Malicious Security for SCALES

Outsourced Computation with Ephemeral Servers

Anasuya Acharaya[1], Carmit Hazay[2], Vladimir Kolesnikov[3], and Manoj Prabhakaran[4]

[1] Bar-Ilan University, acharya@biu.ac.il
[2] Bar-Ilan University, carmit.hazay@biu.ac.il
[3] Georgia Institute of Technology, kolesnikov@gatech.edu
[4] Indian Institute of Technology Bombay, mp@cse.iitb.ac.in

Abstract. SCALES (Small Clients And Larger Ephemeral Servers) model is a recently proposed model for MPC (Acharya et al., TCC 2022). While the SCALES model offers several attractive features for practical large-scale MPC, the result of Acharya et al. only offered semi-honest secure protocols in this model.

We present a new efficient SCALES protocol secure against malicious adversaries, for general Boolean circuits. We start with the base construction of Acharya et al. and design and use a suite of carefully defined building blocks that may be of independent interest. The resulting protocol is UC-secure without honest majority, with a CRS and bulletin-board as setups, and allows publicly identifying deviations from correct execution.

Keywords: SCALES · MPC with Ephemeral Servers · Malicious Security · Dishonest Majority · Rerandomizable Garbled Circuits

1 Introduction

Secure Multi-Party Computation (MPC) has been steadily transitioning from theory to practice, especially in settings involving two or a small number of parties. However, making MPC practical for large scale tasks, possibly involving a large number of input providers, remains an outstanding challenge. Over the last few years, the question of practical large scale MPC has led to several approaches and ideas [BGG+20, GHK+21, GHM+21, GMPS21, CGG+21, RS22, AHKP22, KRY22, BDO23].

This work is motivated by one of the more recent proposals, called SCALES [AHKP22], which offers several attractive features for practical large scale MPC. Unfortunately, the results of [AHKP22] were primarily confined to the semi-honest setting, which is not sufficient in many real-life scenarios. In this work, we propose a new protocol in the SCALES model with security against active corruption. In fact, our protocol also allows publicly identifying corrupt behavior, offering the best possible level of security when a majority of the participants can be corrupt.

Before proceeding further, we discuss some of the key features of SCALES that makes it potentially the model of choice in many practical situations. SCALES is an acronym for "Small Clients And Larger Ephemeral Servers," which captures two of the main features of the model.

- *Small Clients* refers to the fact that the complexity of each party with an input is proportional only to its own input, and does not depend on the computational complexity of the function being securely evaluated, or even the number of other input clients. This is important in applications where a large number of clients supply their inputs to a task (e.g., computing statistics, or data models).
- *Ephemeral Servers* refers to the fact that (apart from the small clients) the protocol relies only on servers which are stateless, and which can join the protocol at any point; at that point they send out a single message and exit. The complexity of each server can be linear in the complexity of the function being securely evaluated.[5]
- *No Private Communication, No Secret Committees.* In SCALES all communication is over a public bulletin board. Also, there are no secret committees available at the start of the protocol (which could have enabled primitives like target-anonymous channels).
- *No Honest Majority.* In practice, honest majority assumption presents several difficulties. For one, sophisticated (rational) players may not carry out expensive computations unless they are likely to be caught (as evidenced by the concerns that have emerged in the context of blockchain miners). Secondly, for an honest majority in the universe of players to be reflected (allowing for a margin) in a committee with all but negligible probability, the committee will have to be quite large. As such, SCALES model requires only one honest server participating in (each phase of) the computation.
- *Constant Number of Rounds.* Another attractive feature of a SCALES protocol is that the number of rounds in the protocol is constant – i.e., independent of the complexity of the function being evaluated. It is solely determined by the number of servers who participate, which is in turn determined by the need to have at least one honest server participating in the computation.

Ephemeral Servers and Non-Ephemeral Clients. The motivation for ephemeral servers – originally expounded in the development of the YOSO (You Only Speak Once) model [GHK+21] – is security in the face of adaptive corruption. Specifically, if all the (relatively small number of) servers in an MPC protocol execution become known to the adversary, it could adaptively target and corrupt *all* of them, making any form of security impossible. An innovative solution proposed in the YOSO model is to use ephemeral parties [GHK+21]. The idea is that these parties will remain publicly unknown – and protected from targeted corruption – until they have "spoken;" however, by the time they have spoken,

[5] An alternate expansion of SCALES could be "Small Clients and *Linear-complexity* Ephemeral Servers," to emphasize the fact that the servers are not arbitrarily complex.

they would have erased their state, and are no longer attractive targets for adaptive corruption.

Unfortunately, requiring that all parties be ephemeral leads to major efficiency bottlenecks. In particular, protocols in the YOSO model rely on several instances of expensive *target anonymous channels* for every round of communication. SCALES model introduced a practically meaningful relaxation of the YOSO requirement, to allow all the communication in the protocol to be directly over a simple bulletin board. For this, SCALES distinguishes MPC servers from MPC clients (input providers). Typically, from an application's point of view, the clients are not anonymous, and are already exposed to targeted corruption attempts. Hence the SCALES model does not require the clients to speak only once. However, it will be required that corrupting a client affects only its own input, and does not compromise the security of the other clients.

With this relaxation, protocols that are fundamentally different from (and much faster than) YOSO protocols can be used to achieve SCALES. In particular, [AHKP22] constructs concretely efficient *semi-honest* secure protocols based on rerandomizable circuit garbling.

SCALES against malicious adversaries. While [AHKP22] points out that in principle, generic NIZK can be used to convert their garbled circuits based SCALES protocol to be secure against active corruption, this is not concretely efficient and is far from practical due to the complexity of the statements to be proven. On the other hand, it is not evident how to adapt techniques available for using garbled circuits against active corruption (without using generic NIZK) can be exploited, as they require more interaction than is allowed in the SCALES model.

To overcome this challenge, our starting point is the observation that the garbled circuits used in [AHKP22] are "rerandomizable" and we can exploit this to obtain a fairly efficient 3-round ZK proof (in the CRS model) for correctness of garbling (and rerandomizing). We go on to develop techniques for efficiently incorporating this protocol into a SCALES protocol. In the resulting protocol, not only do we not require an honest majority, but also we ensure that any corruption by a server is publicly detectable. Since this allows a larger application to impose penalties to those who deviate from the protocol, it is arguably a pragmatic approach to guaranteeing output delivery, instead of assuming an honest majority.

1.1 Summary of Our Contributions and Protocol Highlights

Our main contribution is the following:

> *We present the first maliciously secure protocol with ephemeral servers and SCALES-like efficiency, that is resilient to adaptive corruptions of a majority of the computing parties. Further, in this protocol, any deviation from correct execution can be publicly identified.*

Here by SCALES-like efficiency, we mean that the total communication and computation in the protocol scales linearly with size of the computation, while that of each input provider scales only proportional to its own input.

We obtain our result in the SCALES model, which was originally introduced in [AHKP22], and is extended to the setting of active corruption here. The protocol is UC-secure with only a CRS and a bulletin-board as setups, under a fairly general adaptive corruption model. (See later for more details.) As part of achieving this result, we build several components which are potentially of independent interest:

1. A new ZK proof of knowledge for *proving the correctness of garbling schemes* based on rerandomizable garbling schemes.
2. New primitives – *updatable OT, verifiable updatable OT*, and *distributed committed-index OT* – and efficient instantiations for them.

Zero-Knowledge Proof for Correctness of Garbling. As mentioned above, as a contribution of independent interest, we construct an efficient zero-knowledge proof of knowledge for proving correctness of garbling a circuit *and* the use of the corresponding input wire labels as sender inputs to OT. This can be used in a 2PC setting to get a general garbling-based protocol with a poly-logarithmic multiplicative overhead in the security parameter over the semi-honest communication and computation complexity. It prevents selective abort attacks and implicitly removes the need for the garbler and evaluator input consistency checks required in existing cut-and-choose techniques for maliciously secure garbling. We construct this ZK proof from a Σ-protocol that uses the (rerandomizable) garbling scheme in a black-box manner. If desired, this protocol can be converted into an NIZK protocol in the random oracle model. (We remind the reader that our SCALES protocol does not rely on random oracles.)

Protocol Structure. Our malicious SCALES protocol is in the CRS model and operates over a bulletin board, in three phases: function computation phase, verification phase, and decoding phase. In each phase, each client posts an initial message and then, one by one, each ephemeral server participating in that phase posts a single message. While all the involved servers are ephemeral, each client speaks thrice. Our protocol computes arbitrary functions with security with abort in the presence of an adversary that can adaptively corrupt all-but-one of the clients and all-but-one of the servers in each phase. Further, as all communication in SCALES is over the bulletin-board, the correctness of the complete transcript is publicly verifiable, and any deviation publicly identified. While technically guaranteed output delivery is not achievable in our all-but-one corruption setting, this identification guarantee is practically almost as good, since penalties can be levied on parties found deviating. The overall cost this protocol incurs, over the semi-honest SCALES protocol of [AHKP22], is a polylogarithmic multiplicative overhead in the security parameter in both computation and communication. Further, the soundness parameter is tunable, so that with a constant soundness error, it can yield a *covertly secure* SCALES protocol with a *constant* multiplicative communication overhead over the semi-honest protocol.

1.2 Related Work

Related to our work are both YOSO and SCALES models, as well as other MPC schemes with specialized communication patterns.

SCALES MPC. The most relevant prior work is SCALES [AHKP22]. We build on their foundations and lift their semi-honest protocol into the malicious setting. Doing so efficiently is technically challenging. Clearly, simply applying a generic NIZK proof to every message works and preserves the SCALES communication pattern. However, it is also concretely extremely inefficient.

On the other hand, standard efficient malicious-GC techniques, such as cut-and-choose and authenticated garbling, seem to fail to work in our constrained setting for several reasons. Firstly, standard GC protocols are designed for 2PC. Known multi-party extensions, such as BMR [BMR90], require multiple rounds between parties, violating SCALES requirements. Similarly, cut-and-choose too does not naturally fit with the SCALES-like communication pattern: communication-efficient cut-and-choose [Lin13, HKE13] requires several rounds of interaction to achieve 2^{-s} soundness error by communicating s GCs. We show that SCALES malicious security *could* be achieved in the style of cut-and-choose, using the distributed committed OT primitive we introduce (see Section 1.4), albeit with a lower efficiency (achieving only $2^{-0.31s}$ soundness error using s GCs).

Our main protocol avoids this penalty by using an altogether different approach that exploits the rerandomizability of the underlying GCs (achieving 2^{-s} soundness error with s GCs).

YOSO. Recall that one of our goals is to outsource MPC to a relatively small number of servers, while remaining secure against an active and adaptive adversary who can potentially corrupt every server that it finds out to be part of the protocol. This can be achieved by the recently introduced YOSO framework [GHK+21], where the parties only speak once, and the computation is evolved using a sequence of elected committees.

However, [GHK+21] and related works incur impractically large costs on various fronts. For one, these protocols use several large (hidden) committees, with expensive "encryption to the future" channels, where each committee needs to have an honest majority. Supporting encryption to the future turns out to be quite expensive. Secondly, for security against active corruption, these works use generic NIZK protocols in proportion to the size of the original computation. Thirdly, these protocols have a round complexity proportional to the depth of the computation, inherited from the underlying protocols and this is a major overhead as each round (involving a committee selection) is highly expensive. [6] Finally,

[6] While [KRY22] offers YOSO protocols which are constant round, these are meant to be theoretical feasibility results (no concrete cost analysis is available) that build on expensive base protocols, and further use their next message functions in a non-black-box manner.

these protocols suffer inherent inefficiencies when Boolean logic is involved, say, in the form of conditionals (as opposed to purely arithmetic computation). The SCALES protocol in this work avoids each of these issues. This is discussed in more detail in Section 1.3.

Alternate MPC Models. Goyal et al. [GMPS21] explore feasibility of blockchain-assisted MPC: input parties put the input and garblings of an MPC protocol's next-message function in conditional storage and retrieval systems (CSaRs). These are posted on the blockchain that executes this MPC protocol by processing the garbled next-message functions. It ensures no interaction between the parties, at the cost of having them perform protocol-size-dependent computation and engaging a powerful (and expensive) blockchain setup. Two-round MPC [GGHR14, GS18, BJKL21, BGSZ22] involves input parties posting two rounds of messages to a bulletin board, based on which the output can be publicly computed. However, the input parties incur communication and computation costs proportional to the circuit size of an underlying MPC protocol. The Fluid MPC model [CGG+21] allows parties to dynamically join and leave the protocol computation; however, the identities of future parties joining the protocol need to be known in advance. It also relies on the assumption that the adversary can corrupt only a minority of the servers in each committee. A recent work [RS22] extends Fluid MPC to the dishonest majority setting, though it still does not meet the YOSO requirement.

Other Techniques for Malicious Security. Our custom zero-knowledge proof, exploiting the explainability of RGS and UOT, avoids several disadvantages of alternate generic approaches. There are a few significant approaches to ensuring correctness of garbling that we need to compare with: using zero-knowledge proofs [HIMV19, ASH+20], cut-and-choose techniques [LP09], and authenticated garbling [YWZ20, IKO+11].

Firstly, using a generic NIZK proof system comes with the disadvantage that it is non-black-box in the underlying primitives and incurs a high computation overhead. A circuit for verifying the statement that a GC is a correct garbling of a function (or a correct rerandomization of another GC) and that the inputs to UOT are consistent with the GC, is much larger than the GC itself. In particular, this circuit incorporates the underlying cryptographic primitives. For example, the only known instantiation of RGS from [AHKP22] uses [BHHO08] as the underlying encryption scheme, and involves $O(\kappa^2)$ group exponentiations per gate. Further, this circuit needs to be encoded as low-level algebraic circuits, R1CS programs, etc., before the proofs are created, resulting in a prover complexity that is super-linear in the GC size and concretely very high. Even in the Random Oracle (RO) model (which we do not resort to), wherein ZK-SNARKs can virtually eliminate the communication and verifier complexity, the high prover complexity remains. In contrast, our proof operates in the standard model (without RO) and is black-box in the RGS, and the computational (and communication) overhead in the final SCALES protocol compared to the semi-honest version in [AHKP22] is simply a multiplicative factor of s, for 2^{-s} security error.

Another alternative, which is black-box in the underlying garbling scheme is to use cut-and-choose techniques. Although cut-and-choose is black-box in the underlying cryptographic primitive, they are not directly compatible with the SCALES model. This could be addressed using the DCOT primitive we develop, in the same way we convert our Σ protocols into the SCALES model. Even so, they incur a multiplicative $3.2s$ factor for achieving 2^{-s} soundness error while our techniques are simpler and incur only a factor of s for the same soundness.

Finally, authenticated garbling techniques cannot be used in SCALES protocols due to the need for multiple rounds of communication.

1.3 Comparison with YOSO Protocols

While YOSO and SCALES models share the key feature of ephemeral servers, they remain incomparable to each other on certain other aspects. While YOSO protocols allow even input clients to speak only once, in many practical settings, SCALES protocols can offer much better efficiency and practicality. We elaborate on this below.

Target Anonymous Channels. Typical protocols in the YOSO model involve secret committees, and require expensive point-to-point target anonymous channels for every round of communication, between every pair of "current" committee member and a "future" committee member. This is exacerbated by the fact that the most efficient YOSO protocols have round complexity proportional to the depth of the circuits. Further, this affects the flexibility of these protocols by requiring that the target anonymous channels are setup sufficiently in advance.

SCALES protocols are fundamentally different in that they do not involve private communication to the servers to be selected in the future, and there is no need to setup expensive target anonymous channels. Indeed, the SCALES model allows servers to join the protocol truly on the fly, without any parties being aware of a server's existence prior to it speaking.

Rounds as a resource. A significant advantage of our work compared to the results in the YOSO model is the round complexity and the corresponding concrete costs. While one can instantiate YOSO with constant-round protocols (like the BMR protocol), that incurs heavy computational costs due to generic ZK proofs; indeed, [KRY22] proposes several such protocols as a proof of concept, focusing on providing end-to-end proofs. But these protocols build on expensive base protocols, and further use their next message functions in a non-black-box manner. On the other hand, the more practical instantiations of YOSO which use underlying protocols whose round complexity scales with the function's multiplicative depth (GMW, CDN, etc.) do allow for (relatively) efficient arithmetic computations, but incurs round costs.

In the setting of YOSO and SCALES, where players self-select and communicate over a bulletin board (blockchain), the round latency is daunting, ranging from several seconds to minutes. To highlight the relative cost of latency even in high-speed networks, we point out that recent experiments reported in [YPHK23]

show $\approx 1600\times$ improvement by switching from a GMW-based multi-round solution to a constant-round GC-based protocol, with the primary factor behind the speed up being network latency. Instantiating YOSO with multi-round protocols will incur similar penalties.

Computational costs. In terms of total computational costs, for a Boolean circuit of size s and a security parameter κ executed by n servers, our total computational cost is $O(ns\kappa^3)$. In YOSO_{CDN}, with a YOSO committee size N (a function of the security parameter), *each role* requires $O(N^2)$ computation, plus a generic non-black-box NIZK for proving correctness of various operations (decryption, encryption, threshold key reconstruction, resharing, and partial decryption) in a Paillier encryption scheme.

Boolean vs. Arithmetic Computation. Our protocol is geared towards secure evaluation of boolean circuits, rather than arithmetic computation. While arithmetic circuits can offer better efficiency in some applications, this is not generically the case. This is because control flow, such as branching, is conditioned on Boolean values, conversion to which is expensive when an arithmetic circuit is used. Programs where arithmetic operations are interspersed with conditionals (e.g., GCD computation and, more generally, programs best executed as CPU-emulation) are more efficiently represented as boolean circuits rather than arithmetic circuits, as has also been observed in the context of recent high-performing interactive ZK systems [YHKD22].

Guaranteed Output Delivery vs. Identifiability. Many of the YOSO protocols support guaranteed output delivery, whereas our SCALES protocol settles for identifiability of corrupt parties. However, a closer look reveals that the SCALES approach is arguably more practical. Firstly, guaranteed output delivery requires an honest majority assumption, which not only limits its applicability, but also, even when it applies, severely affects efficiency compared to the SCALES requirement that only one server needs to be honest in each phase.[7] Secondly, when MPC is deployed in a larger setting, often identifiability can already be used to incentivize honest behavior (e.g., via smart contracts that penalize parties identified as corrupt) and thereby guarantee output delivery.

1.4 Technical Overview

We outline the ideas and building blocks used for our maliciously secure SCALES protocol. Similarly to the semi-honest protocol of [AHKP22], our construction is based on rerandomizable garbling. To efficiently lift semi-honest to malicious security, we design zero-knowledge (ZK)-friendly protocols while preserving the SCALES communication pattern. We structure the overview as follows

[7] E.g., when 1/4 of the pool is corrupted, selecting 20 parties at random from the pool drives down the probability of not having an honest party to 2^{-40}; even with over a 100 parties, the probability of not having an honest majority stays above 2^{-30}.

(cf Figure 1): we review our starting point, semi-honest construction [AHKP22]. We then highlight several lower-level obstacles to be resolved for getting malicious security. For each, we introduce a corresponding functionality, discuss technical issues, and sketch our approach to efficient instantiation. Finally, we discuss how to put it all together.

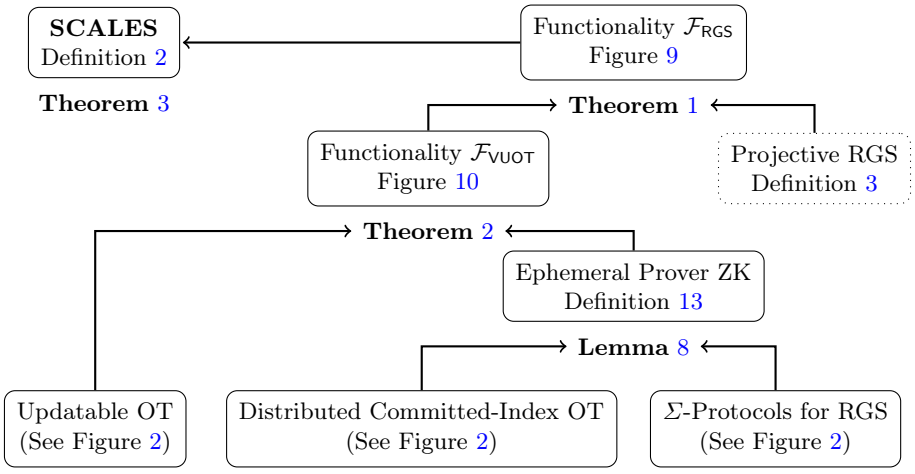

Fig. 1 Overview of our main construction. Dotted box indicates a primitive from prior work. We introduce and build the three lower boxes: Updatable OT, Distributed Committed-Index OT, and Σ-protocols for correct function encoding. This Figure 1 is the roadmap for obtaining our maliciously-secure SCALES protocol of Figure 26.

The $\mathcal{F}_{\mathsf{RGS}}$ Abstraction. Our starting point is the semi-honest SCALES protocol of [AHKP22] which uses Rerandomizable Garbling Schemes (RGS). We introduce a functionality $\mathcal{F}_{\mathsf{RGS}}$ (Figure 9) that captures the core of this protocol, and then UC-securely implement it (as opposed to only with semi-honest security).

We briefly recall the structure of the protocol in [AHKP22]: A server creates a garbled circuit (GC), which is then rerandomized by a sequence of servers, one at a time. The final evaluation is carried out on the last GC. The input labels of this last GC are published by the input clients: before the GCs are constructed, they commit to their inputs via the first message of a 2-round OT protocol, and using the second-round messages in the OT protocol posted along with each rerandomized GC, each client can keep track of the input labels of the latest GC.

$\mathcal{F}_{\mathsf{RGS}}$ carries out the generation and rerandomization of the garbled circuits, and publishes the correct labels of the final GC. It is a reactive functionality, which first receives the function inputs from the clients. Then it accepts the randomness used to create a GC of f from the first server, constructs this GC, and makes it available to all parties. $\mathcal{F}_{\mathsf{RGS}}$ also gives each client the active label for its own input. Next, it accepts randomness from each successive server to

rerandomize the prior GC, makes the new GC available to all parties, and each client receives the active labels for its input. After the final rerandomization, all active labels corresponding to the final garbling are given to all parties.

To realize the $\mathcal{F}_{\mathsf{RGS}}$ functionality with malicious security, we will need to enhance the RGS-based construction of [AHKP22] with a zero-knowledge proof facility compatible with the SCALES communication pattern.

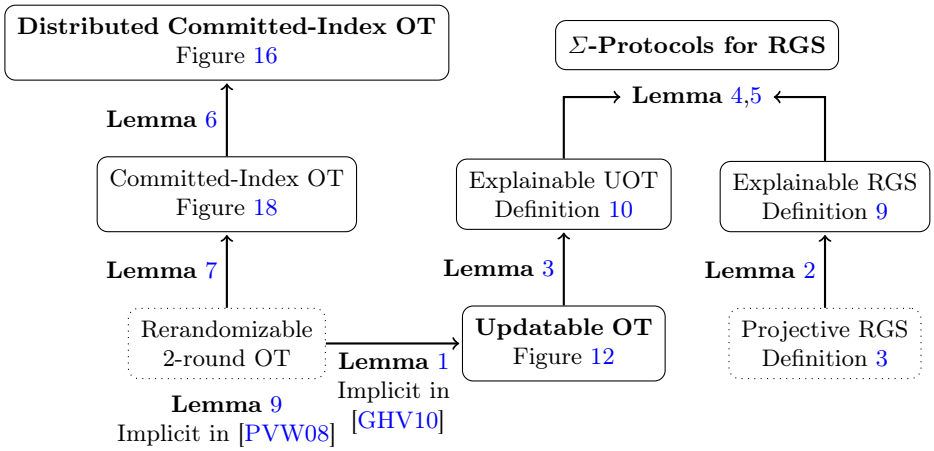

Fig. 2 Overview of constructing the building blocks. Dotted boxes indicate primitives from prior work. This shows the roadmap for obtaining our 3 main building-blocks shown in Figure 1 from Projective RGS and a 2-round OT protocol.

Updatable Oblivious Transfer (UOT). A key ingredient in implementing the $\mathcal{F}_{\mathsf{RGS}}$ functionality is a variant of the OT functionality called $\mathcal{F}_{\mathsf{UOT}}$, which involves one receiver, one sender and multiple *updaters*. This functionality implements a sequence of OTs with the same receiver and the same choice bit. While the sender gets to input a set of strings (s_0^0, s_1^0) to the functionality (just like in OT), and the j^{th} updater's input to OT is defined as $(s_0^j, s_1^j) = (\sigma_j(s_0^{j-1}), \sigma_j(s_1^{j-1}))$, where σ_j is an update function that the updater can pick from a suitably defined function class (which for us will be related to the rerandomization admitted by the RGS). The receiver learns s_b^j for all j, where b is its input. Further, to facilitate public evaluation in the overall protocol, $\mathcal{F}_{\mathsf{UOT}}$ *publishes* the last of these strings, s_b^d, where d is the total number of updates.

A UOT protocol is a UC-secure realization of $\mathcal{F}_{\mathsf{UOT}}$ with additional structural restrictions to fit into the SCALES model. In Section 5.1, we present such a protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid based on the 2-round OT protocol of [PVW08]. This resembles the construction of 'rerandomizable OT' in [GHV10]. Further, to implement the final step of publishing the last received string (denoted as s_b^d above), we rely on a NIZK protocol. As the statement proven using this NIZK

protocol is small (depends only on one instance of the OT protocol per string received), we rely on a generic NIZK PoK protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid [GHV10].

Verifiable Updatable OT (VUOT). To ensure correct garbling/rerandomizing and that the sender's and updaters' inputs to the UOT protocol are derived during the garbling and rerandomizing process, we abstract *verifiable updatable oblivious transfer* as a functionality $\mathcal{F}_{\mathsf{VUOT}}$. This forms a convenient abstraction from which the $\mathcal{F}_{\mathsf{RGS}}$ functionality can be readily realized.

The $\mathcal{F}_{\mathsf{VUOT}}$ functionality is parameterized by two relations and it internally incorporates a UOT functionality. The relations tie the inputs of the sender or the updater with a public piece of information that is maintained across multiple rounds of updates. In our case, this will be a rerandomizable garbled circuit (GC). The first relation will be used to check if the function is garbled correctly and if the sender's inputs in UOT are the input labels formed in this process; the second relation will be used to check if the GC instance is a correct rerandomization of an existing garbling and if the updater's UOT inputs are indeed those transformations created in this process. $\mathcal{F}_{\mathsf{VUOT}}$ executes as follows: For a function f with m input bits, first each client (receiver in $\mathcal{F}_{\mathsf{VUOT}}$) sends its input bits to the functionality. Then the first encoder (sender in $\mathcal{F}_{\mathsf{VUOT}}$) garbles f, and sends this garbling, the randomness used to garble, and the complete set of labels for all input wires to the functionality. The functionality also uses this information to verify the first relation; then each client receives its active label and the garbling is publicly posted. For each successive rerandomization of the GC, $\mathcal{F}_{\mathsf{VUOT}}$ is used in a similar fashion, but with the second relation verifying that updated OT secrets are consistent with the rerandomization of the GC.

In addition to the UOT protocol, realizing $\mathcal{F}_{\mathsf{VUOT}}$ would require a mechanism to verify the validity of the two internal relations. This needs to be done in a way that preserves the SCALES communication pattern: having a stateless prover "speak once." We abstract this mechanism as an Ephemeral Prover Zero-Knowledge (EPZK) proof protocol and discuss its building blocks below.

Sigma Protocols for RGS. As a starting point, we construct efficient Σ-protocols for proving that garbling and rerandomizing in RGS were performed correctly. Our basic idea for proving garbling correctness is as follows (proving rerandomization is similar): the Prover (RGS garbler) generates and publishes the garbling F. Simultaneously, it rerandomizes F and publishes this regarbling F_{proof}. Based on a Boolean challenge from the verifier, it opens randomness showing either correct regarbling $F \mapsto F_{\mathsf{proof}}$, or an *explanation* of F_{proof} as a correct garbling of f directly from some randomness. Crucially, the original garbling F remains unopened and suitable for secure evaluation. Soundness is based on the fact that if F is not correctly constructed, the prover cannot succeed in both the cases and will be caught with probability 1/2 (this can be amplified by repetition). The actual proof additionally needs to assure correctness of the UOT execution with inputs that are consistent with the (re)garbling. Details can be found in Section 5.2.

We introduce *explainability* as a property of rerandomizable objects and define a special class of RGS (including the construction in [AHKP22]) that satisfies this. Informally, explainability is a property by which a rerandomized garbling of a function f can be explained as a fresh garbling of f. This is done by defining an algorithm that takes both the garbling and rerandomizing randomnesses and composes them to form randomness that is indistinguishable from fresh garbling randomness. An explainable UOT protocol is described similarly: for an updater's message in a UOT protocol, the sender's and updater's randomness can be composed to form randomness that explains this as a fresh message by a sender using the receiver's first message and the rerandomized labels as inputs. Details of both these definitions can be found in Section 5.2. These facilitate the construction of the Σ-protocols as outlined above.

Why not a generic proof? Our proof, exploiting the explainability of RGS and UOT, avoids several disadvantages of alternate generic approaches – like using generic zero-knowledge proofs [HIMV19, ASH+20], cut-and-choose techniques [LP09], and authenticated garbling [YWZ20, IKO+11]. See Section 1.2 for a detailed discussion.

Ephemeral Prover Zero-Knowledge Protocols. The Σ protocols above cannot be directly used in a SCALES protocol: they involve the prover (in our case, a server) speaking twice, retaining state in between. What we require instead is a proof system in which the prover speaks only once. This *does not* need to be a NIZK, as the verifier is allowed to send a message prior to the prover's message. We call such a proof system an Ephemeral Prover ZK (EPZK) proof system.

Following a similar outline as that in [JKKR17], we observe that we can eliminate the first round message in a Σ protocol, using a 2-round UC-secure OT protocol, as follows: Recall that in a Σ protocol, the prover sends a message a, the verifier responds with a random challenge $b \in \{0, 1\}$, and then the prover replies with a message c. After computing a, the prover can compute two responses, c_0 and c_1, so that c_b is the response on challenge b. In the new protocol, the verifier starts by sending the first OT message playing the role of a receiver with choice bit b. The prover computes (a, c_0, c_1) and responds with (a, m) where m is the second message in the OT protocol corresponding to sender's inputs (c_0, c_1). The verifier can recover c_b from OT, and complete the verification of the Σ protocol.

Using a 2-round UC-secure OT protocol (in the CRS model) above, and running several (poly-logarithmic in the security parameter many) parallel executions of this protocol, we obtain a UC-secure protocol for the ZK functionality, as can be modularly analyzed by considering the protocol in the OT-hybrid model.

While the above yields an EPZK proof system, we will in fact be interested in ZK proofs given by an a priori unknown number of provers, and involving multiple verifying clients (namely the input clients) so that unless all the clients are corrupt, the functionality guarantees the correctness of all the claims being proved. Also, we will avoid the need for separate proofs given by each server to each client, using a version of OT that we discuss next.

Distributed Committed-Index OT (DCOT). As mentioned, a Σ protocol can be turned into a UC-secure ZK proof in the OT-hybrid model, and further by composing with a 2-round OT protocol, we obtain a 2-round ZK proof protocol. However, since the SCALES model does not insist that the number of servers (who will need to give proofs to the clients) is *a priori* fixed, we need an OT functionality in which many senders can take part for the same choice bit sent by a receiver. Unlike in OT, this functionality makes the choice bit and the chosen string publicly available at the end. This is captured as "Committed Index OT," and can be readily implemented by having all the senders use the same first message from the receiver, in a 2-round OT protocol (see Section 5.3).

Another significant efficiency issue stems from the need to give the proof to multiple verifiers. While repeating the proof separately works, it results in one of the most expensive parts of the overall protocol being replicated by a factor equal to the number of input parties. To avoid this, we define and implement a new functionality called DCOT which allows for a single proof that remains sound as long as at least one of the verifying clients is honest.

The DCOT functionality is similar to COT, except that there are multiple receivers and the choice bit is uniformly random even if an adversary corrupts all-but-one of the receivers. We give a protocol for DCOT in the COT-hybrid model, which, when composed with a 2-round COT protocol, yields a 2-round protocol. In this protocol, each chooser C_i samples a bit b_i as its input to a session of the COT protocol (with all the senders). Then each sender samples two random masks k_0 and k_1 and publishes $(s_0 \oplus k_0, s_1 \oplus k_1)$ where (s_0, s_1) is its input. Let $\Delta = k_0 \oplus k_1$. Corresponding to each of the m choosers C_i, it samples a random string L_0^i under the constraint that the XOR of these strings is k_0. It sets each string $L_1^i = L_0^i \oplus \Delta$. Note that for $b = \bigoplus_{i\in[m]} b_i$, it holds that $k_b = \bigoplus_{i\in[m]} L_{b_i}^i$. Then the sender uses (L_0^i, L_1^i) as its inputs for the COT instance with the chooser C_i, for each of the m choosers C_i. The chosen strings $L_{b_i}^i$ are made publicly available, from which k_b can be publicly computed.

Maliciously Secure SCALES Protocol. Tracing back the above discussion, the Σ protocols for RGS can be turned into EPZK protocols for the same relations. This can then be used to obtain a UC-secure protocol for $\mathcal{F}_{\mathsf{VUOT}}$, and then for $\mathcal{F}_{\mathsf{RGS}}$ for evaluating any function, and this can then be used to obtain a UC-secure SCALES protocol. In Figure 3, we summarize this protocol, omitting the abstractions of $\mathcal{F}_{\mathsf{VUOT}}$ and $\mathcal{F}_{\mathsf{RGS}}$ (which are helpful for a modular proof). The protocol has three phases with each requiring at least one honest participating server. This protocol can also be relaxed to provide covert security by reducing the number of parallel instances of the Σ-protocol executed in EPZK to one. We refer the reader to Appendix C for details on the covertly secure protocol.

