, \vec{v}, f)$ in the strong secure channel model. We can construct a two-party protocol π' with P'_1 and P'_2, that is compiled by a reduction compiler from the protocol π. From Lemma 1, the protocol π' realizes the functionality $(G_{2.a}, \vec{v}_{2.a}, f')$, in contradiction to the assumption. Therefore, the size-hiding class $(G, \vec{v})$ is infeasible in the strong secure channel model. □

Theorem 2 is proven by Lemmas 6 and 7.

5.5 Protocol with Private Output Size

In this subsection, we construct a size-hiding protocol where some parties must not learn the output size; see Protocol 3. (The case of public output size is described in Section 5.3.) Note that it is not superior to Protocol 2 since these

[8] Note that, in the two-party setting, the strong secure channel model and the secure channel model are essentially the same.

size-hiding conditions are different. Interestingly, the underlying idea of the protocol is completely different from Protocol 2; see Section 1.2.

Building block – GMW protocol. Goldreich et al. [GMW87] constructed a general MPC protocol that is secure in the presence of HBRC adversaries[9] corrupting up to $n-1$ of n parties, and showed that it can be compiled to a protocol that is secure in the presence of malicious adversaries. Our protocol uses the HBRC protocol in order to compute the desired function by all servers. For simplicity, we use *GMW protocol* to denote a protocol that realizes the following functionality.

- Input: Each party is given a secret share of x_j for all $j = 1, \cdots, n$.
- Output: All parties output $f(x_1, \cdots, x_n)$.

Protocol 3 *Suppose that parties $P_1, P_2, \cdots, P_n$ have inputs $x_1, x_2, \cdots, x_n$, respectively, and there are some forbidden parties. The function f to be computed has a polynomial p s.t. $|f(x'_1, \cdots, x'_n)| < p(|x'_1|, \cdots, |x'_n|)$ for all $x'_i \in \{0,1\}^*$. Let p' be a polynomial such that $p'(x) = p(x, x, \cdots, x)$. For a ciphertext $c = (c_1, \cdots, c_\ell)$ (each c_i is a ciphertext of 1-bit message), $[c^{\mathsf{out}}]_{\ell'}$ ($\ell' \le \ell$) denotes $c' = (c_1, \cdots, c_{\ell'})$. Without loss of generality, we can assume $1 \in I_\perp$ and $2 \in I_{\mathsf{p}}$. The protocol proceeds as follows.*

1. *All permitted parties invoke a ThrGen protocol with inputs 1^κ, and then each permitted party P_i obtains the public key pk and a share of the secret key sk_i. Then, P_2 sends the public key pk to all forbidden parties.*
2. *Each party P_i computes shares of additive secret sharing $\{r_{i,j}\}_{j \in I_\perp}$, where $\{r_{i,j} \in \{0,1\}^{|x_i|}$, whose secret is x_i, i.e., $r_{i_1,j} \oplus r_{i_2,j} \oplus \cdots \oplus r_{i_t,j} = x_i$ ($\{i_1, \cdots, i_t\} = I_\perp$). Then, P_i computes $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(r_{i,j})$, and sends $c^{\mathsf{in}}_{(i,j)}$ to all forbidden parties.*
3. *The forbidden parties homomorphically evaluate a GMW protocol computing $f(\vec{x})$ of length $L = p'(\max(|x_1|, \cdots, |x_n|))$ using FHE, i.e., all messages in the execution are encrypted by FHE and all computations are done by homomorphic evaluation. As the output of the protocol, they obtain a ciphertext c^{out}.*
4. *The party P_1 constructs a circuit C_{size} that takes $x \in \{0,1\}^L$ as an input, and outputs $|x| \in \{0,1\}^{(\log \kappa)^2}$. Then, P_1 computes $c^{\mathsf{size}} \leftarrow \mathsf{Eval}_{pk}(C_{\mathsf{size}}, c^{\mathsf{out}})$, and sends c^{size} to all permitted parties.*
5. *Let $\sigma_i = \max\{|x_j| \,|\, (j,i) \in E(G)\}$, and let $L_i = p'(\sigma_i)$. (It must hold $L_i \le L$ for all i by the definition.) For every $i \in I_{\mathsf{p}}$, the party P_1 homomorphically evaluates a max function for $[c^{\mathsf{out}}]_{L_i}$ and $\mathsf{Enc}_{pk}(0^{L_i})$, and obtains a ciphertext c^{out}_i. Then, P_1 sends c^{out}_i to P_i for every $i \in I_{\mathsf{p}}$.*
6. *All permitted parties invoke a ThrDec protocol with inputs 1^κ and c^{size}, and obtain the decrypted value ℓ.*
7. *Each permitted party P_i sends $[c^{\mathsf{out}}_i]_\ell$ to the party P_2. (If the length of the ciphertext is less than ℓ, then the party uses a padding with zero ciphertexts up to ℓ.) The party P_2 computes c^{out} by homomorphic evaluation of a max function from $[c^{\mathsf{out}}_1]_\ell, \cdots, [c^{\mathsf{out}}_n]_\ell$, and sends it to all permitted parties.*

