R3PO: Reach-Restricted Reactive Program Obfuscation and Its Applications

Kaartik Bhushan[1](✉), Sai Lakshmi Bhavana Obbattu[2], Manoj Prabhakaran[1], and Rajeev Raghunath[1]

[1] IIT Bombay, Mumbai, Mumbai, India
{kbhushan,mp,mrrajeev}@cse.iitb.ac.in
[2] IIT (BHU) Varanasi, Varanasi, India

Abstract. In recent breakthrough results, novel use of grabled circuits yielded constructions for several primitives like Identity-Based Encryption (IBE) and 2-round secure multi-party computation, based on standard assumptions in public-key cryptography. While the techniques in these different results have many common elements, these works did not offer a modular abstraction that could be used across them.

Our main contribution is to introduce a novel notion of obfuscation, called Reach-Restricted Reactive-Program Obfuscation (R3PO) that captures the essence of these constructions, and exposes additional capabilities. We provide a powerful composition theorem whose proof fully encapsulates the use of garbled circuits in these works.

As an illustration of the potential of R3PO, and as an important contribution of independent interest, we present a variant of Multi-Authority Attribute-Based Encryption (MA-ABE) that can be based on (single-authority) CP-ABE in a blackbox manner, using only standard cryptographic assumptions (e.g., DDH) in addition. This is in stark contrast to the existing constructions for MA-ABE, which rely on the random oracle model and supports only limited policy classes.

1 Introduction

Consider the following approach to Identity-Based Encryption (IBE):

- The master key pair is a verification/signing key pair for a signature scheme.
- The decryption key for an identity is simply a signature on the identity.
- The ciphertext is an *obfuscation* of the following program: it checks if its input is a valid signature on a target identity, and if so, it outputs the message.

With the right notion of obfuscation, as we shall see, *this construction indeed translates to a secure IBE scheme!* Further, such an obfuscation can be instantiated using standard cryptographic assumptions like DDH, based on the tools in [19,20].

The motivation of this work comes from the breakthrough results of [6,16,20,27]. These results were surprising not only because of the end results, but

Q. Tang and V. Teague (Eds.): PKC 2024, LNCS 14603, pp. 61–91, 2024.
https://doi.org/10.1007/978-3-031-57725-3_3

also because the central tools involved – garbled circuits, oblivious transfer, smooth projective hash functions, etc. – were all well known for a long time. The power behind these results lay in a *machinery* that carefully meshed these tools together.

However, this line of works has lacked reusable high-level abstractions, even as the low-level techniques were clearly similar across multiple works. Even the few abstractions of this machinery that appeared subsequently, e.g., in the form of hash-garbling [23], were not comprehensive enough to capture the multifarious applications of the machinery itself.

The main contribution of this work is to develop a versatile abstraction of the common machinery underlying the above works, and take it beyond the current set of applications. Our abstraction involves a strong form of obfuscation, which can be realized for *programs that are appropriately sampled.* The obfuscation formulation gives an intuitive description of potential solutions, and facilitates realizing it via a *novel composition theorem.* This not only aids in understanding the current constructions better, but also shows the way to new applications. As an illustration, *and as an important contribution of independent interest*, we present a variant of Multi-Authority Attribute-Based Encryption (MA-ABE) that can be based on (single-authority) ABE in a blackbox manner, using only standard cryptographic assumptions (e.g., DDH) in addition. This is in stark contrast to the constructions available for the original formulation of MA-ABE, which rely on specific assumptions, are for special restricted policy classes and/or are in the random oracle model [15,17,40,46].

1.1 Our Contributions

Our contributions are in two parts – (1) developing a powerful new framework to capture several important results from the recent literature, and (2) using it to construct a multi-authority version of ABE.

The R3PO Framework. Our primary contribution is to develop the notion of *Reach-Restricted Reactive Program Obfuscation* (R3PO) that modularly encapsulates and extends the powerful techniques behind the surprising results of [6,16,20,24,27]. Our definition of R3PO allows an intuitive description of prior constructions like IBE [20] (with an easy extension to Identity-Based Functional Encryption [54]), 2-Round MPC [6,27], and RBE [24], all of them using obfuscation of natural reactive programs.

- We present a **library** of useful R3PO schemes. The library includes obfuscations for non-reactive programs that check a commitment opening, verify a signature, and verify the (partial) opening of a hashed value.
- We present a **composition theorem** that can be used to obtain an R3PO scheme for a reactive program from R3PO schemes for smaller (non-reactive) programs (like those in the library above) into which it *decomposes.*
- For each of the applications we consider (as well as some of the programs in the library), we define an appropriate reactive program, construct an R3PO for it and use it to complete our construction. The requisite R3PO maybe

directly available from the library, or is constructed using the composition theorem.

The grabled circuit technique is entirely encapsulated within the proof of the composition theorem above. This is in contrast with prior work that used these techniques, where the proof would use a sequence of indistinguishability arguments interleaving garbled circuit simulation with other arguments specific to the construction. Indeed, one of the main technical challenges we overcome is to allow disentangling the garbled circuits from the other cryptographic elements, in these security proofs, using a strong simulation based definition of R3PO and our novel notion of *decomposition*.

Private Multi-Authority ABE. Another important contribution of this work is a new version of *Multi-Authority Attribute-Based Encryption* (called Private Multi-Authority ABE or p-MA-ABE), and a construction for it, conceived in terms of an R3PO.

The motivation for p-MA-ABE stems from the natural use-case for MA-ABE (or even ABE) where a user has privacy requirements against an attribute authority (e.g., they may want to obtain attributes corresponding to a city and a state that they consider their primary home, but without revealing the name of those locations to the authority). Correspondingly, the authority would be willing to issue attributes that satisfy a (possibly private) *attribute-granting policy* (e.g., issue the attributes for any one state and any one city within that state). The privacy requirement is that the authority (or authorities) shall not learn *anything* about the attributes of a user, and the user shall not learn anything about the attribute-granting policy, beyond whether the policy is met by the attribute set.

Now, a non-private MA-ABE (or ABE) scheme can be easily converted into a private version, via secure 2-party computation of a function to which the user's input is their attribute request, and the authority's input is its master secret key and its attribute-granting policy. Since such a 2PC protocol can be implemented in two rounds (e.g., a simple protocol based on Yao's Garbled Circuit works, as we consider the authorities to be honest-but-curious), this only requires the user to send a single message to the server – which we call an *attribute request* – before the server responds.

p-MA-ABE captures this trade-off: allow the user to initiate the contact with the authority,[1] and in return obtain a strong privacy guarantee. Though the above transformation shows that standard MA-ABE can be easily turned into p-MA-ABE, the former is known to be realizable only for very limited functions and in the random oracle model. In contrast, our results show that p-MA-ABE is as widely realizable as ABE itself!

We give a construction for p-MA-ABE from any (single-authority, ciphertext-policy) ABE scheme in a blackbox manner, using R3PO for (non-reactive) programs for signature checking and commitment opening, that is provably secure

[1] We remark that in a practical situation, this extra round comes at virtually no cost, since anyway a user would first need to establish a secure channel and authenticate itself with the authority before receiving its credentials.

in the standard model. The scheme supports general access policies as supported by the underlying ABE scheme, and is policy-hiding if the ABE is policy-hiding.

1.2 Related Work

We mention a few related works below, and discuss how R3PO relate to other notions in Sect. 2.7.

Obfuscation. A large variety of notions of obfuscation have been studied in the literature leading to several important breakthroughs along the way (e.g., [2,4,5,7,22,31,35–37,41,49,51]). Like R3PO, many of these require security only when the program being obfuscated is generated appropriately [4,7,33,51].

Composition. Composition has been considered in the context of cryptographic protocols, leading up to UC security and its variants [3,11–13,18,21,32,45,47,52], as well as alternate approaches like Constructive Cryptography [44]. Composition for obfuscation has received far less attention, although it was explicitly considered in an early work [43].

Garbled Circuits. Garbled circuits were conceived by Yao [53]. The techniques of chaining multiple garbled circuits appeared in garbled RAM schemes [25,26,28,42], and later several results like Laconic OT [16], IBE from DDH [19,20], 2-round MPC [6,27], and several extensions of these works have all relied on these techniques.

Multi-Authority Attribute Based Encryption (MA-ABE). The notion of Ciphertext Policy-Attribute Based Encryption (CP-ABE) was introduced in [48] and formally defined in [30]. There is a rich sequence of works realizing ABE, based on lattice based (LWE) [9,29] and pairing based assumptions [30,34,39]. But for MA-ABE, first proposed in [15], realizations so far have been limited. In the standard GID model, [40] formalized the notion of decentralized MA-ABE (where in, no trusted setup algorithm other than a common reference string is allowed), and gave a scheme for it under appropriate bilinear maps assumptions in the random oracle model (supporting general policy structures). A sequence of works culminated in [17], where they gave a scheme under the Learning With Errors (LWE) assumption in the random oracle model for policies corresponding to DNF formulae. Concurrently, [46] modified the definition to consider sender security (policy hiding) as well as receiver security (attribute hiding), and gave a construction for it under the k-linear assumption in the random oracle model for a special subset of policies. More recently, [50] gave a (current state-of-the-art) construction for MA-ABE in the plain model for subset policies (including DNF formulae) from the new evasive LWE assumption. Their construction however requires a global setup.

[38] proposed a variant of MA-ABE called the OT model. It is a relaxed model where there is no global identity fixed for the users. However, as pointed out in [17], this allows multiple users to pool their attributes, defeating one of the main goals of ABE. Our model has a global identity that an authority would incorporate into the key issued for a party, and as captured in the security definition, the user can combine only attributes that are issued for the same

global id. Another drawback of [38] was that it used a global setup; we do not. Our setup is local to each authority (as in the global id model). Our model much more closely resembles the standard global id model, but with an additional key request step in the syntax. On the other hand, our results are much stronger than those available in the standard model (which are in the random oracle model and/or for limited function classes). We also offer further flexibility by not requiring each attribute to be attached to a unique authority.

We also note that MA-ABE can be modeled as an appropriate functionality in the framework of public-key Multi-Party Functional Encryption (MPFE) [1]. Their work gives a construction for public-key MPFE for general functionalities. However, this does not yield the result in our work due to the following limitations. Their construction uses an interactive setup, forcing the authorities to be aware of and interact with each other, while we require the MA-ABE authorities to only use "local" setup. Further, their construction is based on Multi-Input Functional Encryption for general functionalities (which is a strong assumption that implies iO). In contrast, we rely only on ABE and standard assumptions. Indeed, the main motivation behind R3PO and the entire line of work leading to it, is to be able to base various cryptographic schemes on simpler assumptions, and to avoid the need for assumptions like iO.

2 Technical Overview

2.1 Motivating Examples

We start with a few motivating constructions, along the lines of the IBE construction mentioned at the beginning of this paper, which we seek to base on our new notion of obfuscation. In the general case, we would be obfuscating a *reactive* program (or more specifically, a *Moore machine*), which at each step, accepts an input, updates its state, and produces an output based on the new state.

Identity-Based Functional Encryption. IBFE is an extension of IBE where each identity id is associated with a unique function f_{id} (not known to the encryptor), so that when an encryption of a message m addressed to id is decrypted using the key for f_{id}, one receives $f_{\mathsf{id}}(m)$. An IBFE scheme can be obtained by simply modifying the IBE scheme above so that the obfuscated program takes a signature on (id, f) (where id is already fixed in the program, but f is not), and transitions to a state encoding f, where it outputs $f(m)$.