Maliciously Secure SCALES Protocol in the Random Oracle Model. A simpler and more efficient SCALES protocol can be achieved in the Random Oracle Model (ROM). This protocol requires only two client messages: one at the beginning of the encoding phase and the decoding phase respectively. In the protocol in

Maliciously Secure SCALES Protocol

1. **Encoding Phase:**
   - Each client creates the first message of the UOT protocol as a receiver, using its input bit as the choice bit. **It also creates the first message of the EPZK protocol.** These are posted on the bulletin-board.
   - The first encoder garbles an RGS for the function f. For each input wire, it uses the two labels to create the senders' message of the UOT protocol. **It creates the EPZK server message for proof of correct garbling.** The garbled circuit, **the EPZK message,** and the set of UOT second messages are posted onto the bulletin-board.
   - Every successive encoder reads the previous GC and UOT message set. It rerandomizes the GC and updates the UOT messages so the labels match the new garbling. **It creates the EPZK server message for proof of correct rerandomizing.** The new garbled circuit, **EPZK message,** and set of UOT messages are posted on the bulletin-board.
2. **Verification Phase:**
   - **Each client posts the second EPZK message.**
   - **For multiple rounds, a server verifies the EPZK transcript correctness and posts a bit indicating correct verification.**
3. **Decoding Phase:**
   - **If all verifiers accept,** each client decodes the last UOT message and posts the final label on the bulletin-board. **This is accompanied by a CRS-based NIZK indicating honest decoding.**
   - For multiple rounds, a decoder reads and **verifies each client's last message,** uses the labels to evaluate the last GC, and posts the result.
   - **If all decoders post the same output,** this is accepted by all parties.

Fig. 3 An Informal Description of a Maliciously Secure SCALES Protocol with three phases – an encoding phase, a verification phase, and a decoding phase. Each phase requires at least one honest participating server. The elements of the protocol that are relevant to security against active corruption are shown bold and in color.

Figure 3, when all the participants have access to a public random oracle RO, all encoder ZK-proofs in the EPZK protocol can be made NIZK proofs. This is done by taking all the parallely invoked Σ-protocols for correct garbling (resp. rerandomizing) for an encoder and applying the Fiat-Shamir transform on them collectively. This eliminates the need for the DCOT protocol completely and hence the clients' involvement in EPZK. The protocol is summarized in Figure 4.

2 The SCALES Model

We describe the SCALES model and define a SCALES protocol family for a function family $\mathcal{F}$ where the size of each function representation is parameterized by k. The model involves servers, clients and a public append-only bulletin-board.

Maliciously Secure SCALES Protocol in the ROM

1. **Encoding Phase:**
   - Each client creates the UOT protocol first message as a receiver, using its input bit as the choice bit and posts it on the bulletin-board.
   - The first encoder garbles an RGS for the function f. For each input wire, it uses the two labels to create the UOT protocol senders' message. **It uses the RO to create the EPZK message (the NIZK proof) for correct garbling.** The GC, **NIZK proof,** and set of UOT second messages are posted onto the bulletin-board.
   - Every successive encoder reads the previous garbling and UOT message set posted. It rerandomizes the GC and updates the UOT messages so that the labels match the new garbling. **It uses the RO to create the NIZK proof for correct rerandomizing.** The new GC, **NIZK proof,** and set of UOT messages are posted on the bulletin-board.
2. **Verification Phase:**
   - **For multiple rounds, a server uses the RO to verify the NIZK proof for all encoders and posts a bit.**
3. **Decoding Phase:**
   - **If all verifiers accept,** each client decodes the last UOT message and posts the final label on the bulletin-board, **along with an RO-based NIZK for honest decoding.**
   - For multiple rounds, a decoder reads and **verifies each client's last message,** uses the labels to evaluate the last GC, and posts the result.
   - **If all decoders post the same output,** this is accepted by all parties.

Fig. 4 An Informal Description of a Maliciously Secure SCALES Protocol in the Random Oracle Model with three phases – an encoding phase, a verification phase, and a decoding phase. Each phase requires at least one honest participating server. The elements of the protocol that are relevant to security against active corruption are shown bold and in color.

All the communication between the participating parties take place over this bulletin-board.

Definition 1. *Let $\mathcal{F} = \{f_k | k \in \mathbb{N}\}$ be a function family with members $f_k : \{0,1\}^{m(k)} \to \{0,1\}^{l(k)}$ that have circuit representation $\mathbf{C}_k$. Let $x = \{x_i\}_{i\in[m]}$ be the function input. For $c \in \mathbb{N}$, a protocol family $\{\Pi_k | k \in \mathbb{N}\}$ is said to be a c-phase **SCALES protocol family** for $\mathcal{F}$ if, for each k, the protocol Π_k has the following structure.*

- ***Participants:***
  - *Clients $\mathcal{C} = \{C_i\}_{i\in[m]}$ where C_i has input bit x_i.*
  - *Ephemeral server pool $\mathcal{S}$ with constant fraction σ of honest servers.*
  - *Bulletin-board $\mathcal{B}$ to which any party can write. Anything written onto $\mathcal{B}$ is visible to all parties.*

– ***Communication Pattern:*** *For each phase $p \in [c]$, let s_p be the number of servers participating in computation in that phase. Each phase p has $s_p + 1$ rounds where,*
  - *In round 0, all clients $C_i \in \mathcal{C}$ simultaneously post a message m_i^p onto $\mathcal{B}$*
  - *$\forall$ round $j \in [s_p]$, a randomly selected server $S_j^p \in \mathcal{S}$ reads $\mathcal{B}$, silently performs computation, erases its state, and posts a message m_j^p onto $\mathcal{B}$*

– ***Complexity:***
  - *Each client $C_i \in \mathcal{C}$ can perform computation proportional to a security parameter κ and $\max_{p\in[c]}(s_p)$, but independent of $|\mathbf{C}_k|$*
  - *Each server S_j^p participating in Π_k can perform $O(|\mathbf{C}_k|) = \text{poly}(k)$ computation*

In Definition 1 note that all participating servers are stateless and the number of times a (stateful) client posts a message is a small constant c. The definition remains silent about an additional privacy property of a SCALES protocol since this would differ depending on the type of adversary being considered. Note that the above definition is a compiler for the circuit representation of any function into a SCALES protocol.

The SCALES protocol constructed in [AHKP22] is secure against semi-honest adversaries and consisted of 2 phases. In the first phase, after each client has posted a message, a set of $s_1 = d$ ephemeral servers $\{E_j\}_{j\in[d]}$ operated as encoders to compute a garbled circuit. In the next phase, the clients' message provided the input labels corresponding to the final garbled circuit. Then, $s_2 = 1$ server is tasked with evaluating the garbled circuit. In this protocol, security holds when the semi-honest adversary corrupts all-but-one participating server in the encoding phase and the decoding server. It can also corrupt, statically or adaptively, up to all the input clients.

Maliciously Secure SCALES. In this work we present a SCALES protocol that is secure in the presence of malicious adversaries. The protocol consists of 3 phases: the first phase consists of a client message followed by a set of ephemeral server messages acting as *encoders*; the second phase consists of a client message followed by a set of servers that take the role of *verifiers*; and the third phase consists of the last client message followed by a set of *decoders*.

Definition 2. *Let $\mathcal{F}_{\mathcal{B}}$ be the Bulletin-Board functionality (Figure 6) and $\mathcal{F}_{\mathsf{CRS}}$ be a Common Reference String functionality (Figure 7). A SCALES protocol family (Definition 1) $\{\Pi_k | k \in \mathbb{N}\}$ for a function family $\{f_k | k \in \mathbb{N}\}$ is said to be a* ***maliciously secure SCALES protocol family*** *if, for each $k \in \mathbb{N}$, Π_k is a UC-secure protocol for the functionality $\mathcal{F}_{\mathsf{SCALES}}^{f_k}$ (Figure 5) in the $(\mathcal{F}_{\mathcal{B}}, \mathcal{F}_{\mathsf{CRS}})$-hybrid against a malicious PPT adversary that is allowed all-but-one adaptive corruption of clients, and, for each phase, all-but-one adaptive corruption with erasures of participating servers.*

3 Preliminaries

We begin by recapping relevant definitions of objects from prior works. We refer the reader to Appendix A for additional preliminaries. We denote by κ a computational security parameter and s a statistical security parameter.

3.1 Rerandomizable Garbling Schemes

We re-state the definition of projective Rerandomizable Garbling Schemes [AHKP22].

Definition 3. *A **Projective Rerandomizable Garbling Scheme** for function family $\mathcal{F}$ with input domain $\{0,1\}^m$, and a leakage function $\phi : \mathcal{F} \to \{0,1\}^*$, is a tuple $\mathsf{GS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ of PPT algorithms satisfying:*

- ***Correctness**: For every $f \in \mathcal{F}$ and input $x \in \{0,1\}^m$,*

$$\begin{aligned}\Pr[y = f(x) \ : (F,e) \leftarrow \mathsf{Gb}(f),\ X = \mathsf{En}(e,x),\ y = \mathsf{Ev}(F,X) \qquad] = 1\\ \Pr[y = f(x) \ : (F,e) \leftarrow \mathsf{Gb}(f), \qquad (F',\pi_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(F),\\ X' = \mathsf{En}(\pi_{\mathsf{En}}(e),x),\ y = \mathsf{Ev}(F',X') \qquad] = 1\end{aligned}$$

- ***Privacy**: $\forall f_0, f_1 \in \mathcal{F}$ s.t. $\phi(f_0) = \phi(f_1)$, and $\forall x_0, x_1 \in \{0,1\}^m$ s.t. $f_0(x_0) = f_1(x_1)$,*

$$\{F_0, X_0\}_{(F_0,e_0)\leftarrow\mathsf{Gb}(f_0), X_0=\mathsf{En}(e_0,x_0)} \overset{c}{\approx} \{F_1, X_1\}_{(F_1,e_1)\leftarrow\mathsf{Gb}(f_1), X_1=\mathsf{En}(e_1,x_1)}$$

- ***Rerand-Privacy**: For the space of random tapes R in Gb, $\forall f \in \mathcal{F}, x \in \{0,1\}^m$,*

$$\{r, F_0, X_0\}_{\substack{r\leftarrow R,\ (F,e)\leftarrow\mathsf{Gb}(f;r),\\ (F_0,\pi_{\mathsf{En}})\leftarrow\mathsf{Rerand}(F),\ X_0=\mathsf{En}(\pi_{\mathsf{En}}(e),x)}} \overset{c}{\approx} \{r, F_1, X_1\}_{\substack{r\leftarrow R,(F_1,e_1)\leftarrow\mathsf{Gb}(f),\\ X_1=\mathsf{En}(e_1,x)}}$$

- ***Projective Encoding**: The function $\mathsf{En} : \mathcal{E} \times \{0,1\}^m \to \mathcal{Z}^m$ is projective:*
  - *$\forall x = \{x_i\}_{i\in[m]}, x' = \{x'_i\}_{i\in[m]} \in \{0,1\}^m$, $\forall e \in \mathcal{E}$, $\{L_i\}_{i\in[m]} = \mathsf{En}(e,x)$ and $\{L_i^{'}\}_{i\in[m]} = \mathsf{En}(e,x')$ such that $\forall i \in [m]$, if $x_i = x'_i$ then $L_i = L_i^{'}$.*
  - *$\pi_{\mathsf{En}} = \{\sigma_i\}_{i\in[m]}$ produced by Rerand is such that $\forall x \in \{0,1\}^m$, $\forall e \in \mathcal{E}$, $\{L_i\}_{i\in[m]} = \mathsf{En}(e,x)$ and $\{L_i^{'}\}_{i\in[m]} = \mathsf{En}(\pi_{\mathsf{En}}(e),x)$ where $\sigma_i(L_i) = L_i^{'}$.*

[AHKP22] contains a construction for projective RGS based on garbled circuits for which we refer the reader to Appendix A (Construction 1). We also instantiate a variant of projective RGS with this in our final SCALES protocol.

3.2 Oblivious Transfer

Oblivious Transfer (OT) is a 2-party functionality $\mathcal{F}_{\mathsf{OT}}$ between a sender S and a receiver R. The sender S has inputs $a_0, a_1 \in \{0,1\}$, and the receiver R has $b \in \{0,1\}$. $\mathcal{F}_{\mathsf{OT}}$ takes these inputs and gives a_b to R. We require an OT protocol with additional features as described in Definition 4.

Definition 4. *Let $\mathcal{F}_{\mathsf{CRS}}$ (Figure 7) be a Common Reference String functionality. A tuple of PPT functions $\mathbf{OT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_3)$ is a 2-round maliciously secure bit-OT scheme with reusable first message and rerandomizable second message if the following holds,*

Functionality $\mathcal{F}^{f}_{\mathsf{SCALES}}$

Let $f : \{0,1\}^m \to \{0,1\}^l$ be a function. Let $\mathcal{A}$ be a PPT ideal-world adversary. Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the clients. The functionality $\mathcal{F}^{f}_{\mathsf{SCALES}}$ works as follows:

- $\mathcal{F}^{f}_{\mathsf{SCALES}}$ accepts an input bit x_i from each $C_i \in \mathcal{C}$
- $\mathcal{F}^{f}_{\mathsf{SCALES}}$ computes $f(x)$ and gives $f(x)$ to $\mathcal{A}$
- if $\mathcal{A}$ sends OK to $\mathcal{F}^{f}_{\mathsf{SCALES}}$, it sends $f(x)$ to $\mathcal{C}$

Fig. 5 SCALES Functionality

Bulletin-Board $\mathcal{F}_{\mathcal{B}}$

Let $\mathcal{P}$ be the set of all parties. Let $\mathsf{publish}$ denote the action of the functionality sending a message to all parties. The bulletin-board $\mathcal{F}_{\mathcal{B}}$ works as follows:

- $\mathcal{F}_{\mathcal{B}}$ initializes its transcript to $\tau = \{\}$ and $c = 0$
- Each time $\mathcal{F}_{\mathcal{B}}$ receives message m from party P_i,
  - $\mathcal{F}_{\mathcal{B}}$ sets $c = c + 1$ and $\tau[c] = (m, P_i)$
  - $\mathcal{F}_{\mathcal{B}}$ $\mathsf{publish}$ c
- If a party P_i sends read to $\mathcal{F}_{\mathcal{B}}$, the functionality sends τ to P_i
- For $c' \in [c]$, if a party P_i sends (read, c') to $\mathcal{F}_{\mathcal{B}}$, the functionality sends $\tau[c']$ to P_i

Fig. 6 Bulletin-Board

Common Reference String $\mathcal{F}^{\mathsf{D}}_{\mathsf{CRS}}$

Let $\mathcal{P}$ be the set of all parties. Let D denote a distribution over strings $\{0,1\}^*$ (implicitly parameterized by a security parameter κ). The functionality $\mathcal{F}^{\mathsf{D}}_{\mathsf{CRS}}$ works as follows:

- Initialization:
  - Sample $s \leftarrow \mathsf{D}$
- If a party P_i sends read to $\mathcal{F}^{\mathsf{D}}_{\mathsf{CRS}}$, the functionality sends s to P_i

Fig. 7 Common Reference String

1. ***Reusable first message****: for a receiver R with input $b \in \{0,1\}$ and the receiver's message $\mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b)$ can be used by multiple senders S_i with input bits $(a^i_0, a^i_1) \in \{0,1\}^2$ operating as,*

$$\begin{aligned} \mathcal{F}^{\mathsf{CRSgen}}_{\mathsf{CRS}}: &\quad \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa) \\ R: &\quad (\mathsf{Aux}, \mathbf{m_1}) \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b) \end{aligned}$$

Multi-OT $\mathcal{F}_{\mathsf{multiOT}}$

Let R be a receiver with input $b \in \{0,1\}$ and $\{S_i\}_{i\in[\ell]}$ be ℓ senders where each S_i has inputs $(a_0^i, a_1^i) \in \{0,1\}^2$. The functionality $\mathcal{F}_{\mathsf{multiOT}}$ operates as follows:

- $\mathcal{F}_{\mathsf{multiOT}}$ accepts b from R
- $\forall i \in [\ell]$, on receiving (a_0^i, a_1^i) from S_i, $\mathcal{F}_{\mathsf{multiOT}}$ gives a_b^i to R

Fig. 8 Multi-OT Functionality

$$\forall i \in [\ell], S_i : \quad \mathbf{m_2^i} \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (a_0^i, a_1^i))$$
$$\forall i \in [\ell], R : \quad a_b^i \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathbf{m_2^i}, \mathsf{Aux})$$

The above protocol UC-securely realizes $\mathcal{F}_{\mathsf{multiOT}}$ (Figure 8) in the $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}$-hybrid when a malicious PPT adversary corrupts any subset of $\{S_i\}_{i\in[\ell]}$ statically and R adaptively.

2. ***Rerandomizable second message**: Let R be the space of randomness used in OT_2. There exists a composition operation $\diamond$ such that $\forall r, r' \in R$, $r^* = r' \diamond r \in R$ and $\forall \mathbf{m_1}$ it holds that,*

$$\mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (a_0, a_1); r^*) = \mathsf{OT}_3(\mathsf{CRS}; \mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (a_0, a_1); r'); r)$$

if $r \leftarrow R$ or $r' \leftarrow R$ is sampled uniformly, r^* is uniformly distributed in R.

In the malicious setting, in the CRS model, we create such an OT protocol from the OT protocol in [PVW08]. This instantiation can be found in Appendix B along with a proof that it satisfies Definition 4.

3.3 Zero-Knowledge Proofs

Zero-Knowledge Proofs are an interaction between a prover P and a verifier V such that the interaction convinces V that a secret input x of P belongs to a public language L but the interaction reveals nothing beyond that fact.

Definition 5. *A pair of machines (P, V) is a **computational zero-knowledge** proof for a language L if for every PPT V, the following holds:*

- ***Perfect Completeness:** For every $x \in L$,*

$$\Pr[\langle P, V\rangle(x) = 1] = 1$$

- ***ϵ-Soundness:** For every $x \notin L$, and every P^*,*

$$\Pr[\langle P^*, V\rangle(x) = 1] \leq \epsilon$$

- ***Zero-Knowledge:** For every PPT V^* there exists a PPT S^* such that,*

$$\{\langle P, V^*\rangle(x)\}_{x\in L} \overset{\mathrm{c}}{\approx} \{S^*(x)\}_{x\in L}$$

In this work, we construct zero-knowledge proofs for correct garbling and correct rerandomizing for Construction 1 and use these in the SCALES protocols. We do so by first constructing Σ-protocols [CDS94] and then converting them into zero-knowledge protocols where the prover sends only one message.

Definition 6. *A* Σ***-protocol*** $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ *for a statement* $x \in L$ *with witness* w *is an interaction between a prover* P *and verifier* V *of the form:*

1. P *computes* $\mathsf{Aux}, a \leftarrow P_1(x, w)$ *and sends the message* a *to* V
2. V *computes* $b \leftarrow V_1(\cdot)$*, a uniformly random* b *to send to* P
3. P *sends the last message* $c = P_2(a, b, x, w)$ *to* V
4. V *outputs* $V_2(a, b, c, x) \in \{0, 1\}$

Such a protocol must satisfy the following properties:

- ***Completeness****: For every* $x \in L$*,* $a \leftarrow P_1(x, w)$*, and* $c = P_2(a, b, x, w)$*,*

$$\Pr[V_2(a, b, c, x) = 1] = 1$$

- ***Special Soundness****: Let* w *be a witness proving* $x \in L$*,* $a \leftarrow P_1(x, w)$*. Then for any set of transcripts* (a, b, c) *and* (a, b', c') *s.t.* $b \neq b'$*,* $V_2(a, b, c, x) = 1$ *and* $V_2(a, b', c', x) = 1$*, there exists a PPT algorithm,*

$$w \leftarrow \mathsf{Extract}(a, b, c, b', c', x)$$

- ***Special Honest-Verifier Zero Knowledge (SHVZK)****: There exists a PPT algorithm,*

$$\tau = (a, b, c) \leftarrow \mathsf{Sim}(1^\kappa, x, b)$$

such that $1 = V_2(a, b, c, x)$*. For all* $b \in \{0, 1\}^\ell$*, the distribution* $\tau \leftarrow \mathsf{Sim}(1^\kappa, x, b)$ *is computationally indistinguishable from the distribution of* $\tau' = (a', b, c')$ *obtained from an honest execution of* $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ *for a statement* $x \in L$ *with witness* w*.*

4 Rerandomizable Garbling Functionality

Previous work in [AHKP22] designs a semi-honest secure SCALES protocol based on RGS (Definition 3). Following their premise, in this work we design an RGS-based malicious secure SCALES protocol. In order to realize this, we first formalize a functionality $\mathcal{F}_{\mathsf{RGS}}$ in the multiparty setting (Figure 9). For modularity, we replace the bulletin-board functionality $\mathcal{B}$ with the command $\mathsf{publish}$ that makes a message available to all parties in the system. The final protocol implements this functionality in the presence of a malicious adversary as in the definition of SCALES (Definition 2). We realize $\mathcal{F}_{\mathsf{RGS}}$ using a projective RGS (Definition 3) and a verifiable updatable oblivious transfer (VUOT) functionality.

Rerandomizable Garbling Functionality $\mathcal{F}_{\mathsf{RGS}}$

Let $f : \{0,1\}^m \to \{0,1\}^l$ be a function. Let $\mathcal{S}$ be a set of servers and $\{E_j\}_{j\in[d+1]} \subset \mathcal{S}$ be the set of $d+1$ total encoding servers. Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the clients where each C_i has input bit x_i. Let $\mathsf{publish}$ denote the action of $\mathcal{F}_{\mathsf{RGS}}$ making a message visible to all parties. Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be a projective RGS (Definition 3). $\mathcal{F}_{\mathsf{RGS}}$ is a reactive functionality as follows:

1. Commit Phase:
   - receive input bit x_i from each client $C_i \in \mathcal{C}$
2. Initial Encoding:
   - receive input randomness r_0 and function f from E_0
   - compute $(F_0, e_0) = \mathsf{RGS.Gb}(f; r_0)$ and $\mathsf{publish}$ F_0
   - parse $\{L^0_{i,0}, L^0_{i,1}\}_{i\in[m]} = e_0$ and give each L^0_{i,x_i} to client C_i
3. Successive Encoding $j \in [d]$:
   - receive input randomness r_j from E_j
   - compute $(F_j, \pi_j) = \mathsf{RGS.Rerand}(F_{j-1}; r_j)$ and $\mathsf{publish}$ F_j
   - parse $\{L^j_{i,0}, L^j_{i,1}\}_{i\in[m]} = \pi_j(e_{j-1}) = e_j$ and give each L^j_{i,x_i} to C_i
4. $\mathsf{publish}$ $X = \{L^d_{i,x_i}\}_{i\in[m]}$

At the end of the execution, any party can compute $f(x) = \mathsf{RGS.Ev}(F_d, X)$

Fig. 9 Rerandomizable Garbling Functionality

4.1 Verifiable Updatable Oblivious Transfer

We formally define a functionality $\mathcal{F}_{\mathsf{VUOT}}$ for Verifiable Updatable Oblivious Transfer (VUOT) in Figure 10. $\mathcal{F}_{\mathsf{VUOT}}$ operates in phases and involves multiple receivers, one sender, and multiple *updaters*. It is parameterized by two public relations $\mathcal{R}_0$ and $\mathcal{R}_1$. Each receiver C_i has a bit x_i that is input to $\mathcal{F}_{\mathsf{VUOT}}$ in the 'Commit Phase'. The sender S's input is a set of strings: a pair (s^0_i, s^1_i) corresponding to each receiver C_i; a statement α_0 and witness w_0. This is input to $\mathcal{F}_{\mathsf{VUOT}}$ in the 'Initial OT' phase. Then $\mathcal{F}_{\mathsf{VUOT}}$ checks the relation $\mathcal{R}_0$ over all of the sender's inputs, gives each C_i the string $s^{x_i}_i$, and reveals α_0 to all parties.

Each updater U_j has as input a set of functions: one σ^j_i corresponding to each receiver C_i; a statement α_j and witness w_j. This is input to $\mathcal{F}_{\mathsf{VUOT}}$ in the j^{th} 'Successive Update' phase. $\mathcal{F}_{\mathsf{VUOT}}$ checks the relation $\mathcal{R}_1$ over all of U_j's inputs and α_{j-1} from the previous phase. The functionality gives each C_i the string $s_{i,j}$ that is σ^j_i applied to the output to C_i in the previous phase. It also reveals α_j to all parties. After the final update phase, $\mathcal{F}_{\mathsf{VUOT}}$ reveals to all parties a bit indicating whether all updaters and sender satisfied their respective relations. If this is true, it also makes the set of final receiver outputs visible to all parties.

4.2 Realizing $\mathcal{F}_{\mathsf{RGS}}$

We use a projective RGS (Definition 3) and the functionality $\mathcal{F}_{\mathsf{VUOT}}$ (Figure 10) to instantiate $\mathcal{F}_{\mathsf{RGS}}$ (Figure 9). The functionality $\mathcal{F}_{\mathsf{VUOT}}$ needs to be closely tied

Verifiable Updatable Oblivious Transfer $\mathcal{F}_{\mathsf{VUOT}}$

Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the set of receivers where C_i has input $x_i \in \{0,1\}$.
Let S be a sender with inputs $\{s_i^0, s_i^1\}_{i\in[m]}$ where each string $s_i^b \in \{0,1\}^\ell$.
Let $\mathcal{R}_0$ be a relation with statement α_0 and witness w_0 s.t.,

$$\{0,1\} \leftarrow \mathcal{R}_0(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]})$$

Let $\mathcal{U} = \{U_j\}_{j\in[d]}$ be updaters s.t. U_j has $\{\sigma_i^j\}_{i\in[m]}$ and $\sigma_i^j : \{0,1\}^\ell \to \{0,1\}^\ell$.
Let $\mathcal{R}_1$ be a relation s.t. for each $j \in [d]$ with statement α_j and witness w_j,

$$\{0,1\} \leftarrow \mathcal{R}_1(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]})$$

Let $\mathsf{publish}$ denote the action of $\mathcal{F}_{\mathsf{VUOT}}$ making a message visible to all parties.
$\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_0,\mathcal{R}_1}$ is a reactive functionality that operates as follows:

1. Commit Phase:
   - receive input x_i from each receiver $C_i \in \mathcal{C}$
2. Initial OT:
   - receive inputs $\{s_i^0, s_i^1\}_{i\in[m]}$ and α_0, w_0 from S
   - compute $b_0 \leftarrow \mathcal{R}_0(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]})$
   - for all $i \in [m]$, give $s_{i,0} = s_i^{x_i}$ to receiver C_i
   - $\mathsf{publish}$ α_0
3. Successive Update $j \in [d]$:
   - receive inputs $\{\sigma_i^j\}_{i\in[m]}$ and α_j, w_j from U_j
   - compute $b_j \leftarrow \mathcal{R}_1(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]})$
   - for all $i \in [m]$, give $s_{i,j} = \sigma_i^j(s_{i,j-1})$ to receiver C_i
   - $\mathsf{publish}$ α_j
4. $\mathsf{publish}$ $b = \bigwedge_{j=0}^{d} b_j$ and, if $b = 1$, then $\mathsf{publish}$ $\{s_{i,d}\}_{i\in[m]}$

Fig. 10 Verifiable Updatable Oblivious Transfer

to the RGS. In particular, in $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ let $\mathsf{RGS.Gb}$ produce a projective input encoding $e = \{L_i^0, L_i^1\}_{i\in[m]}$. Then in $\mathcal{F}_{\mathsf{VUOT}}$, the input of the sender S is $\{s_i^0, s_i^1\}_{i\in[m]} = \{L_i^0, L_i^1\}_{i\in[m]}$ and relation $\mathcal{R}_0$ becomes:

$$\begin{aligned}&\mathcal{R}_{\mathsf{Gb}}(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]})\\ &= \begin{cases} 1 \text{ if } (F, e) = \mathsf{RGS.Gb}(f; r) \text{ where } F = \alpha_0, r = w_0, e = \{s_i^0, s_i^1\}_{i\in[m]} \\ 0 \text{ otherwise} \end{cases}\end{aligned} \tag{1}$$

Let $\mathsf{RGS.Rerand}$ produce input transformations of the form $\pi_{\mathsf{En}} = \{\lambda_i\}_{i\in[m]}$. Then for $\mathcal{F}_{\mathsf{VUOT}}$, the input of each updater U_j is $\{\sigma_i^j\}_{i\in[m]} = \{\lambda_i\}_{i\in[m]}$. Further, the relation $\mathcal{R}_1$ is of the following form:

$$\begin{aligned}&\mathcal{R}_{\mathsf{Rerand}}(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]})\\ &= \begin{cases} 1 \text{ if } (F, \pi) = \mathsf{RGS.Rerand}(F'; r) \text{ where } \begin{cases} F' = \alpha_{j-1}, F = \alpha_j, \\ r = w_j, \pi = \{\sigma_i^j\}_{i\in[m]} \end{cases} \\ 0 \text{ otherwise} \end{cases}\end{aligned} \tag{2}$$

Protocol Realizing $\mathcal{F}_{\mathsf{RGS}}$

Let f be the function and $x = \{x_i\}_{i\in[m]}$ be its input. Let $\mathcal{S}$ be a set of servers and $\{E_j\}_{j\in[d]} \subset \mathcal{S}$ be the encoding servers. $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the clients where each C_i has input bit x_i. Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be a projective RGS (Definition 3) and $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_{\mathsf{Gb}},\mathcal{R}_{\mathsf{Rerand}}}$ be a VUOT functionality (Figure 10).

1. Commit Phase:
   - Each client $C_i \in \mathcal{C}$ gives x_i to $\mathcal{F}_{\mathsf{VUOT}}$
2. Initial Encoding:
   - E_0 samples r_0 and computes $(F_0, e_0) = \mathsf{RGS.Gb}(f; r_0)$
   - E_0 sends (F_0, r_0, e_0) to $\mathcal{F}_{\mathsf{VUOT}}$ as in relation $\mathcal{R}_{\mathsf{Gb}}$ (Equation 1)
   - parse $e_0 = \{L_i^0, L_i^1\}_{i\in[m]}$ and for all $i \in [m]$, $\mathcal{F}_{\mathsf{VUOT}}$ gives L_{i,x_i}^0 to C_i
   - $\mathcal{F}_{\mathsf{VUOT}}$ publishes $F_0 = \alpha_0$
3. Successive Encoding:
   - E_j samples r_j and computes $(F_j, \pi_j) = \mathsf{RGS.Rerand}(F_{j-1}; r_j)$
   - E_j sends (F_j, r_j, π_j) to $\mathcal{F}_{\mathsf{VUOT}}$ as in relation $\mathcal{R}_{\mathsf{Rerand}}$ (Equation 2)
   - parse $\pi_j = \{\lambda_i\}_{i\in[m]}$ and for all $i \in [m]$, $\mathcal{F}_{\mathsf{VUOT}}$ gives $L_{i,x_i}^j = \lambda_i(L_{i,x_i}^{j-1})$ to C_i
   - $\mathcal{F}_{\mathsf{VUOT}}$ publishes $F_j = \alpha_j$
4. $\mathcal{F}_{\mathsf{VUOT}}$ publishes $b = \bigcap_{j=0}^{d} b_j$
   - if $b = 0$, then honest parties ABORT
   - if $b = 1$, then $\mathcal{F}_{\mathsf{VUOT}}$ publishes $X = \{L_{i,x_i}^d\}_{i\in[m]}$
5. Decoding Phase:
   - Any $S \in \mathcal{S}$ can compute $f(x) = \mathsf{RGS.Ev}(F_d, X)$

Fig. 11 Construction for $\mathcal{F}_{\mathsf{RGS}}$ in the $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_{\mathsf{Gb}},\mathcal{R}_{\mathsf{Rerand}}}$-hybrid

Theorem 1. *Let* $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ *be a projective RGS (Definition 3). Then the protocol in Figure 11 UC-securely realizes* $\mathcal{F}_{\mathsf{RGS}}$ *(Figure 9) in the* $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_{\mathsf{Gb}},\mathcal{R}_{\mathsf{Rerand}}}$*-hybrid, where* $\mathcal{F}_{\mathsf{VUOT}}$ *(Figure 10) is parameterized by relations* $\mathcal{R}_{\mathsf{Gb}}$ *(Equation 1) and* $\mathcal{R}_{\mathsf{Rerand}}$ *(Equation 2).*

Proof. We need to show that the functionality $\mathcal{F}_{\mathsf{RGS}}$ (Figure 9) is realized by the protocol in Figure 11. Let $\mathcal{B}$ be the bulletin-board. Let $\mathcal{A}$ be a PPT malicious adversary. Among the parties that $\mathcal{A}$ corrupts, let $C_S \subset \mathcal{C}$ be the set of statically corrupted clients. Let $C_A \subset \mathcal{C} - C_S$ denote the set of clients that $\mathcal{A}$ adaptively corrupts after the 'commit phase' in $\mathcal{F}_{\mathsf{RGS}}$. Let $S \subset \{E_0, \ldots, E_d\}$ be the set of statically corrupted encoding servers such that at least one server is honest. Note that $\mathcal{A}$ may also corrupt other servers than the encoding server and it has access to everything on $\mathcal{B}$.

Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be a projective RGS as in Definition 3. Let $\mathcal{F}_{\mathsf{VUOT}}$ be the functionality realizing VUOT (Figure 10). Then in the $\mathcal{F}_{\mathsf{VUOT}}$-hybrid let Sim be a PPT simulator for the protocol in Figure 11 that operates as follows:

1. In the 'commit phase',
   - Sim emulates the 'commit phase' of $\mathcal{F}_{\mathsf{VUOT}}$ and for each statically corrupted client $C_i \in C_S$, it accepts x_i for $\mathcal{A}$.
   - Sim passes $\{x_i\}_{C_i\in C_S}$ to $\mathcal{F}_{\mathsf{RGS}}$.

2. For every client $C_i \in C_A$ that $\mathcal{A}$ adaptively corrupts,
   - Sim sends i to $\mathcal{F}_{\mathsf{RGS}}$ and receives x_i.
   - Sim gives x_i to $\mathcal{A}$.
3. In the 'initial encoding' phase, if E_0 is corrupted,
   - Sim emulates 'initial OT' of $\mathcal{F}_{\mathsf{VUOT}}$ and accepts (F_0, r_0, e_0) from $\mathcal{A}$.
   - Sim emulates $\mathcal{F}_{\mathsf{VUOT}}$ and computes $b_0 = 1$ if $(F_0, e_0) == \mathsf{RGS.Gb}(f; r_0)$, and $b_0 = 0$ otherwise.
   - If $b_0 = 0$, it gives $\perp$ to $\mathcal{F}_{\mathsf{RGS}}$ and the execution aborts.
   - Otherwise, it gives r_0 and f, that is publicly known, to $\mathcal{F}_{\mathsf{RGS}}$ that computes $(F_0, e_0) = \mathsf{RGS.Gb}(f; r_0)$ and posts F_0 onto $\mathcal{B}$.
   - $\mathcal{F}_{\mathsf{RGS}}$ gives $\{L^0_{i,x_i} \in e_0\}_{C_i \in C_S \cup C_A}$ to Sim, and Sim gives this to $\mathcal{A}$.