[9] It is well known that the protocol is secure against HBC adversaries. However, it is also secure against HBRC adversaries since the simulation algorithm does not have to choose random tapes by itself.

8. *All permitted parties invoke a $\mathsf{ThrDec}_{I_\mathsf{f}}$ protocol with inputs 1^κ and c^{out}, and all full-output parties obtain the decrypted value $z \in \{0,1\}^\ell$. All full-output parties output z, and the other parties output nothing. The protocol terminates.*

Lemma 8 (Security of Protocol 3). *Let $(G, \vec{v})$ be a size-hiding class with n parties, which satisfies the conditions stated in Theorem 3. Let $f(x_1, \cdots, x_n)$ be any n-ary polynomial-time computable function. Assuming the existence of threshold FHE, Protocol 3 realizes $(G, \vec{v}, f)$ in the strong secure channel model.*

Proof. Given 1^κ, I, inputs $\vec{x}_I$, input sizes $\{1^{|x_j|} \big| (j,i) \in E(G)\}_{i \in I}$, the output $f(\vec{x})$ if $I \cap I_\mathsf{f} \neq \emptyset$, and random tapes $\vec{r}_I$ produced by a randomness producer, the simulator $\mathcal{S}$ works as follows. First, $\mathcal{S}$ computes $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, chooses $sk_i \overset{\mathrm{U}}{\leftarrow} \{0,1\}^{|sk|}$ for every $i \in I$, and simulates ThrGen protocol with keys $(pk, sk_i)_{i \in I}$. Next, for every $i \in \{1, \cdots, n\}$ and every $j \in I$, $\mathcal{S}$ computes $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|-|x_i|}||1||x_i)$ if it holds $i \in I$ and $|x_j| \geq |x_i|$, otherwise, $c^{\mathsf{in}}_{(i,j)} = \mathsf{Enc}_{pk}(0^{|x_j|+1})$. Then, $\mathcal{S}$ computes c^{size}_i and c^{out}_i for every $i \in I$, and evaluates c^{size} and c^{out}. $\mathcal{S}$ simulates threshold decryption protocols for c^{size} and c^{out}. Finally, $\mathcal{S}$ outputs views of corrupted parties generated as above. The views generated by $\mathcal{S}$ are indistinguishable from the views in a real execution of the protocol due to the IND-CPA security of threshold FHE and the security of threshold key generation and decryption protocols. □

Proof. Let I_1 and I_2 be a partition of $I = I_1 \cup I_2$ such that $I_1 \subset I_\perp$ and $I_2 \subset I_\mathsf{p}$. We consider the following cases: (1) $I_1 \subsetneq I_\perp$ and $I_2 = I_\mathsf{p}$, (2) $I_1 = \emptyset$ and $I_2 = I_\mathsf{p}$, (3) $I_1 = I_\perp$ and $I_2 \subsetneq I_\mathsf{p}$, (4) $I_1 \subsetneq I_\perp$ and $I_2 \subsetneq I_\mathsf{p}$, (5) $I_1 = \emptyset$ and $I_2 \subsetneq I_\mathsf{p}$, (6) $I_1 = I_\perp$ and $I_2 = \emptyset$, and (7) $I_1 \subsetneq I_\perp$ and $I_2 = \emptyset$. We show the simulator in the cases of (1) and (3). It is easy to adapt the proof to the other cases.