IBFE has been explored in a prior work [54], but their definition is incomparable to our notion above. On the one hand, their definition does not allow the adversary to obtain any function keys – under any IDs – for a function f such that $f(m_0)$ and $f(m_1)$ are not equal; on the other hand, it is not made very clear if the adversary is restricted to obtaining a single function key for the challenge id, as is the case in our definition. Finally, they offer a construction for the primitive only for a very restricted class of functions, while our construction supports general functionalities.

2-Round MPC. Following the constructions in [6,27], an underlying (multi-round) MPC protocol can be reinterpreted as evaluating a blinded circuit, in which each boolean gate is owned by a party, and the protocol amounts to evaluating the wires of this circuit publicly. The wire values are public, but each gate is private to its owner.

The 2-round MPC constructed from the blinded circuit is as follows. In the first round, each party broadcasts a commitment to the 4 bits (separately) of the truth table of each of its gates. In the second round, each party broadcasts the obfuscation of the following reactive program:

- The program maintains a public state consisting of all the wire values of the circuit, evaluated thus far.
- If the next gate is owned by another party, the program accepts as input the output wire value of the gate, along with an opening of the corresponding commitment in the gate. If the opening verifies, it updates its state to correspond to having evaluated this wire. It produces no output for this transition.
- If the next gate is owned by this party, then it takes no input, transitions to a state that includes the output wire value of this gate, and outputs the opening of the corresponding commitment.

Finally, given these obfuscated reactive programs, the parties evaluate the blinded circuit gate by gate, at each step first running the program from the owner of the gate, and then feeding its inputs to all the other programs.

Laconic OT. This is a version of OT in which the receiver has a vector D of choice-bits, which it commits to by sending a short string y to the sender. Later, on input (i, x_0, x_1), the sender should send a string to the receiver from which the latter should learn only x_{D_i}.

We consider the following implementation of Laconic OT: Using a hash that supports "selective opening" of a bit in the hashed string, with a collision resistance guarantee that prevents opening any bit in two different ways, the receiver hashes D to obtain y. On input (i, x_0, x_1), the sender obfuscates the following (small) program and sends it over to the receiver: The program accepts as input an opening of y at position i to a bit b, and if the opening is valid, then it outputs x_b.

Each of the above simplistic constructions relied on an intuitive notion of "obfuscation." In the sequel, we develop a formal notion of obfuscation which will let us make the above descriptions precise, while retaining their simplicity. Importantly, our new obfuscation notion is indeed realizable in all the above cases, using the same standard cryptographic assumptions as in the prior works which introduced these constructions.

2.2 Defining R3PO

At a high-level, we consider obfuscation of *reactive programs.* A reactive program (a finite-state machine, or more precisely, a Moore machine) takes inputs over multiple rounds, updating its state and producing an output based on the state at each round. It is specified by a start state, a transition function π and a message function μ, so that, on reaching a state σ, the program outputs $\mu(\sigma)$.

Before discussing the definitions, it will be useful to have a couple of running examples in mind. In these examples, μ is arbitrary (and secret), and a public π is as specified below.

- **Commitment.** π_c incorporates a commitment string c. On input d at the start state, if d decommits c to m, then π_c transitions to a state σ_m encoding m.
- **Signature.** A signature verification key vk is encoded in the start state σ_{vk} of π (denoted as $\pi[\sigma_{\mathsf{vk}}]$), from where, given a valid signature on a message m as input, it transitions to a state σ_m encoding m.

These are both instances of "one-step programs" which have transitions only out of the start state. (We shall later explain the slightly different choices for how the values c and vk are incorporated into π in the two cases.) In these examples, π is not hidden, and the goal of obfuscating such a program would be to hide μ. More generally, π and μ can both have secrets in them (when defining reactive programs formally, we will denote them as $\pi^{(\alpha)}$ and $\mu^{(\beta)}$, where α and β are the secrets).

Reach Extraction and Simulation. Our simulation-based notion of obfuscation requires that a "reach-extractor" should exist for the program being obfuscated. A reach extractor would predict all the states of a reactive program that are reachable using inputs that can be efficiently computed by any adversary. Then, the obfuscation of the program should be simulated using only the outputs produced by the program at those states. We elaborate on reach-exaction and the rest of the simulation below.

Reach Extractability. Which states in a program π are efficiently reachable is a consequence of *the process that generates the program* (analogous to how an "evasive program" being evasive is a consequence of sampling it from a distribution). This process involves a *generator* G and an *adversary* Q. A *reach extractor* for an adversary Q is a program that passively (possibly in a non-blackbox manner) observes Q as it interacts with G, and then predicts (a superset of) the set of states that the adversary will be able to reach in the program output by G. This prediction is made explicitly in the form of inputs to a (possibly different) reactive program $\sqcap$ that will reach all the states reachable by the adversary, and perhaps more. Here we allow the extractor to specify $\sqcap$, which belongs to a transition function family $\mathring{\mathcal{P}}$ that may be different from the transition program family $\mathcal{P}$ that is obfuscated. We refer to this as the "reach bounding" guarantee of the reach extractor.

We illustrate a reach extractor for the two running examples.

- **Commitment:** G accepts a commitment string c from Q, and then outputs π_c. A reach extractor can extract a value m from the commitment, either when Q is semi-honest, or when a setup is used that the extractor can control. Now, m is not a decommitment as expected by π_c. Instead, we allow the extractor to specify a different program $\sqcap_m$ which accepts m itself as the input and transitions to σ_m.

This extractor is reach bounding, because, due to the binding property of the commitment scheme, the only state Q could reach in π_c is also σ_m.
- **Signature:** In this case, G internally samples a pair $(\mathsf{sk}, \mathsf{vk})$ of signing and verification keys. It sends vk to Q, and further may answer signature requests by Q. An extractor can collect all the signatures Q receives from G and output them as a reach-bounding set of inputs for $\Pi = \pi[\sigma_{\mathsf{vk}}]$. Note that here the program family to be used by the extractor $\mathring{\mathcal{P}}$ is the same as the one being obfuscated $\mathcal{P}$ (and it has only one program in it but with various start states). The reach bounding property follows from unforgeability of the signature scheme.

Simulation. An obfuscator for a generator R3PO security definition requires that a 2-stage simulation exists for any adversary Q, as follows:

- Stage 1: After Q finishes interacting with G, a reach extractor observing Q specifies a set of reachable states (in the form of a program Π and inputs to it).
- Stage 2: Given the output of the original message function μ on those states, a simulated obfuscation is produced. This should be indistinguishable from the obfuscation of the reactive program produced by G, even given auxiliary information output by both G and Q.

Note that this is a *stronger notion of simulation than even VBB obfuscation*, which only requires the simulation of one predicate at a time, rather than a simulation of the entire obfuscated program. Indeed, requiring such a simulator would typically entail that the program is learnable and hence trivial to obfuscate. What keeps our definition from becoming trivial is the fact that *the extracted inputs are a function of the program generation process*, and are not available to the obfuscator.

Reach Restriction. The final component in our definition of R3PO is in the form of an additional requirement on the reach extractor in Stage 1 above. This requirement stems from the "one-time" nature of Yao's Garbled Circuits, a key ingredient in the constructions that we wish to capture. Intuitively, these constructions require that an adversary can evaluate any garbled circuit on only one set of inputs. We incorporate a corresponding *reach restriction* requirement into our definition of reach extractability of a reactive program (Definition 3), which leads to the name *reach-restricted reactive programs* (R3P).[2]

To define reach restriction, we require the state space of the reactive programs to be *a priori* partitioned into a polynomial number of parts, $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_N$. Then, informally, the reach restriction property of a reactive program is that no efficient adversary would be able to find inputs that take π to *two different states that belong to the same part*. Formally, the reach restriction property is imposed on the reachable states produced by the reach-extractor.

We return to our running examples.

[2] Formally, we do not define R3P, but only an R3P Generator, as a program generator (Definition 2) that has a reach extractor.

- **Commitment:** We let Σ_1 consist only of the start state and Σ_2 consist of all states of the form σ_m. Since the extractor outputs only one message m, the reach restriction property already holds.
- **Signature:** We let Σ_1 consist of all the potential start states σ_{vk} and Σ_2 consist of all states of the form σ_m (the two kind of states are encoded so that $\Sigma_1 \cap \Sigma_2 = \emptyset$). To be reach restricting, we will require that the generator G gives out at most one signature. Further, we would want to enforce that breaking reach restriction in Π must correspond to forging signatures with respect to the key sampled by G. This is enforced by keeping vk in the start state of $\pi[\mathsf{vk}]$ (rather than in the transition function itself), which in turn forces Π to use the same start state and hence the same verification key.

2.3 R3PO Composition Theorem

As noted earlier, a major motivation of this work is to encapsulate a range of powerful techniques using garbled circuits in a reusable form. This result takes the form of a composition theorem, which allows obfuscating a reactive program via an obfuscation of its various components.

The high-level idea is to view a reactive program $\pi \in \mathcal{P}$, over a state space $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_N$ as consisting of separate programs $\widehat{\pi}_1, \ldots, \widehat{\pi}_N$, such that $\widehat{\pi}_i$ is identical to π on states $\sigma \in \Sigma_i$, and in other states it ignores all inputs (i.e., remains at the same state). Let $\mathcal{P}_i$ denote the class of such programs $\widehat{\pi}_i$. W.l.o.g. (due to reach-restriction), we require π to not have any transitions between states in the same part, and hence each $\widehat{\pi}_i$ is a "one-step" (or non-reactive) program that halts after its first transition out of the start state. However, attempting to formalize this leads to a couple of conundrums.

Conundrum 1: Dynamically Determined Programs. As a naïve starting point, one could try building an obfuscator for $\mathcal{P}$ from obfuscators for $\mathcal{P}_i$. However, this runs into an immediate problem: When executing a program in $\mathcal{P}$, the state reached in Σ_i is dynamically determined by the inputs used, whereas when obfuscating a program in $\mathcal{P}_i$, its start state needs to be fixed. The resolution of this conundrum, which goes back to [16,19,20,25,26,28,42], is to provide a **garbled circuit** that can dynamically compute the obfuscation of $\widehat{\pi}_i[\sigma_i]$ with the correct start state σ_i; the input to this garbled circuit would be the labels encoding σ_i, which in turn would be released by the obfuscation of a previous program $\pi_j[\sigma_j]$ on an input x such that $\pi_j(\sigma_j, x) = \sigma_i$.

However, the price we pay for using garbled circuits is that only one set of labels can be made available to the adversary for each garbled circuit, in turn resulting in the reach-restriction requirement.

Conundrum 2: Intertwined Generators. Recall that to formalize reach restriction, our definition needed to take into account the generators. Now, when we try to map the different parts of a single reactive program as being generated by multiple generators, *the generators can become deeply intertwined*, sharing secret keys and state variables. Further, the program generated by one generator needs to have a start state that is determined by the outputs produced by programs in other parts. So it may not always be possible to view a (reach-restricted) reactive program produced by a generator as the composition of single-step reactive programs produced by separate generators.

The resolution to this conundrum is to require some additional relation between the generator for the reactive program and the generators for the one-step programs. This leads us to the notion of *decomposition.*

Decomposition. Unlike in the case of MPC protocols, wherein the subprotocols are explicitly executed by a composite protocols, a reactive program generator need not have "sub-generators" running within it. Indeed, this presents a challenge to composition that is *fundamentally different from composition in MPC.*

Our novel solution is to define decomposition in terms of a *bisimulation* requirement. Roughly, for G to decompose into a smaller generator H (and additional computation), we require that G *can be viewed* as H via a simulator, and *vice versa.* More precisely, we require that there be two simulators J and Z such that $\boxed{\begin{array}{c}\boxed{G}\\ J\end{array}}$ (denoting that J internally runs G as a black box) and $\boxed{\begin{array}{c}Z\\ \boxed{H}\end{array}}$ are indistinguishable from each other from the point of view of any adversary Q (or more precisely, for $\boxed{\begin{array}{c}Q\\ W\end{array}}$, where the wrapper W is also part of the simulation). This by itself can be trivially arranged by letting $J = H$ and $Z = G$. We need to further capture the requirement that the program $\widehat{\pi}$ produced by H corresponds to a single step in the program π produced by G. More precisely, the state space of the generator H corresponds to a *part* Σ_i of the state space of G, and we require that the start state of $\widehat{\pi}$ is the same as the only state in Σ_i that is reachable in π.