   If E_0 is honest, $\mathcal{F}_{\mathsf{RGS}}$ posts F_0 onto $\mathcal{B}$ and Sim gets for each corrupted C_i the label $L^0_{i,x_i} \in e_0$, which it passes to $\mathcal{A}$.
4. In the 'successive encoding' phase, if E_j is corrupted,
   - Sim emulates 'successive update' of $\mathcal{F}_{\mathsf{VUOT}}$ and gets (F_j, r_j, π_j) from $\mathcal{A}$.
   - Sim, emulating $\mathcal{F}_{\mathsf{VUOT}}$, sets $b_j = 1$ if $(F_j, \pi_j) == \mathsf{RGS.Rerand}(F_{j-1}; r_j)$, and $b_j = 0$ otherwise.
   - If $b_j = 0$, it gives $\perp$ to $\mathcal{F}_{\mathsf{RGS}}$ and the execution aborts.
   - Otherwise, Sim gives r_j to $\mathcal{F}_{\mathsf{RGS}}$.
   - $\mathcal{F}_{\mathsf{RGS}}$ computes $(F_j, \pi_j) = \mathsf{RGS.Rerand}(F_{j-1}; r_j)$ and posts F_j onto $\mathcal{B}$.
   - $\mathcal{F}_{\mathsf{RGS}}$ gives $\{L^j_{i,x_i}\}_{C_i \in C_S \cup C_A}$ to Sim, where each $L^j_{i,x_i} \in e_j = \pi_j(e_{j-1})$, and Sim gives this to $\mathcal{A}$.

   If E_j is honest, $\mathcal{F}_{\mathsf{RGS}}$ posts F_j onto $\mathcal{B}$ and Sim gets for each corrupted C_i the label $L^j_{i,x_i} \in e_j = \pi_j(e_{j-1})$. Sim gives this to $\mathcal{A}$.
5. Sim emulates $\mathcal{F}_{\mathsf{VUOT}}$ and posts b=1 onto $\mathcal{B}$ and then $X = \{L^d_{i,x_i}\}_{i \in [m]}$ is posted by $\mathcal{F}_{\mathsf{RGS}}$.

Let κ be a computational security parameter. Let $\mathcal{R}$ denote the space of randomness of all parties participating in the protocol and $\vec{r}$ denote the contents of the random tape of all the parties in the real execution. The view of $\mathcal{A}$ produced by Sim is distributed as,

$$\left\{\{x_i\}_{C_i \in C_S \cup C_A}, \{r_j\}_{j \neq j^* \in [d]}, \{F_j, \{L^j_{i,x_i}\}_{C_i \in C_S \cup C_A}\}_{j \in [d]}, b, X\right\}_{\kappa \in \mathbb{N}, \vec{r} \in \mathcal{R}}$$

This is exactly identical to the distribution of the view in the real execution. Hence, Figure 11 securely computes $\mathcal{F}_{\mathsf{RGS}}$ in the $\mathcal{F}_{\mathsf{VUOT}}$-hybrid.

It now remains to show how $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_{\mathsf{Gb}}, \mathcal{R}_{\mathsf{Rerand}}}$ (Figure 10) is UC-securely realized. However, before diving into the construction itself, we first define and construct its key building-blocks.

5 Constructing Verifiable Updatable OT

In this section we construct a protocol that realizes $\mathcal{F}_{\mathsf{VUOT}}$ (Figure 10) in the presence of malicious adversaries. This requires three major building blocks. We define them in their full generality and present protocols for these that when put together realize $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_{\mathsf{Gb}}, \mathcal{R}_{\mathsf{Rerand}}}$ in particular:

Updatable Oblivious Transfer $\mathcal{F}_{\mathsf{UOT}}^{\mathcal{M},\Sigma}$

For a message domain $\mathcal{M}$, let S be a sender with input strings $s_0, s_1 \in \mathcal{M}$. Let C be a chooser with input bit $b \in \{0,1\}$. For function domain Σ, let $\{U_j\}_{j\in[d]}$ be updaters with $\sigma_j \in \Sigma$ s.t. $\sigma_j : \mathcal{M} \to \mathcal{M}$. Let $\mathsf{publish}$ be the action of making a message visible to all parties. The functionality $\mathcal{F}_{\mathsf{UOT}}^{\mathcal{M},\Sigma}$ works as follows:

1. OT Phase:
   - receive input b from C and (s_0, s_1) from S
   - send $s_{b,0} = s_b$ to C
2. Update Phase, for each $j \in [d]$:
   - receive input σ_j from U_j and send $s_{b,j} = \sigma_j(s_{b,j-1})$ to C
3. Reveal Phase:
   - $\mathsf{publish}$ $s_{b,d}$

Fig. 12 Updatable Oblivious Transfer

- **Updatable Oblivious Transfer Functionality** (UOT): In Section 5.1 we define the functionality $\mathcal{F}_{\mathsf{UOT}}$ and present a protocol realizing $\mathcal{F}_{\mathsf{UOT}}$ for a message and function domain compatible with the RGS in Construction 1.
- **Σ-Protocols**: In Section 5.2 we present Σ-Protocols for the relations $\mathcal{R}_{\mathsf{Gb}}$ and $\mathcal{R}_{\mathsf{Rerand}}$. The Σ-Protocol for $\mathcal{R}_{\mathsf{Gb}}$ proves correctness of garbling and that the correct garbling labels are used in the OT phase. The Σ-Protocol for $\mathcal{R}_{\mathsf{Rerand}}$ proves the correctness of rerandomizing and that the correct input label transformation functions are used in the successive update.
- **Distributed Committed-Index Oblivious Transfer Functionality** (DCOT): In Section 5.3 we define a functionality $\mathcal{F}_{\mathsf{DComOT}}$ and construct a protocol that realizes it. The functionality allows multiple receivers to input a random bit each. Then, for multiple rounds, a sender sends two input strings. In the end one string of each sender is publicly revealed and the choice of this string corresponds to the XOR of all the receivers' input choice bits.

In Section 5.4, we combine the last two objects to get Ephemeral Prover Zero-Knowledge $\mathcal{F}_{\mathsf{EPZK}}$. This is a functionality parameterized by two relations $\mathcal{R}_0$ and $\mathcal{R}_1$. It involves multiple provers P_j each with a statement α_j and witness w_j as inputs where, in the end, $\mathcal{F}_{\mathsf{EPZK}}$ outputs a bit indicating whether all the provers' inputs satisfy $\mathcal{R}_0$ or $\mathcal{R}_1$ as required. In Section 5.5 we present a protocol that realizes $\mathcal{F}_{\mathsf{VUOT}}$ in the $\mathcal{F}_{\mathsf{EPZK}}$-hybrid using the UOT protocol as a building block.

5.1 Updatable Oblivious Transfer

In this section, we define and construct a protocol for Updatable Oblivious Transfer (UOT). The UOT functionality $\mathcal{F}_{\mathsf{UOT}}$ is described in Figure 12. It involves a chooser C with a bit b, a sender S with input strings $s_0, s_1 \in \mathcal{M}$ and multiple updaters $\{U_j\}_{j\in[d]}$ where each U_j has a function $\sigma_j \in \Sigma$. In the 'OT

phase' $\mathcal{F}_{\mathsf{UOT}}$ accepts the chooser's and sender's inputs and returns the string s_b to the chooser. In every 'Update Phase', each updater U_j sends σ_j to $\mathcal{F}_{\mathsf{UOT}}$ which gives to C the output of σ_j applied to the output received in the previous phase. Finally, in the 'Reveal Phase' $\mathcal{F}_{\mathsf{UOT}}$ makes the final output to C available to all parties.

We realize $\mathcal{F}_{\mathsf{UOT}}$ in a way that requires the sender and each updater to send only one message each. The chooser sends one message at the beginning and one at the end of the protocol and the resulting protocol is secure in the presence of a PPT malicious adversary adaptively corrupting the chooser, and all-but-one of the sender and updaters. We abstract such a protocol as in Definition 7.

Definition 7. *Let $\mathcal{M}$ be the message space for the sender S and Σ be the function space for each updater in $\{U_j\}_{j\in[d]}$ such that each $\sigma \in \Sigma$ is of the form $\sigma : \mathcal{M} \to \mathcal{M}$. The tuple $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ is a **SCALES Protocol for UOT** if the following protocol UC-securely realizes $\mathcal{F}_{\mathsf{UOT}}^{\mathcal{M},\Sigma}$ (Figure 12) in the $(\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}, \mathcal{F}_{\mathcal{B}})$-hybrid in the presence of any PPT adversary $\mathcal{A}$ that can adaptively corrupt C maliciously and statically corrupt all-but-one of $\{S\} \cup \{U_j\}_{j\in[d]}$ semi-honestly:*

Inputs: $b \in \{0,1\}$ for receiver C; $(s_0, s_1) \in \mathcal{M}^2$ for S; σ_i for each updater U_i.

$$
\begin{aligned}
\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}: &\quad \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa) \\
C: &\quad \mathsf{Aux}, \mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b) && \mathbf{m_1} \to \mathcal{F}_{\mathcal{B}} \\
S: &\quad \mathbf{m_2^0} \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0, s_1)) && \mathbf{m_2^0} \to \mathcal{F}_{\mathcal{B}} \\
C: &\quad M \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathsf{Aux}, \mathbf{m_2^0}) \\
&\quad \{s_b, \bot\} \leftarrow \mathsf{OT}_{\mathsf{vrfy}}(\mathsf{CRS}; M) && \textit{output } s_b \\
\forall j \in [d]\ U_j: &\quad \mathbf{m_2^j} \leftarrow \mathsf{OT}_{\mathsf{updt}}(\mathsf{CRS}; \mathbf{m_2^{j-1}}, \sigma_j) && \mathbf{m_2^j} \to \mathcal{F}_{\mathcal{B}} \\
C: &\quad M_j \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathsf{Aux}, \mathbf{m_2^j}) && M_d \to \mathcal{F}_{\mathcal{B}} \\
&\quad \{s_{b,j}, \bot\} \leftarrow \mathsf{OT}_{\mathsf{vrfy}}(\mathsf{CRS}; M_j) && \textit{output } s_{b,j} \\
\textit{public}: &\quad \{s_{b,d}, \bot\} \leftarrow \mathsf{OT}_{\mathsf{vrfy}}(\mathsf{CRS}; M_d) && \textit{output } s_{b,d}
\end{aligned}
$$

where $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}$ (Figure 7) is the Common Reference String functionality and $\mathcal{F}_{\mathcal{B}}$ (Figure 6) is the Bulletin-Board functionality.

[GHV10] constructs a protocol for 'rerandomizable OT' in the semi-honest setting that achieves a similar functionality as UOT for a restricted class of messages and function space. Our protocol for $\mathcal{F}_{\mathsf{UOT}}$ closely follows their construction.

Definition 8. *Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be a projective rerandomizable garbling scheme (Definition 3) with encoding set $\mathcal{E} = \mathcal{Z}^{2m}$ and domain of offset functions $\Pi = \{\pi_{\mathsf{En}} : \mathcal{Z}^2 \to \mathcal{Z}^2\}^m$. Then UOT is a UOT protocol (Definition 7) with sender's message space $\mathcal{M}$ and updater's function space Σ is RGS-**Compatible** if it holds that $\mathcal{Z} \subseteq \mathcal{M}$ and $\pi_{\mathsf{En}} \subseteq \Sigma$.*

Constructing UOT. We construct UOT for the sender's message domain $\mathcal{M}$ being the set of balanced binary strings: $s_0, s_1 \in \{0,1\}^\ell$ with $\frac{\ell}{2}$ 0s and the rest 1s. The function family for each updater U_j is the family of permutations $\Sigma = \{\sigma : [\ell] \to [\ell]\}$. This suffices since we only require a UOT protocol that is *compatible* with the projective RGS in Construction 1. That is, the UOT inputs of the sender are labels of an input wire of the RGS. The inputs of each updater is a function $\sigma \in \Sigma$ in the space of transformations applied to the labels. Following [GHV10], we realize $\mathcal{F}_{\mathsf{UOT}}$ from 2-round bit-OT protocol (Definition 4).

Protocol Realizing $\mathcal{F}_{\mathsf{UOT}}$

Let $\mathbf{OT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_3)$ be a 2-round maliciously secure OT protocol (Definition 4). Let $\mathsf{NIZK} = (\mathsf{CRSgen}, \mathsf{Prove}, \mathsf{Verify})$ be an NIZK proof $\mathsf{CRS}_{\mathsf{NIZK}} \leftarrow \mathsf{CRSgen}(1^\kappa)$, $\Pi \leftarrow \mathsf{Prove}(\mathsf{CRS}_{\mathsf{NIZK}}; \mathsf{Aux}, b; m_1, m_2, a_b)$ and $\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{CRS}_{\mathsf{NIZK}}; m_1, m_2, a_b, \Pi)$ for $\mathcal{R}^{\mathbf{OT}}$:

$$\exists r, \mathsf{Aux}, b \text{ s.t.} \quad \mathsf{Aux}, m_1 \leftarrow \mathbf{OT}.\mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}, r; b) \quad a_b = \mathbf{OT}.\mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{OT}}; m_2, \mathsf{Aux})$$

Let $\mathcal{M}$ be the set of ℓ-bit balanced binary strings and Σ be the family of permutations on ℓ positions. Then $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ is defined as follows:

- $\mathsf{CRS} \leftarrow \mathsf{UOT}.\mathsf{CRSgen}(1^\kappa)$

$$\mathsf{CRS}_{\mathsf{OT}} \leftarrow \mathbf{OT}.\mathsf{CRSgen}(1^\kappa)$$
$$\mathsf{CRS}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK}.\mathsf{CRSgen}(1^\kappa) \qquad \mathsf{CRS} = (\mathsf{CRS}_{\mathsf{OT}}, \mathsf{CRS}_{\mathsf{NIZK}})$$

- $\mathsf{Aux}, \mathbf{m_1} \leftarrow \mathsf{UOT}.\mathsf{OT}_1(\mathsf{CRS}; b)$

$$\mathsf{Aux}, \mathbf{m_1} \leftarrow \mathbf{OT}.\mathsf{OT}_1(\mathsf{CRS}_{\mathsf{OT}}; b)$$

- $\mathbf{m_2^0} \leftarrow \mathsf{UOT}.\mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0, s_1))$

$$s_0, s_1 = \{a_0^i\}_{i\in[\ell]}, \{a_1^i\}_{i\in[\ell]}$$
$$\forall i \in [\ell], m_2^i \leftarrow \mathbf{OT}.\mathsf{OT}_2(\mathsf{CRS}_{\mathsf{OT}}; \mathbf{m_1}, (a_0^i, a_1^i)) \qquad \mathbf{m_2^0} = \{m_2^i\}_{i\in[\ell]}$$

- $\mathbf{m_2^j} \leftarrow \mathsf{UOT}.\mathsf{OT}_{\mathsf{updt}}(\mathsf{CRS}; \mathbf{m_2^{j-1}}, \sigma_j)$

$$\mathbf{m_2^{j-1}} = \{m_2^i\}_{i\in[\ell]} \qquad \sigma_j : [\ell] \to [\ell] \in \Sigma$$
$$\forall i \in [\ell], m_2^{i'} \leftarrow \mathbf{OT}.\mathsf{OT}_3(\mathsf{CRS}_{\mathsf{OT}}; m_2^i) \qquad \mathbf{m_2^j} = \sigma_j(\{m_2^{i'}\}_{i\in[\ell]})$$

- $M_j \leftarrow \mathsf{UOT}.\mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathsf{Aux}, \mathbf{m_2^j}) \qquad \mathbf{m_2^j} = \{m_2^i\}_{i\in[\ell]}$

$$\forall i \in [\ell], a_b^i \leftarrow \mathbf{OT}.\mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{OT}}; m_2^i, \mathsf{Aux})$$
$$\forall i \in [\ell], \Pi_i \leftarrow \mathsf{NIZK}.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{NIZK}}; \mathsf{Aux}, b; m_1, m_2^i, a_b^i)$$
$$s_b = \{a_b^i\}_{i\in[\ell]} \qquad \Pi = \{\Pi_i\}_{i\in[\ell]} \qquad M_j = (s_b, \Pi)$$

- $s_b \leftarrow \mathsf{UOT}.\mathsf{OT}_{\mathsf{vrfy}}(\mathsf{CRS}; M_j)$

$$M_j = (s_b = \{a_b^i\}_{i\in[\ell]}, \{\Pi_i\}_{i\in[\ell]})$$
$$\forall i \in [\ell], b_i \in \{0,1\} \leftarrow \mathsf{NIZK}.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{NIZK}}; \mathbf{m_1}, m_2^i, a_b^i, \Pi_i) \qquad \text{return } \begin{cases} s_b & \text{if } \bigcap_{i\in[\ell]} b_i = 1 \\ \perp & \text{otherwise} \end{cases}$$

Fig. 13 Construction for $\mathcal{F}_{\mathsf{UOT}}$ using $\mathbf{OT}$

Lemma 1. *Let $\mathcal{M} \subset \{0,1\}^\ell$ be the set of balanced binary strings and $\Sigma = \{\sigma : [\ell] \to [\ell]\}$ be the family of permutations on bit-positions. Let* $\mathbf{OT}$ =

$(\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_3)$ *be a 2-round maliciously secure OT protocol (Definition 4) in the* $\mathcal{F}_{\mathsf{CRS}}^{\mathbf{OT}.\mathsf{CRSgen}}$*-hybrid and* $\mathsf{NIZK} = (\mathsf{CRSgen}, \mathsf{Prove}, \mathsf{Verify})$ *be an NIZK proof system for* $\mathcal{R}^{\mathbf{OT}}$ *in the* $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{NIZK}.\mathsf{CRSgen}}$*-hybrid. Figure 13 is a UC-secure protocol that realizes* $\mathcal{F}_{\mathsf{UOT}}^{\mathcal{M},\Sigma}$ *(Figure 12 – Definition 7).*

Proof. Figure 13 contains a UOT protocol constructed using a 2-round malicious-secure bit-OT protocol and an NIZK proof system. Let $\mathbf{OT}$ be the OT protocol as in (Definition 4). Since the protocol is a 2-round secure protocol in the presence of reuse of the first OT message, there exists a PPT simulator $\mathsf{Sim}_{\mathsf{multiOT}}$ that can simulate the view of a malicious PPT adversary that statically corrupts an arbitrary subset of the senders and adaptively corrupts the receiver, in a way that is computationally indistinguishable from the real view.

Let $\mathsf{NIZK} = (\mathsf{Prove}, \mathsf{Verify})$ be an NIZK proof system for the relation,

$$\begin{aligned} \exists \mathsf{Aux}, b \text{ s.t. } \mathsf{Aux}, m_1 \leftarrow & \mathbf{OT}.\mathsf{OT}_1(b) \\ a_b = & \mathbf{OT}.\mathsf{OT}_{\mathsf{fin}}(m_2, \mathsf{Aux}) \end{aligned}$$

where $w = (\mathsf{Aux}, b)$ is the witness and $\alpha = (\mathbf{OT}, \mathbf{m_1}, \mathbf{m_2}, a_b)$ is the statement. There exists a PPT simulator $\mathsf{Sim}_{\mathsf{NIZK}}(\alpha)$ that can produce a view whose distribution is computationally indistinguishable from the real NIZK proof.

Let $\mathcal{M} \subset \{0,1\}^\ell$ be the set of balanced binary strings and $\Sigma = \{\sigma : [\ell] \to [\ell]\}$ be the family of permutations on ℓ-bit-positions. We need to show that the protocol in Figure 13 is a protocol that realizes $\mathcal{F}_{\mathsf{UOT}}$ (Figure 12) as in Definition 7 with respect to sender's input space $\mathcal{M}$ and updaters' function space Σ. Let $\mathsf{Sim}_{\mathsf{UOT}}$ be a PPT simulator in the ideal world that operates as follows:

1. In the 'OT Phase', if C is statically corrupted:
   - $\mathsf{Sim}_{\mathsf{UOT}}$ receives $\mathbf{m_1}$ from the PPT adversary $\mathcal{A}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ invokes $\mathsf{Sim}_{\mathsf{multiOT}}$ to extract b from a corrupt receiver.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ passes b into $\mathcal{F}_{\mathsf{UOT}}$.

   otherwise, if C is initially honest:
   - $\mathsf{Sim}_{\mathsf{UOT}}$ invokes $\mathsf{Sim}_{\mathsf{multiOT}}$ to simulate $\mathbf{m_1^*}$ for an honest receiver.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ posts $\mathbf{m_1^*}$ to $\mathcal{B}$.
2. Continuing the 'OT Phase', if S is statically corrupted:
   - $\mathsf{Sim}_{\mathsf{UOT}}$ receives $\mathbf{m_2^0}$ from the adversary $\mathcal{A}$ and parses $\mathbf{m_2^0} = \{m_2^i\}_{i\in[\ell]}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ has $s_0 = \{a_0^i\}_{i\in[\ell]}$ and $s_1 = \{a_1^i\}_{i\in[\ell]}$ and passes it into $\mathcal{F}_{\mathsf{UOT}}$.
3. If C is adaptively corrupted:
   - $\mathsf{Sim}_{\mathsf{UOT}}$ gets the input b from $\mathcal{F}_{\mathsf{UOT}}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ invokes $\mathsf{Sim}_{\mathsf{multiOT}}$ to derive r for an adaptively corrupted receiver
   - $\mathsf{Sim}_{\mathsf{UOT}}$ passes (r, b) to $\mathcal{A}$.
4. At the end of the 'OT phase', if C is corrupted and S is honest,
   - $\mathsf{Sim}_{\mathsf{UOT}}$ gets $s_b^0 = s_b$ from $\mathcal{F}_{\mathsf{UOT}}$ and parses $s_b^0 = \{a_b^i\}_{i\in[\ell]}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ continues $\mathsf{Sim}_{\mathsf{multiOT}}$ with $\{a_b^i\}_{i\in[\ell]}$ to generate $\{m_2^{i*}\}_{i\in[\ell]}$ for ℓ honest senders.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ posts $\mathbf{m_2^{0*}} = \{m_2^{i*}\}_{i\in[\ell]}$ to $\mathcal{B}$.
5. For each $j \in [d]$, in the 'Update Phase', if U_j is statically corrupted:

- $\mathsf{Sim}_{\mathsf{UOT}}$ receives $\mathbf{m_2^j}$ from $\mathcal{A}$ and parses $\mathbf{m_2^j} = \{m_2^{i'}\}_{i\in[\ell]}$.
- $\mathsf{Sim}_{\mathsf{UOT}}$ has σ and passes it to $\mathcal{F}_{\mathsf{UOT}}$.

6. At the end of the 'Update Phase', if C is corrupted and $\mathrm{U}_{j'}$ is honest,
   - $\mathsf{Sim}_{\mathsf{UOT}}$ gets s_b^j from $\mathcal{F}_{\mathsf{UOT}}$ and parses $s_b^j = \{a_b^i\}_{i\in[\ell]}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ completes $\mathsf{Sim}_{\mathsf{multiOT}}$ with $\{a_b^i\}_{i\in[\ell]}$ corresponding to ℓ honest senders and simulates $\{m_2^{i*}\}_{i\in[\ell]}$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ posts $\mathbf{m_2^{j*}} = \{m_2^{i*}\}_{i\in[\ell]}$ to $\mathcal{B}$.
7. If C is honest in the 'Reveal Phase',
   - $\mathsf{Sim}_{\mathsf{UOT}}$ receives s_b^d from $\mathcal{F}_{\mathsf{UOT}}$ and parses $s_b^d = \{a_b^i\}_{i\in[\ell]}$.
   - $\forall i \in [\ell]$, $\mathsf{Sim}_{\mathsf{UOT}}$ invokes $\Pi_i^* \leftarrow \mathsf{Sim}_{\mathsf{NIZK}}(\mathsf{OT}, \mathbf{m_1}, m_2^{i'}, a_b^i)$.
   - $\mathsf{Sim}_{\mathsf{UOT}}$ posts $s_b^d, \{\Pi_i^*\}_{i\in[\ell]}$ to $\mathcal{B}$.

The complete view of the protocol produced by $\mathsf{Sim}_{\mathsf{UOT}}$ consists of,

$$
\begin{cases} \mathbf{m_1^*} & \text{if C is honest} \\ \mathbf{m_1}, b, \{s_b^j\}_{j\in[d]\cup\{0\}} & \text{if C is statically corrupted} \end{cases}
$$

$$
\begin{cases} \mathbf{m_2^{0*}} & \text{if S is honest} \\ \mathbf{m_2^0}, s_0, s_1 & \text{if S is corrupted} \end{cases}
$$

$$
\forall j \in [d] \begin{cases} \mathbf{m_2^{j*}} & \text{if } \mathrm{U}_j \text{ is honest} \\ \mathbf{m_2^j}, \sigma_j & \text{if } \mathrm{U}_j \text{ is corrupted} \end{cases}
$$

$$
\begin{cases} b, r, \{s_b^j\}_{j\in[d]\cup\{0\}} & \text{if C is adaptively corrupted} \\ s_b^d, \{\Pi_i^*\}_{i\in[\ell]} & \text{if C is honest} \end{cases}
$$

Without loss of generality, consider the view when S and all updaters U_j are honest, and C is adaptively corrupted at the end of the execution. Let κ be a computational security parameter. Let $\mathcal{R}$ denote the space of randomness of all parties participating in the protocol and $\vec{r}$ denote the contents of the random tape of all parties and simulator. The view for this execution is distributed as,

$$
\left\{(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \mathbf{m_1^*}, \mathbf{m_2^{0*}}, \{\mathbf{m_2^{j*}}\}_{j\in[d]}, \{\Pi_i^*\}_{i\in[\ell]}\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}
$$

Out of all corruption strategies, the distribution of this view differs the most from the distribution of the real view in the protocol,

$$
\left\{(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \mathbf{m_1}, \mathbf{m_2^0}, \{\mathbf{m_2^j}\}_{j\in[d]}, \{\Pi_i\}_{i\in[\ell]}\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}
$$

This distribution can be shown to be indistinguishable from that of the real view using UOT by reducing to the indistinguishability of the real and simulated views of the bit-OT protocol **OT**, and the NIZK proof NIZK. Consider the following set of hybrids:

- Hybrid H_0. This is the distribution of the output of $\mathsf{Sim}_{\mathsf{UOT}}$. It differs from the view of the real execution in that all the messages in the transcript are simulated messages.

$$
\begin{aligned}
\mathsf{H}_0 = \big\{&(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \\
&\mathbf{m_1^*}, \mathbf{m_2^{0*}}, \{\mathbf{m_2^{j*}}\}_{j\in[d]}, \{\Pi_i^*\}_{i\in[\ell]}\big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}
\end{aligned}
$$

– Hybrid H_1. This is the distribution output by an intermediate hybrid experiment in which the sender S's message $\mathbf{m_2^0}$ is generated as in the real execution but everything else is generated as in $\mathsf{Sim}_{\mathsf{UOT}}$.

$$\mathsf{H}_1 = \big\{(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \\ \mathbf{m_1^*}, \mathbf{m_2^0}, \{\mathbf{m_2^{j*}}\}_{j\in[d]}, \{\Pi_i^*\}_{i\in[\ell]}\big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_2. This is the distribution output by an intermediate hybrid experiment in which both the chooser C's first message $\mathbf{m_1}$ and the sender S's message $\mathbf{m_2^0}$ is generated as in the real execution but everything else is generated as in $\mathsf{Sim}_{\mathsf{UOT}}$.

$$\mathsf{H}_2 = \big\{(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \\ \mathbf{m_1}, \mathbf{m_2^0}, \{\mathbf{m_2^{j*}}\}_{j\in[d]}, \{\Pi_i^*\}_{i\in[\ell]}\big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_{j+2}. For each $j \in [d]$, this is the distribution output by an intermediate hybrid experiment in which C's first message, S's message and the messages of the first j updaters are created as in the real execution but everything else is generated as in $\mathsf{Sim}_{\mathsf{UOT}}$.

$$\mathsf{H}_{j+2} = \big\{(s_0, s_1), \{\sigma_{j'}\}_{j'\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \\ \mathbf{m_1}, \mathbf{m_2^0}, \{\mathbf{m_2^{j'}}\}_{j'\le j}, \{\mathbf{m_2^{j'*}}\}_{j'>j\in[d]}, \{\Pi_i^*\}_{i\in[\ell]}\big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

The last of these hybrids has all the messages in the bit-OT protocol as messages from the real protocol.

– Hybrid H_{d+2+i}. For each $i \in [\ell]$, this is the distribution output by an intermediate hybrid experiment in which all the messages up to the last updater's message is generated as in the real execution of the protocol. In the chooser's final message, the first i instances of the NIZK proof is generated as in the real execution and the rest is generated as in the simulation.

$$\mathsf{H}_{d+2+i} = \big\{(s_0, s_1), \{\sigma_j\}_{j\in[d]}, b, r, \{s_b^j\}_{j\in[d]\cup\{0\}}, \\ \mathbf{m_1}, \mathbf{m_2^0}, \{\mathbf{m_2^j}\}_{j\in[d]}, \{\Pi_{i'}\}_{i'\le i}, \{\Pi_{i'}^*\}_{i'>i\in[\ell]}\big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

The last hybrid is the distribution as in the real protocol.

Claim 1 *Assuming that the bit-OT protocol* **OT** *is secure upon reusing the first OT message in the presence of a PPT malicious adversary adaptively corrupting the receiver, the views in hybrid distributions* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. The hybrids H_0 and H_1 differ only in the way that the honest sender's message is generated. In H_1 it is $\mathbf{m_2^0}$, generated as in the real execution of the protocol. In H_0 it is the simulated message $\mathbf{m_2^{0*}}$. This can be parsed as, $\mathbf{m_2^{0*}} = \{m_2^{i*}\}_{i\in[\ell]}$ where this is the output of the bit-OT simulator $\mathsf{Sim}_{\mathsf{multiOT}}$ when the receiver's message is $\mathbf{m_1^*}$ and the OT output is $\{a_b^i\}_{i\in[\ell]}$. Since the

bit-OT scheme **OT** is secure upon the reuse of the first OT message in the presence of a PPT malicious adversary adaptively corrupting the receiver, it follows that the distribution of the view produced by $\mathsf{Sim}_{\mathsf{multiOT}}$ is computationally indistinguishable from that of the real view. However, if there existed a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_0 and H_1 with non-negligible advantage ϵ, then $\mathcal{A}$ can be used in a black-box way as a subroutine by a PPT distinguisher D to distinguish between the output of $\mathsf{Sim}_{\mathsf{multiOT}}$ and the real OT view upon reuse of the first message.

D works by sending the challenger $\mathcal{C}$ the sender's OT input bits $\{a_0^i, a_1^i\}_{i\in[\ell]}$ and the OT first message $\mathbf{m_1^*}$. $\mathcal{C}$ replies with a message m that D uses to create the rest of the view as in the simulation $\mathsf{Sim}_{\mathsf{UOT}}$. If $m = \mathbf{m_2^{0*}}$ as output by $\mathsf{Sim}_{\mathsf{multiOT}}$, the view created is from H_0. Otherwise, if $m = \mathbf{m_2^0}$, the view created is from H_1. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution D has the same advantage as $\mathcal{A}$, which is non-negligible. However, since the OT protocol is secure against a corrupt receiver R when the first OT message is reused, no such D can exist and so no such $\mathcal{A}$ can exist.

Claim 2 *Assuming that the bit-OT protocol* **OT** *is secure against a PPT malicious adversary statically corrupting the sender, the views in hybrid distributions* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. If there exists a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_1 and H_2 with non-negligible advantage, then it can be used in a black-box way by a PPT distinguisher D that needs to distinguish between a real and simulated view of a corrupted sender S in the bit-OT protocol **OT**. D works by sending the challenger $\mathcal{C}$ the receiver's OT input bit b. $\mathcal{C}$ replies with a message m that D uses to create the rest of the view as in the simulation $\mathsf{Sim}_{\mathsf{UOT}}$. If $m = \mathbf{m_1^*}$, the view created is from H_1. Otherwise, if $m = \mathbf{m_1}$, the view created is from H_2. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution D has the same advantage as $\mathcal{A}$, which is non-negligible. However, since the OT protocol is secure against a corrupted sender S, no such D can exist and so no such $\mathcal{A}$ can exist.

Claim 3 *Assuming that the bit-OT protocol* **OT** *is secure against a PPT malicious adversary statically corrupting the receiver, the views in hybrid distributions* H_{j+2} *and* H_{j+1} *are computationally indistinguishable.*

Proof. For each $j \in [d]$, the hybrids H_{j+1} and H_{j+2} differ only in one honest updater's message. In H_{j+2} it is $\mathbf{m_2^j}$, generated as in the real execution of the protocol. In H_{j+1} it is the simulated message $\mathbf{m_2^{j*}}$. This can be parsed as, $\mathbf{m_2^{j*}} = \{m_2^{i*}\}_{i\in[\ell]}$ that is the output of the bit-OT protocol simulator $\mathsf{Sim}_{\mathsf{multiOT}}$ when the receiver's message is $\mathbf{m_1}$ and the OT output is $\{a_b^i\}_{i\in[\ell]}$. Like in the proof of Claim 1, since the bit-OT scheme **OT** is secure upon the reuse of the first OT message in the presence of a PPT malicious adversary statically corrupting the receiver, it follows that the distribution of the view produced by $\mathsf{Sim}_{\mathsf{multiOT}}$ is computationally indistinguishable from that of the real view. This is distributed identically to a view in which the message $\mathbf{m_2^j}$ is generated as in the real execution

of the UOT protocol owing to the fact that the second OT message in $\mathbf{OT}$ is rerandomizable. That is, the outputs of $\mathbf{OT}.\mathsf{OT}_3$ used in the execution of UOT is identically distributed to that of $\mathbf{OT}.\mathsf{OT}_2$ in the above distribution. Therefore, if there exists a PPT adversary $\mathcal{A}$ that can distinguish between H_{j+1} and H_{j+2} with non-negligible advantage, it can be used in a black-box way by a PPT distinguisher D to distinguish between the output of $\mathsf{Sim}_{\mathsf{multiOT}}$ and the real OT view upon reuse of the first message.

D works by sending the challenger $\mathcal{C}$ the sender's OT input bits $\{a_0^i, a_1^i\}_{i\in[\ell]}$ (that are the output after permuting using σ_j) and the OT first message $\mathbf{m_1}$. D also creates the messages $\mathbf{m_2^0}, \{\mathbf{m_2^{j'}}\}_{j'<j}$ as in the real execution of UOT. $\mathcal{C}$ replies with a message m that D uses to create the rest of the view as in the simulation $\mathsf{Sim}_{\mathsf{UOT}}$. If $m = \mathbf{m_2^{j*}}$ as output by $\mathsf{Sim}_{\mathsf{multiOT}}$, the view created is from H_{j+1}. Otherwise, if $m = \mathbf{m_2^j}$, the view created is from H_{j+2}. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution D has the same advantage as $\mathcal{A}$, which is non-negligible. However, since the OT protocol is secure against such a corrupt receiver R, no such D can exist and so no such $\mathcal{A}$ can exist.