We construct the simulator $\mathcal{S}$ in the case of (1), where all clients and some servers are corrupted. Given 1^κ, I, inputs $\vec{x}_I$, all input sizes $\{1^{|x_1|}, \cdots, 1^{|x_n|}\}$, the output $f(\vec{x})$, and random tapes $\vec{r}_I$ produced by a randomness producer, the simulator $\mathcal{S}$ works as follows. First, $\mathcal{S}$ invokes a threshold key generation protocol, and obtains a public key pk and all shares of the secret key. Second, $\mathcal{S}$ computes secret shares of x_i for $i \in I$ and $0^{|x_i|}$ for $i \notin I$, and encrypts them by threshold FHE. Next, $\mathcal{S}$ computes $c^{\mathsf{out}} = \mathsf{Enc}_{pk}(f(\vec{x}))$ padded with zeroes up to appropriate length and simulates an encrypted GMW protocol on the input ciphertexts and c^{out}. Then, $\mathcal{S}$ invokes threshold decryption protocols. Finally, $\mathcal{S}$ outputs views of corrupted parties generated as above. The view generated by $\mathcal{S}$ and the view in a real execution are indistinguishable due to the security of the GMW protocol.

Next we construct the simulator $\mathcal{S}$ in the case of (3), where all servers and some clients are corrupted. Given 1^κ, $I = I_1 \cup I_2$, inputs $\vec{x}_I$, all input sizes $\{1^{|x_1|}, \cdots, 1^{|x_n|}\}$, the output $f(\vec{x})$, and random tapes $\vec{r}_I$ produced by a randomness producer, the simulator $\mathcal{S}$ works as follows. First, $\mathcal{S}$ computes $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, chooses $sk_i \overset{\mathrm{U}}{\leftarrow} \{0,1\}^{|sk|}$ for all $i \in I_2$, and simulates a threshold key generation protocol with the keys. Second, $\mathcal{S}$ computes secret shares of x_i for $i \in I$ and $0^{|x_i|}$ for $i \notin I$, and encrypts them by threshold FHE. Next, $\mathcal{S}$ homomorphically executes GMW protocol, and obtains the output ciphertext c^{out}. Then, $\mathcal{S}$ computes c^{size} honestly, and simulates threshold decryption protocols. Finally, $\mathcal{S}$ outputs views of corrupted parties generated as above. The view generated by $\mathcal{S}$ and the view in a real execution are indistinguishable due to the IND-CPA security of FHE and the security of threshold protocols. □

5.6 Infeasibility Result with Private Output Size

We show that, when some parties must not learn the output size, the condition stated in Theorem 3 is a necessary and sufficient condition under which a general size-hiding protocol can exist. In this subsection, in order to prove this, we show that if the condition is not satisfied, a general size-hiding protocol does not exist.

Lemma 9. *Let $(G, \vec{v})$ be a size-hiding class with n parties, where the output size is private. The size-hiding class $(G, \vec{v})$ is infeasible in the strong secure channel model if there exists a vertex $i^* \in V(G)$ such that $\vec{v}[i^*] = \bot$, which satisfies one of the following conditions:*

1. *There exists a vertex $j^* \in V(G)$ such that $(j^*, i^*) \notin E(G)$.*
2. *There is no edge $(i^*, j) \in E(G)$ such that $\vec{v}[j] \neq \bot$.*

Proof. Let $(G, \vec{v})$ be a size-hiding class that satisfies the former condition. Without loss of generality, we can assume $i^* = 1$ and $j^* = 2$, i.e., $\vec{v}[1] = \bot$ and $(2, 1) \notin E(G)$. Let f' be a two-ary function such that the functionality $(G_{1.d}, \vec{v}_{1.d}, f')$ cannot be realized in the (strong) secure channel model, and let f be an n-ary function such that $f(x_1, x_2, \cdots, x_n) = f'(x_1, x_2)$. Assume by contradiction that there exists n-party protocol π that realizes $(G, \vec{v}, f)$ in the strong secure channel model. Now we construct a two-party protocol π' with $P'_1 = \{P_1\}$ and $P'_2 = \{P_2, \cdots, P_n\}$ that is compiled by a reduction compiler from the protocol π. Since P'_1 must not know both the output size and the input size $|x_2|$, the protocol π' realizes $(G_{1.d}, \vec{v}_{1.d}, f')$ in the (strong) secure channel model, in contradiction to the infeasibility of f'. Therefore, the size-hiding class $(G, \vec{v})$ is infeasible in the strong secure channel model.