Now, by requiring this, we require J to know the reachable state in π produced by G. While this is possible in some cases (e.g., when the reachable state is determined by a signed message sent by G), in certain other cases it is not possible (e.g., when it is determined by a message hidden in a commitment). To accommodate these different situations, we allow J to obtain this information from the wrapper W, which is in turn allowed to obtain this from a reach-extractor for G (or more precisely, from a "partial" reach extractor which only extracts the reach within Σ_i).

Finally, for use in our composition theorem, we shall require a uniformly sampled message function to be associated with the reactive program produced by J. (While the definition of decomposition allows arbitrary message function class here, the composition theorem is for decomposition that uses a particular message function class.)

We refer the reader to Sect. 4.1 for a more detailed discussion and a precise definition of decomposition.

Composition. Having defined decomposition, we turn to stating and proving the composition theorem. Informally, it states that if a generator G decomposes into generators $(H_1, \ldots, H_N)$ (for a partition of its state space $(\Sigma_1, \ldots, \Sigma_N)$), and if each H_i has an R3PO scheme $\mathcal{O}_i$, then there is one for G as well. The construction uses garbled circuits, following the outline at the beginning of this section. The final obfuscation consists of one garbled circuit GC_i for each part

Σ_i, such that on reaching $\sigma \in \Sigma_i$, an evaluator would have the labels that encode σ as input for GC_i, and GC_i would then output $\mu(\sigma)$ as well as an obfuscation $\mathcal{O}_i(\widehat{\pi}_i[\sigma], \widehat{\mu}_i)$ (using a hard-coded random tape). Feeding an input x to this obfuscated program will release the labels for the state $\widehat{\pi}_i(\sigma, x) = \pi(\sigma, x)$.

To prove that this construction yields an R3PO for G, we use a sequence of hybrids that would replace one garbled circuit at a time with a simulated one, which in turn outputs not the actual obfuscation $\mathcal{O}_i(\widehat{\pi}_i[\sigma], \widehat{\mu}_i)$, but a simulated one. At each step, we will be able to apply the decomposition guarantee (using an inductively maintained partial reach extractor) to go from G to $\boxed{\begin{array}{c}\boxed{G}\\ J_i\end{array}}$ to $\boxed{\begin{array}{c}Z_i\\ \boxed{H_i}\end{array}}$, wherein we use the R3PO guarantee to replace the actual obfuscation used to simulate GC_i with a simulated one (while also extending the partial extractor); then we move back from $\boxed{\begin{array}{c}Z_i\\ \boxed{H_i}\end{array}}$ to $\boxed{\begin{array}{c}\boxed{G}\\ J_i\end{array}}$ and then G.

2.4 R3PO Library

We present R3PO schemes for a few basic program classes which can be combined together in a variety of constructions.

- *Commitment-Opening.* This is similar to the running example presented above. In the full version, we realize the R3PO for a couple of flavors of this (UC secure commitment, and "weakly secure" commitment that is suitable for semi-honest committers), based on the standard assumption of 2-round OT.
- *Signature-Checking.* We provide an R3PO for signature-checking programs as in the running example. To facilitate full security in applications like IBE and IBFE, we support *puncturable* signature schemes.[3] We instantiate a puncturable signature scheme and give an R3PO scheme for this program family assuming an OTSE scheme in the full version.
- *Hash-Opening.* This is similar to the commitment opening reactive program, but with a compressing hash instead of a binding commitment. The R3PO for this program class can be constructed from Laconic OT [16]. Alternately, we can use our composition theorem to bootstrap from an R3PO for the same class instantiated with a factor-2 compressing laconic OT (see Sect. 2.5 below).
- *ϵ-Transition* While specifying reactive programs using the above building blocks, often it is useful to transition from state reached via one building block to a state that is suitable as the start state of another building block. ϵ-transitions provide the essential syntactic sugar to enable this. R3PO for an ϵ-transition is implemented using a garbled circuit.

[3] In our constructions of IBE and IBFE, the ciphertext corresponds to an obfuscated program. For full security, the adversary must be allowed to make key queries even after receiving the ciphertext. But, no interaction is allowed between the program generator and the adversary after the program has been generated. Hence, we consider a generator which gives out an appropriately punctured signing key before the interaction finishes.

2.5 Applications: The Different Ways of Using R3PO

Our R3PO library and our composition theorem form a versatile toolkit for instantiating new and old constructions. There are a few different ways in which they can be put to use.

Off-the-Shelf Without Composition. In certain cases, the components in our library are already powerful enough off-the-shelf to yield a construction for a desired application. An illustrative example is that an R3PO for (puncturable) signature-checking can be used to construct an IBFE scheme (and, as a special case, IBE), as sketched in Sect. 2.1 and elaborated in the full version. The security proof is fairly direct, by using a generator for the R3PO that models the security experiment of IBFE.

Using Composition. We illustrate a typical "workflow" for using the R3PO composition theorem in a higher-level application. We use the example of Laconic OT [16], which is one of the early constructions that form the inspiration for this work. For the sake of readability we use slightly imprecise terminology.

- We start by identifying a reactive program family, such that an R3PO for it directly yields our application.In the case of laconic OT, this reactive program traverses a pre-determined path along a Merkle tree, with states holding the hash value at each node, and making a transition if the input "explains" the hash at that node. The Merkle tree uses an underlying hash scheme which compresses by a factor of 2.
- We consider the one-step restrictions of this reactive program as another reactive program family, and carry out the following two steps:
  - We show that the original reactive programs can be decomposed into its one-step restrictions. This involves matching the definition of decomposition with straightforward constructions.
  - We give an R3PO scheme for these one-step restrictions. This can be directly based on the construction in [16] for factor-2-compression laconic OT (not involving garbled circuit chaining).
- Then we *simply invoke our composition theorem* to obtain an R3PO for the original program family.
- We package the original reactive program as another one-step program, so that it can be included in our R3PO library for various applications (see the full version). Laconic OT is a direct consequence of an R3PO for this one-step program.

Another example of this workflow is in the construction of the R3PO for signature-checking that was mentioned above as part of our library (where the smaller non-reactive programs used correspond to one-time signature checking).

R3PO as a Component. In the above examples, once an R3PO is constructed, the final application is fairly immediately realized. However, it is also possible to use R3PO as a component in a larger construction, wherein the step from R3PO to the final security proof may be non-trivial. The proof may involve multiple hybrids, with R3PO security used to replace a real obfuscation in one hybrid with the simulated obfuscation in the next. One such example is the 2-round MPC protocol of [6,27], which we rederive in the full version using R3PO for commitment opening. In this construction, as sketched in Sect. 2.1, several programs obfuscated using R3PO are involved. The security of R3PO can be used to move to an "ideal" execution of the MPC protocol (or more precisely, build a simulator for the 2-round MPC using the simulators of R3PO and the simulator for the underlying MPC protocol).

Our main application of p-MA-ABE (discussed below) also falls into this category, where the security of the final construction depends on several components, one of which is an R3PO scheme. This construction also illustrates the possibility of combining multiple library components (commitment-opening and signature-checking) in the same reactive program.

2.6 Private Multi-Authority ABE

In this section, we give a brief overview of the new variant of MA-ABE that we introduce, called *Private Multi-Authority ABE* (p-MA-ABE), and the main ideas behind our construction for it. Our construction is intuitive in terms of an obfuscation of a reactive program, and can indeed be realized using R3PO. The flexibility of the new framework allows a relatively easy construction, using existing ABE schemes, and with a robust security definition. The full description can be found in Sect. 5.

Defining p-MA-ABE: The setting of p-MA-ABE (as well as MA-ABE) involves a set of mutually distrusting authorities (say $A_1, \ldots, A_\mathsf{N}$), a sender and a receiver. The algorithms in an p-MA-ABE scheme are as follows:

- Setup: At the start of the execution, each authority A_i does a local (decentralized) setup to generate its public and secret keys $(\mathsf{mpk}_i, \mathsf{msk}_i)$ and shares the public key mpk_i with the other users in the system.
- Key-Request: a receiver can construct a set of key-requests $\mathsf{req} = (\mathsf{req}_1, \ldots, \mathsf{req}_\mathsf{N})$ for a global identifier gid and attribute set $\bar{\mathsf{x}}$ from the public keys. It can then submit a key-request query (of the form $(\mathsf{gid}, \mathsf{req}_i)$) to an authority A_i and get back a key-component $\mathsf{sk}_{\mathsf{gid},\mathsf{req}_i}$ from A_i (req will hide $\bar{\mathsf{x}}$).
- Key-Gen: an authority A_i receives as input a key-request req_i for a global identifier gid, and outputs a key-component $\mathsf{sk}_{\mathsf{gid},\mathsf{req}_i}$ that incorporates an attribute-granting policy Θ_i^{gid} (which, for simplicity, we do not consider a secret).
- Encryption: a sender can encrypt a message m with a ciphertext policy ϕ, using the public keys of the authorities to produce a ciphertext $\mathsf{ct}_{m,\phi}$.
- Decryption: a receiver can decrypt a ciphertext $\mathsf{ct}_{m,\phi}$ using key components of the form $(\mathsf{sk}_{\mathsf{gid},\mathsf{req}_1}, \cdots, \mathsf{sk}_{\mathsf{gid},\mathsf{req}_\mathsf{N}})$ where all the requests req_i were generated using gid and $\bar{\mathsf{x}}$ such that $\Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$ for all i, and $\phi(\bar{\mathsf{x}}) = 1$.

Compared to the original definition of MA-ABE, there are two main differences in p-MA-ABE: Firstly, since the attributes are to be kept private even from the authorities, there is a key-request step, wherein the user generates the key-request messages to all the authorities based on its desired set of attributes. Secondly, we allow each authority to use an arbitrary attribute-granting policy, which depends on the entire attribute vector.[4]

We define security w.r.t. a corruption model where the adversary is allowed to maliciously corrupt the receivers and semi-honestly corrupt any subset of authorities. If a receiver is honest, we require that the key-request req reveals nothing about $\bar{\mathsf{x}}$ to the authorities, even if all of them collude. When the receiver is corrupt, we guarantee that, for any choice of (ϕ, m_0, m_1), the adversary cannot distinguish between the encryptions of m_0 and m_1 w.r.t. a policy ϕ, unless for a pair $(\mathsf{gid}, \bar{\mathsf{x}})$ such that $\Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$ for *all* honest authorities A_i, $\phi(\bar{\mathsf{x}}) = 1$ and the adversary sent a valid key request for $(\mathsf{gid}, \bar{\mathsf{x}})$ (i.e., a request that can be produced by the Key-Request algorithm on those inputs) to at least one honest authority (and it could have sent it to the others as well).