Claim 4 *Assuming that the NIZK proof scheme* NIZK *is a computational zero-knowledge proof, the views in hybrid distributions* H_{d+2+i} *and* H_{d+1+i} *are computationally indistinguishable.*

Proof. If there exists a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_{d+2+i} and H_{d+1+i}, then $\mathcal{A}$ can be used in a black-box way by a PPT distinguisher D that distinguishes between a real NIZK proof Π_i and a simulated proof Π_i^* output by $\mathsf{Sim}_{\mathsf{NIZK}}$. D works by generating the view up to Π_{i-1} as in the real protocol execution of UOT. It sends the challenger $\mathcal{C}$ the NIZK inputs $a_b^i, \mathsf{Aux}, b, \mathbf{m_1}, \mathbf{m_2^i}$. $\mathcal{C}$ replies with a message m that D uses to create the rest of the view as in the simulation. If $m = \Pi_i^*$ as output by $\mathsf{Sim}_{\mathsf{NIZK}}$, the view created is from H_{d+1+i}. Otherwise, if $m = \Pi_i$, the view created is from H_{d+2+i}. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution D has the same advantage as $\mathcal{A}$, which is non-negligible. However, since the NIZK proof system is secure against a corrupt verifier, no such D can exist and so no such $\mathcal{A}$ can exist.

Since none of the listed set of hybrids is distinguishable, it follows that the distributions of the real view of the protocol UOT and the simulated view are computationally indistinguishable. Hence, the protocol UOT securely realizes $\mathcal{F}_{\mathsf{UOT}}$ in the presence of malicious adversaries in the SCALES model.

5.2 Σ-Protocols for Correct Function Encoding

Given a projective RGS (Definition 3) and a compatible UOT protocol UOT (Definition 8), it remains to check the relations $\mathcal{R}_{\mathsf{Gb}}, \mathcal{R}_{\mathsf{Rerand}}$ as follows:

$$\begin{aligned}
&\mathcal{R}_{\mathsf{Gb}}(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]}): \qquad f, \{\mathbf{m_1^i}\}_{i\in[m]}, \text{ are known}\\
&\quad w_0 = r_0, \{r_i\}_{i\in[m]} \quad \alpha_0 = F_0, \mathbf{OT}_0, \mathsf{CRS} \quad \{s_i^0, s_i^1\}_{i\in[m]} = \{L_{0,i}^0, L_{0,i}^1\}_{i\in[m]} = e_0\\
&\text{s.t.} \qquad \begin{cases} F_0, e_0 = \mathsf{RGS.Gb}(f; r_0) \\ \mathbf{OT}_0 = \{\mathbf{m_2^{0,i}} = \mathsf{UOT.OT}_2(\mathsf{CRS}; \mathbf{m_1^i}; L_{0,i}^0, L_{0,i}^1; r_i)\}_{i\in[m]} \end{cases}
\end{aligned} \tag{3}$$

$$
\begin{aligned}
&\mathcal{R}_{\mathsf{Rerand}}(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]}) : \\
&\quad w_j = r_j, \{r_i\}_{i\in[m]} \qquad \{\sigma_i^j\}_{i\in[m]} = \{\lambda_i\}_{i\in[m]} = \pi_j \\
&\qquad \alpha_j = F_j, \mathbf{OT}_j \qquad \alpha_{j-1} = F_{j-1}, \{\mathbf{m_2^{j-1,i}}\}_{i\in[m]}, \mathsf{CRS} \\
&\text{s.t.} \qquad \begin{cases} F_j, \pi_j = \mathsf{RGS.Rerand}(F_{j-1}; r_j) \\ \mathbf{OT}_j = \{\mathbf{m_2^{j,i}} = \mathsf{UOT.OT_{updt}}(\mathsf{CRS}; \mathbf{m_2^{j-1,i}}; \lambda_i; r_i)\}_{i\in[m]} \end{cases}
\end{aligned} \tag{4}
$$

Explainability. We define 'explainability' as a property of rerandomizable objects. Informally, an explainable RGS has a function $\mathsf{Explain}$ that creates randomness r explaining the rerandomized GC as a correct garbling of the function f, hiding the randomnesses used in garbling and rerandomizing individually.

Definition 9. *Let* $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ *be a projective RGS (Definition 3). Let* R *be the space of randomness for garbling and rerandomizing.* RGS *is* ***Explainable*** *if there exists a function* $\mathsf{Explain}(\cdot)$ *s.t. for all* $f \in \mathcal{F}$ *and* $x \in \{0,1\}^m$*,*

– ***Explainable Garbling Privacy****:*

$$
\begin{aligned}
&\{f, X, F, F', r^*\}_{r,r' \leftarrow R; (e,F)=\mathsf{Gb}(f;r); X=\mathsf{En}(e,x); F'=\mathsf{Rerand}(F;r'); r^*=\mathsf{Explain}(r,r')} \\
\overset{c}{\approx} &\{f, X, F, G, s\}_{r,s \leftarrow R; (e,F)=\mathsf{Gb}(f;r); X=\mathsf{En}(e,x); G=\mathsf{Gb}(f;s)}
\end{aligned}
$$

– ***Explainable Rerandomizing Privacy****:*

$$
\begin{aligned}
&\{F^*, X, F, F', r^*\}_{\substack{r,r' \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi)=\mathsf{Rerand}(F^*;r); X=\mathsf{En}(\pi(e),x); \\ F'=\mathsf{Rerand}(F;r'); r^*=\mathsf{Explain}(r,r')}} \\
\overset{c}{\approx} &\{F^*, X, F, G, s\}_{r,s \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi)=\mathsf{Rerand}(F^*;r); X=\mathsf{En}(\pi(e),x); G=\mathsf{Rerand}(F^*;s)}
\end{aligned}
$$

Lemma 2. *The RGC in Construction 1 is an Explainable RGS (Definition 9).*

A detailed proof for this can be found in Appendix A.3. Similarly to the above, we also define 'explainability' for a UOT protocol. Here, we require that the function $\mathsf{Explain}$ composes the sender and updater's randomnesses.

Definition 10. *A UOT protocol (Definition 7)* $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT_{updt}}, \mathsf{OT_{fin}}, \mathsf{OT_{vrfy}})$ *is* ***Explainable*** *if there exists a PPT function* $\mathsf{Explain}(\cdot)$ *s.t.* $\forall s_0, s_1 \in \mathcal{M}, \sigma \in \Sigma$*, and the space of randomness* R_{OT} *used in* OT_2 *and* $\mathsf{OT_{updt}}$*,*

– ***Update Explainability****:*

$$
\begin{aligned}
&\{\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_2^1}, r^*\}_{\substack{r,r' \leftarrow R_{\mathsf{OT}}; \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa); \mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b); \\ \mathbf{m_2}=\mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0,s_1); r); \mathbf{m_2^1}=\mathsf{OT_{updt}}(\mathsf{CRS}; \mathbf{m_2}, \sigma; r'); r^*=\mathsf{Explain}(r,r'; \sigma)}} \\
\overset{c}{\approx} &\{\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_2'}, s\}_{\substack{r,s \leftarrow R_{\mathsf{OT}}; \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa); \mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b); \\ \mathbf{m_2}=\mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0,s_1); r); \mathbf{m_2'}=\mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (\sigma(s_0), \sigma(s_1)); s)}}
\end{aligned}
$$

– ***Sequential Update Explainability****:*

$$
\begin{aligned}
&\{\mathbf{m_2}, \mathbf{m_2'}, \mathbf{m_2''}, r^*\}_{\substack{r,r' \leftarrow R_{\mathsf{OT}}; \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa); \mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b); \mathbf{m_2} \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0,s_1)); \\ \mathbf{m_2'}=\mathsf{OT_{updt}}(\mathsf{CRS}; \mathbf{m_2}, \sigma; r); \mathbf{m_2''}=\mathsf{OT_{updt}}(\mathsf{CRS}; \mathbf{m_2'}, \sigma'; r'); r^*=\mathsf{Explain}(r,r'; \sigma')}} \\
\overset{c}{\approx} &\{\mathbf{m_2}, \mathbf{m_2'}, \mathbf{m_2^*}, s\}_{\substack{r,s \leftarrow R_{\mathsf{OT}}; \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa); \mathbf{m_1} \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b); \mathbf{m_2} \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathbf{m_1}, (s_0,s_1)); \\ \mathbf{m_2'}=\mathsf{OT_{updt}}(\mathsf{CRS}; \mathbf{m_2}, \sigma; r); \mathbf{m_2^*}=\mathsf{OT_{updt}}(\mathsf{CRS}; \mathbf{m_2}, (\sigma \circ \sigma'); s)}}
\end{aligned}
$$

Lemma 3. *The UOT protocol in Figure 13 is Explainable (Definition 10).*

Proof. We already know from the proof of Lemma 1 that Figure 13 is an updatable OT protocol for sender's message space $\mathcal{M} \subset \{0,1\}^\ell$, the set of balanced binary strings, and the family of permutations on bit-positions $\mathcal{F} = \{\sigma : [\ell] \to [\ell]\}$ as the updaters' function family. Let $\mathbf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_3)$ be the underlying 2-round malicious secure bit-OT protocol (Definition 4). Consider the following function $\mathsf{Explain}$:

- parse the inputs $\sigma : [\ell] \to [\ell]$, $r = \{r_i\}_{i\in[\ell]}$ and $r' = \{r'_i\}_{i\in[\ell]}$
- for each $i \in [\ell]$, compute $r^*_i = r_{\sigma[i]} \diamond r'_i$
- output $r^* = \{r^*_i\}_{i\in[\ell]}$

Due to the functions in $\mathbf{OT}$, for any honestly created UOT first message $\mathbf{m_1}$,

$$\begin{aligned}&\mathsf{OT}_2(\mathbf{m_1}, (\sigma(s_0), \sigma(s_1)); \mathsf{Explain}(r, r'; \sigma))\\ =&\mathsf{OT}_{\mathsf{updt}}(\mathsf{OT}_2(\mathbf{m_1}, (s_0, s_1); r), \sigma; r')\end{aligned}$$

Furthermore, for any subsequent honestly created UOT message $\mathbf{m_2}$,

$$\begin{aligned}&\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2}, (\sigma \circ \sigma'); \mathsf{Explain}(r, r'; \sigma))\\ =&\mathsf{OT}_{\mathsf{updt}}(\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2}, \sigma; r), \sigma'; r')\end{aligned}$$

We now need to show that *update explainability* (Definition 10) holds:

$$\begin{aligned}&\{\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_2^1}, r^*\}_{\substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(b);\mathbf{m_2}=\mathsf{OT}_2(\mathbf{m_1},(s_0,s_1);r)\\ \mathbf{m_2^1}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2},\sigma;r');r^*=\mathsf{Explain}(r,r';\sigma)}}\\ \overset{c}{\approx}&\{\mathbf{m_1}, \mathbf{m_2}, \mathbf{m'_2}, s\}_{\substack{r,s\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(b);\mathbf{m_2}=\mathsf{OT}_2(\mathbf{m_1},(s_0,s_1);r)\\ \mathbf{m'_2}=\mathsf{OT}_2(\mathbf{m_1},(\sigma(s_0),\sigma(s_1));s)}}\end{aligned}$$

That is, for all inputs (b, s_0, s_1), given the first two messages $\mathbf{m_1}$ and $\mathbf{m_2}$ of the updatable OT protocol in Figure 13, a message $\mathbf{m_2^1}$ created by updating the second message, along with the composed randomness r^*, is indistinguishable from a fresh message $\mathbf{m'_2}$ using OT_2 and fresh randomness s. Both cases use the same s_0, s_1 and σ, but in the latter case, $\mathbf{m'_2}$ is independent of $\mathbf{m_2}$. These distributions are indistinguishable due to the fact that both $\mathbf{m_2^1} = \{m_2^{i'}\}_{i\in[\ell]}$ and $\mathbf{m'_2} = \{m_2^i\}_{i\in[\ell]}$ encode $(\sigma(s_0), \sigma(s_1))$ and it holds that for all $i \in [\ell]$, $m_2^{i'}$ and m_2^i come from the same distributions owing to the rerandomizing property of $\mathbf{OT}.\mathsf{OT}_3$ in the bit-OT construction. Their joint distributions with the respective randomnesses r^* and s are also indistinguishable. The randomness r' is sampled uniformly at random and so $r^* = \mathsf{Explain}(r, r'; \sigma)$ is also uniformly random in R_{OT}, regardless of how r is sampled. The freshly picked s is uniformly random as well. So both distributions given above are identically distributed.

It remains to show that *sequential update explainability* (Definition 10) also holds:

$$\begin{aligned}&\{\mathbf{m_2}, \mathbf{m'_2}, \mathbf{m''_2}, r^*\}_{\substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(b);\mathbf{m_2}\leftarrow\mathsf{OT}_2(\mathbf{m_1},(s_0,s_1));\\ \mathbf{m'_2}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2},\sigma;r);\mathbf{m''_2}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m'_2},\sigma';r');\\ r^*=\mathsf{Explain}(r,r';\sigma')}}\end{aligned}$$

$$\overset{c}{\approx}\{\mathbf{m_2}, \mathbf{m'_2}, \mathbf{m^*_2}, s\}_{r,s\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(b);\mathbf{m_2}\leftarrow\mathsf{OT}_2(\mathbf{m_1},(s_0,s_1));\ \mathbf{m'_2}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2},\sigma;r);\ \mathbf{m^*_2}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2},(\sigma\circ\sigma');s)}$$

That is, for all inputs (b, s_0, s_1, σ), given the message $\mathbf{m_2}$ and $\mathbf{m'_2}$ updated from it using the updatable OT protocol in Figure 13, a message $\mathbf{m''_2}$ created by updating $\mathbf{m'_2}$, along with the composed randomness r^*, is indistinguishable from a message $\mathbf{m^*_2}$ using $\mathsf{OT}_{\mathsf{updt}}$ on $\mathbf{m_2}$ and fresh randomness s. Both cases use the same σ and σ', but in the latter case, $\mathbf{m^*_2}$ is independent of $\mathbf{m'_2}$. These distributions are indistinguishable due to the fact that both $\mathbf{m''_2} = \{m_2^{i'}\}_{i\in[\ell]}$ and $\mathbf{m^*_2} = \{m_2^i\}_{i\in[\ell]}$ apply $(\sigma \circ \sigma')$ to $\mathbf{m_2}$ and it holds that for all $i \in [\ell]$, $m_2^{i'}$ and m_2^i come from the same distributions owing to the rerandomizing property of $\mathbf{OT}.\mathsf{OT}_3$ in the bit-OT construction. Their joint distributions with the respective randomnesses r^* and s are also indistinguishable. The randomness r' is sampled uniformly at random and so $r^* = \mathsf{Explain}(r, r'; \sigma)$ is also uniformly random, regardless of how r is sampled. The freshly picked s is uniformly random as well. This completes the proof.

Σ-Protocol for $\mathcal{R}_{\mathsf{Gb}}$. We present a Σ-Protocol for relation $\mathcal{R}_{\mathsf{Gb}}$ (Equation 3). Let $\mathsf{GS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be an explainable RGS (Definition 9). Let $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ be a compatible explainable UOT protocol (Definition 10). Consider $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ as in Figure 14.

Protocol Σ-protocol for $\mathcal{R}_{\mathsf{Gb}}$

The prover P has witness $w = (r_0, e_0, \{r_i\}_{i\in[m]})$. Both the prover P and verifier V have $\alpha = (f, \{\mathbf{m_1^i}\}_{i\in[m]}, F_0, \mathbf{OT}_0, \mathsf{CRS})$.

1. $P_1(\alpha, w)$:
   - sample r^* and $\{r_i^*\}_{i\in[m]}$ and compute $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) = \mathsf{RGS.Rerand}(F_0; r^*)$
   - parse $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ and compute $\{\mathbf{m_i^{proof}} = \mathsf{UOT.OT}_{\mathsf{updt}}(\mathsf{CRS}; \mathbf{m_2^{0,i}}; \sigma_i; r_i^*)\}_{i\in[m]}$
   - output $a = (F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]})$
2. $V_1(\cdot)$: output $b \leftarrow \{0, 1\}$
3. $P_2(a, b, \alpha, w)$:
   - if $b = 0$, output $c = (r^*, \{r_i^*\}_{i\in[m]})$
   - if $b = 1$, compute $r' = \mathsf{RGS.Explain}(r_0, r^*)$ and $\{r'_i = \mathsf{UOT.Explain}(r_i, r_i^*; \sigma_i)\}_{i\in[m]}$
     - output $c = (r', \{r'_i\}_{i\in[m]})$
4. $V_2(a, b, c, \alpha)$:
   - if $b = 0$,
     - output 1 if $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) == \mathsf{RGS.Rerand}(F_0; r^*)$, and $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ s.t. $\forall i \in [m]$, $\mathbf{m_i^{proof}} == \mathsf{UOT.OT}_{\mathsf{updt}}(\mathsf{CRS}; \mathbf{m_2^{0,i}}; \sigma_i; r_i^*)$; and 0 otherwise
   - if $b = 1$,
     - output 1 if $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}(e_0)) == \mathsf{RGS.Gb}(f; r')$, and $\pi_{\mathsf{proof}}(e_0) = \{\sigma_i(L_i^0), \sigma_i(L_i^1)\}_{i\in[m]}$ s.t. $\forall i \in [m]$, $\mathbf{m_i^{proof}} == \mathsf{UOT.OT}_2(\mathsf{CRS}; \mathbf{m_1^i}; \sigma_i(L_i^0), \sigma_i(L_i^1); r'_i)$; and 0 otherwise

Fig. 14 Σ-protocol for Correct Garbling

Lemma 4. *If* RGS *is an explainable RGS (Definition 9) and* UOT *is a compatible explainable UOT protocol (Definition 10) then Figure 14 is a Σ-protocol for the relation $\mathcal{R}_{\mathsf{Gb}}$ (Equation 3) with soundness error $\frac{1}{2}$.*

Proof. In Figure 14, Let $\mathsf{UOT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ be an explainable UOT protocol (Definition 10). Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be an explainable RGS (Definition 9). *Completeness* (Definition 6) holds since for a correct statement, both when $b = 0$ and $b = 1$, the verification will output 1 by construction of an Explainable RGS and Explainable UOT.

To show that *special soundness* (Definition 6) also holds, consider:

$$\begin{aligned} F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]} &= a \leftarrow P_1(\alpha, w) \\ r^*, \{r_i^*\}_{i\in[m]} &= c_0 \leftarrow P_2(a, 0, \alpha, w) \\ r', \{r_i'\}_{i\in[m]} &= c_1 \leftarrow P_2(a, 1, \alpha, w) \\ \text{where, } r' &= \mathsf{RGS.Explain}(r_0, r^*) \\ \forall i \in [m], r_i' &= \mathsf{UOT.Explain}(r_i, r_i^*, \sigma_i) \end{aligned}$$

and let the PPT function $\mathsf{Extract}(a, 0, c_0, 1, c_1, \alpha)$ works as follows:

- given r^* and r', compute the garbling randomness r_0
- for each $i \in [m]$, given r_i^* and r_i', compute the updatable OT randomness r_i
- compute $(F_0, e_0) = \mathsf{RGS.Gb}(f; r_0)$
- return $w = (r_0, e_0, \{r_i\}_{i\in[m]})$

It remains to show that Figure 14 satisfies *Special Honest-Verifier Zero Knowledge* (SHVZK - Definition 6). For this, let $\mathsf{Sim}(1^\kappa, \alpha, b)$ be a PPT algorithm that operates as follows:

- Parse $\alpha = (\{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}\}_{i\in[m]}, f, F_0)$.
- If $b = 0$, sample r^* and compute $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) \leftarrow \mathsf{RGS.Rerand}(F_0; r^*)$. Parse $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ and for all $i \in [m]$, sample r_i^* and compute $\mathbf{m_i^{proof}} = \mathsf{UOT.OT}_{\mathsf{updt}}(\mathbf{m_2^{0,i}}; \sigma_i; r_i^*)$. Output,

$$a = (F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]}), \quad b = 0, \quad c = (r^*, \{r_i^*\}_{i\in[m]})$$

- Otherwise if $b = 1$, sample s and compute $(F'_{\mathsf{proof}}, e'_{\mathsf{proof}}) \leftarrow \mathsf{RGS.Gb}(f; s)$. Parse $e'_{\mathsf{proof}} = \{L_i'^0, L_i'^1\}_{i\in[m]}$ and for all $i \in [m]$, sample s_i and compute $\mathbf{m_i'^{proof}} = \mathsf{UOT.OT}_2(\mathbf{m_1^i}; L_i'^0, L_i'^1; s_i)$. Output,

$$a = (F'_{\mathsf{proof}}, \{\mathbf{m_i'^{proof}}\}_{i\in[m]}), \quad b = 1, \quad c = (s, \{s_i\}_{i\in[m]})$$

When $b = 0$, the output of the simulation is exactly identical to the real execution of the Σ-protocol. When $b = 1$, the views are computationally indistinguishable. In the latter case, in both the real and simulated views $\alpha = (\{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}\}_{i\in[m]}, f, F_0)$. In the real execution, the first message a contains a rerandomized circuit F_{proof} and updated OT messages $\{\mathbf{m_i^{proof}}\}_{i\in[m]}$ and the last message c contains the composed randomnesses for both $(r^*, \{r_i^*\}_{i\in[m]})$. In the simulated view, a contains a fresh garbling F'_{proof} and OT messages $\{\mathbf{m_i'^{proof}}\}_{i\in[m]}$ created directly from the

first OT message, while c contains fresh randomness used for both the garbling and OT messages $(s, \{s_i\}_{i\in[m]})$. It remains to show that,

$$\left\{\{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}, \mathbf{m_i^{proof}}, r_i^*\}_{\substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1^i}\leftarrow \mathsf{OT}_1(x_i);\\ i\in[m], \mathbf{m_2^{0,i}}=\mathsf{OT}_2(\mathbf{m_1^i},(L_i^0,L_i^1);r);\\ \mathbf{m_i^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{0,i}},\sigma_i;r');\\ r_i^*=\mathsf{Explain}(r,r';\sigma_i)}}, \right.$$
$$\left.\{f, F_0, F_{\mathsf{proof}}, r^*\}_{\substack{r,r'\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_0;r');r^*=\mathsf{Explain}(r,r')}}\right\}$$
$$\overset{c}{\approx} \left\{\{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}, \mathbf{m_i'^{proof}}, s_i\}_{\substack{r,s_i\leftarrow R_{\mathsf{OT}};\mathbf{m_1^i}\leftarrow \mathsf{OT}_1(x_i);\\ i\in[m], \mathbf{m_2^{0,i}}=\mathsf{OT}_2(\mathbf{m_1^i},(L_i^0,L_i^1);r)\\ \mathbf{m_i'^{proof}}=\mathsf{OT}_2(\mathbf{m_1^i},(\sigma_i(L_i^0),\sigma_i(L_i^1));s_i)}}, \right.$$
$$\left.\{f, F_0, F'_{\mathsf{proof}}, s\}_{\substack{r,s\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F'_{\mathsf{proof}}=\mathsf{Gb}(f;s)}}\right\}$$

Consider the following sequence of hybrids:

- Hybrid H_0. This is the same distribution as the LHS of the above equation. It is distributed as the output of the real execution when $b = 1$.

$$\mathsf{H}_0 = \left\{\{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}, \mathbf{m_i^{proof}}, r_i^*\}_{\substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1^i}\leftarrow \mathsf{OT}_1(x_i);\\ i\in[m], \mathbf{m_2^{0,i}}=\mathsf{OT}_2(\mathbf{m_1^i},(L_i^0,L_i^1);r);\\ \mathbf{m_i^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{0,i}},\sigma_i;r');\\ r_i^*=\mathsf{Explain}(r,r';\sigma_i)}}, \right.$$
$$\left.\{f, F_0, F_{\mathsf{proof}}, r^*\}_{\substack{r,r'\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_0;r');r^*=\mathsf{Explain}(r,r')}}\right\}$$

- Hybrid H_i. For each $i \in [m]$, this is an intermediate hybrid which is distributed as the output of the real execution when $b = 1$, up to the $m - i^{th}$ updated OT message $\mathbf{m_i^{proof}}$. All the UOT messages beyond this are generated as in the simulation.

$$\mathsf{H}_i = \left\{\{\mathbf{m_1^{i'}}, \mathbf{m_2^{0,i'}}, \mathbf{m_{i'}^{proof}}, r_{i'}^*\}_{\substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1^{i'}}\leftarrow \mathsf{OT}_1(x_{i'});\\ i'\leq m-i, \mathbf{m_2^{0,i'}}=\mathsf{OT}_2(\mathbf{m_1^{i'}},(L_{i'}^0,L_{i'}^1);r);\\ \mathbf{m_{i'}^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{0,i'}},\sigma_{i'};r');\\ r_{i'}^*=\mathsf{Explain}(r,r';\sigma_{i'})}}, \right.$$
$$\{\mathbf{m_1^{i'}}, \mathbf{m_2^{0,i'}}, \mathbf{m_{i'}'^{proof}}, s_{i'}\}_{\substack{r,s_{i'}\leftarrow R_{\mathsf{OT}};\mathbf{m_1^{i'}}\leftarrow \mathsf{OT}_1(x_{i'});\\ i'>m-i\in[m], \mathbf{m_2^{0,i'}}=\mathsf{OT}_2(\mathbf{m_1^{i'}},(L_{i'}^0,L_{i'}^1);r)\\ \mathbf{m_{i'}'^{proof}}=\mathsf{OT}_2(\mathbf{m_1^{i'}},(\sigma_{i'}(L_{i'}^0),\sigma_{i'}(L_{i'}^1));s_{i'})}},$$
$$\left.\{f, F_0, F_{\mathsf{proof}}, r^*\}_{\substack{r,r'\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_0;r');r^*=\mathsf{Explain}(r,r')}}\right\}$$

Note that the last of these hybrids has all UOT messages in the 3^{rd} message of the proof as messages generated in the simulation.

$$\mathsf{H}_m = \Bigg\{ \{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}, \mathbf{m_i'^{proof}}, s_i\}_{\substack{i\in[m],\\ r,s_i\leftarrow R_{\mathsf{OT}};\mathbf{m_1^i}\leftarrow\mathsf{OT}_1(x_i);\\ \mathbf{m_2^{0,i}}=\mathsf{OT}_2(\mathbf{m_1^i},(L_i^0,L_i^1);r)\\ \mathbf{m_i'^{proof}}=\mathsf{OT}_2(\mathbf{m_1^i},(\sigma_i(L_i^0),\sigma_i(L_i^1));s_i)}} , \{f, F_0, F_{\mathsf{proof}}, r^*\}_{\substack{r,r'\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_0;r');r^*=\mathsf{Explain}(r,r')}} \Bigg\}$$

– Hybrid H_{m+1}. This is the last hybrid that is distributed as the RHS of the distribution above and is the output of the simulator Sim when $b = 1$. This differs from the previous hybrid in that the garbling in the 3^{rd} message of the proof is a fresh garbling of f using independent randomness s.

$$\mathsf{H}_{m+1} = \Bigg\{ \{\mathbf{m_1^i}, \mathbf{m_2^{0,i}}, \mathbf{m_i'^{proof}}, s_i\}_{\substack{i\in[m],\\ r,s_i\leftarrow R_{\mathsf{OT}};\mathbf{m_1^i}\leftarrow\mathsf{OT}_1(x_i);\\ \mathbf{m_2^{0,i}}=\mathsf{OT}_2(\mathbf{m_1^i},(L_i^0,L_i^1);r)\\ \mathbf{m_i'^{proof}}=\mathsf{OT}_2(\mathbf{m_1^i},(\sigma_i(L_i^0),\sigma_i(L_i^1));s_i)}} , \{f, F_0, F'_{\mathsf{proof}}, s\}_{\substack{r,s\leftarrow R; F_0=\mathsf{Gb}(f;r);\\ F'_{\mathsf{proof}}=\mathsf{Gb}(f;s)}} \Bigg\}$$

Claim 5 *Assuming that* UOT *is an explainable UOT protocol (Definition 10) satisfying update explainability, then for all* $i \in [m]$*, the hybrid distributions* H_i *and* H_{i-1} *are computationally indistinguishable.*

Proof. Note that the hybrid distributions H_i and H_{i-1} differ only in that in H_{i-1}, proof message $\mathbf{m'^{proof}_{m-i}}$ is computed directly from $\mathbf{m_1^{m-i}}$ using fresh randomness s_{m-i}, as in the simulation. In H_i, proof message $\mathbf{m^{proof}_{m-i}}$ is computed as an update from $\mathbf{m_2^{0,m-i}}$ using randomness r^*_{m-i}, as in the real execution. If there existed a PPT adversary $\mathcal{A}$ that can distinguish between these hybrids with non-negligible advantage, then this can be used in a black-box way by a PPT distinguisher D to violate *update explainability* of UOT.

D knows RGS, UOT, the function f, input x and the index i. It computes F_0, F_{proof} and r^* as in the real execution of the Σ-protocol for when $b = 1$. Then for all $i' < m - i$, it computes $(\mathbf{m_1^{i'}}, \mathbf{m_2^{0,i'}}, \mathbf{m_{i'}^{proof}}, r^*_{i'})$ using e_0 and π_{proof} as in the real execution as well. For all $i' > m - i \in [m]$, it computes $(\mathbf{m_1^{i'}}, \mathbf{m_2^{0,i'}}, \mathbf{m_{i'}'^{proof}}, s_{i'})$ as in the simulation. It then gives the tuple $(\mathbf{m_1^{m-i}}, \mathbf{m_2^{0,m-i}}, x_{m-i} \in x, (L^0_{m-i}, L^1_{m-i}) \in e_0, \sigma_{m-i} \in \pi_{\mathsf{proof}})$ to the challenger $\mathcal{C}$, along with the randomness r used to create $\mathbf{m_2^{0,m-i}}$ from $\mathbf{m_1^{m-i}}$. The challenger then returns a message m. If $m = (\mathbf{m_1^{m-i}}, \mathbf{m_2^{0,m-i}}, \mathbf{m^{proof}_{m-i}}, r^*_{m-i})$, then this tuple, along with the view already generated, would belong to the hybrid distribution H_i. Otherwise, if $m = (\mathbf{m_1^{m-i}}, \mathbf{m_2^{0,m-i}}, \mathbf{m'^{proof}_{m-i}}, s_{m-i})$, then this would belong to the hybrid distribution H_{i-1}. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution, D would have the same advantage as $\mathcal{A}$, which is non-negligible. However, since UOT is an explainable UOT protocol and *update explainability* holds, it follows that no such D can exist and so no such $\mathcal{A}$ can exist.

Claim 6 *Assuming that* RGS *is an explainable RGS (Definition 9) satisfying explainable garbling privacy, then the hybrid distributions* H_m *and* H_{m+1} *are computationally indistinguishable.*

Proof. Note that the hybrid distributions H_m and H_{m+1} differ only in that in H_{m+1}, the proof garbling F'_{proof} is computed directly from f using fresh randomness s, as in the simulation. In H_m, the proof garbling F_{proof} is computed by rerandomizing F_0 and r^* is the composed randomness as in the real execution. If there existed a PPT adversary $\mathcal{A}$ that can distinguish between these hybrids with non-negligible advantage, then this can be used in a black-box way by a PPT distinguisher D to violate *explainable garbling privacy* of RGS.

D knows RGS, UOT and the function f and input x. It computes the garbling F_0 and sends the tuple (f, F_0, e_0, x) to the challenger $\mathcal{C}$ along with the garbling randomness r. The challenger then returns a message m containing f, F_0, X another garbling F' and randomness r' such that $(F', e') \leftarrow \mathsf{Gb}(f; r')$. D uses e' to generate the UOT messages for the proof as fresh OT sender's messages derived from $\{\mathbf{m_1^i}\}_{i\in[m]}$. All these, except X, complete view is sent to $\mathcal{A}$ and D outputs whatever $\mathcal{A}$ outputs. Note that if $(F', r') == (F'_{\mathsf{proof}}, s)$ is a fresh garbling of f, then the above distribution is as in H_{m+1}. Otherwise, if $(F', r') == (F_{\mathsf{proof}}, r^*)$ is a rerandomized garbling from F_0 and the composed randomness, the distribution is as in hybrid H_m. In this execution, D would have the same advantage as $\mathcal{A}$, which is non-negligible. However, since RGS is an explainable RGS and *explainable garbling privacy* holds, it follows that no such D or $\mathcal{A}$ can exist.

Since none of the adjacent pairs of hybrids listed above are distinguishable and there are polynomial number of hybrids in the security parameter κ, it follows that the distributions of the real and simulated executions are computationally indistinguishable.

Σ-Protocol for $\mathcal{R}_{\mathsf{Rerand}}$. For an explainable RGS (Definition 9) and compatible explainable UOT protocol (Definition 10), consider the Σ-protocol in Figure 15.

Protocol Σ-protocol for $\mathcal{R}_{\mathsf{Rerand}}$

The prover P has witness $w = (r_j, \pi_j, \{r_i\}_{i\in[m]})$. Both the prover P and verifier V have $\alpha = (F_{j-1}, \{\mathbf{m_2^{j-1,i}}\}_{i\in[m]}, F_j, \mathbf{OT}_j, \mathsf{CRS})$.

1. $P_1(\alpha, w)$:
   - sample r^* and $\{r_i^*\}_{i\in[m]}$ and compute $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) = \mathsf{RGS.Rerand}(F_j; r^*)$
   - parse $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ and compute $\{\mathbf{m_i^{proof}} = \mathsf{UOT.OT_{updt}}(\mathsf{CRS}; \mathbf{m_2^{j,i}}; \sigma_i; r_i^*)\}_{i\in[m]}$
   - output $a = (F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]})$
2. $V_1(\cdot)$: output $b \leftarrow \{0, 1\}$
3. $P_2(a, b, \alpha, w)$:
   - if $b = 0$, output $c = (r^*, \{r_i^*\}_{i\in[m]})$
   - if $b = 1$, compute $r' = \mathsf{RGS.Explain}(r_j, r^*)$ and $\{r'_i = \mathsf{UOT.Explain}(r_i, r_i^*; \sigma_i)\}_{i\in[m]}$
     - output $c = (r', \{r'_i\}_{i\in[m]})$
4. $V_2(a, b, c, \alpha)$:
   - if $b = 0$,
     - output 1 if $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) == \mathsf{RGS.Rerand}(F_j; r^*)$, and $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ s.t. $\forall i \in [m]$, $\mathbf{m_i^{proof}} == \mathsf{UOT.OT_{updt}}(\mathsf{CRS}; \mathbf{m_2^{j,i}}; \sigma_i; r_i^*)$
     - otherwise, output 0
   - if $b = 1$,

- output 1 if $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}(\pi_j)) == \mathsf{RGS.Rerand}(F_{j-1}; r')$, and $\pi_{\mathsf{proof}}(\pi_j) = \{\sigma'_i \circ \sigma_i\}_{i\in[m]}$ s.t. $\forall i \in [m]$, $\mathbf{m_i^{proof}} == \mathsf{UOT.OT_{updt}}(\mathsf{CRS}; \mathbf{m_2^{j-1,i}}; \sigma'_i \circ \sigma_i; r'_i)$
- otherwise, output 0

Fig. 15 Σ-protocol for Correct Rerandomizing

Lemma 5. *If* RGS *is an explainable RGS (Definition 9) and* UOT *is a compatible explainable UOT protocol (Definition 10) then Figure 15 is a Σ-protocol for the relation $\mathcal{R}_{\mathsf{Rerand}}$ (Equation 4) with soundness error $\frac{1}{2}$.*

Proof. In Figure 15, Let $\mathsf{UOT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT_{updt}}, \mathsf{OT_{fin}}, \mathsf{OT_{vrfy}})$ be an explanable UOT protocol (Definition 10). Let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ be an explanable RGS (Definition 9). *Completeness* (Definition 6) holds since for a correct statement, both when $b = 0$ and $b = 1$, the verification will output 1 by construction of Explanable RGS and Explanable UOT.