Let $(G, \vec{v})$ be a size-hiding class that satisfies the latter condition. Let P'_1 be a subset of parties $P'_1 = \{P_i | \vec{v}[i] = \bot\}$, and let $P'_2 = \{P_1, \cdots, P_n\} \setminus P'_1$. Without loss of generality, we can assume $i^* = 1$ and $P_2 \in P'_2$. Let f' be a two-ary function such that $(G_{1.b}, \vec{v}_{1.b}, f')$ cannot be realized in the (strong) secure channel model, and let f be an n-ary function such that $f(x_1, x_2, \cdots, x_n) = f'(x_1, x_2)$. Assume by contradiction that there exists n-party protocol π that realizes $(G, \vec{v}, f)$ in the strong secure channel model. Now we construct a two-party protocol π' with P'_1 and P'_2, that is compiled by a reduction compiler from the protocol π. Since P'_1 must not learn the output size, and P'_2 must not learn the input size $|x_1|$, the protocol π' realizes $(G_{1.b}, \vec{v}_{1.b}, f')$ in the (strong) secure channel model, in contradiction to the infeasibility of f'. Therefore, the size-hiding class $(G, \vec{v})$ is infeasible in the strong secure channel model. □

Theorem 3 is proven by Lemmas 8 and 9.

Acknowledgment The authors would like to thank members of the study group "Shin-Akarui-Angou-Benkyou-Kai" for the valuable discussions and helpful comments, and thank the anonymous reviewers for their comments. This work was supported in part by JSPS KAKENHI Grant Number 26330151 and JSPS A3 Foresight Program.

References

ACT11. Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (if) size matters: Size-hiding private set intersection. In *Public Key Cryptography*, pages 156–173, 2011.

AJL+12. Gilad Asharov, Abhishek Jain, Adriana Lopez-Alt, Eran Trómer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *EUROCRYPT*, pages 483–501, 2012.

Cachin04. Christian Cachin. An information-theoretic model for steganography. Inf. Comput., volume 192, number 1, pages 41–56, 2004.

COV15. Melissa Chase, Rafail Ostrovsky, and Ivan Visconti. Executable Proofs, Input-Size Hiding Secure Computation and a New Ideal World. In *EUROCRYPT*, pages 532–560, 2015.

CV12. Melissa Chase, and Ivan Visconti. Secure Database Commitments and Universal Arguments of Quasi Knowledge. In *CRYPTO*, pages 236–254, 2012.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229, 1987.

Gol04. Odded Goldreich. Foundation of Cryptography Volume 2 Basic Applications. Cambridge University Press, 2004.

HAL09. Nicholas J. Hopper, Luis von Ahn, and John Langford. Provably Secure Steganography. In IEEE Trans. Computers, volume 58, number 5, pages 662–676, 2009.

HW15. Pavel Hubácek and Daniel Wichs. On the Communication Complexity of Secure Function Evaluation with Long Output. In ITCS, pages 163–172, 2015.

IP07. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *TCC*, pages 575–594, 2007.

LNO13. Yehuda Lindell, Kobbi Nissim, and Claudio Orlandi. Hiding the Input-Size in Secure Two-Party Computation. In *ASIACRYPT*, pages 421–440, 2013. A full version is available at IACR ePrint Archive.

MRK03. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *FOCS*, pages 80–91, 2003.

Appendix

Honest-But-Randomness-Controlling Model

In this section, we show that the honest-but-randomness-controlling (HBRC) model is truly stronger than the honest-but-curious (HBC) model. We also explain the relation between the HBRC model and the honest-but-deterministic

(HBD) model proposed by [HW15][10]. To clarify the difference among them, we describe them in a standard setting (not size-hiding settings).

The view of the party P_i during an execution of π with inputs $\vec{x}$ is defined as $\mathrm{view}_i^\pi(\vec{x}) = (x_i, r_i, m_{i_1}, \cdots, m_{i_t})$, where r_i is its internal coin tosses and m_{i_j} is the j-th message that was received by P_i in the protocol execution. We also use $\mathrm{view}_i^\pi(\vec{x})|_{r_i} = (x_i, m_{i_1}, \cdots, m_{i_t})$ to denote $\mathrm{view}_i^\pi(\vec{x})$ on given randomness r_i. Here, if the length of r_i is shorter than the length of its internal randomness, its internal randomness is $r_i||0^k$ for appropriate $k \in \mathbb{N}$.