A p-MA-ABE Scheme: Our scheme is easily described in terms of obfuscating a reactive program. The key-request req_i is a commitment to $\bar{\mathsf{x}}$ (using a common random string in the public-key of A_i). The key issued by each authority is the obfuscation of a reactive program; the reactive programs by the different authorities "talk" to each other and confirm that they all agree on granting the same attribute $\bar{\mathsf{x}}$ to gid, and if so, issue standard CP-ABE keys for $\bar{\mathsf{x}}$. More precisely, the reactive program $(\pi^{(\alpha)}, \mu^{(\beta)})$ works as follows (with A_i's CP-ABE master secret-key constituting the secret β; α can be empty, or alternately, can be used to store Θ_i^{gid} privately):

– at the start state accepts a decommitment for req_i and transitions to a state with $\bar{\mathsf{x}}$. There, if $\Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$, then it outputs a signature on $(\mathsf{gid}, \bar{\mathsf{x}})$, using A_i's signature key.
– Then, it moves through $\mathsf{N} - 1$ states accepting signatures on $(\mathsf{gid}, \bar{\mathsf{x}})$ from all the other servers.
– On reaching the last of these states, it outputs a CP-ABE key for the attribute $\bar{\mathsf{x}}$, under a (standard) CP-ABE scheme for which A_i is the authority.

If $\Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$ for all i, then the receiver can obtain the CP-ABE keys for $\bar{\mathsf{x}}$ under all the authorities. Now, to encrypt a message under a policy ϕ, one simply secret-shares m into N shares, and encrypts each share under the CP-ABE public key of the corresponding authority.

Note that if even one (honest) authority's key component is missing, no honest authority's CP-ABE key can be obtained. This is crucial because the CP-ABE keys do not involve gid and cannot prevent the use of keys obtained using multiple gids.

Using the composition theorem, we show that there is an R3PO for a suitably defined generator that models the p-MA-ABE security game. (As it turns out, we need to do this for two different generators to handle two different hybrids;

[4] In particular, it captures the standard formulation of MA-ABE, where each authority A_i "owns" a set of attributes and its key-issuing decision is based on the values of the attributes it owns.

the first hybrid does not rely on the unforgeability of the signatures, and lets the adversary specify all the signing keys. We show that the same obfuscator $\mathcal{O}$ is an R3PO scheme for both the generators.)

2.7 Comparison of R3PO with Existing Primitives

It is instructive to compare R3PO with various existing primitives and techniques.

Hash Garbling. This abstraction from [23] gives a similar interface to R3PO *for a specific class of program generators*, namely, "Hash Opening." More precisely, hash garbling involves a hash-opening check as well as a circuit evaluation, which corresponds to a reactive program that carries out a hash-opening transition followed by an epsilon transition (that evaluates the circuit).

{Batch, Hash, Chameleon, OneTime Signature}-Encryption. These flavors of encryption that were introduced in prior work [8,19,20] correspond to R3PO schemes for one-step programs that are included in our library (Hash Opening and Signature Checking). While these original definitions differ in their details, R3PO provides a simulation-based definition that can be uniformly used in all their applications.

Witness Encryption over Commitments (cWE). cWE, recently defined in [10], is quite similar to an R3PO for "Commitment Opening" followed by an epsilon transition. It was instantiated from Oblivious Transfer (OT) and garbled circuits, just as the R3PO scheme obtained directly from our library and the composition theorem is.

Garbled Circuit Chaining. The *technique* of garbled circuit chaining has appeared in a long line of works [6,20,24,27,42]. We note that R3PO allows different one-step programs (for example combining commitment and signature in p-MA-ABE), while all prior works used garbled circuit chaining with links that correspond to a single cryptographic element. Also, as already mentioned, the prior works do not separate out the chaining from the cryptographic elements that are chained together.

Obfuscation Notions. Our notion of R3PO is different in many ways from the other notions of obfuscation. Many notions of obfuscation are either unrealizable in general or inhabit "obfustopia," requiring a combination of relatively strong assumptions, and are not practical in terms of efficiency [2,4,5,7,7,14,35]. But there are a few exceptions for specialized applications, like obfuscation of reencryption based on bilinear pairings [33] or compute-and-compare obfuscation based on LWE [51]. R3PO could be considered to be in the latter group, but with a much richer class of applications compared to the others.

The original notions of obfuscation require worst-case security, but there are several others, including obfuscation of evasive circuit families [4], strong iO [7], reencryption obfuscation [33], compute-and-compare obfuscation [51], etc. which require only *distributional security*, when the program being obfuscated is sampled from distributions with particular properties. Again, R3PO falls into the latter class here, with the sampling process being interactive.

3 The R3PO Framework

3.1 Reactive Programs and Generators

Below we define a reactive program as a stateful machine that takes inputs, transitions its state and produces outputs as a function of its state. Formally, such a program consists of a deterministic transition function π and a deterministic message function μ, both of which can be parameterized by (secret) values α, β (hardwired into circuits $\pi^{(\cdot)}$, $\mu^{(\cdot)}$ respectively).

Definition 1 (Reactive Program over ($\mathcal{X}$, Σ, $\mathcal{A}$, $\mathcal{B}$, M). A *reactive program* $(\pi^{(\alpha)}, \mu^{(\beta)})$, with input alphabet $\mathcal{X}$, a state-space Σ, a start-state $\mathsf{start} \in \Sigma$ and secret spaces $\mathcal{A}$, $\mathcal{B}$ is specified by a deterministic *transition* program $\pi^{(\alpha)} : \Sigma \times \mathcal{X} \to \Sigma$ parameterized by a secret $\alpha \in \mathcal{A}$ and a deterministic[5] message function $\mu^{(\beta)} : \Sigma \to \mathsf{M}$ parameterized by a secret $\beta \in \mathcal{B}$, that on input sequence $(x_1, \ldots, x_\ell) \in \mathcal{X}$, *reaches* a state σ_ℓ, where $\sigma_i = \pi^{(\alpha)}(\sigma_{i-1}, x_i)$ for $i = 1, \ldots, \ell$ and $\sigma_0 = \mathsf{start}$, and *outputs* a message $\mu^{(\beta)}(\sigma_\ell)$. We also define $\text{REACH}_{\pi^{(\alpha)}}(x_1, \ldots, x_\ell) = \{\sigma_0, \cdots, \sigma_\ell\}$ and $\overline{\pi}^{(\alpha)}(x_1, \ldots, x_\ell) = \sigma_\ell$. $\triangleleft$

Reactive programs have an associated implicit security parameter κ; specifically, we require that the states in Σ and secrets in $\mathcal{A}$, $\mathcal{B}$ are represented as binary strings of length polynomial in the security parameter κ, and the functions $\pi^{(\alpha)}$ and $\mu^{(\beta)}$ are polynomial in κ. Throughout the rest of the paper, we shall omit κ and implicitly refer to "polynomial in κ" as simply being polynomial.
Partition Function and Program Class. A transition function class $\mathcal{P}$ refers to a set of transition functions along with an associated partition function $\mathcal{I}$ that maps states to integers, i.e., $\mathcal{I} : \Sigma \to [N]$ for some positive integer N. We say that $\mathcal{I}$ partitions the state space Σ into $\Sigma_1, \cdots, \Sigma_N$ where,

$$\Sigma_i = \mathcal{I}^{-1}(i) := \{\sigma \mid \sigma \in \Sigma,\ \mathcal{I}(\sigma) = i\}$$

Unless otherwise stated, the start state of a reactive program is assumed to be in Σ_1. We say that a transition function $\pi^{(\alpha)}$ is **tree-ordered** with respect to $\mathcal{I}$, if the directed graph over $[N]$ (each partition as a vertex) with an edge-set

$$\{(i,j) \mid \exists \text{ distinct } \sigma \in \Sigma_i,\ \sigma' \in \Sigma_j,\ \exists x \in \mathcal{X} \text{ s.t } \pi^{(\alpha)}(\sigma, x) = \sigma'\}$$

is a tree, and all its edges (i,j) satisfy $i < j$. That is, for any partition j, there is at most a single partition $i < j$ from which states in partition j can be transitioned to. Further, we say that a transition function class $\mathcal{P}$ is tree-ordered if every $\pi^{(\alpha)} \in \mathcal{P}$ is tree-ordered w.r.t. the partition associated with $\mathcal{P}$.

A program class $(\mathcal{P}, \mathcal{M})$ is a set of reactive programs $(\pi^{(\alpha)}, \mu^{(\beta)})$ with $\pi^{(\alpha)} \in \mathcal{P}$ and $\mu^{(\beta)} \in \mathcal{M}$.

[5] As described in the full version, restricting our obfuscations to deterministic message functions is without loss of generality, even if we are interested in randomized message functions.

Reactive Program Generator. We now describe the process which generates a reactive program to be obfuscated. A PPT program G (which we call the generator) interacts with a PPT program Q (which we call the adversary) over many rounds; at the end G outputs a reactive program $(\pi^{(\alpha)}, \mu^{(\beta)})$. Both G and Q are also allowed to produce auxiliary outputs.

Definition 2 (($\mathcal{P}$,$\mathcal{M}$)-Generator G). A $(\mathcal{P}, \mathcal{M})$-*generator* G for a transition function class $\mathcal{P}$ and message function class $\mathcal{M}$ is a PPT interactive program that interacts with an arbitrary PPT program Q. We write

$$\Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G;\ a_Q\Big) \leftarrow \langle G : Q \rangle$$

to indicate that at the end of the interaction, G outputs $\Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G\Big)$ and Q outputs a_Q (where $\pi^{(\alpha)} \in \mathcal{P}$, $\mu^{(\beta)} \in \mathcal{M}$). A *generator class* is simply a set of generators. $\triangleleft$

An adversary class $\mathcal{Q}$ is simply a set of adversaries Q. Some useful adversary classes depending on the application are: set of all PPT machines (for active corruption) and set of "semi-honest" PPT machines which follow a given protocol. We also consider adversary classes with *setup*. For any T that is a program in a setup class $\mathcal{T}$, we use Q^T to denote an adversary Q that gets oracle access to an honest execution of T.

3.2 Reach Extractor

To define a reach extractor, we introduce some notation. We write $Q \hat{|} \mathcal{E}$ to denote a composite machine in which $\mathcal{E}$ semi-honestly runs Q internally in a straight-line manner (where $\mathcal{E}$ can read the internal state of Q), letting Q directly communicate externally (with a generator). $\mathcal{E}$ produces the final auxiliary output. For an adversary class with a setup, given an adversary Q^T in the composite machine $Q^T \hat{|} \mathcal{E}$, $\mathcal{E}$ is allowed to replace T with any program from $\mathcal{T}$. For example, to capture common reference strings as a setup, $\mathcal{T}$ would correspond to $\{\mathsf{Setup}, \mathsf{Setup}_{\mathrm{Sim}}\}$, where Setup is the standard setup algorithm and $\mathsf{Setup}_{\mathrm{Sim}}$ produces a simulated CRS.