To show that *special soundness* (Definition 6) also holds, consider:

$$\begin{aligned} F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]} &= a \leftarrow P_1(\alpha, w) \\ r^*, \{r_i^*\}_{i\in[m]} &= c_0 \leftarrow P_2(a, 0, \alpha, w) \\ r', \{r'_i\}_{i\in[m]} &= c_1 \leftarrow P_2(a, 1, \alpha, w) \\ \text{where, } r' &= \mathsf{RGS.Explain}(r, r^*) \\ \forall i \in [m], r'_i &= \mathsf{UOT.Explain}(r_i, r_i^*, \sigma_i) \end{aligned}$$

and let the PPT function $\mathsf{Extract}(a, 0, c_0, 1, c_1, \alpha)$ works as follows:

- given r^* and r', compute r_j
- for each $i \in [m]$, given r_i^* and r'_i, compute the updatable OT randomness r_i
- compute $(F_j, \pi_j) = \mathsf{RGS.Rerand}(F_{j-1}; r_j)$
- return $w = (r_j, \pi_j, \{r_i\}_{i\in[m]})$

It remains to show that Figure 15 satisfies *Special Honest-Verifier Zero Knowledge* (SHVZK - Definition 6). For this, let $\mathsf{Sim}(1^\kappa, \alpha, b)$ be a PPT algorithm that operates as follows:

- Parse $\alpha = (\{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}\}_{i\in[m]}, F_{j-1}, F_j)$
- If $b = 0$, sample r^* and compute $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) \leftarrow \mathsf{RGS.Rerand}(F_j; r^*)$. Parse $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ and for each $i \in [m]$, sample r_i^* and compute $\mathbf{m_i^{proof}} = \mathsf{UOT.OT_{updt}}(\mathbf{m_2^{j,i}}; \sigma_i; r_i^*)$. Output,

$$a = (F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]}), \quad b = 0, \quad c = (r^*, \{r_i^*\}_{i\in[m]})$$

- Otherwise if $b = 1$, sample s and compute $(F_{\mathsf{proof}}, \pi_{\mathsf{proof}}) \leftarrow \mathsf{RGS.Rerand}(F_{j-1}; s)$. Parse $\pi_{\mathsf{proof}} = \{\sigma_i\}_{i\in[m]}$ and for each $i \in [m]$, sample s_i and compute $\mathbf{m_i^{proof}} = \mathsf{UOT.OT_{updt}}(\mathbf{m_2^{j-1,i}}; \sigma_i; s_i)$. Output,

$$a = (F_{\mathsf{proof}}, \{\mathbf{m_i^{proof}}\}_{i\in[m]}), \quad b = 1, \quad c = (s, \{s_i\}_{i\in[m]})$$

When $b = 0$, the simulation is exactly identical to the real execution of the Σ-protocol. When $b = 1$, the views are computationally indistinguishable. In the latter case, in both the real and simulated views the statement is $\alpha = (\{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}\}_{i\in[m]}, F_{j-1}, F_j)$. In the real execution, the first message a contains a garbling F_{proof} rerandomized from F_j and OT messages $\{\mathbf{m_i^{proof}}\}_{i\in[m]}$ updated from $\{\mathbf{m_2^{j,i}}\}_{i\in[m]}$. The last message c contains the composed randomnesses for both $(r^*, \{r_i^*\}_{i\in[m]})$. In the simulated view, a contains F'_{proof} rerandomized from F_{j-1} and OT messages $\{\mathbf{m_i'^{proof}}\}_{i\in[m]}$ created directly from $\{\mathbf{m_2^{j-1,i}}\}_{i\in[m]}$, while c contains fresh randomness used for both the garbling and OT messages $(s, \{s_i\}_{i\in[m]})$. It remains to show that,

$$\left\{\{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}, \mathbf{m_i^{proof}}, r_i^*\}_{i\in[m],\ \substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(x_i);\\ \mathbf{m_2^{j-1,i}}\leftarrow\mathsf{OT}_2(\mathbf{m_1},(L^0_{i,j-1},L^1_{i,j-1}));\\ \mathbf{m_2^{j,i}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}},\sigma_i^j;r);\\ \mathbf{m_i^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j,i}},\sigma_i;r');\\ r_i^*=\mathsf{Explain}(r,r';\sigma_i)}} ,\right.$$
$$\left.\{F_{j-1}, F_j, F_{\mathsf{proof}}, r^*\}_{r,r'\leftarrow R;F_{j-1}\leftarrow\mathsf{Gb}(f);F_j=\mathsf{Rerand}(F_{j-1};r);\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_j;r');r^*=\mathsf{Explain}(r,r')}\right\}$$
$$\overset{c}{\approx}\left\{\{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}, \mathbf{m_i'^{proof}}, s_i\}_{i\in[m],\ \substack{r,s_i\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(x_i);\\ \mathbf{m_2^{j-1,i}}\leftarrow\mathsf{OT}_2(\mathbf{m_1},(L^0_{i,j-1},L^1_{i,j-1}));\\ \mathbf{m_2^{j,i}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}},\sigma_i^j;r);\\ \mathbf{m_i'^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}},(\sigma_i^j\circ\sigma_i);s_i)}} ,\right.$$
$$\left.\{F_{j-1}, F_j, F'_{\mathsf{proof}}, s\}_{r,s\leftarrow R;F_{j-1}\leftarrow\mathsf{Gb}(f);F_j=\mathsf{Rerand}(F_{j-1};r);\ F'_{\mathsf{proof}}=\mathsf{Rerand}(F_{j-1};s)}\right\}$$

Consider the following sequence of hybrids:

– Hybrid H_0. This is the same distribution as the LHS of the above equation. It is distributed as the output of the real execution when $b = 1$.

$$\mathsf{H}_0 = \left\{\{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}, \mathbf{m_i^{proof}}, r_i^*\}_{i\in[m],\ \substack{r,r'\leftarrow R_{\mathsf{OT}};\mathbf{m_1}\leftarrow\mathsf{OT}_1(x_i);\\ \mathbf{m_2^{j-1,i}}\leftarrow\mathsf{OT}_2(\mathbf{m_1},(L^0_{i,j-1},L^1_{i,j-1}));\\ \mathbf{m_2^{j,i}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}},\sigma_i^j;r);\\ \mathbf{m_i^{proof}}=\mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j,i}},\sigma_i;r');\\ r_i^*=\mathsf{Explain}(r,r';\sigma_i)}} ,\right.$$
$$\left.\{F_{j-1}, F_j, F_{\mathsf{proof}}, r^*\}_{r,r'\leftarrow R;F_{j-1}\leftarrow\mathsf{Gb}(f);F_j=\mathsf{Rerand}(F_{j-1};r);\ F_{\mathsf{proof}}=\mathsf{Rerand}(F_j;r');r^*=\mathsf{Explain}(r,r')}\right\}$$

– Hybrid H_i. For each $i \in [m]$, this is an intermediate hybrid which is distributed as the output of the real execution when $b = 1$, up to the $m - i^{th}$ updated OT message $\mathbf{m_i^{proof}}$. All the UOT messages beyond this are generated as in

the simulation.

$$\mathsf{H}_i = \Bigg\{ \{\mathbf{m_2^{j-1,i'}}, \mathbf{m_2^{j,i'}}, \mathbf{m_{i'}^{proof}}, r^*_{i'}\} \quad \substack{r,r' \leftarrow R_{\mathsf{OT}}; \mathbf{m_1} \leftarrow \mathsf{OT}_1(x_{i'}); \\ \mathbf{m_2^{j-1,i'}} \leftarrow \mathsf{OT}_2(\mathbf{m_1}, (L^0_{i',j-1}, L^1_{i',j-1})); \\ i' \le m-i, \quad \mathbf{m_2^{j,i'}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i'}}, \sigma^j_{i'}; r); \\ \mathbf{m_{i'}^{proof}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j,i'}}, \sigma_{i'}; r'); \\ r^*_{i'} = \mathsf{Explain}(r, r'; \sigma_{i'})} ,$$

$$\{\mathbf{m_2^{j-1,i'}}, \mathbf{m_2^{j,i'}}, \mathbf{m_{i'}^{\prime proof}}, s_{i'}\} \quad \substack{r, s_{i'} \leftarrow R_{\mathsf{OT}}; \mathbf{m_1} \leftarrow \mathsf{OT}_1(x_{i'}); \\ \mathbf{m_2^{j-1,i'}} \leftarrow \mathsf{OT}_2(\mathbf{m_1}, (L^0_{i',j-1}, L^1_{i',j-1})); \\ i' > m-i \in [m], \quad \mathbf{m_2^{j,i'}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i'}}, \sigma^j_{i'}; r); \\ \mathbf{m_{i'}^{\prime proof}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i'}}, (\sigma^j_{i'} \circ \sigma_{i'}); s_{i'})} ,$$

$$\{F_{j-1}, F_j, F_{\mathsf{proof}}, r^*\}_{\substack{r,r' \leftarrow R; F_{j-1} \leftarrow \mathsf{Gb}(f); F_j = \mathsf{Rerand}(F_{j-1}; r); \\ F_{\mathsf{proof}} = \mathsf{Rerand}(F_j; r'); r^* = \mathsf{Explain}(r, r')}} \Bigg\}$$

Note that the last of these hybrids has all UOT messages in the 3^{rd} message of the proof as messages generated in the simulation.

$$\mathsf{H}_m = \Bigg\{ \{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}, \mathbf{m_i^{\prime proof}}, s_i\} \quad \substack{r, s_i \leftarrow R_{\mathsf{OT}}; \mathbf{m_1} \leftarrow \mathsf{OT}_1(x_i); \\ \mathbf{m_2^{j-1,i}} \leftarrow \mathsf{OT}_2(\mathbf{m_1}, (L^0_{i,j-1}, L^1_{i,j-1})); \\ i \in [m], \quad \mathbf{m_2^{j,i}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}}, \sigma^j_i; r); \\ \mathbf{m_i^{\prime proof}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}}, (\sigma^j_i \circ \sigma_i); s_i)}$$

$$\{F_{j-1}, F_j, F_{\mathsf{proof}}, r^*\}_{\substack{r,r' \leftarrow R; F_{j-1} \leftarrow \mathsf{Gb}(f); F_j = \mathsf{Rerand}(F_{j-1}; r); \\ F_{\mathsf{proof}} = \mathsf{Rerand}(F_j; r'); r^* = \mathsf{Explain}(r, r')}} \Bigg\}$$

– Hybrid H_{m+1}. This is the last hybrid that is distributed as the RHS of the distribution above and is the output of the simulator Sim when $b = 1$. This differs from the previous hybrid in that the garbling in the 3^{rd} message of the proof is a rerandomized garbling from F_{j-1} using independent randomness s.

$$\mathsf{H}_{m+1} = \Bigg\{ \{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}, \mathbf{m_i^{\prime proof}}, s_i\} \quad \substack{r, s_i \leftarrow R_{\mathsf{OT}}; \mathbf{m_1} \leftarrow \mathsf{OT}_1(x_i); \\ \mathbf{m_2^{j-1,i}} \leftarrow \mathsf{OT}_2(\mathbf{m_1}, (L^0_{i,j-1}, L^1_{i,j-1})); \\ i \in [m], \quad \mathbf{m_2^{j,i}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}}, \sigma^j_i; r); \\ \mathbf{m_i^{\prime proof}} = \mathsf{OT}_{\mathsf{updt}}(\mathbf{m_2^{j-1,i}}, (\sigma^j_i \circ \sigma_i); s_i)}$$

$$\{F_{j-1}, F_j, F'_{\mathsf{proof}}, s\}_{\substack{r,s \leftarrow R; F_{j-1} \leftarrow \mathsf{Gb}(f); F_j = \mathsf{Rerand}(F_{j-1}; r); \\ F'_{\mathsf{proof}} = \mathsf{Rerand}(F_{j-1}; s)}} \Bigg\}$$

Claim 7 *Assuming that* UOT *is an explainable UOT protocol (Definition 10) satisfying explainable update privacy, then for all* $i \in [m]$*, the hybrid distributions* H_i *and* H_{i-1} *are computationally indistinguishable.*

Proof. Note that the hybrid distributions H_i and H_{i-1} differ only in that in H_{i-1}, proof message $\mathbf{m_{m-i}^{\prime proof}}$ is computed directly from $\mathbf{m_2^{j-1,m-i}}$ using fresh randomness s_{m-i}, as in the simulation. In H_i, proof message $\mathbf{m_{m-i}^{proof}}$ is computed as an update from $\mathbf{m_2^{j,m-i}}$ using randomness r^*_{m-i}, as in the real execution. If

there existed a PPT adversary $\mathcal{A}$ that can distinguish between these hybrids with non-negligible advantage, then this can be used in a black-box way by a PPT distinguisher D to violate *sequential update explainability* of UOT.

D knows RGS, UOT, the prior garbling F_{j-1}, $\{\mathbf{m_2^{j-1,i'}}\}_{i' \in [m]}$, the input x and the index i. It computes F_j, F_{proof} and r^* as in the real execution of the Σ-protocol for when $b = 1$. Then for all $i' < m - i$, it computes $(\mathbf{m_2^{j-1,i'}}, \mathbf{m_2^{j,i'}}, \mathbf{m_{i'}^{proof}}, r^*_{i'})$ using π_j and π_{proof} as in the real execution as well. For all $i' > m - i \in [m]$, it computes $(\mathbf{m_2^{j-1,i'}}, \mathbf{m_2^{j,i'}}, \mathbf{m'^{proof}_{i'}}, s_{i'})$ as in the simulation. It then gives the tuple $(\mathbf{m_2^{j-1,m-i}}, \mathbf{m_2^{j,m-i}}, \sigma^j_{m-i} \in \pi_j, \sigma_{m-i} \in \pi_{\mathsf{proof}}, x_{m-i} \in x, L^{x_{m-i}}_{m-i,j-1})$ to the challenger $\mathcal{C}$, along with the randomness r used to create $\mathbf{m_2^{j,m-i}}$ from $\mathbf{m_2^{j-1,m-i}}$. The challenger then returns a message m. If $m = (\mathbf{m_2^{j-1,m-i}}, \mathbf{m_2^{j,m-i}}, \mathbf{m^{proof}_{m-i}}, r^*_{m-i})$, then this, along with the view already generated would belong to the hybrid distribution H_i. Otherwise, if $m = (\mathbf{m_2^{j-1,m-i}}, \mathbf{m_2^{j,m-i}}, \mathbf{m'^{proof}_{m-i}}, s_{m-i})$, then this, along with the view already generated would belong to the hybrid distribution H_{i-1}. D gives this to $\mathcal{A}$ and outputs whatever it outputs. In this execution, D would have the same advantage as $\mathcal{A}$, which is non-negligible. However, since UOT is an explainable UOT protocol and *sequential update explainability* holds, it follows that no such D can exist and so no such $\mathcal{A}$ can exist.

Claim 8 *Assuming that* RGS *is an explainable RGS (Definition 9) satisfying sequential update explainability, then the hybrid distributions* H_m *and* H_{m+1} *are computationally indistinguishable.*

Proof. Note that the hybrid distributions H_m and H_{m+1} differ only in that in H_{m+1}, the proof garbling F'_{proof} is computed directly from F_{j-1} using fresh randomness s, as in the simulation. In H_m, the proof garbling F_{proof} is computed by rerandomizing F_j and r^* is the composed randomness as in the real execution. If there existed a PPT adversary $\mathcal{A}$ that can distinguish between these hybrids with non-negligible advantage, then this can be used in a black-box way by a PPT distinguisher D to violate *explainable rerandomizing privacy* of RGS.

D knows RGS, UOT and the prior garbling F_{j-1}, input x and input encoding information e_{j-1}. It computes the garbling F_j and sends the tuple $(F_{j-1}, x, e_{j-1}, F_j, \pi_j)$ to the challenger $\mathcal{C}$ along with the rerandomizing randomness r. The challenger then returns a message m containing F_{j-1}, F_j, X_j another garbling F' and randomness r' such that $(F', \pi') \leftarrow \mathsf{Rerand}(F_{j-1}; r')$. D uses π' to generate the UOT messages for the proof as fresh OT updater's messages derived from $\{\mathbf{m_2^{j-1,i}}\}_{i \in [m]}$. This complete view, except X_j, is sent to $\mathcal{A}$ and D outputs whatever $\mathcal{A}$ outputs. Note that if $(F', r') == (F'_{\mathsf{proof}}, s)$ is rerandomized from F_{j-1}, then the above distribution is as in H_{m+1}. Otherwise, if $(F', r') == (F_{\mathsf{proof}}, r^*)$ is a rerandomized garbling from F_j and the composed randomness, the distribution is as in hybrid H_m. In this execution, D would have the same advantage as $\mathcal{A}$, which is non-negligible. However, since RGS is an explainable RGS and *explainable rerandomizing privacy* holds, it follows that no such D can exist and so no such $\mathcal{A}$ can exist.

Distributed Committed-Index OT $\mathcal{F}_{\mathsf{DComOT}}$

Let $\{S_j\}_{j\in[d]\cup\{0\}}$ be the senders where each S_j has strings (s_0^j, s_1^j). Let $\{C_i\}_{i\in[m]}$ be choosers where each C_i has bit b_i. Let $\mathsf{publish}$ denote the action of $\mathcal{F}_{\mathsf{DComOT}}$ making a message visible to all parties. $\mathcal{F}_{\mathsf{DComOT}}$ works as follows:

1. Choosers' Commit Phase:
   - $\forall i \in [m]$, receive input b_i from C_i
2. $\forall j \in [d] \cup \{0\}$, Senders' Commit Phase:
   - receive input (s_0^j, s_1^j) from S_j
3. Reveal Phase:
   - $\mathsf{publish}$ $(\{s_b^j\}_{j\in[d]}, \{b_i\}_{i\in[m]})$, where $b = \bigoplus_{i=0}^{m} b_i$

Fig. 16 Distributed Committed-Index OT

Since none of the adjacent pairs of hybrids listed above are distinguishable and there are polynomial number of hybrids in the security parameter κ, it follows that the distributions of the real and simulated executions are computationally indistinguishable.

5.3 Distributed Committed-Index Oblivious Transfer

In this section we define and build the last key building-block for $\mathcal{F}_{\mathsf{VUOT}}$: a 'Distributed Committed-Index OT' (DCOT) $\mathcal{F}_{\mathsf{DComOT}}$. This is a functionality between $d+1$ senders and m choosers. First, each chooser C_i samples a random bit b_i and inputs to the functionality. Each sender S_j gives 2 strings s_0^j, s_1^j as its input and the goal is for the functionality to make publicly visible: s_b^j, for a choice bit b that combines the bits of all the choosers: $b = \bigoplus_{i=0}^{m} b_i$. Figure 16 describes this functionality $\mathcal{F}_{\mathsf{DComOT}}$. We need a protocol for this where each server, acting as a sender, speaks once and the clients, acting as choosers, post at most one message before and after, followed by public verification and decoding.

Definition 11. *The tuple* $\mathsf{DComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ *is a* ***Maliciously Secure SCALES Protocol for DCOT*** *if the following protocol UC-securely realizes* $\mathcal{F}_{\mathsf{DComOT}}$ *(Figure 16) in the* $(\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}, \mathcal{F}_{\mathcal{B}})$*-hybrid in the presence of any PPT malicious adversary* $\mathcal{A}$ *that can corrupt all-but-one client* $\{C_i\}_{i\in[m]}$ *adaptively and all-but-one of* $\{S_j\}_{j\in[d]\cup\{0\}}$ *statically.*

Inputs: $b_i \in \{0,1\}$ *for each chooser* C_i, $(s_0^j, s_1^j) \in \mathcal{M}^2$ *for each* S_j

$$
\begin{aligned}
\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}} : &\quad \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa) \\
\forall i \in [m], C_i : &\quad \mathsf{Aux}_i, \mathsf{m}_1^i \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b_i) && \mathsf{m}_1^i \rightarrow \mathcal{F}_{\mathcal{B}} \\
\forall j \in [d] \cup \{0\}, S_j : &\quad \mathsf{m}_2^j \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \{\mathsf{m}_1^i\}_{i\in[m]}, (s_0^j, s_1^j)) && \mathsf{m}_2^j \rightarrow \mathcal{F}_{\mathcal{B}} \\
\forall i \in [m], C_i : &\quad \mathsf{m}_3^i \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathsf{m}_2^j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i) && \mathsf{m}_3^i \rightarrow \mathcal{F}_{\mathcal{B}}
\end{aligned}
$$

$$\begin{aligned} \textit{public}: \{\{s_b^j\}_{j\in[d]\cup\{0\}}, \leftarrow \mathsf{OT}_{\mathsf{dec}}(\mathsf{CRS}; \{\mathsf{m}_1^{\mathsf{i}}\}_{i\in[m]}, \\ \{b_i\}_{i\in[m]}\} \qquad\qquad \{\mathsf{m}_2^{\mathsf{j}}\}_{j\in[d]\cup\{0\}}, \{\mathsf{m}_3^{\mathsf{i}}\}_{i\in[m]}) \\ \textit{output}\ \{\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}\} \end{aligned}$$

where $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}$ (Figure 7) is the Common Reference String functionality and $\mathcal{F}_{\mathcal{B}}$ (Figure 6) is the Bulletin-Board functionality.

We build DComOT (Definition 11) using multiple instances of 'Committed-Index OT' protocols. The 'Committed-Index OT' (COT) $\mathcal{F}_{\mathsf{ComOT}}$ (Figure 18 – Definition 12) is a simplification of the above functionality to the single-chooser case. Such a protocol ComOT as in Definition 12 can be realized using a 2-round OT with a reusable first message (Definition 4). This is done by simply letting the first and second messages in ComOT be the first and second messages in the OT protocol. In the third message $\mathsf{ComOT}.\mathsf{OT}_{\mathsf{fin}}$, the chooser outputs its choice bit and the randomness used to compute the first message. Finally, in $\mathsf{ComOT}.\mathsf{OT}_{\mathsf{dec}}$ any party can use the COT transcript so far to derive the receiver's OT output.

A Distributed Committed-Index OT Protocol. Given a COT protocol $\mathsf{ComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ as above, it can be extended to a DCOT protocol (Definition 11). Consider the protocol below in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid.

Lemma 6. *Figure 17 UC-securely realizes $\mathcal{F}_{\mathsf{DComOT}}$ (Definition 11 – Figure 16) in the $(\mathcal{F}_{\mathsf{ComOT}}, \mathcal{F}_{\mathcal{B}})$-hybrid.*

Protocol Realizing $\mathcal{F}_{\mathsf{DComOT}}$

For each $i \in [m]$, let $\mathcal{F}_{\mathsf{ComOT}}^i$ (Figure 18) realize COT. Let $\mathcal{F}_{\mathcal{B}}$ be the bulletin-board functionality. Let $\{S_j\}_{j\in[d]\cup\{0\}}$ be the senders where each S_j has strings (s_0^j, s_1^j). Let $\{C_i\}_{i\in[m]}$ be choosers where each C_i has bit b_i.

- **Choosers' Commit Phase**:
  - $\forall i \in [m], C_i$ sends b_i to $\mathcal{F}_{\mathsf{ComOT}}^i$
- $\forall j \in [d] \cup \{0\}$, **Sender's commit phase**:
  - S_j has inputs $s_0^j, s_1^j \in \{0,1\}^\ell$ and samples $\Delta \leftarrow \{0,1\}^\ell$
  - $\forall i \in [m]$, S_j samples $L_0^{i,j} \leftarrow \{0,1\}^\ell$ and computes $L_1^{i,j} = L_0^{i,j} \oplus \Delta$
  - $\forall i \in [m]$, S_j sends $L_0^{i,j}, L_1^{i,j}$ to $\mathcal{F}_{\mathsf{ComOT}}^i$
  - S_j computes, $c_0^j = s_0^j\big(\bigoplus_{i=1}^m L_0^{i,j}\big)$ and $c_1^j = s_1^j\big(\bigoplus_{i=1}^m L_0^{i,j}\big) \oplus \Delta$
  - S_j sends (c_0^j, c_1^j) to $\mathcal{F}_{\mathcal{B}}$
- **Reveal Phase**:
  - $\forall i \in [m]$, $\mathcal{F}_{\mathsf{ComOT}}^i$ publishes $(\{L_{b_i}^{i,j}\}_{j\in[d]\cup\{0\}}, b_i)$
  - compute $b = \bigoplus_{i=1}^m b_i$ and $\forall j \in [d] \cup \{0\}, s_b^j = c_b^j\big(\bigoplus_{i=1}^m L_{b_i}^{i,j}\big)$
  - output $(\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]})$

Fig. 17 Protocol Realizing $\mathcal{F}_{\mathsf{DComOT}}$ in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid

Committed-Index OT $\mathcal{F}_{\mathsf{ComOT}}$

Let $\{S_j\}_{j\in[d]\cup\{0\}}$ be the senders where each S_j has input strings (s_0^j, s_1^j). Let C be the chooser with input bit b. Let $\mathsf{publish}$ denote the action of $\mathcal{F}_{\mathsf{ComOT}}$ making a message visible to all parties. $\mathcal{F}_{\mathsf{ComOT}}$ works as follows:

1. Chooser's Commit Phase:
   - receive input b from C
2. $\forall j \in [d] \cup \{0\}$ Senders' Commit Phase:
   - receive input (s_0^j, s_1^j) from S_j
3. Reveal Phase:
   - $\mathsf{publish}$ $(\{s_b^j\}_{j\in[d]\cup\{0\}}, b)$

Fig. 18 Committed-Index OT

We prove that DComOT (Figure 17) is a secure protocol for $\mathcal{F}_{\mathsf{DComOT}}$ in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid. Before getting into the details of the proof, it becomes necessary to first formally define the Committed-Index OT functionality $\mathcal{F}_{\mathsf{ComOT}}$.

Definition 12. *The tuple* $\mathsf{ComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ *is a* ***Maliciously Secure SCALES Protocol for COT*** *if the following protocol UC-securely realizes* $\mathcal{F}_{\mathsf{ComOT}}$ *(Figure 18) in the* $(\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}, \mathcal{F}_{\mathcal{B}})$*-hybrid in the presence of any PPT malicious adversary* $\mathcal{A}$ *that can corrupt* C *adaptively and all-but-one of* $\{S_j\}_{j\in[d]\cup\{0\}}$ *statically.*

Inputs: $b \in \{0,1\}$ *for chooser* C, $(s_0^j, s_1^j) \in \mathcal{M}^2$ *for each* S_j

$$
\begin{aligned}
\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}: &\quad \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa) \\
C: &\quad \mathsf{Aux}, \mathsf{m}_1 \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b) && \mathsf{m}_1 \rightarrow \mathcal{F}_{\mathcal{B}} \\
\forall j \in [d]\cup\{0\}, S_j: &\quad \mathsf{m}_2^j \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathsf{m}_1, (s_0^j, s_1^j)) && \mathsf{m}_2^j \rightarrow \mathcal{F}_{\mathcal{B}} \\
C: &\quad \mathsf{m}_3 \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathsf{m}_2^j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}) && \mathsf{m}_3 \rightarrow \mathcal{F}_{\mathcal{B}} \\
\mathit{public}: &\quad \{\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}\} \leftarrow \mathsf{OT}_{\mathsf{dec}}(\mathsf{CRS}; \mathsf{m}_1, \{\mathsf{m}_2^j\}_{j\in[d]\cup\{0\}}, \mathsf{m}_3) \\
&\quad \mathit{output}\ \{\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}\}
\end{aligned}
$$

where $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}$ *(Figure 7) is the Common Reference String functionality and* $\mathcal{F}_{\mathcal{B}}$ *(Figure 6) is the Bulletin-Board functionality.*

We construct ComOT as in Definition 12 using a 2-round OT with a reusable first message (Definition 4).

Protocol Realizing $\mathcal{F}_{\mathsf{ComOT}}$

Let $\mathbf{OT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}})$ be a 2-round OT protocol with reusable first message (Definition 4). Then $\mathsf{ComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ operates as follows:

- $\mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa)$

  $\mathsf{CRS} \leftarrow \mathbf{OT}.\mathsf{CRSgen}(1^\kappa)$
- Chooser's commit phase: $\mathsf{Aux}, \mathsf{m}_1 \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b)$

  $(\mathsf{m}_1, \mathsf{Aux}) \leftarrow \mathbf{OT}.\mathsf{OT}_1(\mathsf{CRS}, r; b)$ s.t. $r \in \mathsf{Aux}$
- $\forall j \in [d] \cup \{0\}$, Sender's commit phase: $\mathsf{m}_2^j \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \mathsf{m}_1; s_0^j, s_1^j)$

  $s_0^j, s_1^j = \{a_0^i\}_{i\in[\ell]}, \{a_1^i\}_{i\in[\ell]}$

  $\forall i \in [\ell], \mathbf{m_2^i} \leftarrow \mathbf{OT}.\mathsf{OT}_2(\mathsf{CRS}; \mathsf{m}_1, a_0^i, a_1^i)$

  $\mathsf{m}_2^j = \{\mathbf{m_2^i}\}_{i\in[\ell]}$
- Reveal Phase: $\mathsf{m}_3 \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathsf{m}_2^j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux})$

  $\mathsf{m}_3 = (b, r)$
- $\{(\{s_b^j\}_{j\in[d]\cup\{0\}}, b), \bot\} \leftarrow \mathsf{OT}_{\mathsf{dec}}(\mathsf{CRS}; \mathsf{m}_1, \{\mathsf{m}_2^j\}_{j\in[d]\cup\{0\}}, \mathsf{m}_3)$

  $\mathsf{m}_3 = (b, r)$ and $\mathsf{Aux} \leftarrow r$

  $\forall j \in [d] \cup \{0\}$, $\mathsf{m}_2^j = \{\mathbf{m_2^i}\}_{i\in[\ell]}$

  $\forall i \in [\ell], a_b^i \leftarrow \mathbf{OT}.\mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathbf{m_2^i}, \mathsf{Aux})$

  $s_b^j = \{a_b^i\}_{i\in[\ell]}$

  output $\begin{cases} (\{s_b^j\}_{j\in[d]\cup\{0\}}, b) & \text{if } (\mathbf{m_1}, \mathsf{Aux}) = \mathsf{ComOT}.\mathsf{OT}_1(\mathsf{CRS}, r; b) \\ \bot & \text{otherwise} \end{cases}$

Fig. 19 Protocol realizing $\mathcal{F}_{\mathsf{ComOT}}$ using $\mathbf{OT}$

Lemma 7. *If* OT *is a 2-round OT protocol (Definition 4) with reusable first message that is secure against adaptive receiver corruption and static sender corruption in the malicious setting, then Figure 19 is a maliciously secure SCALES protocol UC-securely realizing* $\mathcal{F}_{\mathsf{ComOT}}$ *(Figure 18 – Definition 12) in the* $\mathcal{F}_{\mathsf{multiOT}}$*-hybrid.*

Proof Outline. We need to show that given a 2-round maliciously secure OT protocol with reusable first message, that is secure also in the presence of adaptive receiver corruption, the COT protocol in Figure 19 UC-securely realizes $\mathcal{F}_{\mathsf{ComOT}}$ (Definition 12 – Figure 18) in the presence of a malicious PPT adversary. We present this proof in the $\mathcal{F}_{\mathsf{multiOT}}$-hybrid.

In COT, the PPT malicious adversary $\mathcal{A}$ can corrupt C adaptively and up to all-but-one of the senders statically. This corresponds to corrupting R adaptively and for $(d+1)\ell$ senders in the bit-OT protocol, statically corrupting up to all-but-ℓ-consecutive senders, where ℓ is the length of the senders' strings in COT. The simulator for this protocol Sim works by emulating $\mathcal{F}_{\mathsf{multiOT}}$ corresponding to this corruption set. The view produced by the COT simulator can be shown as identically distributed to the real view in the indistinguishable from the real view in the $\mathcal{F}_{\mathsf{multiOT}}$-hybrid.

Proof. Let $\mathbf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}})$ be a 2-round maliciously secure OT protocol that is secure against adaptive receiver corruption and under reuse of the first OT message. Let $\mathcal{F}_{\mathsf{multiOT}}$ be the corresponding functionality that it realizes.

For the Committed-Index OT protocol ComOT as in Figure 19, let $\mathcal{A}$ be a malicious PPT adversary. $\mathcal{A}$ can corrupt the chooser C either statically or adaptively after it sends the first message, and all-but-one senders in $\mathcal{S}$. Consider the following PPT simulator $\mathsf{Sim}_{\mathsf{ComOT}}$:

1. if C is statically corrupted:
   - $\mathsf{Sim}_{\mathsf{ComOT}}$, emulating $\mathcal{F}_{\mathsf{multiOT}}$, receives b from the PPT adversary $\mathcal{A}$.
   - $\mathsf{Sim}_{\mathsf{ComOT}}$ passes b into $\mathcal{F}_{\mathsf{ComOT}}$.

   otherwise, if C is initially honest $\mathsf{Sim}_{\mathsf{ComOT}}$ does nothing.
2. $\forall j \in [d] \cup \{0\}$, if S_j is statically corrupted:
   - $\mathsf{Sim}_{\mathsf{ComOT}}$, emulating $\mathcal{F}_{\mathsf{multiOT}}$ for a set of ℓ corrupt senders, $\{a_0^i, a_1^i\}_{i\in[\ell]}$.
   - $\mathsf{Sim}_{\mathsf{ComOT}}$ passes $s_0 = \{a_0^i\}_{i\in[\ell]}$ and $s_1 = \{a_1^i\}_{i\in[\ell]}$ into $\mathcal{F}_{\mathsf{ComOT}}$.
3. if C is adaptively corrupted:
   - $\mathsf{Sim}_{\mathsf{ComOT}}$ gets b from $\mathcal{F}_{\mathsf{ComOT}}$ and emulating $\mathcal{F}_{\mathsf{multiOT}}$, sends b to $\mathcal{A}$.
4. if C is honest in the 'reveal phase':
   - $\mathcal{F}_{\mathsf{ComOT}}$ posts $(\{s_b^j\}_{j\in[d]\cup\{0\}}, b)$ to $\mathcal{B}$.