Definition 5 (HBC) *Let f be a polynomial-time computable n-ary function. We say that π securely computes f in the HBC model if there exists a PPT $\mathcal{S}$ such that for every $I \subsetneq \{1, \cdots, n\}$, every polynomials $q_1, q_2, \cdots, q_n$, $\left\{\mathcal{S}(1^\kappa, I, \vec{x}_I, \vec{y}_I)\right\}_{\kappa,\vec{x}} \overset{c}{\equiv} \left\{\mathrm{view}_I^\pi(\vec{x})\right\}_{\kappa,\vec{x}}$, where $x_1 \in \{0,1\}^{q_1(\kappa)}, \cdots, x_n \in \{0,1\}^{q_n(\kappa)}$.*

Definition 6 (HBRC Model) *Let f be a polynomial-time computable n-ary function. We say that a PPT $\mathcal{R}$ is a* randomness producer *if $\mathcal{R}(1^\kappa, I)$ outputs a vector of strings $\vec{r}_I = (r_{i_1}, \cdots, r_{i_t}) \in (\{0,1\}^*)^{|I|}$ for all $I = \{i_1, \cdots, i_t\} \subsetneq \{1, \cdots, n\}$. We say that π securely computes f in the HBRC model if for every randomness producer $\mathcal{R}$ there exists a PPT $\mathcal{S}$ such that for every $I \subsetneq \{1, \cdots, n\}$, every polynomials $q_1, \cdots, q_n$, $\left\{\mathcal{S}(1^\kappa, I, \vec{x}_I, \vec{y}_I, \vec{r}_I \leftarrow \mathcal{R}(1^\kappa, I))\right\}_{\kappa,\vec{x}} \overset{c}{\equiv} \left\{\mathrm{view}_I^\pi(\vec{x})|_{\vec{r}_I}\right\}_{\kappa,\vec{x}}$, where $x_1 \in \{0,1\}^{q_1(\kappa)}, \cdots, x_n \in \{0,1\}^{q_n(\kappa)}$.*

Definition 7 (HBD Model) *Let f be a polynomial-time computable n-ary function. We say that π securely computes f in the HBD model if there exists a PPT $\mathcal{S}$ such that for every $I \subsetneq \{1, \cdots, n\}$, every polynomials $q_1, q_2, \cdots, q_n$, $\left\{\mathcal{S}(1^\kappa, I, \vec{x}_I, \vec{y}_I)\right\}_{\kappa,\vec{x}} \overset{c}{\equiv} \left\{\mathrm{view}_I^\pi(\vec{x})|_0\right\}_{\kappa,\vec{x}}$, where $x_1 \in \{0,1\}^{q_1(\kappa)}, \cdots, x_n \in \{0,1\}^{q_n(\kappa)}$.*

It is trivial that the security in the HBRC model implies the security in the HBD model. Moreover, the HBRC model implies the HBC model by the following Theorem.

Theorem 4 *Let f be a polynomial-time computable n-ary function. If a protocol π securely computes f in the HBRC model then it also securely computes f in the HBC model.*

Proof. For simplicity, we consider the case of $n = 2$ and that one party is corrupted. We note that the general case $n > 2$ can be proven in the same way.

Assume that a protocol π is secure in the HBRC model. We construct a PPT $\mathcal{S}'$ that produces $\mathrm{view}^\pi(\vec{x})$ given an input $(1^\kappa, x, y)$. $\mathcal{S}'$ computes $T = p(\kappa, |x|, |y|)$ and generates an uniformly random string $r \in \{0,1\}^T$. Then, $\mathcal{S}'$ invokes $\mathcal{S}$ on inputs $(1^\kappa, x, y, r)$ and outputs the same output as $\mathcal{S}$. Therefore, if a protocol is secure in the HBRC model then it is also secure in the HBC model. □

[10] In original definition in [HW15], the model captures precomputation settings. Our formalization does not include a precomputation for simplicity.