$\mathcal{E}$ is a valid reach-extractor if the following hold: in the IDEAL interaction, $\mathcal{E}$ observes the adversary Q and produces an extra output (Π, X^*) such that the states reached in Π using X^* (that is, $\text{REACH}_{\Pi}(X^*)$) is an upper bound on what D can reach in $\pi^{(\alpha)}$; further the output is such that it reaches at-most a single state in each partition.[6]

Definition 3 (Reach-Extractor for Q w.r.t. $(\mathcal{G}, \mathring{\mathcal{P}})$). A *reach-extractor* for an adversary $Q \in \mathcal{Q}$ w.r.t. a $(\mathcal{P}$,$\mathcal{M})$-generator class $\mathcal{G}$ and a transition function class $\mathring{\mathcal{P}}$, is a PPT program $\mathcal{E}$ such that, for all $G \in \mathcal{G}$ and PPT D, the

[6] As discussed just before Sect. 2.3, the reach-restriction condition is to enable our composition theorem (which depends on the use of garbled circuits).

output X produced by the following two experiments are indistinguishable:

$$\begin{array}{ll} \textsc{real}(G, Q, D): & \Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G;\ a_Q\Big) \leftarrow \langle G : Q \rangle \\ & \text{output } X \leftarrow D(\pi^{(\alpha)}, \mu^{(\beta)}, a_G, a_Q) \\ \hline \textsc{ideal}(G, Q\hat{|}\mathcal{E}, D): & \Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G;\ a_Q, \sqcap, X^*\Big) \leftarrow \langle G : Q\hat{|}\mathcal{E} \rangle \\ & \text{output } X \leftarrow D(\pi^{(\alpha)}, \mu^{(\beta)}, a_G, a_Q) \end{array}$$

and further the following hold:

– In $\textsc{ideal}(G, Q\hat{|}\mathcal{E}, D)$, $\sqcap \in \mathring{\mathcal{P}}$.
– Suppose $\mathcal{I}$ partitions Σ into N parts $\Sigma_1, \cdots, \Sigma_N$. Then
  - **Reach-Bound**: For all $i \in [N]$, $\Pr[(\textsc{reach}_{\pi^{(\alpha)}}(X) \cap \Sigma_i) \nsubseteq (\textsc{reach}_{\sqcap}(X^*) \cap \Sigma_i)]$ is negligible.
  - **Reach-Restriction**: For all $i \in [N]$, $|\textsc{reach}_{\sqcap}(X^*) \cap \Sigma_i| \leq 1$. $\triangleleft$

Real and Ideal Program Classes. Note that, in the above definition, $\mathcal{E}$ in the ideal world is allowed to extract an idealized reactive program $\sqcap \in \mathring{\mathcal{P}}$ to describe the set of states reachable by the adversary Q. While in many of our examples, $\mathring{\mathcal{P}}$ is the same as $\mathcal{P}$, the class of "real world" reactive programs being obfuscated, this is not mandatory. This flexibility in the ideal world can help with enabling reach extraction while remaining useful in a higher level application. Please refer to the full version for more details.

3.3 Reach-Restricted Reactive Program Obfuscation

Recall that, our goal in obfuscating a reactive program is to hide the parameters α, β, except for the states an adversary can reach. Let $\mathcal{E}$ be a reach-extractor for Q w.r.t. $\mathcal{G}$ s.t. $\mathcal{E}$ outputs $\sqcap$, X^* and $\textsc{reach}_{\sqcap}(X^*)$ bounds the reach of the adversary in $\pi^{(\alpha)}$. Then, we define a secure obfuscation as requiring a simulator Sim which, given only the circuits $\pi^{(\cdot)}$, $\mu^{(\cdot)}$ and the reachable states $(\overline{x}, \mu^{(\beta)}(\sqcap(\overline{x})))$ for input sequences $\overline{x} \in X^*$, can output an obfuscation indistinguishable from a real obfuscation.

Definition 4 (R3PO scheme $\mathcal{O}$ for $(\mathcal{G},\mathcal{Q},\mathring{\mathcal{P}})$). A PPT program $\mathcal{O}$ is an *Reach-Restricted Reactive Program Obfuscation (R3PO) scheme* for a $(\mathcal{P}, \mathcal{M})$-Generator class $\mathcal{G}$ and transition function class $\mathring{\mathcal{P}}$, if the following hold: [7]

– **Correctness:** For all $\pi^{(\alpha)} \in \mathcal{P}$, $\mu^{(\beta)} \in \mathcal{M}$, $\rho \leftarrow \mathcal{O}(\pi^{(\alpha)}, \mu^{(\beta)})$, and $\overline{x} \in \mathcal{X}^*$, it holds that $\rho(\overline{x}) = \mu^{(\beta)}(\overline{\pi}^{(\alpha)}(\overline{x}))$.
– **Security:** There exists a PPT program Sim s.t. $\forall Q \in \mathcal{Q}$, there exists a reach-extractor $\mathcal{E}$ w.r.t. $(\mathcal{G}, \mathring{\mathcal{P}})$, so that $\forall\ G \in \mathcal{G}$, the outputs of the following two experiments are indistinguishable:

[7] We assume that the programs $\pi^{(\alpha)}, \mu^{(\beta)}, \mathcal{I}, \mathcal{O}$ are all specified as circuits. Further, $\pi^{(\alpha)}$ and $\mu^{(\beta)}$ are given as circuits for $\pi^{(\cdot)}$ and $\mu^{(\cdot)}$ (resp.), which take α and β (resp.) as an input.

$$\text{REAL}(G,Q): \quad \begin{aligned} &\Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G;\, a_Q\Big) \leftarrow \langle G : Q\rangle \\ &\rho \leftarrow \mathcal{O}(\pi^{(\alpha)}, \mu^{(\beta)}) \\ &\text{output } (\rho, a_G, a_Q) \end{aligned}$$

$$\text{IDEAL}(G,Q): \quad \begin{aligned} &\Big((\pi^{(\alpha)}, \mu^{(\beta)}), a_G;\, a_Q, \Pi, X^*\Big) \leftarrow \langle G : \hat{Q|}\mathcal{E}\rangle \\ &\rho \leftarrow \mathsf{Sim}\left(\pi^{(\cdot)}, \mu^{(\cdot)}, \Pi, \{\overline{x}, \mu^{(\beta)}(\Pi(\overline{x}))\}_{\overline{x}\in X^*}\right) \\ &\text{output } (\rho, a_G, a_Q) \end{aligned}$$

◁

4 A Composition Theorem for R3PO

We now describe our composition theorem that enables building an R3PO for a generator class from R3POs for generator classes that produces smaller “one-step” (or non-reactive) programs. First we formalize the notion of decomposition.

4.1 Decomposition

The goal of decomposition is to view the transition function π of a reactive program produced by a generator G, as consisting of several one-step transitions π_i of reactive programs produced by generators H_i. Below, we define the notion of a σ-restriction of π at a state σ.

One-Step Restriction of a Transition Function. Given a reactive program's transition function $\pi^{(\alpha)}$ and one of its states σ, we define a *one-step σ -restriction of* $\pi^{(\alpha)}$ as a transition function $\widehat{\pi}_\sigma^{(\alpha)}$ with start state σ, where

$$\widehat{\pi}_\sigma^{(\alpha)}(\sigma', x) = \begin{cases} \pi^{(\alpha)}(\sigma, x) & \text{if } \sigma' = \sigma \\ \sigma & \text{otherwise.} \end{cases}$$

(i.e., in $\widehat{\pi}_\sigma^{(\alpha)}$, the only transitions allowed are from its start state σ).

Note that the state space Σ of π can be exponentially large in κ, and correspondingly π consists of that many one-step transition functions. When decomposing π, we will group them into polynomially many classes of transition functions, using the partition $\mathcal{I}$ of the state space, $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_N$ associated with π. This imposes the following structure on the class of transition functions $\mathcal{P}$ to which π belongs.

The transition function class $\mathcal{P}_1 \times \cdots \times \mathcal{P}_N$. For any set of N classes $\mathcal{P}_1, \cdots, \mathcal{P}_N$ over the (same) state space Σ and partition function $\mathcal{I} : \Sigma \to [N]$, we define $\mathcal{P}_1 \times \cdots \times \mathcal{P}_N$ to consist of transition functions $\pi^{(\alpha)}$ such that for each state $\sigma \in \Sigma_i$, the one-step σ-restriction of $\pi^{(\alpha)}$ is in $\mathcal{P}_i$. That is, for all $\sigma \in \Sigma$ and inputs x, $\pi^{(\alpha)}(\sigma, x) = \widehat{\pi}_\sigma^{(\alpha)}(\sigma, x)$ where $\widehat{\pi}_\sigma^{(\alpha)} \in \mathcal{P}_{\mathcal{I}(\sigma)}$.

Though π can have exponentially many states, we would like to view it as composed of N transition functions, $\widehat{\pi}_{\sigma_i} \in \mathcal{P}_i$, where $\sigma_i \in \Sigma_i$. Thanks to the reach-restriction requirement on the reactive programs that we are interested in, for each i, there would indeed be only one state $\sigma_i \in \Sigma_i$ that we need to consider. However, recall that π is dynamically generated by a generator G interacting with an adversary Q, and the reachable states in π are determined by this interaction. So the decomposition should be framed at the level of the interactive generators, rather than individual transition functions.

This leads us to a bi-simulation based definition of decomposition that views the generator G as incorporating another generator H (which produces one-step programs), and gives a two-way equivalence between them. To formalize this notion of bi-simulation, we introduce the following notation of composite machines.

Composite Machines. It will be convenient to define a few different ways in which a program (a generator or an adversary) can be wrapped by another program. As described below, a generator will be wrapped by a blackbox simulator, J or Z.[8] We also introduce a non-blackbox wrapper W which will be used to adapt an adversary Q (that expects to interact with G) so that it can interact with both G and H.

– For a generator H, we write $\boxed{\begin{smallmatrix} Z \\ \boxed{H} \end{smallmatrix}}$ to denote a composite machine in which Z runs H internally in a blackbox straight-line manner. The reactive program output by the composite generator is produced H, and the auxiliary output produced by it contains outputs from both H and Z. H may communicate with Z, and further the composite machine can communicate externally as described shortly. The running time of $\boxed{\begin{smallmatrix} Z \\ \boxed{H} \end{smallmatrix}}$ is bounded by that of H plus an *additive* $\mathsf{poly}(\kappa)$ overhead that depends on Z.

– For a generator G, we write $\boxed{\begin{smallmatrix} \boxed{G} \\ J \end{smallmatrix}}$ to denote a slightly different composite machine, which is similar to $\boxed{\begin{smallmatrix} J \\ \boxed{G} \end{smallmatrix}}$ (G is run internally by J in a blackbox straight-line manner, and the auxiliary information is produced by both) but the reactive program it produces is output by J. The external communication pattern is also different as described below.

– For an adversary Q, we write $\boxed{\begin{smallmatrix} \boxed{Q} \\ W \end{smallmatrix}}$ to denote a composite machine in which W internally runs Q in a straight-line manner with additive overhead, *but W can read the internal state of Q*. The auxiliary output of this composite machine is the entire view of W (which includes the auxiliary information a_Q produced by Q). The communication pattern is described below.

– Each of $\boxed{\begin{smallmatrix} \boxed{G} \\ J \end{smallmatrix}}$, $\boxed{\begin{smallmatrix} Z \\ \boxed{H} \end{smallmatrix}}$ and $\boxed{\begin{smallmatrix} \boxed{Q} \\ W \end{smallmatrix}}$ has three external communication channels – one used by the internal machine (shown boxed) and the other two by the wrapper machine. In all machines the "middle" channel is used by the wrapper (J, Z and W, respectively); the "top" channel is used by G, Z and Q (resp.); the "bottom" channel is used by J, H and W (resp.). Note that when $\boxed{\begin{smallmatrix} \boxed{G} \\ J \end{smallmatrix}}$ is

[8] Looking ahead, the role of J below is to simulate the presence of a one-step generator H when the actual execution involves the generator G, and the role of Z is to simulate the presence of G when the actual execution involves H.

connected to $\boxed{\begin{array}{c}\fbox{Q}\\ W\end{array}}$, Q directly interacts with G, whereas when $\boxed{\begin{array}{c}Z\\ \fbox{H}\end{array}}$ is connected to $\boxed{\begin{array}{c}\fbox{Q}\\ W\end{array}}$, Q interacts with Z.