The complete view of the protocol produced by $\mathsf{Sim}_{\mathsf{ComOT}}$ consists of,

$$\forall j \in [d] \cup \{0\} \begin{cases} b & \text{if C is statically corrupted} \\ s_0^j, s_1^j & \text{if } \mathrm{S}_j \text{ is corrupted} \\ \{s_b^j\}_{j\in[d]\cup\{0\}}, b \end{cases}$$

This view is identical to that in the real execution in the $\mathcal{F}_{\mathsf{multiOT}}$-hybrid. Hence, the protocol ComOT securely realizes $\mathcal{F}_{\mathsf{ComOT}}$ in the presence of malicious adversaries in the SCALES model.

We are now ready to prove that DComOT (Figure 17) is a secure protocol for $\mathcal{F}_{\mathsf{DComOT}}$ against all-but-one statically corrupted senders and all-but-one choosers that can be adaptively corrupted by a malicious adversary, in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid.

Proof Outline. Let $\mathcal{A}$ be a PPT adversary that can statically corrupt all-but-one of the senders and adaptively corrupt all-but-one of the choosers. We need to show that DComOT is a secure protocol for $\mathcal{F}_{\mathsf{DComOT}}$ in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid. This would imply that the DCOT protocol as in Figure 20 securely realizes $\mathcal{F}_{\mathsf{DComOT}}$ in the plain model given a COT protocol as in Definition 12 for $\mathcal{F}_{\mathsf{ComOT}}$.

For DCOT, the PPT simulator $\mathsf{Sim}_{\mathsf{DComOT}}$, in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid would work by first collecting all the choice bits b_i from each statically corrupt client by emulating $\mathcal{F}_{\mathsf{ComOT}}$. These are forwarded to $\mathcal{F}_{\mathsf{DComOT}}$ in the 'commit-phase'. If a server is corrupted, it would receive the 2 ciphertexts (c_0^j, c_1^j) posted and for all $i \in [m]$, the labels (L_0^i, L_1^i) by emulating $\mathcal{F}_{\mathsf{ComOT}}$. The simulator checks if the labels are well formed and extracts (s_0^j, s_1^j) to send to $\mathcal{F}_{\mathsf{DComOT}}$. If a server is not corrupted, it samples the ciphertexts (c_0^j, c_1^j) uniformly at random and posts

these. For each client that is adaptively corrupted, $\mathcal{F}_{\mathsf{DComOT}}$ reveals their choice bits to the simulator and this is given to the adversary. In the 'reveal-phase' when $\mathcal{F}_{\mathsf{DComOT}}$ reveals $\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}$ to the simulator, for each $j \in [d] \cup \{0\}$, it prepares $k_b^j = c_b^j \oplus s_b^j$. It samples each $L_{b_i}^i$ at random with the constraint that $k_b^j = \bigoplus_{i=1}^m L_{b_i}^i$. Then is posts for all $i \in [m]$, $L_{b_i}^i, b_i$. This completes the simulation.

This simulated view is identically distributed to the real view of the protocol, and hence the protocol is secure.

Proof. Let $\mathcal{A}$ be a PPT malicious adversary in the real world that can statically corrupt all-but-one sender and adaptively corrupt all-but-one choosers. We need to show that Figure 17 is a secure protocol for $\mathcal{F}_{\mathsf{DComOT}}$ in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid.

Let C_S be the set of choosers that are statically corrupted and C_A be the set of adaptively corrupted choosers. Let $\mathcal{S}^*$ be the set of corrupted senders. $\mathcal{A}$ can also corrupt any number of parties that are not the senders or the choosers. The PPT simulator $\mathsf{Sim}_{\mathsf{DComOT}}$ in the $\mathcal{F}_{\mathsf{ComOT}}$-hybrid would work as follows:

1. for all corrupted clients $C_i \in C_S$,
   - get b_i by emulating the i^{th} instance of $\mathcal{F}_{\mathsf{ComOT}}$.
   - pass b_i to $\mathcal{F}_{\mathsf{DComOT}}$.
2. for each $j \in [d] \cup \{0\}$, if S_j is corrupted:
   - receive the ciphertexts (c_0^j, c_1^j) posted by $\mathcal{A}$.
   - for all $i \in [m]$, get labels (L_0^i, L_1^i) by emulating the i^{th} instance of $\mathcal{F}_{\mathsf{ComOT}}$.
   - if $\nexists \Delta$ such that $\forall i \in [m], L_0^i \oplus L_1^i = \Delta$, send $\perp$ to $\mathcal{F}_{\mathsf{DComOT}}$.
   - otherwise, compute $L_0^i \oplus L_1^i = \Delta$, $k_0^j = \bigoplus_{i=1}^m L_0^i$ and $k_1^j = k_0^j \oplus \Delta$.
   - extract $s_0^j = c_0^j \oplus k_0^j$ and $s_1^j = c_1^j \oplus k_1^j$ and pass (s_0^j, s_1^j) to $\mathcal{F}_{\mathsf{DComOT}}$.

   if S_j is honest:
   - sample c_0^j and c_1^j uniformly at random and post it.
3. for each adaptively corrupted client $C_i \in C_A$,
   - $\mathcal{A}$ sends i to $\mathsf{Sim}_{\mathsf{DComOT}}$ and this is passed to $\mathcal{F}_{\mathsf{DComOT}}$.
   - $\mathcal{F}_{\mathsf{DComOT}}$ gives b_i that it gives to $\mathcal{A}$, emulating the i^{th} instance of $\mathcal{F}_{\mathsf{ComOT}}$.
4. when $\mathcal{F}_{\mathsf{DComOT}}$ reveals $(\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]})$, for each honest S_j:
   - compute $k_b^j = c_b^j \oplus s_b^j$.
   - for all $i \in [m]$, sample $L_{b_i}^i$ at random but satisfying $k_b^j = \bigoplus_{i=1}^m L_{b_i}^i$.
   - for all $i \in [m]$, emulate the i^{th} instance of $\mathcal{F}_{\mathsf{ComOT}}$ and post $(L_{b_i}^i, b_i)$.

Let κ be a computational security parameter. Let $\mathcal{R}$ be the space of randomness for all participating parties and let $\vec{r}$ denote the contents of the random tapes of all these parties. The view of in the above simulation is distributed as:

$$\Big\{\{s_0^j, s_1^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}, \{c_0^j, c_1^j, \{L_0^i, L_1^i\}_{i\in[m]}\}_{S_j\in\mathcal{S}^*}, \\ \{c_0^j, c_1^j, \{L_{b_i}^i\}_{i\in[m]}\}_{S_j\notin\mathcal{S}^*}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

This simulated view is identically distributed to the real view of the protocol, and hence the protocol is secure.

Distributed Committed-Index OT in the Plain Model.

Protocol Realizing $\mathcal{F}_{\mathsf{DComOT}}$

Let $\mathsf{ComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ be a COT protocol (Definition 12) where sender's inputs $\in \{0,1\}^\ell$. The protocol
$\mathsf{DComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ operates as follows:

- $\mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa)$

  $\mathsf{CRS} \leftarrow \mathsf{ComOT.CRSgen}(1^\kappa)$
- Choosers' Commit Phase:
  $\forall i \in [m], (\mathsf{Aux}_i, \mathsf{m}_1^{\mathsf{i}}) \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b_i)$

  $(\mathsf{Aux}_i, \mathsf{m}_1^{\mathsf{i}}) \leftarrow \mathsf{ComOT.OT}_1(\mathsf{CRS}; b_i; r_i)$
- $\forall j \in [d] \cup \{0\}$, Sender's commit phase:
  $\mathsf{m}_2^{\mathsf{j}} \leftarrow \mathsf{OT}_2(\mathsf{CRS}; \{\mathsf{m}_1^{\mathsf{i}}\}_{i\in[m]}; s_0^j, s_1^j)$

  $$s_0^j, s_1^j \in \{0,1\}^\ell$$
  $$\text{sample } \Delta \leftarrow \{0,1\}^\ell$$
  $$\forall i \in [m], \quad L_0^{i,j} \leftarrow \{0,1\}^\ell, \quad L_1^{i,j} = L_0^{i,j} \oplus \Delta$$
  $$\forall i \in [m], \quad \mathsf{m}_2^{\mathsf{i}} \leftarrow \mathsf{ComOT.OT}_2(\mathsf{CRS}; \mathsf{m}_1^{\mathsf{i}}; L_0^{i,j}, L_1^{i,j})$$
  $$c_0^j = s_0^j\Big(\bigoplus_{i=1}^m L_0^{i,j}\Big), \quad c_1^j = s_1^j\Big(\bigoplus_{i=1}^m L_0^{i,j}\Big) \oplus \Delta$$
  $$\mathsf{m}_2^{\mathsf{j}} = (c_0^j, c_1^j, \{\mathsf{m}_2^{\mathsf{i}}\}_{i\in[m]})$$
- Reveal Phase:
  $\mathsf{m}_3^{\mathsf{i}} \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathsf{m}_2^{\mathsf{j}}\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i)$

  $$\forall j \in [d] \cup \{0\}, \mathsf{m}_2^{\mathsf{i,j}} = \mathsf{m}_2^{\mathsf{i}} \in \mathsf{m}_2^{\mathsf{j}}$$
  $$\mathsf{m}_3^{\mathsf{i}} \leftarrow \mathsf{ComOT.OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathsf{m}_2^{\mathsf{i,j}}\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i)$$
- $\{(\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]}), \perp\} \leftarrow \mathsf{OT}_{\mathsf{dec}}^{\mathsf{CRS}}(\{\mathsf{m}_1^{\mathsf{i}}\}_{i\in[m]}, \{\mathsf{m}_2^{\mathsf{j}}\}_{j\in[d]\cup\{0\}}, \{\mathsf{m}_3^{\mathsf{i}}\}_{i\in[m]})$

  $\forall i \in [m], j \in [d] \cup \{0\}, \quad \mathsf{m}_2^{\mathsf{i,j}} = \mathsf{m}_2^{\mathsf{i}} \in \mathsf{m}_2^{\mathsf{j}}$

  $\forall i \in [m], \{(\{L_{b_i}^{i,j}\}_{j\in[d]\cup\{0\}}, b_i), \perp\} \leftarrow \mathsf{ComOT.OT}_{\mathsf{dec}}(\mathsf{CRS}; \mathsf{m}_1^{\mathsf{i}}, \{\mathsf{m}_2^{\mathsf{i,j}}\}_{j\in[d]\cup\{0\}}, \mathsf{m}_3^{\mathsf{i}})$

  $$b = \bigoplus_{i=1}^m b_i$$
  $$\forall j \in [d] \cup \{0\}, \quad s_b^j = c_b^j\Big(\bigoplus_{i=1}^m L_{b_i}^{i,j}\Big)$$

  output $(\{s_b^j\}_{j\in[d]\cup\{0\}}, \{b_i\}_{i\in[m]})$

Fig. 20 Protocol Realizing $\mathcal{F}_{\mathsf{DComOT}}$ in the Plain Model

Ephemeral Prover Zero-Knowledge Proof Functionality $\mathcal{F}_{\mathsf{EPZK}}$

Let $\{P_j\}_{j\in[d]\cup\{0\}}$ be the provers where P_0 has statement α_0 and witness w_0 for relation $\mathcal{R}_0$ and for $j \in [d]$, each P_j has input statement α_j and witness w_j for relation $\mathcal{R}_1$. Let $\mathsf{publish}$ denote the action of $\mathcal{F}_{\mathsf{EPZK}}$ making a message visible to all parties. The functionality $\mathcal{F}_{\mathsf{EPZK}}$ works as follows:

1. $\forall j \in [d] \cup \{0\}$ Prover's Commit Phase:
   - receive input (α_j, w_j) from P_j and $\mathsf{publish}$ α_j
2. Reveal Phase:
   - compute $b_0 = \mathcal{R}_0(\alpha_0, w_0)$ and $\forall j \in [d]$, compute $b_j = \mathcal{R}_1(\alpha_j, \alpha_{j-1}, w_j)$
   - $\mathsf{publish}$ $b = \bigwedge_{j\in[d]\cup\{0\}} b_j$

Fig. 21 Ephemeral Prover Zero-Knowledge Proof Functionality

5.4 Ephemeral Prover Zero-Knowledge Proofs

Given a DCOT protocol (Definition 11 – Figure 16), the Σ-protocols in Section 5.2 can be converted into an ephemeral prover zero-knowledge proof in the SCALES model. Consider a EPZK functionality as in Figure 21. Recall that Σ-protocols (Definition 6) have a three-message-transcript: (a, b, c). In our SCALES protocol, we need the encoding ephemeral server to be the prover. However, it can only send one message, as opposed to the two messages required in the Σ-protocol. An Ephemeral Prover Zero-Knowledge Proof is formalized in Definition 13.

Definition 13. *For $\{S_j\}_{j\in[d]\cup\{0\}}$, each proving server S_j has witness w_j and statement α_j. For public relations $\mathcal{R}_0, \mathcal{R}_1$, S_0 needs to prove $\mathcal{R}_0(\alpha_0, w_0) = 1$ and all other S_j prove $\mathcal{R}_1(\alpha_j, w_j) = 1$. Let $\{C_i\}_{i\in[m]}$ be a set of choosers. An* ***ephemeral prover zero-knowledge proof*** *is a tuple $\mathsf{EPZK} = (\mathsf{CRSgen}, \mathsf{EPZK}_1, \mathsf{EPZK}_P, \mathsf{EPZK}_2, \mathsf{EPZK}_V)$ that UC-securely realizes $\mathcal{F}_{\mathsf{EPZK}}$ (Figure 21) in the $(\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}, \mathcal{F}_{\mathcal{B}})$-hybrid in the presence of a PPT malicious adversary $\mathcal{A}$ corrupting all-but-one client $\{C_i\}_{i\in[m]}$ adaptively and all-but-one of $\{S_j\}_{j\in[d]\cup\{0\}}$ statically.*

$$
\begin{aligned}
\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}: &\quad \mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa) \\
\forall i \in [m], C_i: &\quad \mathsf{Aux}_i, m_1^i \leftarrow \mathsf{EPZK}_1(\mathsf{CRS}) && m_1^i \rightarrow \mathcal{F}_{\mathcal{B}} \\
\forall j \in [d] \cup \{0\}, S_j: &\quad \pi_j \leftarrow \mathsf{EPZK}_P(\mathsf{CRS}; \alpha_j, w_j, \{m_1^i\}_{i\in[m]}) && \pi_j \rightarrow \mathcal{F}_{\mathcal{B}} \\
\forall i \in [m], C_i: &\quad m_2^i \leftarrow \mathsf{EPZK}_2(\mathsf{CRS}; \{\pi_j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i) && m_2^i \rightarrow \mathcal{F}_{\mathcal{B}} \\
\textit{public}: &\quad \{0,1\} \leftarrow \mathsf{EPZK}_V(\mathsf{CRS}; \{m_1^i\}_{i\in[m]}, \{\pi_j\}_{j\in[d]\cup\{0\}}, \\
&\qquad\qquad \{m_2^i\}_{i\in[m]}, \{\alpha_j\}_{j\in[d]\cup\{0\}}) \\
&\quad \textit{output } \{0,1\}
\end{aligned}
$$

where $\mathcal{F}_{\mathsf{CRS}}^{\mathsf{CRSgen}}$ (Figure 7) is the Common Reference String functionality and $\mathcal{F}_{\mathcal{B}}$ (Figure 6) is the Bulletin-Board functionality.

The verifier's role is split into one that provides the challenge and one that verifies the proof. We delegate the former role in our protocol to the input clients.

Each client picks a challenge and the final challenge used is an XOR of all the individual inputs. The latter role of verifying the proof itself is delegated to other ephemeral servers that post a bit to $\mathcal{B}$ (indicating an accept or reject).

Lemma 8. *If* Sigma *is a* Σ*-protocol with soundness error* $\frac{1}{2}$ *for the relations* $(\alpha_0, w_0) \in \mathcal{R}_0$ *and* $\forall j \in [d], (\alpha_j, w_j) \in \mathcal{R}_1$*, then Figure 22 realizes* $\mathcal{F}_{\mathsf{EPZK}}$ *(Figure 21) in the* $(\mathcal{F}_{\mathsf{DComOT}}, \mathcal{F}_{\mathcal{B}})$*-hybrid, where* $\mathcal{F}_{\mathcal{B}}$ *is the Bulletin-Board functionality (Figure 6) and* $\mathcal{F}_{\mathsf{DComOT}}$ *is the Distributed Committed-Index OT functionality (Figure 16).*

Protocol Realizing $\mathcal{F}_{\mathsf{EPZK}}$

Let s be a statistical security parameter. Let $\mathcal{F}_{\mathsf{DComOT}}$ (Figure 16) be the distributed committed-index OT functionality with $\mathcal{F}^k_{\mathsf{DComOT}}$ denoting the k^{th} instance of DComOT. Let $\mathcal{F}_{\mathcal{B}}$ be the bulletin-board functionality. Let $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ be a Σ-protocol with soundness $\frac{1}{2}$ where $a \leftarrow P_1(x, w)$, $b \in \{0,1\} \leftarrow V_1(\cdot)$, $c \leftarrow P_2(a, b, x, w)$, $\{0,1\} \leftarrow V_2(a, b, c, x)$. Consider the following protocol realizing $\mathcal{F}_{\mathsf{EPZK}}$ in the $\mathcal{F}_{\mathsf{DComOT}}$-hybrid:

1. $\forall i \in [m]$, each client C_i does the following:
   - $\forall k \in [s]$, sample bit b_i^k and give to $\mathcal{F}^k_{\mathsf{DComOT}}$ in the 'commit-phase'
2. $\forall j \in [d] \cup \{0\}$, the server S_j (prover) does the following:
   - $\forall k \in [s]$ compute $a_j^k \leftarrow \mathsf{Sigma}.P_1(\alpha_j, w_j)$
   - $\forall k \in [s]$ compute $s_0^{j,k} = \mathsf{Sigma}.P_2(\alpha_j, w_j, a_j^k, 0)$ and $s_1^{j,k} = \mathsf{Sigma}.P_2(\alpha_j, w_j, a_j^k, 1)$
   - $\forall k \in [s]$ pass $(s_0^{j,k}, s_1^{j,k})$ to $\mathcal{F}^k_{\mathsf{DComOT}}$
   - give $(\alpha_j, \{a_j^k\}_{k\in[s]})$ to $\mathcal{F}_{\mathcal{B}}$
3. each $\mathcal{F}^k_{\mathsf{DComOT}}$ publishes $(\{s_b^{j,k}\}_{j\in[d]\cup\{0\}}, \{b_i^k\}_{i\in[m]})$ in the 'reveal-phase'
4. each server (verifier) does the following:
   - $\forall k \in [s]$ compute $b^k = \bigoplus_{i=1}^m b_i^k$
   - send $\{\mathsf{Sigma}.V_2(a_j^k, b^k, s_{b^k}^{j,k}, \alpha_j)\}_{k\in[s], j\in[d]\cup\{0\}}$ to $\mathcal{F}_{\mathcal{B}}$

Fig. 22 Construction for $\mathcal{F}_{\mathsf{EPZK}}$

We need to show that it securely realizes $\mathcal{F}_{\mathsf{EPZK}}$ in the $\mathcal{F}_{\mathsf{DComOT}}$-hybrid.

Proof Outline. For $(\alpha_0, w_0) \in \mathcal{R}_0$ and for each $j \in [d]$, in the relation $(\alpha_j, w_j) \in \mathcal{R}_1$, let $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ be a Sigma protocol with soundness $\frac{1}{2}$ such that $b \in \{0,1\} \leftarrow \mathsf{Sigma}.V_1(\cdot)$. We need to prove that the above protocol realizes $\mathcal{F}_{\mathsf{EPZK}}$ with negligible soundness error.

Let $\mathcal{A}$ be a PPT adversary that corrupts all-but-one of the provers statically, and all-but-one of the clients adaptively. A simulator $\mathsf{Sim}_{\mathsf{EPZK}}$ for the above protocol would work by first receiving the bits $\{b_i^k\}_{k\in[s]}$ from each statically corrupt client C_i by emulating each $\mathcal{F}^k_{\mathsf{DComOT}}$ in the 'commit-phase'. For each corrupt proving server S_j, it receives $\{s_0^{j,k}, s_1^{j,k}\}_{k\in[s]}$ also by emulating each $\mathcal{F}^k_{\mathsf{DComOT}}$. It reads $\{a_j^k\}_{k\in[s]}$ that the server posts and then it calls the extraction algorithm until it gets a valid witness w_j for at least one $k \in [s]$. Then (α_j, w_j) are passed to $\mathcal{F}_{\mathsf{EPZK}}$. If S_j is honest, first sample $\{b_i^k\}_{k\in s}$ corresponding to the

honest clients C_i and compute $b^k = \oplus_{i\in[m]} b_i^k$. Then use the simulator of the sigma protocol to generate $\{a_j^k, s_{b^k}^{j,k}\}_{k\in[s]}$. Post $\{a_j^k\}_{k\in[s]}$. In the end, once $\mathcal{F}_{\mathsf{EPZK}}$ responds to $\mathsf{Sim}_{\mathsf{EPZK}}$, it posts $\{s_b^{j,k}\}_{j\in[d]\cup\{0\}}, \{b_i^k\}_{i\in[m]}$ on behalf of each $\mathcal{F}_{\mathsf{DComOT}}^k$ execution.

The view produced by this simulation differs from that in the real execution only for the transcripts produced by the Σ-protocol simulation on behalf of the honest provers. Therefore the distribution of both views can be show as computationally indistinguishable by reduction to the indistinguishability of the distribution of the views for the Σ-protocol.

Proof. For $(\alpha_0, w_0) \in \mathcal{R}_0$ and for each $j \in [d]$, in the relation $(\alpha_j, w_j) \in \mathcal{R}_1$, let $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ be a Sigma protocol (Definition 6) with soundness $\frac{1}{2}$ such that $b \in \{0,1\} \leftarrow \mathsf{Sigma}.V_1(\cdot)$. Such a protocol satisfies 'special soundness' and there exists a PPT algorithm,

$$w_j \leftarrow \mathsf{Sigma.Extract}(a, b, c, b', c', \alpha_j)$$

The protocol also satisfies 'special honest-verifier zero-knowledge' so there exists a PPT simulator,

$$(a, b, c) \leftarrow \mathsf{Sigma.Sim}(1^\kappa, \alpha_j, b)$$

Given these building-blocks, we need to prove that the above protocol realizes $\mathcal{F}_{\mathsf{EPZK}}$ with negligible soundness error.

Let $\mathcal{A}$ be a PPT adversary that corrupts all-but-one of the provers statically, and all-but-one of the clients adaptively. Let C_S be the set of statically corrupted clients, C_A be the set of adaptively corrupted clients, and $\mathcal{S}^*$ be the set of corrupted proving servers. The PPT simulator $\mathsf{Sim}_{\mathsf{EPZK}}$ operates as follows:

- $\forall k \in [s]$, $\mathsf{Sim}_{\mathsf{EPZK}}$ emulates $\mathcal{F}_{\mathsf{DComOT}}^k$ to receive $\{b_i^k\}_{C_i\in C_S}$ from $\mathcal{A}$
- $\forall j \in [d] \cup \{0\}$, if S_j is corrupt,
  - $\forall k \in [s]$, $\mathsf{Sim}_{\mathsf{EPZK}}$ emulates $\mathcal{F}_{\mathsf{DComOT}}^k$ and gets $(s_0^{j,k}, s_1^{j,k})$ from $\mathcal{A}$
  - $\mathsf{Sim}_{\mathsf{EPZK}}$ reads $\{a_j^k\}_{\in[s]}$ that $\mathcal{A}$ posts
  - $\forall k \in [s]$, $\mathsf{Sim}_{\mathsf{EPZK}}$ invokes $\{w_j, \perp\} \leftarrow \mathsf{Sigma.Extract}(a_j^k, 0, s_0^{j,k}, 1, s_1^{j,k}, \alpha_j)$ until it gets a valid w_j
  - $\mathsf{Sim}_{\mathsf{EPZK}}$ gives (α_j, w_j) to $\mathcal{F}_{\mathsf{EPZK}}$ that posts α_j

  if S_j is honest,
  - $\mathsf{Sim}_{\mathsf{EPZK}}$ samples the bits $\{b_i^k\}_{k\in[s]}$ for each honest client C_i
  - $\mathcal{F}_{\mathsf{EPZK}}$ publishes α_j
  - $\forall k \in [s]$, $\mathsf{Sim}_{\mathsf{EPZK}}$ computes $b^k = \oplus_{i\in[m]} b_i^k$ and then invokes,

$$(a_j^{k*}, b^k, s_{b^k}^{j,k*}) \leftarrow \mathsf{Sigma.Sim}(1^\kappa, \alpha_j, b^k)$$

  - $\mathsf{Sim}_{\mathsf{EPZK}}$ publishes $\{a_j^{k*}\}_{k\in[s]}$
- for each adaptively corrupted client $C_i \in C_A$,
  - $\mathsf{Sim}_{\mathsf{EPZK}}$ emulates each $\mathcal{F}_{\mathsf{DComOT}}^k$ and gives $\{b_i^k\}_{C_i\in C_A}$ to $\mathcal{A}$
- in the reveal phase for each $\mathcal{F}_{\mathsf{DComOT}}^k$,
  - publish $(\{s_b^{j,k}\}_{j\in[d]\cup\{0\}}, \{b_i^k\}_{i\in[m]})$

Let κ be a computational security parameter. Let $\mathcal{R}$ be the space of randomness for all participating parties and $\vec{r}$ denote the contents of their random tapes. The view produced in the above execution is distributed as,

$$\Big\{\{\alpha_j, w_j\}_{j\in[d]\cup\{0\}}, \{b_i^k\}_{i\in[m],k\in[s]}, \\ \{a_j^k, s_0^{j,k}, s_1^{j,k}\}_{k\in[s],S_j\in\mathcal{S}^*}, \{a_j^{k*}, s_{b^k}^{j,k*}\}_{k\in[s],S_j\notin\mathcal{S}^*}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

Let s be the number of provers not corrupted by $\mathcal{A}$. Consider the following s^2+1 hybrid distributions:

- Hybrid H_0. This is the distribution of the output of $\mathsf{Sim}_{\mathsf{EPZK}}$. This differs from the distribution of the view of $\mathcal{A}$ in the real execution only in that $\mathcal{A}$ receives the output of the Σ-protocol simulator for all honest provers.

$$\mathsf{H}_0 = \Big\{\{\alpha_j, w_j\}_{j\in[d]\cup\{0\}}, \{b_i^k\}_{i\in[m],k\in[s]}, \\ \{a_j^k, s_0^{j,k}, s_1^{j,k}\}_{k\in[s],S_j\in\mathcal{S}^*}, \{a_j^{k*}, s_{b^k}^{j,k*}\}_{k\in[s],S_j\notin\mathcal{S}^*}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

- Hybrid H_{js+k}. For each $j\in[s], k\in[s]$, this is the distribution output by an intermediate hybrid experiment in which all the messages up to the $js+k^{th}$ Σ-protocol execution messages of an honest prover are generated as in the real protocol execution. The rest of the messages are generated as in the simulation.

$$\mathsf{H}_{js+k} = \{\{\alpha_{j'}, w_{j'}\}_{j'\in[d]\cup\{0\}}, \{b_i^{k'}\}_{i\in[m],k'\in[s]}, \\ \{a_{j'}^{k'}, s_0^{j',k'}, s_1^{j',k'}\}_{k'\in[s],S_{j'}\in\mathcal{S}^*}, \{a_j^{k'}, s_{b^{k'}}^{j',k'}\}_{k'<k,(k'=k,j'<j)}, \\ \{a_j^{k'*}, s_{b^{k'}}^{j',k'*}\}_{k'>k,(k'=k,j'\geq j)}\}$$

  The last of the above hybrids H_{ss} is the distribution produced by the real execution of the protocol.

Claim 9 *Assume that* Sigma *is a* Σ*-protocol with soundness* $\frac{1}{2}$ *such that* $b\in\{0,1\}\leftarrow\mathsf{Sigma}.V_1(\cdot)$*, for the relation* $(\alpha_0,w_0)\in\mathcal{R}_0$ *and for each* $j\in[d]$*, the relation* $(\alpha_j,w_j)\in\mathcal{R}_1$*. Then for each* $i\in[ss]$*, the views in the hybrid distributions* H_i *and* H_{i-1} *are computationally indistinguishable.*

Proof. Let $j\in[s], k\in[s]$ be such that $i=js+k$. Then the hybrid distributions H_i and H_{i-1} differ only in the way that the j^{th} honest prover's k^{th} Σ-protocol transcript is produced. In H_i, this is produced as in the real execution of the Σ-protocol. In H_{i-1}, this is produced as given in the output of the simulator $\mathsf{Sigma.Sim}$. Since the Σ-protocol satisfies 'special honest-verifier zero-knowledge', it follows that both these transcripts are computationally indistinguishable.

However, if there existed a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_i and H_{i-1} with non-negligible advantage ϵ, then $\mathcal{A}$ can be used in a black-box way as a subroutine by a PPT distinguisher D to distinguish between the output of $\mathsf{Sigma.Sim}$ and the Σ-protocol. D works by first generating the view of the protocol up to the j^{th} honest prover's $k-1^{th}$ Σ-protocol transcript as in

the real execution. It then gives the statement α_j, witness w_j and the bit b^k to the challenger $\mathcal{C}$ that returns a message m. D then completes the rest of the view of the protocol as in the simulation $\mathsf{Sim}_{\mathsf{EPZK}}$ and gives this to $\mathcal{A}$. Note that if $m = (a_j^{k*}, b^k, s_{b^k}^{j,k*})$ as output by $\mathsf{Sigma.Sim}$ then the complete transcript belongs in the distribution H_{i-1}. Otherwise, if $m = (a_j^k, b^k, s_{b^k}^{j,k})$ as in the real execution of the algorithms in Sigma, then the transcript belongs in the distribution H_i. Finally, D outputs whatever $\mathcal{A}$ outputs. In this experiment D's advantage is the same as that of $\mathcal{A}$, which is non-negligible. However since Sigma is a secure Σ-protocol, no such D can exist and hence no such $\mathcal{A}$ can exist.

Therefore, as none of the listed adjacent hybrids are distinguishable and there are polynomially many such hybrids, it follows that the distributions of the real and simulated transcripts are also computationally indistinguishable.

Realizing EPZK in the Plain Model.

Protocol Realizing $\mathcal{F}_{\mathsf{EPZK}}$

Let s be a statistical security parameter. Let $\mathsf{DComOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{dec}})$ (Definition 11) be a protocol realizing $\mathcal{F}_{\mathsf{DComOT}}$. Let $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ be a Σ-protocol with soundness $\frac{1}{2}$ where $a \leftarrow P_1(x, w)$, $b \in \{0,1\} \leftarrow V_1(\cdot)$, $c \leftarrow P_2(a, b, x, w)$, $\{0,1\} \leftarrow V_2(a, b, c, x)$. Consider the following protocol $\mathsf{EPZK} = (\mathsf{CRSgen}, \mathsf{EPZK}_1, \mathsf{EPZK}_P, \mathsf{EPZK}_2, \mathsf{EPZK}_V)$:

1. $\mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa)$

   $\mathsf{CRS} \leftarrow \mathsf{DComOT.CRSgen}(1^\kappa)$

2. $\forall i \in [m]$, $\mathsf{Aux}_i, m_1^i \leftarrow \mathsf{EPZK}_1(\mathsf{CRS})$

   $\forall k \in [s]$, $b_i^k \leftarrow \{0,1\}$ $\qquad \mathsf{Aux}_i = \{\mathsf{Aux}_i^k\}_{k\in[s]}$

   $\forall k \in [s], \mathsf{Aux}_i^k, \mathbf{m_1^{k,i}} \leftarrow \mathsf{DComOT.OT}_1(\mathsf{CRS}; b_i^k)$ $\qquad m_1^i = \{\mathbf{m_1^{k,i}}\}_{k\in[s]}$

3. $\forall j \in [d] \cup \{0\}$, $\pi_j \leftarrow \mathsf{EPZK}_P(\mathsf{CRS}; \alpha_j, w_j, \{m_1^i\}_{i\in[m]})$

   $\forall k \in [s]$, $a_j^k \leftarrow \mathsf{Sigma}.P_1(\alpha_j, w_j)$

   $\forall k \in [s]$, $s_0^{j,k} \leftarrow \mathsf{Sigma}.P_2(a_j^k, 0, \alpha_j, w_j)$ $\qquad s_1^{j,k} \leftarrow \mathsf{Sigma}.P_2(a_j^k, 1, \alpha_j, w_j)$

   $\forall k \in [s], \mathbf{m_2^{j,k}} \leftarrow \mathsf{DComOT.OT}_2(\mathsf{CRS}; \{\mathbf{m_1^{k,i}}\}_{i\in[m]}; s_0^{j,k}, s_1^{j,k})$

   $\pi_j = (\alpha_j, \{a_j^k, \mathbf{m_2^{j,k}}\}_{k\in[s]})$

4. $\forall i \in [m]$, $m_2^i \leftarrow \mathsf{EPZK}_2(\mathsf{CRS}; \{\pi_j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i)$

   $\forall k \in [s], \mathbf{m_3^{i,k}} \leftarrow \mathsf{DComOT.OT}_{\mathsf{fin}}(\mathsf{CRS}; \{\mathbf{m_2^{j,k}}\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i^k)$ $\qquad m_2^i = \{\mathbf{m_3^{i,k}}\}_{k\in[s]}$

5. $b \leftarrow \mathsf{EPZK}_V(\mathsf{CRS}; \{m_1^i\}_{i\in[m]}, \{\pi_j\}_{j\in[d]\cup\{0\}}, \{m_2^i\}_{i\in[m]}, \{\alpha_j\}_{j\in[d]\cup\{0\}})$
   - $\forall k \in [s]$, compute

     $\{s_{b^k}^{j,k}\}_{j\in[d]\cup\{0\}}$,

     $\{b_i^k\}_{i\in[m]} \leftarrow \mathsf{DComOT.OT}_{\mathsf{dec}}(\mathsf{CRS}; \{\mathbf{m_1^{k,i}}\}_{i\in[m]}, \{\mathbf{m_2^{j,k}}\}_{j\in[d]\cup\{0\}}, \{\mathbf{m_3^{i,k}}\}_{i\in[m]})$

     $b^k = \bigoplus_{i=1}^m b_i^k$ $\qquad \forall j \in [d] \cup \{0\}, \mathbf{b_k^j} = \mathsf{Sigma}.V_2(a_j^k, b^k, s_{b^k}^{j,k}, \alpha_j)$

– output $b = \bigwedge_{k\in[s],j\in[d]\cup\{0\}} \mathbf{b_k^j}$

Fig. 23 Construction for $\mathcal{F}_{\mathsf{EPZK}}$

It is immediate that the protocol in Figure 22 has the same communication pattern as in Definition 13 and is therefore a SCALES protocol.

5.5 Verifiable Updatable OT Protocol

Given all the building blocks described in the previous sections, we are now ready to describe a protocol that realizes $\mathcal{F}_{\mathsf{VUOT}}$ (Figure 10).

Theorem 2. *Let $\mathcal{R}_0^{\mathcal{M}}$ and $\mathcal{R}_1^{\Sigma}$ be two NP-relations. Let $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ be a UOT protocol (Definition 7) that UC-securely realizes $\mathcal{F}_{\mathsf{UOT}}^{\mathcal{M},\Sigma}$. Then the protocol in Figure 24 realizes $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_0^{\mathcal{M}},\mathcal{R}_1^{\Sigma}}$ (Figure 10) in the $(\mathcal{F}_{\mathsf{EPZK}}^{\mathcal{R}_0^{\mathcal{M}},\mathcal{R}_1^{\Sigma}}, \mathcal{F}_{\mathcal{B}})$-hybrid, where these functionalities are as in Figure 21 and Figure 6 respectively.*

Protocol Realizing $\mathcal{F}_{\mathsf{VUOT}}^{\mathcal{R}_0,\mathcal{R}_1}$ in the $(\mathcal{F}_{\mathsf{EPZK}}^{\mathcal{R}_0,\mathcal{R}_1}, \mathcal{F}_{\mathcal{B}})$-hybrid

Let $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ be a UOT protocol (Definition 7) that computes $\mathcal{F}_{\mathsf{UOT}}$ (Figure 12) among 1 sender, 1 receiver and d updaters. Let $\mathcal{F}_{\mathsf{EPZK}}$ (Figure 21) be a zero-knowledge functionality for $d+1$ servers and m challengers. Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the set of receivers where C_i has input $x_i \in \{0,1\}$. Let S be a sender with inputs $\{s_i^0, s_i^1\}_{i\in[m]}$ where each string $s_i^b \in \{0,1\}^\ell$. Let $\mathcal{U} = \{U_j\}_{j\in[d]}$ be updaters s.t. U_j has $\{\sigma_i^j\}_{i\in[m]}$ and each $\sigma_i^j : \{0,1\}^\ell \to \{0,1\}^\ell$. Let $\mathcal{R}_0$ be a relation with statement α_0 and witness w_0 and $\mathcal{R}_1$ be a relation s.t. for each $j \in [d]$, statement α_j and witness w_j,

$$\{0,1\} \leftarrow \mathcal{R}_0(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]})$$
$$\{0,1\} \leftarrow \mathcal{R}_1(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]})$$

Let $\mathcal{F}_{\mathcal{B}}$ be a bulletin-board. The protocol VUOT operates as follows:

$$\mathsf{CRS} \leftarrow \mathsf{UOT.CRSgen}(1^\kappa)$$

1. **Commit Phase:** All receivers $C_i \in \mathcal{C}$ send $\mathbf{m_1^i}$ to $\mathcal{F}_{\mathcal{B}}$ where

$$\mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_1^i} \leftarrow \mathsf{UOT.OT}_1(\mathsf{CRS}; x_i)$$

2. **Initial OT:** Sender S computes,

$$\forall i \in [m], \mathbf{m_2^{0,i}} \leftarrow \mathsf{UOT.OT}_2(\mathsf{CRS}; \mathbf{m_1^i}, (s_i^0, s_i^1))$$
$$\alpha, w \to \mathcal{F}_{\mathsf{EPZK}} \text{ where, } \quad \alpha = \alpha_0, \{\mathbf{m_2^{0,i}}\}_{i\in[m]} \quad w = w_0, \{s_i^0, s_i^1\}_{i\in[m]}$$

$\mathcal{F}_{\mathsf{EPZK}}$ publishes α and all receivers $C_i \in \mathcal{C}$ compute,

$$M_{i,0} \leftarrow \mathsf{UOT.OT}_{\mathsf{fin}}(\mathsf{CRS}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{0,i}})$$
$$\{s_{x_i}^{0,i}, \perp\} \leftarrow \mathsf{UOT.OT}_{\mathsf{vrfy}}(\mathsf{CRS}; M_{i,0})$$

3. **Successive Update:** Each updater $U_j \in \mathcal{U}$ computes,

$$\forall i \in [m], \mathbf{m_2^{j,i}} \leftarrow \mathsf{UOT.OT_{updt}}(\mathsf{CRS}; \mathbf{m_2^{j-1,i}}, \sigma_i^j)$$
$$\alpha, w \rightarrow \mathcal{F}_{\mathsf{EPZK}} \text{ where, } \quad \alpha = \alpha_j, \{\mathbf{m_2^{j,i}}\}_{i\in[m]} \quad w = w_j, \{\sigma_i^j\}_{i\in[m]}$$

$\mathcal{F}_{\mathsf{EPZK}}$ publishes α and all receivers $C_i \in \mathcal{C}$ compute,

$$M_{i,j} \leftarrow \mathsf{UOT.OT_{fin}}(\mathsf{CRS}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{j,i}})$$
$$\{s_{x_i}^{j,i}, \perp\} \leftarrow \mathsf{UOT.OT_{vrfy}}(\mathsf{CRS}; M_{i,j})$$

4. **Verification Phase:** $\mathcal{F}_{\mathsf{EPZK}}$ publishes $\{b_j\}_{j\in[d]\cup\{0\}}$
5. If $b = \bigcap_{j\in[d]\cup\{0\}} b_j == 1$, all receivers $C_i \in \mathcal{C}$ send $M_{i,d}$ to $\mathcal{F}_{\mathcal{B}}$ where

$$M_{i,d} \leftarrow \mathsf{UOT.OT_{fin}}(\mathsf{CRS}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{d,i}})$$

Any party can compute, $\forall i \in [m], \{s_{x_i}^{d,i}, \perp\} \leftarrow \mathsf{UOT.OT_{vrfy}}(\mathsf{CRS}; M_{i,d})$

Fig. 24 Construction for $\mathcal{F}_{\mathsf{VUOT}}$ in the $\mathcal{F}_{\mathsf{EPZK}}$-hybrid

Proof. We need to show that the functionality $\mathcal{F}_{\mathsf{VUOT}}$ (Figure 10) is realized by the protocol in Figure 24. Let $\mathcal{B}$ be the bulletin-board. Let $\mathcal{A}$ be a PPT malicious adversary. Among the parties that $\mathcal{A}$ corrupts, let $C_S \subset \mathcal{R}$ be the set of statically corrupted receivers. Let $C_A \subset \mathcal{R} - C_S$ denote the set of receivers that $\mathcal{A}$ adaptively corrupts after the 'commit phase' in $\mathcal{F}_{\mathsf{VUOT}}$. Let $\mathcal{S} \subset \{S\} \cup \mathcal{U}$ be the set of statically corrupted updaters and sender such that at least one party is honest. Let the honest updater/sender be E_{j*}. Note that $\mathcal{A}$ may also corrupt parties not involved in the computation and it has access to everything on $\mathcal{B}$.

Let $\mathcal{F}_{\mathsf{EPZK}}^{\mathcal{R}_0,\mathcal{R}_1}$ (Figure 21) be the functionality realizing ephemeral-prover zero-knowledge for relations $\mathcal{R}_0$, $\mathcal{R}_1$. Let $\mathsf{UOT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT_{updt}}, \mathsf{OT_{fin}}, \mathsf{OT_{vrfy}})$ be a UOT protocol (Definition 7) realizing the functionality $\mathcal{F}_{\mathsf{UOT}}$ (Figure 12). For $i \in [m]$, let UOT_i be the instance of the UOT functionality with R_i as the receiver and let $\mathsf{Sim}_{\mathsf{UOT}}^i$ be the corresponding PPT simulator that can simulate the view of this protocol in the presence of an adversary corrupting all-but-one updaters and sender statically and the receiver adaptively. Then let Sim be a PPT simulator in the $\mathcal{F}_{\mathsf{EPZK}}$-hybrid that uses $\{\mathsf{Sim}_{\mathsf{UOT}}^i\}_{i\in[m]}$ as a black-box and operates as follows:

1. In the 'commit phase' of the protocol,
   - for each statically corrupted receiver $R_i \in C_S$, Sim calls $\mathsf{Sim}_{\mathsf{UOT}}^i$ and extracts x_i from $\mathcal{A}$.
   - Sim passes $\{x_i\}_{R_i\in C_S}$ to $\mathcal{F}_{\mathsf{VUOT}}$.
2. For every receiver $R_i \in C_A$ that $\mathcal{A}$ adaptively corrupts,
   - Sim sends i to $\mathcal{F}_{\mathsf{VUOT}}$ and receives x_i.
   - Sim gives x_i to $\mathcal{A}$.
3. In the 'initial OT' phase, if S is corrupted,
   - Sim emulates $\mathcal{F}_{\mathsf{EPZK}}$ and receives (α_0, w_0) from $\mathcal{A}$.
     It extracts $\{s_0^i, s_1^i\}_{i\in[m]} \in w_0$ and $\forall i \in [m]$, invoke $\mathsf{Sim}_{\mathsf{UOT}}^i$ with (s_0^i, s_1^i).
   - Sim sends $(\{s_0^i, s_1^i\}_{i\in[m]}, \alpha_0, w_0 p)$ to $\mathcal{F}_{\mathsf{VUOT}}$.
     Otherwise, it sends ABORT.

– Sim computes $b_0 = \mathcal{R}_0(\alpha_0, w_0)$ and posts α_0 onto $\mathcal{B}$.
– Sim uses $\{s_i^0 = s_{x_i}^i\}_{R_i \in C_A \cup C_S}$ to complete executing each $\mathsf{Sim}_{\mathsf{UOT}}^i$.

If S is honest, $\mathcal{F}_{\mathsf{VUOT}}$ posts α_0 onto $\mathcal{B}$ and Sim gets for each corrupted R_i the string $s_{x_i}^i$ using which it completes the execution of $\mathsf{Sim}_{\mathsf{UOT}}^i$ with $\mathcal{A}$.

4. In the 'successive update' phase, if U_j is corrupted,
   – Sim emulates $\mathcal{F}_{\mathsf{EPZK}}$ and receives (α_j, w_j) from $\mathcal{A}$.
   It extracts $\{\sigma_j^i\}_{i\in[m]} \in w_j$ and $\forall i \in [m]$, it continues $\mathsf{Sim}_{\mathsf{UOT}}^i$ with σ_j^i.
   – Sim sends $(\{\sigma_j^i\}_{i\in[m]}, \alpha_j, w_j)$ to $\mathcal{F}_{\mathsf{VUOT}}$.
   Otherwise, it sends ABORT.
   – Sim computes $b_j = \mathcal{R}_1(\alpha_j, w_j)$ and posts α_j onto $\mathcal{B}$.
   It uses $\{s_{x_i}^j = \sigma_j^i(s_i^{j-1})\}_{R_i \in C_A \cup C_S}$ to complete each $\mathsf{Sim}_{\mathsf{UOT}}^i$ with $\mathcal{A}$.

   If U_j is honest, $\mathcal{F}_{\mathsf{VUOT}}$ posts α_j onto $\mathcal{B}$ and Sim gets for each corrupted R_i the string s_i^j using which it completes the execution of $\mathsf{Sim}_{\mathsf{UOT}}^i$ with $\mathcal{A}$.
5. In the 'verification phase',
   – Sim emulates $\mathcal{F}_{\mathsf{EPZK}}$ and posts $b = \bigcap_{j\in[d]\cup\{0\}} b_j$ onto $\mathcal{B}$.
6. If $b = 0$, publish ABORT. Otherwise,
   – For each $i \in [m]$, Sim completes the execution of $\mathsf{Sim}_{\mathsf{UOT}}^i$.

Let κ be the computational security parameter. Let $\mathcal{R}$ be the space of randomness of all participating parties and let $\vec{r}$ be the contents of the random tape of the participating parties. Let us denote by $\tau_{\mathsf{Sim}_{\mathsf{UOT}}^i}$ the transcript output by the execution of $\mathsf{Sim}_{\mathsf{UOT}}^i$. The view of produced by Sim is distributed as,

$$\Big\{\{x_i\}_{i\in[m]}, \{s_0^i, s_1^i\}_{i\in[m]}, \{\sigma_j^i\}_{i\in[m],j\in[d]}, \{\alpha_j, \{s_i^j\}_{R_i\in C_S\cup C_A}\}_{j\in\{0\}\cup[d]},$$
$$\{w_j\}_{j\neq j^*\in[d]}, b, \{s_i^d\}_{i\in[m]}, \{\tau_{\mathsf{Sim}_{\mathsf{UOT}}^i}\}_{i\in[m]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

This is identical to the real view with the exception that all the messages in all the instances of the UOT protocol are simulated. The distribution of the above view can be show as computationally indistinguishable from the real view by considering a set of $m+1$ hybrids as follows:

– Hybrid H_0. This is the distribution of the output of Sim. This differs from the distribution of the view in the real execution only in that all the UOT sub-protocol transcripts are simulated.

$$\mathsf{H}_0 = \Big\{\{x_i\}_{i\in[m]}, \{s_0^i, s_1^i\}_{i\in[m]}, \{\sigma_j^i\}_{i\in[m],j\in[d]}, \{\alpha_j, \{s_i^j\}_{R_i\in C_S\cup C_A}\}_{j\in\{0\}\cup[d]},$$
$$\{w_j\}_{j\neq j^*\in[d]}, b, \{s_i^d\}_{i\in[m]}, \{\tau_{\mathsf{Sim}_{\mathsf{UOT}}^i}\}_{i\in[m]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_k. For each $k \in [m]$, this is the distribution output by an intermediate hybrid experiment in which all the messages of the first k UOT protocol executions are generated as in the real protocol execution. The rest of the messages are generated as in the UOT simulation.

$$\mathsf{H}_k = \Big\{\{x_i\}_{i\in[m]}, \{s_0^i, s_1^i\}_{i\in[m]}, \{\sigma_j^i\}_{i\in[m],j\in[d]}, \{\alpha_j, \{s_i^j\}_{R_i\in C_S\cup C_A}\}_{j\in\{0\}\cup[d]},$$
$$\{w_j\}_{j\neq j^*\in[d]}, b, \{s_i^d\}_{i\in[m]}, \{\tau_{\mathsf{UOT}}^i\}_{i\leq k}, \{\tau_{\mathsf{Sim}_{\mathsf{UOT}}^i}\}_{i>k\in[m]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

The last of the above hybrids H_m is the distribution produced by the real execution of the protocol.

Claim 10 *Assume that* UOT *is a UOT protocol that realizes the functionality* $\mathcal{F}_{\mathsf{UOT}}$ *(Definition 7) in the presence of a PPT malicious adversary corrupting all-but-one of the sender and updaters statically and the receiver adaptively. Then for each* $k \in [m]$*, the views in the hybrid distributions* H_k *and* H_{k-1} *are computationally indistinguishable.*

Proof. The hybrid distributions H_k and H_{k-1} differ only in the way that the k^{th} UOT sub-protocol transcript is produced. In H_k, this is produced as in the real execution of the functions in UOT. In H_{k-1}, this is produced as given in the output of the simulator $\mathsf{Sim}_{\mathsf{UOT}}$. Since UOT securely realizes $\mathcal{F}_{\mathsf{UOT}}$ (Definition 7) in the presence of a PPT malicious adversary corrupting all-but-one of the sender and updaters statically and the receiver adaptively, it follows that both these transcripts are computationally indistinguishable.

However, if there existed a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_k and H_{k-1} with non-negligible advantage ϵ, then $\mathcal{A}$ can be used in a black-box way as a subroutine by a PPT distinguisher D to distinguish between the output of $\mathsf{Sim}_{\mathsf{UOT}}$ and the protocol UOT. D works by first generating the view of the protocol up to the $k-1^{th}$ UOT protocol transcript as in the real execution. It then gives the inputs x_k, (s_0^k, s_1^k) and $\{\sigma_k^j\}_{j\in[d]}$ to the challenger $\mathcal{C}$ that returns a message m. D then completes the rest of the view of the protocol as in the simulation $\mathsf{Sim}_{\mathsf{UOT}}$ and gives this to $\mathcal{A}$. Note that if $m = \tau_{\mathsf{Sim}^k_{\mathsf{UOT}}}$ as output by $\mathsf{Sim}_{\mathsf{UOT}}$ then the complete transcript belongs in the distribution H_{k-1}. Otherwise, if $m = \tau^k_{\mathsf{UOT}}$ as in the real execution of the algorithms in UOT, then the transcript belongs in the distribution H_k. Finally, D outputs whatever $\mathcal{A}$ outputs. In this experiment D's advantage is the same as that of $\mathcal{A}$, which is non-negligible. However since UOT is a secure protocol, no such D can exist and hence no such $\mathcal{A}$ can exist.

Therefore, as none of the listed adjacent hybrids are distinguishable and there are polynomially many such hybrids, it follows that the distributions of the real and simulated transcripts are also computationally indistinguishable.

Verifiable Updatable OT in the Plain Model.

Protocol Realizing $\mathcal{F}^{\mathcal{R}_0,\mathcal{R}_1}_{\mathsf{VUOT}}$

Let $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ be a UOT protocol (Definition 7) that for all $i \in [m]$, computes $\mathcal{F}^i_{\mathsf{UOT}}$ (Figure 12) for 1 sender, 1 receiver and $d+1$ updaters. Let $\mathsf{EPZK} = (\mathsf{CRSgen}, \mathsf{EPZK}_1, \mathsf{EPZK}_P, \mathsf{EPZK}_2, \mathsf{EPZK}_V)$ be a zero-knowledge protocol (Definition 13) for $d+1$ servers and m challengers computing $\mathcal{F}_{\mathsf{EPZK}}$ (Figure 21).
Let $\mathcal{R} = \{R_i\}_{i\in[m]}$ be the set of receivers where R_i has input $x_i \in \{0,1\}$.
Let S be a sender with inputs $\{s_i^0, s_i^1\}_{i\in[m]}$ where each string $s_i^b \in \{0,1\}^\ell$.
Let $\mathcal{R}_0$ be a relation with statement α_0 and witness w_0 s.t.,

$$\{0,1\} \leftarrow \mathcal{R}_0(\alpha_0, w_0, \{s_i^0, s_i^1\}_{i\in[m]})$$

Let $\mathcal{U} = \{U_j\}_{j\in[d]}$ be updaters s.t. U_j has $\{\sigma_i^j\}_{i\in[m]}$ and $\sigma_i^j : \{0,1\}^\ell \to \{0,1\}^\ell$.
For each $j \in [d]$, let $\mathcal{R}_1$ be a relation with statement α_j and witness w_j s.t.,

$$\{0,1\} \leftarrow \mathcal{R}_1(\alpha_{j-1}, \alpha_j, w_j, \{\sigma_i^j\}_{i\in[m]})$$

Let $\mathcal{B}$ be a bulletin-board. The protocol VUOT operates as follows:

$$\mathsf{CRS}_{\mathsf{UOT}} \leftarrow \mathsf{UOT.CRSgen}(1^\kappa)$$
$$\mathsf{CRS}_{\mathsf{EPZK}} \leftarrow \mathsf{EPZK.CRSgen}(1^\kappa)$$

1. **Commit Phase:** All receivers $R_i \in \mathcal{R}$ post $(\mathbf{m_1^i}, m_1^i)$ on to $\mathcal{B}$ where,

$$\mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_1^i} \leftarrow \mathsf{UOT.OT}_1(\mathsf{CRS}_{\mathsf{UOT}}; x_i)$$
$$\mathsf{Aux}_i^{\mathsf{EPZK}}, m_1^i \leftarrow \mathsf{EPZK.EPZK}_1(\mathsf{CRS}_{\mathsf{EPZK}})$$

2. **Initial OT:** Sender S posts $(\alpha_0, \pi_0, \{\mathbf{m_2^{0,i}}\}_{i\in[m]})$ on to $\mathcal{B}$ where,

$$\forall i \in [m], \mathbf{m_2^{0,i}} \leftarrow \mathsf{UOT.OT}_2(\mathsf{CRS}_{\mathsf{UOT}}; \mathbf{m_1^i}, (s_i^0, s_i^1))$$
$$\pi_0 \leftarrow \mathsf{EPZK.EPZK}_P(\mathsf{CRS}_{\mathsf{EPZK}}; \alpha, w, \{m_1^i\}_{i\in[m]})$$
$$\text{where,} \quad \alpha = \alpha_0, \{\mathbf{m_2^{0,i}}, \mathbf{m_1^i}\}_{i\in[m]} \qquad w = w_0, \{s_i^0, s_i^1\}_{i\in[m]}$$

All receivers $R_i \in \mathcal{R}$ compute,

$$M_{i,0} \leftarrow \mathsf{UOT.OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{0,i}})$$
$$\{s_{x_i}^{0,i}, \perp\} \leftarrow \mathsf{UOT.OT}_{\mathsf{vrfy}}(\mathsf{CRS}_{\mathsf{UOT}}; M_{i,0})$$

3. **Successive Update:** Each updater $U_j \in \mathcal{U}$ posts $(\alpha_j, \pi_j, \{\mathbf{m_2^{j,i}}\}_{i\in[m]})$ on to $\mathcal{B}$,

$$\forall i \in [m], \mathbf{m_2^{j,i}} \leftarrow \mathsf{UOT.OT}_{\mathsf{updt}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathbf{m_2^{j-1,i}}, \sigma_i^j)$$
$$\pi_j \leftarrow \mathsf{EPZK.EPZK}_P(\mathsf{CRS}_{\mathsf{EPZK}}; \alpha, w, \{m_1^i\}_{i\in[m]})$$
$$\text{where,} \quad \alpha = \alpha_{j-1}, \alpha_j, \{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}\}_{i\in[m]} \qquad w = w_j, \{\sigma_i^j\}_{i\in[m]}$$

All receivers $R_i \in \mathcal{R}$ compute,

$$M_{i,j} \leftarrow \mathsf{UOT.OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{j,i}})$$
$$\{s_{x_i}^{j,i}, \perp\} \leftarrow \mathsf{UOT.OT}_{\mathsf{vrfy}}(\mathsf{CRS}_{\mathsf{UOT}}; M_{i,j})$$

4. **Verification Phase:** All receivers $R_i \in \mathcal{R}$ post m_2^i to $\mathcal{B}$ where,

$$m_2^i \leftarrow \mathsf{EPZK.EPZK}_2(\mathsf{CRS}_{\mathsf{EPZK}}; \{\pi_j\}_{j\in[d]\cup\{0\}}, \mathsf{Aux}_i^{\mathsf{EPZK}})$$

All verifying servers V_v posts b_v to $\mathcal{B}$ where,

$$b_v \leftarrow \mathsf{EPZK.EPZK}_V(\mathsf{CRS}_{\mathsf{EPZK}}; \{m_1^i\}_{i\in[m]}, \{\pi_j\}_{j\in[d]\cup\{0\}}, \{m_2^i\}_{i\in[m]}, \{\alpha_j\}_{j\in[d]\cup\{0\}})$$

5. If $b = \bigcap_{v\in[c]} b_v == 1$, all receivers $R_i \in \mathcal{R}$ post $M_{i,d}$ to $\mathcal{B}$ where,

$$M_{i,d} \leftarrow \mathsf{UOT.OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{d,i}})$$

Any party can compute,

$$\forall i \in [m], \{s_{x_i}^{d,i}, \perp\} \leftarrow \mathsf{UOT.OT}_{\mathsf{vrfy}}(\mathsf{CRS}_{\mathsf{UOT}}; M_{i,d})$$

Fig. 25 Construction for $\mathcal{F}_{\mathsf{VUOT}}$

6 A Malicious SCALES Protocol

In this section we summarize our protocol with all the compositions implemented. This realization of $\mathcal{F}_{\mathsf{RGS}}$ in the $\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathcal{B}}$-hybrid model is a maliciously secure SCALES protocol computing any functionality f. The protocol (Construction 26) uses an instantiation for $\mathcal{F}_{\mathsf{RGS}}$ (Construction 11) using an RGS and an instantiation of $\mathcal{F}_{\mathsf{VUOT}}$ (Construction 24). It uses the following sub-protocols and objects as building blocks:

- A Projective RGS (Definition 3) object that is also Explanable (Definition 9). Let this be $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$.
- A compatible UOT protocol (Definition 8) that is Explanable (Definition 10). Let $\mathsf{UOT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{updt}}, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_{\mathsf{vrfy}})$ have sender's inputs as the projective encoding, and updaters' functions as the encoding transformations.
- An Ephemeral Prover Zero-Knowledge protocol for proving the correctness of encoding (Definition 13). Let this be $\mathsf{EPZK} = (\mathsf{CRSgen}, \mathsf{EPZK}_1, \mathsf{EPZK}_P, \mathsf{EPZK}_2, \mathsf{EPZK}_V)$.

Protocol Maliciously Secure SCALES Protocol for f

Let κ be a computational security parameter. Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the set of clients where each C_i has input bit x_i. Let $\mathcal{B}$ be the bulletin-board. The functionality $\mathcal{F}_{\mathsf{CRS}}$ generates the following and gives it to all parties:

$$\mathsf{CRS}_{\mathsf{UOT}} \leftarrow \mathsf{UOT.CRSgen}(1^\kappa)$$
$$\mathsf{CRS}_{\mathsf{EPZK}} \leftarrow \mathsf{EPZK.CRSgen}(1^\kappa)$$

1. **Computation phase:**
   - $\forall i \in [m]$, client C_i posts $(\mathbf{m_1^i}, m_1^i)$ onto $\mathcal{B}$ where,

$$\mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_1^i} \leftarrow \mathsf{UOT.OT}_1(\mathsf{CRS}_{\mathsf{UOT}}; x_i)$$
$$\mathsf{Aux}_i^{\mathsf{EPZK}}, m_1^i \leftarrow \mathsf{EPZK.EPZK}_1(\mathsf{CRS}_{\mathsf{EPZK}})$$

   - Ephemeral server E_0 posts $(F_0, \pi_0, \{\mathbf{m_2^{0,i}}\}_{i\in[m]})$ onto $\mathcal{B}$ where,

$$(F_0, e_0) \leftarrow \mathsf{RGS.Gb}(f; r_0) \qquad e_0 = \{\mathsf{L}_{i,0}^0, \mathsf{L}_{i,0}^1\}_{i\in[m]}$$
$$\forall i \in [m], \mathbf{m_2^{0,i}} \leftarrow \mathsf{UOT.OT}_2(\mathsf{CRS}_{\mathsf{UOT}}; \mathbf{m_1^i}, (\mathsf{L}_{i,0}^0, \mathsf{L}_{i,0}^1), r_0^i)$$
$$\pi_0 \leftarrow \mathsf{EPZK.EPZK}_P(\mathsf{CRS}_{\mathsf{EPZK}}; \alpha, w, \{m_1^i\}_{i\in[m]}) \quad \text{for relation } \mathcal{R}_{\mathsf{Gb}} \text{ where,}$$
$$\alpha = F_0, \{\mathbf{m_2^{0,i}}, \mathbf{m_1^i}\}_{i\in[m]} \qquad w = r_0, \{r_0^i\}_{i\in[m]}, \{\mathsf{L}_{i,0}^0, \mathsf{L}_{i,0}^1\}_{i\in[m]}$$

   - For round $j \in [c_1 - 1]$, server E_j posts $(F_j, \pi_j, \{\mathbf{m_2^{j,i}}\}_{i\in[m]})$ to $\mathcal{B}$ where,

$$(F_j, \lambda_j) \leftarrow \mathsf{RGS.Rerand}(F_{j-1}; r_j) \qquad \lambda_j = \{\sigma_i^j\}_{i\in[m]}$$
$$\forall i \in [m], \mathbf{m_2^{j,i}} \leftarrow \mathsf{UOT.OT}_{\mathsf{updt}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathbf{m_2^{j-1,i}}, \sigma_i^j, r_j^i)$$
$$\pi_j \leftarrow \mathsf{EPZK.EPZK}_P(\mathsf{CRS}_{\mathsf{EPZK}}; \alpha, w, \{m_1^i\}_{i\in[m]}) \quad \text{for relation } \mathcal{R}_{\mathsf{Rerand}} \text{ where,}$$
$$\alpha = F_{j-1}, F_j, \{\mathbf{m_2^{j-1,i}}, \mathbf{m_2^{j,i}}\}_{i\in[m]} \qquad w = r_j, \{r_j^i\}_{i\in[m]}, \{\sigma_i^j\}_{i\in[m]}$$

2. **Verification phase:**

– $\forall i \in [m]$, client C_i posts m_2^i onto $\mathcal{B}$ where,

$$m_2^i \leftarrow \mathsf{EPZK.EPZK}_2(\mathsf{CRS}_{\mathsf{EPZK}}; \{\pi_j\}_{j\in[c_1-1]\cup\{0\}}, \mathsf{Aux}_i^{\mathsf{EPZK}})$$

– For each round $k \in [c_2]$, each ephemeral server V_k posts b_k onto $\mathcal{B}$ where,

$$b_k \leftarrow \mathsf{EPZK.EPZK}_V(\mathsf{CRS}_{\mathsf{EPZK}}; \{m_1^i\}_{i\in[m]}, \{\pi_j\}_{j\in[c_1-1]\cup\{0\}}, \{m_2^i\}_{i\in[m]}, \{\alpha_j\}_{j\in[c_1-1]\cup\{0\}})$$

3. **Decoding phase:**

– If $\bigcap_{v\in[c_2]} b_v == 1$, $\forall i \in [m]$, each client C_i posts M_{i,c_1-1} onto $\mathcal{B}$ where,

$$M_{i,c_1-1} \leftarrow \mathsf{UOT.OT}_{\mathsf{fin}}(\mathsf{CRS}_{\mathsf{UOT}}; \mathsf{Aux}_i^{\mathsf{UOT}}, \mathbf{m_2^{c_1-1,i}})$$

– For each round $j \in [c_3]$, each ephemeral server D_j posts $f(x)$ onto $\mathcal{B}$ where,

$$\forall i \in [m], \{L_{i,c_1-1}^{x_i}, \perp\} \leftarrow \mathsf{UOT.OT}_{\mathsf{vrfy}}(\mathsf{CRS}_{\mathsf{UOT}}; M_{i,c_1-1})$$
$$X_{c_1} = \{L_{i,c_1-1}^{x_i}\}_{i\in[m]}$$
$$f(x) = \mathsf{RGS.Ev}(F_{c_1-1}, X_{c_1})$$

Each client reads $\mathcal{B}$ and accepts the final output if all servers in the Decoding phase publish the same output.

Fig. 26 Maliciously Secure SCALES Protocol

Theorem 3. *Assuming the hardness of the DDH problem, Figure 26 is a maliciously secure SCALES protocol (Definition 2) computing any function f.*

The proof of Theorem 3 follows from Theorem 1 – 2 and that the communication pattern is as required in SCALES (Definition 1).

Acknowledgements. Anasuya Acharya and Carmit Hazay were partially supported by the United States-Israel Binational Science Foundation (BSF) through Grant No.2020277. Vladimir Kolesnikov was supported in part by Visa research award, Cisco research award, and NSF awards CNS-2246354, and CCF-2217070. Manoj Prabhakaran was supported in part by the IIT Bombay Trust Lab. Anasuya Acharya, Carmit Hazay and Manoj Prabhakaran were also supported by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Algorand Foundation.

References

AHKP22. Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. SCALES - MPC with small clients and larger ephemeral servers. In *TCC 2022*, pages 502–531, 2022.

ASH+20. Jackson Abascal, Mohammad Hossein Faghihi Sereshgi, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Is the classical GMW paradigm practical? the case of non-interactive actively secure 2pc. In *CCS '20: 2020 ACM SIGSAC*, pages 1591–1605, 2020.

BDO23. Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *CRYPTO 2023*, pages 613–645, 2023.

BGG+20. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC*, pages 260–290, 2020.

BGSZ22. James Bartusek, Sanjam Garg, Akshayaram Srinivasan, and Yinuo Zhang. Reusable two-round MPC from LPN. In *PKC 2022*, pages 165–193, 2022.

BHHO08. Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

BJKL21. Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In *EUROCRYPT 2021*, pages 724–753, 2021.

BMR90. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

CGG+21. Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: secure multiparty computation with dynamic participants. In *CRYPTO*, pages 94–123, 2021.

GGHR14. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014.

GHK+21. Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: you only speak once - secure MPC with stateless ephemeral roles. In *CRYPTO*, pages 64–93, 2021.

GHM+21. Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In *TCC 2021*, pages 32–61, 2021.

GHV10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In *CRYPTO*, pages 155–172, 2010.

GMPS21. Vipul Goyal, Elisaweta Masserova, Bryan Parno, and Yifan Song. Blockchains enable non-interactive MPC. In *TCC 2021*, pages 162–193, 2021.

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018*, pages 468–499, 2018.

HIMV19. Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. In *CCS 2019*, pages 327–344. ACM, 2019.

HKE13. Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO 2013*, pages 18–35, 2013.

IKO+11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *EUROCRYPT 2011*, pages 406–425, 2011.

JKKR17. Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *CRYPTO 2017*, pages 158–189, 2017.

KRY22. Sebastian Kolby, Divya Ravi, and Sophia Yakoubov. Towards efficient YOSO MPC without setup. *IACR Cryptol. ePrint Arch.*, page 187, 2022.
Lin13. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *CRYPTO 2013*, pages 1–17, 2013.
LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *J. Cryptol.*, 22(2):161–188, 2009.
PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.
RS22. Rahul Rachuri and Peter Scholl. Le mans: Dynamic and fluid MPC for dishonest majority. In *CRYPTO 2022*, pages 719–749, 2022.
YHKD22. Yibin Yang, David Heath, Vladimir Kolesnikov, and David Devecsery. EZEE: epoch parallel zero knowledge for ANSI C. In *IEEE EuroS&P 2022*, pages 109–123, 2022.
YPHK23. Yibin Yang, Stanislav Peceny, David Heath, and Vladimir Kolesnikov. Towards generic MPC compilers via variable instruction set architectures (visas). *IACR Cryptol. ePrint Arch.*, page 953, 2023.
YWZ20. Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In *SIGSAC*, pages 1627–1646, 2020.