For the ease of writing expressions, we shall denote $\boxed{\begin{array}{c}\fbox{G}\\ J\end{array}}$ by $G\rfloor\!\rfloor J$, and $\boxed{\begin{array}{c}Z\\ \fbox{H}\end{array}}$ by $H\lfloor\!\lfloor Z$. We will denote $\boxed{\begin{array}{c}\fbox{Q}\\ W\end{array}}$ by $Q\lfloor W$; in fact, we will be interested in $\boxed{\begin{array}{c}\fbox{$Q\hat{\rceil}\mathcal{E}$}\\ W\end{array}}$ (where $Q\hat{\rceil}\mathcal{E}$ itself is a composite machine involving a reach-extractor which interacts with Q as defined in Definition 3); we shall denote it by $Q\hat{\rceil}\mathcal{E}\lfloor W$.

Partial Reach-Extractor: For a valid decomposition, it will be important to have a bi-simulation that maps π to a one-step restriction π_σ such that σ is the unique reachable state in a subset of states Σ_i. To enforce this, we shall rely on an extractor for the adversary Q w.r.t. the generator G that produces π. However, the purpose of decomposition and composition is to be able to obtain an extractor for Q w.r.t. G along with a simulator, as in the definition of R3PO (Definition 4). To break this apparent circularity, we use the notion of a partial reach extractor: An $(i-1)$-partial reach extractor will be sufficient for defining decomposition "at part Σ_i," and it can be extended to an i-partial reach extractor, using the R3PO guarantee for the one-step generator.

Formally, a *t-partial reach-extractor* is defined identically to Definition 3, but with the relaxation that the reach-bound condition needs to hold only for $i \le t$, instead of $i \le N$. (The reach-restriction condition is still required to hold for all $i \in N$.) Thus, an N-partial reach extractor is a "full" reach-extractor.

Now we are ready to state the definition of decomposition. While informally we shall refer to decomposing a reactive program (or even a transition function) to one-step programs, formally, the decomposition is of a generator class to a sequence of generator classes, specified along with corresponding adversary classes and relaxed program classes.

Definition 5 (Decomposition of $(\mathcal{G}, \mathcal{Q})$ to $\mathcal{L}$). Let $\mathcal{G}$ be a $(\mathcal{P}, \mathcal{M})$-generator class where $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_N$ is tree-ordered. Let $\mathcal{L} = (\mathcal{H}, \mathcal{Q}, \mathring{\mathcal{P}})$, where $\mathcal{H} = \{H\lfloor\!\lfloor Z \mid \text{PPT } Z\}$ for a fixed $(\mathcal{P}_i, \mathcal{M}_i)$-generator H, $\mathcal{Q}$ is an adversary class, and $\mathring{\mathcal{P}}$ is a transition function class.

Then, a generator $G \in \mathcal{G}$ is said to be *decomposable at part i to $\mathcal{L}$* if, there exist PPT J, Z, W so that $\forall Q \in \mathcal{Q}$, and all $(i-1)$-partial reach-extractors $\mathcal{E}$ for Q w.r.t. $(\mathcal{G}, \mathring{\mathcal{P}})$, it holds that $Q\hat{\rceil}\mathcal{E}\lfloor W \in \mathcal{Q}$ and:

– **Indistinguishability:** $\langle G\rfloor\!\rfloor J : Q\hat{\rceil}\mathcal{E}\lfloor W\rangle \approx \langle H\lfloor\!\lfloor Z : Q\hat{\rceil}\mathcal{E}\lfloor W\rangle$.

– In $\langle G\rfloor\!\rfloor J : Q\hat{\rceil}\mathcal{E}\lfloor W\rangle$, let the output of G be $((\pi^{(\alpha)}, \mu^{(\beta)}), a_G)$, and of $\mathcal{E}$ be $(a_Q, \sqcap, X^*)$; then J outputs $((\widehat{\pi}_\sigma^{(\alpha)}, \widehat{\mu}_\sigma^{(\widehat{\beta})}), a_J)$ s.t.
  - **Correct One-Step Restriction:** $\text{REACH}_{\sqcap}(X^*) \cap \Sigma_i \subseteq \{\sigma\}$.
  - **Correct Message Function:** $\widehat{\mu}_\sigma^{(\widehat{\beta})} \leftarrow \mathcal{M}_i$ is uniformly sampled at the end of the execution.

$(\mathcal{G}, \mathcal{Q})$ is said to be *decomposable into* $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ if $\forall G \in \mathcal{G},\ i \in [N]$, it holds that $\mathcal{L}_i = (\mathcal{H}_i, \mathcal{Q}_i, \overset{\circ}{\mathcal{P}}_i)$ where $\mathcal{Q}_i \supseteq \mathcal{Q}$, and G is decomposable at part i to $\mathcal{L}_i$. $\lhd$

Above, we require two simulations to produce indistinguishable outputs (which includes their communication, as $Q\widehat{|}\mathcal{E}\lfloor W$ outputs its entire view as part of output), with J mimicking H, and Z mimicking G. The "correct one-step restriction" condition forces J (and hence H) to output a one-step restriction whose start state is the state that is reachable, as reported by a (partial) reach-extractor for G.

4.2 Composition Theorem

Above, decomposition related the transition functions in $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_N$ to those in each $\mathcal{P}_i$. Before stating our composition theorem, we need to specify the message function space $\widehat{\mu}_i$ of these one-step programs as well.

As described in Sect. 2.3, $\widehat{\mu}_i$ should release garbled circuit labels for the state at which it is evaluated. For our purposes, it will be helpful to consider a labeling function (denoted below as $\widehat{\beta}$) which takes the part index i as an input, along with a bit position j and bit value b. Then $\widehat{\mu}_i$ will be of the form $\mathsf{encode}^{\mathcal{I},t}_{\widehat{\beta}}$ defined below, which only retains the part of $\widehat{\beta}$ for parts $i > t$.

Parameters: Program classes $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_N$ with state-space $\Sigma = \{0,1\}^n$, and partition function $\mathcal{I} : \Sigma \to [N]$. A message function class $\mathcal{M}$.

Given One-Step Obfuscators: For each $i \in [N]$, $\mathcal{O}_i$ (taking inputs in $\mathcal{P}_i \times \widehat{\mathcal{M}}_{\mathcal{I},i}$)

Garbling Scheme: Let $(\mathsf{GCGarble}, \mathsf{GCEval})$ be a garbling scheme.

Obfuscator $\mathcal{O}$:

- **Input:** $(\pi^{(\alpha)}, \mu^{(\beta)})$, where $\pi^{(\alpha)} \in \mathcal{P}$, $\mu^{(\beta)} \in \mathcal{M}$.
- Uniformly randomly sample $\widehat{\beta} : [N] \times [n] \times \{0,1\} \to \{0,1\}^\kappa$
- For each $i \in [N]$,
  - ∗ Define $\widehat{\mu}_i$ to be $\mathsf{encode}^{\mathcal{I},i}_{\widehat{\beta}}$.
  - ∗ Define the function f_i as follows (with a fresh random tape hard-coded for $\mathcal{O}_i$ and, if needed, $\mu^{(\beta)}$):

$$f_i(\sigma) \ = \ \left(\mathcal{O}_i(\widehat{\pi}^{(\alpha)}_\sigma, \widehat{\mu}_i),\ \mu^{(\beta)}(\sigma)\right)$$

  - ∗ Let $\mathrm{GC}_i \leftarrow \mathsf{GCGarble}(f_i, \widehat{\beta}_i)$, where $\widehat{\beta}_i(\ell, b) = \widehat{\beta}(i, \ell, b)$ for $\ell \in [n], b \in \{0,1\}$.
- Let $i_0 = \mathcal{I}(\mathsf{start})$, where start is the start-state of $\pi^{(\alpha)}$.
  Output $\left(f_{i_0}(\mathsf{start}), \{\mathrm{GC}_i\}_{i \in [N]\setminus\{i_0\}}\right)$, along with a "driver program."

Fig. 1. Obfuscator $\mathcal{O}$ used to prove Theorem 1.

Definition 6 (Message function space $\widehat{\mathcal{M}}$). Let $\Sigma = \{0,1\}^n$, with a partition function $\mathcal{I} : \Sigma \to [N]$, and $\widehat{\beta} : [N] \times [n] \times \{0,1\} \to \{0,1\}^\kappa$. A *state labeling function* $\mathsf{encode}_{\widehat{\beta}}^{\mathcal{I},t} : \Sigma \to \{0,1\}^{n\kappa}$ is defined as

$$\mathsf{encode}_{\widehat{\beta}}^{\mathcal{I},t}(\sigma) = \begin{cases} \left(\widehat{\beta}(\mathcal{I}(\sigma), 1, a_1), \dots, \widehat{\beta}(\mathcal{I}(\sigma), n, a_n)\right) & \text{if } \mathcal{I}(\sigma) > t, \\ \perp & \text{otherwise,} \end{cases}$$

where $\sigma = (a_1, ..., a_n)$. Then, we define $\widehat{\mathcal{M}} = \bigcup_{\mathcal{I}:\Sigma\to[N], t\in[N]} \widehat{\mathcal{M}}_{\mathcal{I},t}$, where

$$\widehat{\mathcal{M}}_{\mathcal{I},t} \;=\; \left\{ \left(\mathsf{encode}_{\widehat{\beta}}^{\mathcal{I},t}\right) \mid \widehat{\beta} : [N] \times [n] \times \{0,1\} \to \{0,1\}^\kappa \right\}$$

$\lhd$

We are now ready to state our composition theorem.

Theorem 1. *Suppose $\mathcal{G}$ is a $(\mathcal{P}, \mathcal{M})$-generator class that is decomposable into $\boldsymbol{\mathcal{L}} = \{\mathcal{H}_i, \mathcal{Q}_i, \mathring{\mathcal{P}}_i\}_{i\in[N]}$, such that, for each $i \in [N]$, $\mathcal{H}_i$ is a $(\mathcal{P}_i, \widehat{\mathcal{M}}_{\mathcal{I},i})$-generator class and there exists an R3PO scheme $\mathcal{O}_i$ for $(\mathcal{H}_i, \mathcal{Q}_i, \mathring{\mathcal{P}}_i)$. Then there exists an R3PO scheme $\mathcal{O}$ for $(\mathcal{G}, \mathcal{Q}, \mathring{\mathcal{P}})$ where $\mathring{\mathcal{P}} = \mathring{\mathcal{P}}_1 \times \cdots \times \mathring{\mathcal{P}}_N$.*

The obfuscator $\mathcal{O}$ used to prove Theorem 1 is shown in Fig. 1. Please refer to the full version for the proof that it is an R3PO scheme.

5 Private Multi-Authority ABE

In this section, we define Private Multi-Authority ABE and show how to instantiate it from any CP-ABE scheme, using R3PO schemes for commitment-opening and signatures together. Section 2.6 gives an overview of the notion and the construction described below.

5.1 Definition for Private Multi-Authority ABE

Let the authorities in the system be $\mathbb{A}_1, \dots, \mathbb{A}_\mathsf{N}$, s.t. each authority $\mathbb{A}_i$ publishes its public key mpk_i after a local non-interactive setup. Each authority $\mathbb{A}_i$ also has an attribute-granting policy Θ_i^{gid} w.r.t. each gid. A sender encrypts a message m with a ciphertext-policy ϕ under the public keys of all the authorities, s.t. a receiver with global identifier gid and attribute vector $\bar{\mathsf{x}}$ can decrypt it only if it has attribute key for $\bar{\mathsf{x}}$ and each authorities' attribute-granting policies accepts (that is, $\forall i \in [N], \Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$) and the ciphertext-policy accepts (that is, $\phi(\bar{\mathsf{x}}) = 1$). To get the attribute key for attribute vector $\bar{\mathsf{x}}$, the receiver sends a key-request req_i to each authority $\mathbb{A}_i$, gets back a key-component $\mathsf{sk}_{\mathsf{req}_i}$, and combines all the key-components to construct the key $\mathsf{sk}_{\bar{\mathsf{x}}}$.

We define security w.r.t. a corruption model where the adversary is allowed to maliciously corrupt the receiver and semi-honestly corrupt any subset of the authorities. If the receiver is honest, we require that any key-request req reveals nothing about $\bar{\mathsf{x}}$ to the adversary (even if it semi-honestly corrupts all the authorities). If the receiver is corrupt, we require that the adversary is unable to distinguish between encryptions of any m_0 and m_1 w.r.t. a policy ϕ, if it did not send a key-request for any $\bar{\mathsf{x}}$ that satisfies ϕ to the honest authorities.

Definition 7 (Private Multi-Authority ABE (p-MA-ABE)). A p-MA-ABE scheme for N authorities, message space M, class $\mathcal{C}$ of ciphertext-policies and class $\boldsymbol{\Theta}$ of attribute-granting policies, both over n-bit attributes, and global identifiers space GID consists of PPT algorithms as follows:

- $\mathsf{SetupAuth}(1^\kappa) \rightarrow (\mathsf{mpk}, \mathsf{msk})$: On input the security parameter κ, outputs the master keys for an individual authority.
- $\mathsf{Encrypt}\left(\{\mathsf{mpk}_i\}_{i\in[\mathsf{N}]}, \phi, m\right) \rightarrow \mathsf{ct}$: On input the master public keys of all authorities, a policy $\phi : \{0,1\}^n \rightarrow \{0,1\}$ in $\mathcal{C}$ and a message $m \in \mathsf{M}$, outputs a ciphertext ct.
- $\mathsf{KeyRequest}\left(\{\mathsf{mpk}_i\}_{i\in[\mathsf{N}]}, \mathsf{gid}, \bar{\mathsf{x}}\right) \rightarrow (\mathsf{st}, \{\mathsf{req}_i\}_{i\in[\mathsf{N}]})$: On input the master public keys of all authorities, a global identity $\mathsf{gid} \in \mathsf{GID}$ and an attribute vector $\bar{\mathsf{x}} \in \{0,1\}^n$, outputs a recipient state st and requests $\{\mathsf{req}_1, \dots, \mathsf{req}_\mathsf{N}\}$.
- $\mathsf{KeyGen}\left(i, \mathsf{msk}_i, \Theta_i^{\mathsf{gid}}, \mathsf{req}_i, \{\mathsf{mpk}_j\}_{j\in[\mathsf{N}]}\right) \rightarrow \mathsf{sk}_{\mathsf{req}_i}$: On input an authority index $i \in [\mathsf{N}]$, master secret key msk_i, an attribute-granting policy Θ_i^{gid}, a request req_i and the master public keys of all authorities, outputs a key component $\mathsf{sk}_{\mathsf{req}_i}$ or $\perp$.
- $\mathsf{KeyCombine}\left(\mathsf{st}, \{\mathsf{sk}_{\mathsf{req}_i}\}_{i\in[\mathsf{N}]}\right) \rightarrow \mathsf{sk}_{\bar{\mathsf{x}}}$: On input a recipient state st and a set of key components $\{\mathsf{sk}_{\mathsf{req}_i}\}_{i\in[\mathsf{N}]}$, outputs a secret key $\mathsf{sk}_{\bar{\mathsf{x}}}$.
- $\mathsf{Decrypt}\left(\mathsf{sk}_{\bar{\mathsf{x}}}, \mathsf{ct}\right) \rightarrow m$: On input a secret key $\mathsf{sk}_{\bar{\mathsf{x}}}$ and a ciphertext ct, outputs a message m or $\perp$.

The following correctness and security properties are required:

1. **Correctness**: $\forall$ security parameter κ, number of authorities $\mathsf{N} \in \mathbb{N}$, identities $\mathsf{gid} \in \mathsf{GID}$, messages $m \in \mathsf{M}$, ciphertext policies $\phi \in \mathcal{C}$, attribute granting policies $\{\Theta_i^{\mathsf{gid}} \in \boldsymbol{\Theta}\}_{i\in[\mathsf{N}]}$, and attribute vectors $\bar{\mathsf{x}}$ s.t $\phi(\bar{\mathsf{x}}) = 1$ and $\Theta_i^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$ for all $i \in [\mathsf{N}]$, it holds that if:

$$\begin{aligned}
\forall i \in [\mathsf{N}], \quad & (\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{SetupAuth}(1^\kappa) \\
& (\mathsf{st}, \{\mathsf{req}_i\}_{i\in[\mathsf{N}]}) \leftarrow \mathsf{KeyRequest}(\{\mathsf{mpk}_i\}_{i\in[\mathsf{N}]}, \mathsf{gid}, \bar{\mathsf{x}}) \\
\forall i \in [\mathsf{N}], \quad & \mathsf{sk}_{\mathsf{req}_i} \leftarrow \mathsf{KeyGen}\left(i, \mathsf{msk}_i, \Theta_i^{\mathsf{gid}}, \mathsf{req}_i, \{\mathsf{mpk}_i\}_{i\in[\mathsf{N}]}\right) \\
& \mathsf{sk}_{\bar{\mathsf{x}}} \leftarrow \mathsf{KeyCombine}\left(\mathsf{st}, \{\mathsf{sk}_{\mathsf{req}_i}\}_{i\in[\mathsf{N}]}\right)
\end{aligned}$$

then $\Pr\left[\mathsf{Decrypt}\left(\mathsf{sk}_{\bar{\mathsf{x}}}, \mathsf{Encrypt}(\{\mathsf{mpk}_i\}_{i\in[\mathsf{N}]}, \phi, m)\right) = m\right] = 1$.

2. **Security of encryption**: For any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ with semi-honest corruption of any subset of authorities and malicious corruption of the receiver, there exists a negligible function $\mathsf{negl}(.)$ such that the following holds in the experiment $\mathrm{IND}_{\mathrm{mesg}}^{\text{p-MA-ABE}}$ shown in Fig. 2:

$$\Pr[\mathrm{IND}_{\mathrm{mesg}}^{\text{p-MA-ABE}}(\mathcal{A}) = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

3. **Receiver Privacy against Semi-honest Adversary**: For any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ that corrupts each authority in a semi-honest way, there exists a negligible function $\mathsf{negl}(.)$ such that the following holds in the experiment $\mathrm{IND}_{\mathrm{attr}}^{\text{p-MA-ABE}}$ shown in Fig. 2:

$$\Pr[\mathrm{IND}_{\mathrm{attr}}^{\text{p-MA-ABE}}(\mathcal{A}) = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

$\lhd$

5.2 Construction for Private Multi-Authority ABE

In this section, we give a scheme for p-MA-ABE from the following primitives:

- a CP-ABE scheme
- a non-interactive UC secure commitment scheme
- a puncturable signature scheme
- an R3PO scheme $\mathcal{O}_{\text{p-MA-ABE}}$ w.r.t. $(\mathcal{G}^1_{\text{p-MA-ABE}}, \mathcal{Q}_{\text{p-MA-ABE}})$ and $(\mathcal{G}^2_{\text{p-MA-ABE}}, \mathcal{Q}_{\text{p-MA-ABE}})$, which we describe in the full version.

Let the number of authorities be N, space of access policies $\mathcal{C}$ correspond exactly to the policies supported by the underlying single-authority CP-ABE scheme.

Completeness Requirement of Commitment Scheme: We will additionally require that, given a commitment setup $\mathsf{Com.crs}$, for any c, d s.t. $\mathsf{Com.Open}(\mathsf{Com.crs}, c, d) = m$, it holds that (m, c, d) lies in the support of the commit algorithm, that is: there exists r s.t. $(c, d) \leftarrow \mathsf{Com.Commit}(\mathsf{Com.crs}, m; r)$. Any commitment scheme can be enhanced to have this property, by modifying it as follows.

Parameter: Let κ be the security parameter, N be the number of authorities.

Experiment $\mathbf{IND}^{\mathbf{p\text{-}MA\text{-}ABE}}_{\mathbf{mesg}}$

- $(\mathsf{st}_0, H) \leftarrow \mathcal{A}_0(1^\kappa)$.
- $\{(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{SetupAuth}(1^\kappa)\}_{i \in H}$.
- $\left(\mathsf{st}_1, \{\mathsf{mpk}_i\}_{i \in [\mathsf{N}] \setminus H}\right) \leftarrow \mathcal{A}_1(\mathsf{st}_0, \{\mathsf{mpk}_i\}_{i \in H})$.

  Let $O_k(y, \{\Theta_i^{\mathsf{gid}}\}_{i \in H}) := \mathsf{KeyGen}\left(k, \mathsf{msk}_k, \Theta_k^{\mathsf{gid}}, y, \{\mathsf{mpk}_i\}_{i \in [\mathsf{N}]}\right)$, where $y = (\mathsf{gid}, c)$.
- $(\mathsf{st}_2, \phi, m_0, m_1) \leftarrow \mathcal{A}_2^{\{O_k(\cdot,\cdot)\}_{k \in H}}(\mathsf{st}_1)$
- $b \leftarrow \{0,1\}$.
- $\mathsf{ct} \leftarrow \mathsf{Encrypt}\left(\{\mathsf{mpk}_i\}_{i \in [\mathsf{N}]}, \phi, m_b\right)$.
- $b' \leftarrow \mathcal{A}_3^{\{O_k(\cdot)\}_{k \in H}}(\mathsf{st}_2, \mathsf{ct})$.
- If $\mathcal{A}_2$ or $\mathcal{A}_3$ queried any oracle with different key-policies $\{\Theta_i^{\mathsf{gid}}\}_{i \in H}$ for the same globalid gid, output $a \leftarrow \{0,1\}$.

  If $\exists \mathsf{gid}, \bar{\mathsf{x}}$ s.t. $\phi(\bar{\mathsf{x}}) = 1$ and, $\forall k \in H$, $\Theta_k^{\mathsf{gid}}(\bar{\mathsf{x}}) = 1$, and $\exists r, k^* \in H$ s.t. $\mathcal{A}_2$ or $\mathcal{A}_3$ queried the oracle O_{k^*} with req_{k^*}, where $(\mathsf{st}, \{\mathsf{req}_k\}_{k \in [\mathsf{N}]}) \leftarrow \mathsf{KeyRequest}(\{\mathsf{mpk}_k\}_{k \in [\mathsf{N}]}, \mathsf{gid}, \bar{\mathsf{x}};\ r)$, then output $a \leftarrow \{0,1\}$.

  Else, output $a = b \oplus b'$.

Experiment $\mathbf{IND}^{\mathbf{p\text{-}MA\text{-}ABE}}_{\mathbf{attr}}$

- $(\mathsf{st}_0, \{\mathsf{mpk}_i\}_{i \in [\mathsf{N}]}, \bar{\mathsf{x}}_0, \bar{\mathsf{x}}_1, \mathsf{gid}) \leftarrow \mathcal{A}_0(1^\kappa)$.
- $b \leftarrow \{0,1\}$.
- $b' \leftarrow \mathcal{A}_1(\mathsf{st}_0, \mathsf{KeyRequest}(\{\mathsf{mpk}_i\}_{i \in [\mathsf{N}]}, \mathsf{gid}, \bar{\mathsf{x}}_b))$.
- Output $b \oplus b'$.

Fig. 2. p-MA-ABE security experiments.

$$\mathsf{Commit}'(\mathsf{Com.crs}, m; r) = \begin{cases} (c, d) & \text{if } r = 0^k || c || d \text{ and } \mathsf{Com.Open}(\mathsf{Com.crs}, c, d) = m \\ \mathsf{Com.Commit}(\mathsf{Com.crs}, m; r) & \text{otherwise.} \end{cases}$$

We now prove that the protocol in Fig. 3 is in fact a secure p-MA-ABE scheme.

Lemma 1. *If there exists a CP-ABE scheme, a non-interactive UC secure commitment scheme, a puncturable signature scheme, and an R3PO scheme $\mathcal{O}_{p\text{-}MA\text{-}ABE}$ w.r.t. $(\mathcal{G}^1_{p\text{-}MA\text{-}ABE}, \mathcal{Q}_{p\text{-}MA\text{-}ABE})$*[9]*, then there exists a secure p-MA-ABE scheme.*

Please refer to the full version for the full details of the proof.

[9] Which is also an R3PO w.r.t. $(\mathcal{G}^2_{\text{p-MA-ABE}}, \mathcal{Q}_{\text{p-MA-ABE}})$.

Protocol for Private Multi-Authority ABE

Parameter: Let κ be the security parameter, N be the number of authorities.
Primitives:
A CP-ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Encrypt}, \mathsf{ABE.Decrypt})$
A Commitment scheme $\mathsf{Com} = (\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Open})$
A puncturable signature scheme $\mathsf{Sig} = (\mathsf{Sig.gen}, \mathsf{Sig.sign}, \mathsf{Sig.verify})$
An obfuscation scheme $\mathcal{O}_{\text{p-MA-ABE}}$

– p-MA-ABE.$\mathsf{SetupAuth}(1^\kappa)$:
- Sample the common random string $\mathsf{Com.crs} \leftarrow \mathsf{Com.Setup}(1^\kappa)$.
- Sample the signature keys $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.gen}(1^\kappa)$.
- Sample the ABE keys $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\kappa)$.
- Set $\mathsf{mpk} := (\mathsf{Com.crs}, \mathsf{Sig.vk}, \mathsf{ABE.mpk})$ and $\mathsf{msk} := (\mathsf{Sig.sk}, \mathsf{ABE.msk})$.
- Output $(\mathsf{mpk}, \mathsf{msk})$.

– p-MA-ABE.$\mathsf{Encrypt}\,(\mathsf{pk}, \phi, m)$:
- For all $i \in [\mathsf{N}]$, parse mpk_i as $(\mathsf{Com.crs}_i, \mathsf{Sig.vk}_i, \mathsf{ABE.mpk}_i)$.
- Sample $s_1, \ldots, s_\mathsf{N}$ s.t. $s_1 + \ldots + s_\mathsf{N} = m$.
- For all $i \in [\mathsf{N}]$, compute $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Encrypt}(\mathsf{ABE.mpk}_i, \phi, s_i)$.
- Output $\mathsf{ct} := \{\mathsf{ABE.ct}_1, \ldots, \mathsf{ABE.ct}_\mathsf{N}\}$.

– p-MA-ABE.$\mathsf{Decrypt}\,(\mathsf{sk}_{\bar{\mathsf{x}}}, \mathsf{ct})$:
- Parse $\mathsf{sk}_{\bar{\mathsf{x}}}$ as $\{\mathsf{sk}^{\mathsf{ABE}}_{\bar{\mathsf{x}},1}, \ldots, \mathsf{sk}^{\mathsf{ABE}}_{\bar{\mathsf{x}},\mathsf{N}}\}$.
- Parse ct as $\{\mathsf{ABE.ct}_1, \ldots, \mathsf{ABE.ct}_\mathsf{N}\}$.
- For all $i \in [\mathsf{N}]$, set $s_i = \mathsf{ABE.Decrypt}(\mathsf{ABE.mpk}_i, \mathsf{sk}^{\mathsf{ABE}}_{\bar{\mathsf{x}},i}, \mathsf{ABE.ct}_i)$.
- Set $m := s_1 + \ldots + s_\mathsf{N}$.
- Output m.

– p-MA-ABE.$\mathsf{KeyRequest}(\mathsf{pk}, \mathsf{gid}, \bar{\mathsf{x}})$:
- For all $i \in [\mathsf{N}]$, parse mpk_i as $(\mathsf{Com.crs}_i, \mathsf{Sig.vk}_i, \mathsf{ABE.mpk}_i)$,
  compute $(c_i, d_i) \leftarrow \mathsf{Com.Commit}(\mathsf{Com.crs}_i, \mathsf{gid}\|\bar{\mathsf{x}})$
  and set $\mathsf{req}_i := (\mathsf{gid}, c_i)$.
- Set $\mathsf{st} := (\mathsf{gid}, \bar{\mathsf{x}}, \{d_i\}_{i\in[\mathsf{N}]})$.
- Output $(\mathsf{st}, \{\mathsf{req}_i\}_{i\in[\mathsf{N}]})$.

– p-MA-ABE.$\mathsf{KeyGen}(t, \mathsf{msk}_t, \Theta_t^{\mathsf{gid}}, \mathsf{req}_t, \mathsf{pk})$: Computes
- Parse msk as $(\mathsf{Sig.sk}_t, \mathsf{ABE.msk}_t)$.
- For all $i \in [\mathsf{N}]$, parse mpk_i as $(\mathsf{Com.crs}_i, \mathsf{Sig.vk}_i, \mathsf{ABE.mpk}_i)$.
- Parse req_t as (gid, c_t).
- Fix $\pi^{(\alpha)}_{\mathsf{pp}} \in \mathcal{P}_{\text{p-MA-ABE}}$ where $\mathsf{pp} = (t, \mathsf{Com.crs}_t, c_t), \alpha = \Theta_t^{\mathsf{gid}}$, and $\mathsf{start} = \sigma^1_{\mathsf{pk},\mathsf{gid}}$.
- Fix $\mu^{(\beta)} \in \mathcal{M}_{\text{p-MA-ABE}}$ where $\beta = (0, \mathsf{Sig.sk}_t, \mathsf{ABE.msk}_t)$.
- Compute $\mathsf{sk}_{\mathsf{req}_t} \leftarrow \mathcal{O}_{\text{p-MA-ABE}}(\pi^{(\alpha)}_{\mathsf{pp}}, \mu^{(\beta)})$.
- Output $\mathsf{sk}_{\mathsf{req}_t}$.

– p-MA-ABE.$\mathsf{KeyCombine}\left(\mathsf{st}, \{\mathsf{sk}_{\mathsf{req}_i}\}_{i\in[\mathsf{N}]}\right)$:
- Parse st as $(\mathsf{gid}, \bar{\mathsf{x}}, \{d_i\}_{i\in[\mathsf{N}]})$.
- For all $t \in [\mathsf{N}]$, parse $\mathsf{sk}_{\mathsf{req}_t}$ as ρ_t and evaluate as $\tau_t \leftarrow \rho_t(d_t)$[11]
- Finally, for all $t \in [\mathsf{N}]$, compute $\mathsf{sk}^{\mathsf{ABE}}_{\bar{\mathsf{x}},t} \leftarrow \rho_t(\tau_1, \ldots, \tau_\mathsf{N})$.
- Output $\{\mathsf{sk}^{\mathsf{ABE}}_{\bar{\mathsf{x}},t}\}_{t\in[\mathsf{N}]}$.

Fig. 3. A secure p-MA-ABE Protocol. (formally, ρ_t also takes as input a state σ_1 and also outputs a state σ_2. for brevity, we ignore mentioning it here.)

References

1. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13043, pp. 224–255. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90453-1_8
2. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013)
3. Backes, M., Pfitzmann, B., Waidner, M.: A general composition theorem for secure reactive systems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 336–354. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_19
4. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 26–51. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_2
5. Barak, B., et al.: On the (Im)possibility of obfuscating programs. J. ACM **59**(2) (2012). ISSN: 0004-5411
6. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 500–532. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_17
7. Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O.: On virtual grey box obfuscation for general circuits. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 108–125. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_7
8. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 535–564. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_20
9. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: ITCS 2014, pp. 1–12 (2014)
10. Campanelli, M., David, B., Khoshakhlagh, H., Konring, A., Nielsen, J.B.: Encryption to the future: a paradigm for sending secret messages to future (anonymous) committees. Cryptology ePrint Archive, Paper 2021/1423 (2021)
11. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pp. 136–145 (2001)
12. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
13. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 3–22. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_1
14. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_19
15. Chase, M.: Multi-authority attribute based encryption. In: Theory of Cryptography, pp. 515–534 (2007)
16. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 33–65. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_2

17. Datta, P., Komargodski, I., Waters, B.: Decentralized multi-authority ABE for DNFs from LWE. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 177–209. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_7
18. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM J. Comput. **30**, 391–437 (2001)
19. Döttling, N., Garg, S.: From selective IBE to Full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 372–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_13
20. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
21. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM **51**(6), 851–898 (2004)
22. Galbraith, S.D., Zobernig, L.: Obfuscating finite automata. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 90–114. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_4
23. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: removing private-key generator from IBE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 689–718. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03807-6_25
24. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A., Sekar, S.: Registration-based encryption from standard assumptions. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 63–93. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_3
25. Garg, S., Lu, S., Ostrovsky, R.: Black-Box Garbled RAM. Cryptology ePrint Archive, Report 2015/307 (2015)
26. Garg, S., Lu, S., Ostrovsky, R., Scafuro, A.: Garbled RAM from one-way functions. In: STOC 2015, pp. 449–458 (2015)
27. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 468–499. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_16
28. Gentry, C., Halevi, S., Lu, S., Ostrovsky, R., Raykova, M., Wichs, D.: Garbled RAM revisited. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 405–422. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_23
29. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. J. ACM **62**(6), 1–33 (2015)
30. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 89–98 (2006)
31. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055744
32. Dennis Hofheinz and Victor Shoup: GNUC: a new universal composability framework. J. Cryptol. **28**(3), 423–508 (2015)
33. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_13

34. Hohenberger, S., Waters, B.: Attribute-based encryption with fast decryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 162–179. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_11
35. Ishai, Y., Pandey, O., Sahai, A.: Public-Coin Differing-Inputs Obfuscation and Its Applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 668–697. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_26
36. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: STOC 2021, pp. 60–73 (2021)
37. Jutla, C.S., Roy, A.: Shorter Quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_1
38. Kim, S.: Multi-authority attribute-based encryption from LWE in the OT model. Cryptology ePrint Archive, Report 2019/280 (2019)
39. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
40. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_31
41. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 447–462. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_17
42. Lu, S., Ostrovsky, R.: How to garble RAM programs? In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 719–734. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_42
43. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_2
44. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27375-9_3
45. Micciancio, D., Tessaro, S.: An equational approach to secure multi-party computation. In: ITCS 2013, pp. 355–372 (2013)
46. Michalevsky, Y., Joye, M.: Decentralized policy-hiding ABE with receiver privacy. In: Lopez, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018, Part II. LNCS, vol. 11099, pp. 548–567. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98989-1_27
47. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: STOC 2004, pp. 242–251 (2004)
48. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
49. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC 2014, pp. 475–484 (2014)

50. Waters, B., Wee, H., Wu, D.J.: Multi-authority ABE from lattices without random oracles. Cryptology ePrint Archive, Paper 2022/1194 (2022). https://eprint.iacr.org/2022/1194
51. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: FOCS, pp. 600–611 (2017)
52. Wikström, D.: Simplified universal composability framework. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 566–595. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_24
53. Yao, A.C.-C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986), pp. 162–167 (1986)
54. Yun, K., Wang, X., Xue, R.: Identity-based functional encryption for quadratic functions from lattices. In: Naccache, D., et al. (eds.) ICICS 2018. LNCS, vol. 11149, pp. 409–425. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01950-1_24