Appendix

A Additional Preliminaries

Definition 14. *Two probability ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ defined over a finite domain D are **statistically indistinguishable**, denoted $X \overset{s}{\approx} Y$, if every positive polynomial $p(\cdot)$ and all sufficiently large n's,*

$$\Delta(X_n, Y_n) < \frac{1}{p(n)}$$

where,

$$\Delta(X_n, Y_n) = \frac{1}{2} \cdot \sum_{\alpha\in D} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|$$

Definition 15. *Two probability ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are **computationally indistinguishable**, denoted $X \overset{c}{\approx} Y$, if for every PPT distinguisher D, every positive polynomial $p(\cdot)$ and all sufficiently large n's,*

$$|\Pr[D(X_n, 1^n) = 1] - \Pr[D(Y_n, 1^n) = 1]| < \frac{1}{p(n)}$$

A.1 Key and Message Homomorphic Encryption

We re-state the definition of strong Key and Message Homomorphic Encryption from [AHKP22].

Definition 16. *A **strong key-and-message homomorphic encryption scheme** (strong KMHE) is the set of PPT algorithms* $\mathsf{KMH} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *defined on domains of (private) keys, messages and ciphertexts* $\mathcal{K}, \mathcal{M}, \mathcal{C}$*, a key transformation family* $\mathcal{F}_{key}$*, and a message transformation family* $\mathcal{F}_{msg}$ *(all indexed by an implicit security parameter* κ*) such that the following conditions hold:*

- ***Correctness**:* $\forall m \in \mathcal{M}, k \in \mathcal{K}$*,*

$$\Pr[k \leftarrow \mathsf{Gen}(1^\kappa); \mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

- ***KMH Correctness**:* $\forall m \in \mathcal{M}, k \in \mathcal{K}$*,* $f \in \mathcal{F}_{key}$*,* $g \in \mathcal{F}_{msg}$*,*

$$\Pr[k \leftarrow \mathsf{Gen}(1^\kappa); \mathsf{Dec}(f(k), \mathsf{Eval}(\mathsf{Enc}(k, m), f, g)) = g(m)] = 1$$

- ***Key Privacy**:* $\forall k, k' \leftarrow \mathsf{Gen}(1^\kappa)$*,* $f \in \mathcal{F}_{key}$*,*

$$\{k, f(k)\} \overset{s}{\approx} \{k, k'\}$$

- ***KMH Privacy**:* $\forall$ *PPT adversary* $\mathcal{A}$*, the advantage* $\Pr[b' = b] \leq \frac{1}{2} + \nu(\kappa)$ *for a negligible function* ν *in the following experiment (*κ *being an implicit input to* C *and* $\mathcal{A}$*):*
  1. C *samples a uniform random bit* $b \leftarrow \{0, 1\}$*, keys* $k_0, k_1, k' \leftarrow \mathsf{Gen}(1^\kappa)$*, and* $f \leftarrow \mathcal{F}_{key}$*. It sends* $(k_0, k_1, f(k_1))$ *to* $\mathcal{A}$*.*
  2. *For as many times as* $\mathcal{A}$ *wants:*
     - $\mathcal{A}$ *produces arbitrary* $m, m' \in \mathcal{M}$ *and* $g \in \mathcal{F}_{msg}$*, and computes* $c \leftarrow \mathsf{Enc}(k_0, m)$*. It sends* (c, g, m') *to* C*.*
     - C *sends* c_b *to* $\mathcal{A}$*, where* $c_0 \leftarrow \mathsf{Eval}(c, f, g)$ *and* $c_1 \leftarrow \mathsf{Enc}(k', m')$*.*
  3. $\mathcal{A}$ *outputs* b'*.*

[AHKP22] contains a construction for strong KMHE based on the [BHHO08] encryption scheme.

A.2 Rerandomizable Garbled Circuits

The work in [AHKP22] also shows an instantiation of a projective RGS with rerandomizable garbled circuits (RGC). The construction is based on that in [GHV10] but uses a sharable variant of strong KMHE (Definition 16) as the underlying encryption primitive.

Sharable KMHE (Definition 17) is a strong KMHE that is modified to facilitate garbling and rerandomizing circuits. Like most traditional garbling, the RGC construction we employ also garbles a circuit gate-by-gate. Each gate is garbled by designating labels for both values 0 and 1 of each input and out wire of the gate. Then a set of ciphertexts are created by encrypting an output label using one input label from each wire, according to the truth table of the gate functionality. In our construction, these ciphertexts are created using strong KMHE. Note that since circuit intermediate wires can be both input and output wires of gates, it becomes necessary that the key space $\mathcal{K}$ of strong KMHE be a subset of the message space $\mathcal{M}$. Further, we apply the same operation to all

ciphertexts with the same wire label for rerandomizing. So it is also necessary that $\mathcal{F}_{key}$ be contained in $\mathcal{F}_{msg}$.

Traditional garbling of 2-input gates employs double encryption of the output label using both input labels, creating one ciphertext. However, this would complicate matters if used as it is for RGCs since it requires the ciphertext space to also be within the message space and the domains of $\mathcal{F}_{key}$ and $\mathcal{F}_{msg}$ would need to be readjusted. Therefore, we garble by creating a 2-out-of-2 secret sharing of the output label and then encrypting each share with one input label as the key. This makes it necessary to have efficient sharing and reconstruction algorithm Share and Recon. Share is a randomized algorithm that takes as input an element from $\mathcal{K}$ and outputs two elements from $\mathcal{M}$ such that no element in isolation reveals the input. Recon is deterministic and can derive the input to Share given both shares.

In order to be compatible with rerandomizing, we require two additional properties. Firstly, we need that the shares can be rerandomized. That is, given a pair of shares s_0 and s_1 of a message k, there exist functions h and $\overline{h}$ such that $h(s_0)$ and $\overline{h}(s_1)$ are also a sharing of k. These functions must come from a domain $\mathcal{F}^*_{msg}$ such that when h is picked at random, applying h and $\overline{h}$ induces a fresh sharing of k. Since rerandomizing requires that operations are performed on ciphertexts, it becomes necessary that these operations are homomorphically applied $\mathcal{F}^*_{msg} \subseteq \mathcal{F}_{msg}$.

Secondly, note that when applying function $\sigma \in \mathcal{F}_{key}$ to the message space, it needs to be applied to shares of the label and not the label itself. Therefore, we need the additional property that any such σ applied to the shares translates to it being applied to the reconstructed value as well. Combining the two requirements, Definition 17 states that a distribution containing a sharing of a key k and, for any $\sigma \in \mathcal{F}_{key}$, the sharing of $\sigma(k)$ be identically distributed to one containing a sharing of k and a pair containing σ and h applied to one share and σ and $\overline{h}$ applied to the other.

Definition 17. *A **sharable key-and-message homomorphic encryption scheme** is a set of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Share}, \mathsf{Recon})$ *where* $\mathsf{KMH} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is a strong KMHE scheme as in Definition 16 for domains of (private) keys $\mathcal{K}$, messages $\mathcal{M}$ and ciphertexts $\mathcal{C}$, a key transformation family $\mathcal{F}_{key}$, and a message transformation family $\mathcal{F}_{msg}$ with the additional property that $\mathcal{K} \subseteq \mathcal{M}$ and $\mathcal{F}_{key} \subseteq \mathcal{F}_{msg}$.*

The scheme has two additional PPT functions (1) $([k]_0, [k]_1) \leftarrow \mathsf{Share}(k)$ that outputs two random shares $[k]_0, [k]_1 \in \mathcal{M}$ of a key $k \in \mathcal{K}$. (2) $k \leftarrow \mathsf{Recon}([k]_0, [k]_1)$ that reconstructs the label k from its shares. Further there exists $\mathcal{F}^*_{msg} \subseteq \mathcal{F}_{msg}$ such that, $\forall \sigma \in \mathcal{F}_{key}$, $\forall h \in \mathcal{F}^*_{msg}, \exists \overline{h} \in \mathcal{F}^*_{msg}$ s.t. $\forall k \in \mathcal{K}$,

$$\begin{aligned}
&\left\{[k]_0, [k]_1, [\sigma(k)]_0, [\sigma(k)]_1\right\}_{\substack{([k]_0,[k]_1)\leftarrow\mathsf{Share}(k);\\ ([\sigma(k)]_0,[\sigma(k)]_1)\leftarrow\mathsf{Share}(\sigma(k))}} \\
\equiv &\left\{[k]_0, [k]_1, h(\sigma([k]_0)), \overline{h}(\sigma([k]_1))\right\}_{([k]_0,[k]_1)\leftarrow\mathsf{Share}(k)}
\end{aligned}$$

Circuit Notation. For a function $f : \{0,1\}^m \to \{0,1\}^l$, a boolean circuit that computes it is denoted by $C = (\mathcal{W}, I, O, \mathcal{G})$. $\mathcal{W}$ is the set of all wires and $I \subset \mathcal{W}$ and $O \subset \mathcal{W}$ are the set of input and output wires respectively. Within $\mathcal{W}$, $I = (w_1, \cdots, w_m)$ are the m input wires, $w_{m+1}, \cdots, w_{m+p}$ are the p internal wires, and $O = (w_{m+p+1}, \cdots, w_{m+p+l})$ are the l output wires. These make $v = m+p+l$ total wires. $\mathcal{G} = (g_{m+1}, ..., g_{m+q})$ is the set of gates. Each $g_i = (w_\ell, w_r, w_i, op)$ is a binary gate where w_ℓ and w_r are the left and right input wires respectively, w_i is the output wire (uniquely defined by the gate index), and op represents the gate functionality (AND, XOR, etc.).

Construction 1 *We denote by* $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{Rerand})$ *a rerandomizable garbling scheme where all the algorithms are instantiated with a sharable KMHE (Definition 17) scheme* KMH *as the underlying encryption scheme.*

1. *The garbling algorithm* $\mathsf{Gb}(C, 1^\kappa)$*:*
   - *For every wire* $w_i \in \mathcal{W} - O$*, sample labels* $L^0_{w_i}, L^1_{w_i} \leftarrow \mathsf{KMH.Gen}(1^\kappa)$.
   - *For every output wire* $w_i \in O$*, use the same labels* $\mathsf{L}_0, \mathsf{L}_1 \in \mathcal{K}$ *across all output wires. These are publicly known.*
   - *For every gate* $g_i = (w_\ell, w_r, w_i, op) \in \mathcal{G}$*, for each of the 4 rows, let* $([L^b_{w_i}]_0, [L^b_{w_i}]_1) \leftarrow \mathsf{KMH.Share}(L^b_{w_i})$ *be the shares of one of* g_i*'s output labels for* $b \in \{0,1\}$ *and* π *be a permutation on four positions. Then the garbling of gate* g_i *can be defined as:*

$$G_i = \begin{bmatrix} \pi[0,0] : \mathsf{KMH.Enc}(L^0_{w_\ell}, [L^{op(0,0)}_{w_i}]_0), \mathsf{KMH.Enc}(L^0_{w_r}, [L^{op(0,0)}_{w_i}]_1) \\ \pi[0,1] : \mathsf{KMH.Enc}(L^0_{w_\ell}, [L^{op(0,1)}_{w_i}]_0), \mathsf{KMH.Enc}(L^1_{w_r}, [L^{op(0,1)}_{w_i}]_1) \\ \pi[1,0] : \mathsf{KMH.Enc}(L^1_{w_\ell}, [L^{op(1,0)}_{w_i}]_0), \mathsf{KMH.Enc}(L^0_{w_r}, [L^{op(1,0)}_{w_i}]_1) \\ \pi[1,1] : \mathsf{KMH.Enc}(L^1_{w_\ell}, [L^{op(1,1)}_{w_i}]_0), \mathsf{KMH.Enc}(L^1_{w_r}, [L^{op(1,1)}_{w_i}]_1) \end{bmatrix}$$

   - *Output* $\hat{\mathcal{C}} = ((G_1, \cdots, G_q), (\mathsf{L}_0, \mathsf{L}_1))$ *and* $\mathcal{L} = \{L^0_{w_i}, L^1_{w_i}\}_{w_i \in I}$.
2. *The encoding algorithm* $\mathsf{En}(\mathcal{L}, x)$ *gets a set of input labels* $\mathcal{L}$ *and the function input* $x = (x_1, \cdots, x_m)$ *and outputs* $\mathcal{I} = \{L^{x_i}_{w_i}\}_{w_i \in I}$.
3. *The evaluation algorithm* $\mathsf{Ev}(\hat{\mathcal{C}}, \mathcal{I})$ *:*
   The algorithm works gate by gate, by decrypting each row in the garbled gate.[8] *The resulting plaintexts are combined to the output label using* $\mathsf{KMH.Recon}$*. Evaluating a gate lets us derive one label for a wire in the circuit. Following the terminology of [LP09], this label is termed the active label of that wire. Such a label is also derived for each output wire of the circuit and this belongs in the set* $(\mathsf{L}_0, \mathsf{L}_1)$ *and can be mapped to output values 0 or 1. This set of labels yields the function's output* $f(x)$.
4. *The rerandomizing algorithm* $\mathsf{Rerand}(\hat{\mathcal{C}})$ *:*
   - *For all wires* $w_i \in \mathcal{W} - O$*, sample* $\sigma_i \leftarrow \mathcal{F}_{key}$.
   - *For all output wires* $w_i \in O$*, let* $\sigma_i = \mathsf{I}$ *be the identity function.*

[8] We assume that the evaluator identifies the valid output label by adding a fixed suffix to the plaintext as suggested originally in [LP09].

– *For all gates* $g_i \in \mathcal{G}$*, let* $(\sigma_\ell, \sigma_r, \sigma_i)$ *correspond to the wires* (w_ℓ, w_r, w_i)*. Let* π_i *be a permutation on four elements. Sample* $h_0, h_1, h_2, h_3 \leftarrow \mathcal{F}^*_{msg}$ *and derive* $\overline{h_0}, \overline{h_1}, \overline{h_2}, \overline{h_3} \in \mathcal{F}^*_{msg}$*. In order to rerandomize* G_i *into* G^*_i*, the following is carried out:*

$$G'_i = \begin{bmatrix} \pi_i[0,0] : \mathsf{KMH.Eval}(c_{0,0}, \sigma_\ell, \sigma_i), \mathsf{KMH.Eval}(c_{0,1}, \sigma_r, \sigma_i) \\ \pi_i[0,1] : \mathsf{KMH.Eval}(c_{1,0}, \sigma_\ell, \sigma_i), \mathsf{KMH.Eval}(c_{1,1}, \sigma_r, \sigma_i) \\ \pi_i[1,0] : \mathsf{KMH.Eval}(c_{2,0}, \sigma_\ell, \sigma_i), \mathsf{KMH.Eval}(c_{2,1}, \sigma_r, \sigma_i) \\ \pi_i[1,1] : \mathsf{KMH.Eval}(c_{3,0}, \sigma_\ell, \sigma_i), \mathsf{KMH.Eval}(c_{3,1}, \sigma_r, \sigma_i) \end{bmatrix} \text{ where } G_i = \begin{bmatrix} c_{0,0}, c_{0,1} \\ c_{1,0}, c_{1,1} \\ c_{2,0}, c_{2,1} \\ c_{3,0}, c_{3,1} \end{bmatrix}$$

$$G^*_i = \begin{bmatrix} \mathsf{KMH.Eval}(c'_{0,0}, \mathsf{I}, h_0), \mathsf{KMH.Eval}(c'_{0,1}, \mathsf{I}, \overline{h_0}) \\ \mathsf{KMH.Eval}(c'_{1,0}, \mathsf{I}, h_1), \mathsf{KMH.Eval}(c'_{1,1}, \mathsf{I}, \overline{h_1}) \\ \mathsf{KMH.Eval}(c'_{2,0}, \mathsf{I}, h_2), \mathsf{KMH.Eval}(c'_{2,1}, \mathsf{I}, \overline{h_2}) \\ \mathsf{KMH.Eval}(c'_{3,0}, \mathsf{I}, h_3), \mathsf{KMH.Eval}(c'_{3,1}, \mathsf{I}, \overline{h_3}) \end{bmatrix} \text{ where } G'_i = \begin{bmatrix} c'_{0,0}, c'_{0,1} \\ c'_{1,0}, c'_{1,1} \\ c'_{2,0}, c'_{2,1} \\ c'_{3,0}, c'_{3,1} \end{bmatrix}$$

– *Output* $\hat{\mathcal{C}}' = ((G^*_1, \cdots, G^*_q), (\mathsf{L}_0, \mathsf{L}_1))$ *and* $\Pi = \{\sigma_i\}_{w_i \in I}$*.*

A.3 Proof of Lemma 2

Proof. For RGCs in Construction 1, let $\mathsf{KMH} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Share}, \mathsf{Recon})$ be a Sharable Strong KMHE scheme (Definition 17). Then, for $F = \mathsf{Gb}(f; r)$, the garbling randomness r is comprised of:

– $\forall$ wire $w \in \mathcal{W}$, labels $L^0_w, L^1_w \leftarrow \mathsf{Gen}(1^\kappa)$.
– $\forall$ gate $g \in \mathcal{G}$, the sharing randomness $\{s^g_i\}_{i\in[4]}$ used in Share.
– $\forall$ gate $g \in \mathcal{G}$, the encryption randomness $\{r^g_{i0}, r^g_{i1}\}_{i\in[4]}$ used in Enc.

For $F = \mathsf{Rerand}(F^*; r')$, the randomness r' is comprised of the following:

– $\forall$ wire $w \in \mathcal{W}$, a transformation function $\sigma_w \leftarrow \mathcal{F}_{key}$.
– $\forall$ gate $g \in \mathcal{G}$, the share randomizing functions $\{h_i \in \mathcal{F}^*_{msg}\}_{i\in[4]}$.
– $\forall$ gate $g \in \mathcal{G}$, the rerandomizing randomness $\{r^g_{i0}, r^g_{i1}\}_{i\in[4]}$ used in Eval.

Given (r, r') such that $F = \mathsf{Gb}(f; r)$ and $F' = \mathsf{Rerand}(F; r')$, let $\mathsf{Explain}(r, r')$ be a function that operates as follows:

– parse $r = (\{L^0_w, L^1_w\}_{w\in\mathcal{W}}, \{s^g_i\}_{i\in[4],g\in\mathcal{G}}, \{r^g_{i0}, r^g_{i1}\}_{i\in[4],g\in\mathcal{G}})$.
– parse $r' = (\{\sigma_w\}_{w\in\mathcal{W}}, \{h_i\}_{i\in[4],g\in\mathcal{G}}, \{r'^g_{i0}, r'^g_{i1}\}_{i\in[4],g\in\mathcal{G}})$.
– return r^* where,

$$r^* = (\{\sigma_w(L^0_w), \sigma_w(L^1_w)\}_{w\in\mathcal{W}}, \{h_i(s^g_i)\}_{i\in[4],g\in\mathcal{G}}, \{r^g_{i0} + r'^g_{i0}, r^g_{i1} + r'^g_{i1}\}_{i\in[4],g\in\mathcal{G}})$$

For the above function $\mathsf{Explain}$, $F' = \mathsf{Gb}(f, r^*)$ by construction (Construction 1). We now show that *explainable garbling privacy* is satisfied:

$$\{f, X, F, F', r^*\}_{\substack{r,r' \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ F' = \mathsf{Rerand}(F;r'); r^* = \mathsf{Explain}(r,r')}}$$

$$\overset{c}{\approx} \{f, X, F, G, s\}_{\substack{r,s \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ G = \mathsf{Gb}(f;s)}}$$

Consider the following set of hybrids:

– Hybrid H_0. This is the same distribution as the LHS of the above equation. It consists of a function f, a garbling F of f, the set of active input labels X corresponding to F, a garbling F' that is rerandomized from F and the composed randomness r^* that is output from $\mathsf{Explain}$.

$$\mathsf{H}_0 = \left\{f, X, F, F', r^*\right\}_{\substack{r,r' \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ F' = \mathsf{Rerand}(F;r'); r^* = \mathsf{Explain}(r,r')}}$$

– Hybrid H_1. This is a hybrid distribution consisting of a function f and a garbling F of f, with input labels X as in H_0. This distribution additionally contains randomness r_1 from the distribution,

$$R_1 = \left\{\{\sigma_w(L_w^0), \sigma_w(L_w^1)\}_{w \in \mathcal{W}}, \{s_i^{*g}\}_{i \in [4], g \in \mathcal{G}}, \right. \\ \left. \{r_{i0}^g + r_{i0}'^g, r_{i1}^g + r_{i1}'^g\}_{i \in [4], g \in \mathcal{G}}\right\}_{\substack{r,r' \leftarrow R; \\ \{L_w^0, L_w^1\}_{w \in \mathcal{W}}, \{r_{i0}^g, r_{i1}^g\}_{i \in [4], g \in \mathcal{G}} \in r; \\ \{\sigma_w\}_{w \in \mathcal{W}}, \{r_{i0}'^g, r_{i1}'^g\}_{i \in [4], g \in \mathcal{G}} \in r'}}$$

and a garbling F_1 of f generated using the randomness r_1. Note that r_1 is different from $r^* \in \mathsf{H}_0$ in that the randomness used in sharing labels $\{s_i^{*g}\}_{i \in [4], g \in \mathcal{G}}$ is different and freshly sampled.

$$\mathsf{H}_1 = \left\{f, X, F, F_1, r_1\right\}_{\substack{r \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ r_1 \leftarrow R_1; F_1 = \mathsf{Gb}(f;r_1)}}$$

– Hybrid H_2. This is a hybrid distribution consisting of a function f, a garbling F of f, with input labels set X as in H_1. This distribution additionally contains randomness r_2 that is distributed as,

$$R_2 = \left\{\{\sigma_w(L_w^0), \sigma_w(L_w^1)\}_{w \in \mathcal{W}}, \{s_i^{*g}\}_{i \in [4], g \in \mathcal{G}}, \right. \\ \left. \{r_{i0}^{g*}, r_{i1}^{g*}\}_{i \in [4], g \in \mathcal{G}}\right\}_{\substack{r,r' \leftarrow R; \\ \{L_w^0, L_w^1\}_{w \in \mathcal{W}} \in r; \\ \{\sigma_w\}_{w \in \mathcal{W}} \in r'}}$$

and a garbling F_2 of f generated using the randomness r_2. Note that r_2 is different from $r_1 \in \mathsf{H}_1$ in that the randomness used for encryption is freshly sampled.

$$\mathsf{H}_2 = \left\{f, X, F, F_2, r_2\right\}_{\substack{r \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ r_2 \leftarrow R_2; F_2 = \mathsf{Gb}(f;r_2)}}$$

– Hybrid H_3. This is a hybrid distribution consisting of a function f, a garbling F of f, with input labels set X as in H_2. It additionally contains randomness s that is completely freshly sampled and a garbling G of f generated using s. It differs from r_2 in that the labels for each wire is a freshly sampled label, instead of being a rerandomized label. This is also the distribution in the RHS of the equation for *explainable garbling privacy*.

$$\mathsf{H}_3 = \left\{f, X, F, G, s\right\}_{\substack{r,s \leftarrow R; (e,F) = \mathsf{Gb}(f;r); X = \mathsf{En}(e,x); \\ G = \mathsf{Gb}(f;s)}}$$

Claim 11 *The hybrid distributions* H_0 *and* H_1*, and* H_1 *and* H_2 *are identically distributed.*

Proof. The hybrid distributions H_0 and H_1 differ only in the that the sharing randomness is $\{h_i(s_i^g)\}_{i\in[4],g\in\mathcal{G}} \in r^* \in \mathsf{H}_0$ and $\{s_i^{*g}\}_{i\in[4],g\in\mathcal{G}} \in r_1 \in \mathsf{H}_1$. In both hybrids, this sharing randomness originates from the same distribution due to a property of sharable KMHE. So it follows that $r^*, r_1 \leftarrow R$. In both hybrids, the elements f, F and X are generated in the same way. In H_0, $F' = \mathsf{Gb}(f; r^*)$ and in H_1, $F_1 = \mathsf{Gb}(f; r_1)$. Since both garbling randomnesses are identically distributed, it follows that the garblings are as well. Therefore the hybrid distributions H_0 and H_1 on the whole are identically distributed.

The hybrids H_1 and H_2 are identically distributed as they only differ since the randomness used for encryption within $r_1 \in \mathsf{H}_1$ is a composition of randomnesses used for encryption and homomorphic operations, whereas that in $r_2 \in \mathsf{H}_2$ is fresh randomness. These come from the same distribution and induce identical distributions on the ciphertexts produced.

Claim 12 *Assuming* KMH *is a strong KMHE (Definition 16) satisfying Key Privacy, the hybrid distributions* H_2 *and* H_3 *are statistically indistinguishable.*

Proof. The hybrid distributions H_2 and H_3 are statistically close since their difference originates from the fact that the wire labels are $\{\sigma_w(L_w^0), \sigma_w(L_w^1)\}_{w\in\mathcal{W}} \in r_2 \in \mathsf{H}_2$ and $\{L_w^{0'}, L_w^{1'}\}_{w\in\mathcal{W}} \in s \in \mathsf{H}_3$. The rest of both the hybrids is created identically. As it follows from the Key Privacy property of strong KMHE (Definition 16) that $\forall w \in \mathcal{W}, b \in \{0,1\}$,

$$\{L_w^b, \sigma_w(L_w^b)\} \overset{\text{s}}{\approx} \{L_w^b, L_w^{b'}\}$$

and there are polynomially many such wires in the security parameter κ, the distributions H_2 and H_3 are also statistically close.

Since none of the above described pairs of adjacent hybrids are distinguishable, it follows that *explainable garbling privacy* holds.

It remains to show that *explainable rerandomizing privacy* (Definition 9) is satisfied. Given (r, r') such that $F = \mathsf{Rerand}(F^*; r)$ and $F' = \mathsf{Rerand}(F; r')$, consider a function $\mathsf{Explain}(r, r')$ that operates as follows:

- parse $r = (\{\sigma_w\}_{w\in\mathcal{W}}, \{h_i^g\}_{i\in[4],g\in\mathcal{G}}, \{r_{i0}^g, r_{i1}^g\}_{i\in[4],g\in\mathcal{G}})$.
- parse $r' = (\{\sigma'_w\}_{w\in\mathcal{W}}, \{h_i'^g\}_{i\in[4],g\in\mathcal{G}}, \{r_{i0}'^g, r_{i1}'^g\}_{i\in[4],g\in\mathcal{G}})$.
- return r^* where,

$$r^* = (\{\sigma'_w \circ \sigma_w\}_{w\in\mathcal{W}}, \{h_i^g \circ h_i'^g\}_{i\in[4],g\in\mathcal{G}}, \{r_{i0}^g + r_{i0}'^g, r_{i1}^g + r_{i1}'^g\}_{i\in[4],g\in\mathcal{G}})$$

For the above function $\mathsf{Explain}$, $F' = \mathsf{Rerand}(F^*, r^*)$ by construction (Construction 1). We need to show that *explainable rerandomizing privacy* is satisfied:

$$\{F^*, X, F, F', r^*\}_{\substack{r,r'\leftarrow R;(F^*,e)\leftarrow\mathsf{Gb}(f);(F,\pi)=\mathsf{Rerand}(F^*;r);\\ X=\mathsf{En}(\pi(e),x);F'=\mathsf{Rerand}(F;r');r^*=\mathsf{Explain}(r,r')}}$$

$$\overset{\text{c}}{\approx}\{F^*, X, F, G, s\}_{\substack{r,s\leftarrow R;(F^*,e)\leftarrow\mathsf{Gb}(f);(F,\pi)=\mathsf{Rerand}(F^*;r);\\ X=\mathsf{En}(\pi(e),x);G=\mathsf{Rerand}(F^*;s)}}$$

Consider the following set of hybrids:

– Hybrid H_0. This is the same distribution as the LHS of the above equation. It consists of a prior garbling F^*, a garbling F rerandomized from F^*, the set of active input labels X corresponding to F, a garbling F' that is rerandomized from F and the composed randomness r^* that is output from $\mathsf{Explain}$.

$$\mathsf{H}_0 = \left\{F^*, X, F, F', r^*\right\}_{\substack{r,r' \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi) = \mathsf{Rerand}(F^*;r); \\ X = \mathsf{En}(\pi(e),x); F' = \mathsf{Rerand}(F;r'); r^* = \mathsf{Explain}(r,r')}}$$

– Hybrid H_1. This is a hybrid distribution consisting of F^*, F and X as in H_0. This distribution additionally contains randomness r_1 from the distribution,

$$R_1 = \left\{\{\sigma'_w \circ \sigma_w\}_{w\in\mathcal{W}}, \{h_i^{*g}\}_{i\in[4],g\in\mathcal{G}}, \right. \\ \left. \{r^g_{i0} + r'^g_{i0}, r^g_{i1} + r'^g_{i1}\}_{i\in[4],g\in\mathcal{G}}\right\}_{\substack{r,r' \leftarrow R; \\ \{\sigma_w\}_{w\in\mathcal{W}}, \{r^g_{i0}, r^g_{i1}\}_{i\in[4],g\in\mathcal{G}} \in r; \\ \{\sigma'_w\}_{w\in\mathcal{W}}, \{r'^g_{i0}, r'^g_{i1}\}_{i\in[4],g\in\mathcal{G}} \in r'}}$$

and a garbling F_1 rerandomized from F^* generated using the randomness r_1. Note that r_1 is different from $r^* \in \mathsf{H}_0$ in that the functions applied on the label shares $\{f_i^{*g}\}_{i\in[4],g\in\mathcal{G}}$ is different and freshly sampled.

$$\mathsf{H}_1 = \left\{F^*, X, F, F_1, r_1\right\}_{\substack{r \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi) = \mathsf{Rerand}(F^*;r); \\ X = \mathsf{En}(\pi(e),x); r_1 \leftarrow R_1; F_1 = \mathsf{Rerand}(F^*;r_1)}}$$

– Hybrid H_2. This is a hybrid distribution consisting of F^*, F and X as in H_1. This distribution additionally contains randomness r_2 that is distributed as,

$$R_2 = \left\{\{\sigma'_w \circ \sigma_w\}_{w\in\mathcal{W}}, \{h_i^{*g}\}_{i\in[4],g\in\mathcal{G}}, \{r^{g*}_{i0}, r^{g*}_{i1}\}_{i\in[4],g\in\mathcal{G}}\right\}_{\substack{r,r' \leftarrow R; \{\sigma_w\}_{w\in\mathcal{W}} \in r; \\ \{\sigma'_w\}_{w\in\mathcal{W}} \in r'}}$$

and a garbling F_2 rerandomized from F^* generated using the randomness r_2. Note that r_2 is different from $r_1 \in \mathsf{H}_1$ in that the randomness used for encryption is freshly sampled.

$$\mathsf{H}_2 = \left\{F^*, X, F, F_2, r_2\right\}_{\substack{r \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi) = \mathsf{Rerand}(F^*;r); \\ X = \mathsf{En}(\pi(e),x); r_2 \leftarrow R_2; F_2 = \mathsf{Rerand}(F^*;r_2)}}$$

– Hybrid H_3. This is a hybrid distribution consisting of F^*, F and X as in H_2. It additionally contains randomness s that is completely freshly sampled and a garbling G rerandomized from F^* generated using s. It differs from r_2 in that the label transformations are freshly sampled, instead of being a composed function of label transformations. This is also the distribution in the RHS of the equation for *explainable rerandomizing privacy.*

$$\mathsf{H}_3 = \left\{F^*, X, F, G, s\right\}_{\substack{r,s \leftarrow R; (F^*,e) \leftarrow \mathsf{Gb}(f); (F,\pi) = \mathsf{Rerand}(F^*;r); \\ X = \mathsf{En}(\pi(e),x); G = \mathsf{Rerand}(F^*;s)}}$$

The hybrids H_0 and H_1 are identically distributed as they only differ since the randomness used for sharing the output labels in each garbled gate within r^* is a composition of the randomnesses used for two stages of rerandomizing

operations, whereas that in r_1 is fresh randomness. These come from the same distribution and induce identical distributions on the ciphertexts produced.

The hybrids H_1 and H_2 are identically distributed as they only differ since the randomness used for encryption within r_1 is a composition of the randomnesses used for homomorphic operations, whereas that in r_2 is fresh randomness. These come from the same distribution and induce identical distributions on the ciphertexts produced.

Finally, the hybrids H_2 and H_3 are indistinguishable since the their difference stems from the fact that the label transformations in s are freshly sampled while those in r_2 are a composition of label transformations. Both these belong in $\mathcal{F}_{key}$ and originate from the same distribution. Hence, rerandomizable garbled circuits as in Construction 1 are Explanable.

B 2-round Malicious OT in the CRS Model

In this section we discuss the oblivious transfer protocol in [PVW08]. It is a 2-round malicious secure OT protocol in the CRS model. We show that it is additionally secure against adaptive receiver corruption. The protocol employs a 'dual-mode cryptosystem' as a building block that we simplify and restate:

Definition 18. *A **Dual-Mode Cryptosystem** consists of a tuple of algorithms:*

- $(\mathsf{CRS}, t) \leftarrow \mathsf{Setup}(1^\kappa)$ *: outputs common reference string* CRS *and trapdoor* t
- $(pk, sk) \leftarrow \mathsf{Gen}(\mathsf{CRS}, \sigma)$ *: for a branch* $\sigma \in \{0,1\}$ *outputs a key pair*
- $c \leftarrow \mathsf{Enc}(\mathsf{CRS}, pk, b, m)$ *: outputs ciphertext* c *encrypting message* m *such that it can only be decrypted if* $b = \sigma$ *that was used to generate* pk
- $m \leftarrow \mathsf{Dec}(sk, c)$ *: outputs message* m *that the ciphertext encrypts if* b *used in the encryption equals* σ *used to create* sk
- $\sigma \leftarrow \mathsf{Branch}(\mathsf{CRS}, t, pk)$ *: outputs the branch* $\sigma \in \{0,1\}$ *used to create* pk
- $(pk, sk_0, sk_1) \leftarrow \mathsf{TrapGen}(\mathsf{CRS}, t)$ *: outputs public key* pk *and two corresponding secret keys, one to decrypt each branch*

The functions need to satisfy the following properties:

- ***Correctness:*** $\forall(\mathsf{CRS}, t) \leftarrow \mathsf{Setup}(1^\kappa), \forall\sigma \in \{0,1\}, (pk, sk) \leftarrow \mathsf{Gen}(\mathsf{CRS}, \sigma)$,
$$\Pr[m = \mathsf{Dec}(sk, \mathsf{Enc}(\mathsf{CRS}, pk, \sigma, m))] = 1$$
- ***Messy Branch Indistinguishability:*** $\forall(\mathsf{CRS}, t) \leftarrow \mathsf{Setup}(1^\kappa), \forall pk, m_0, m_1,$ $\forall b \leftarrow \mathsf{Branch}(\mathsf{CRS}, t, pk)$,
$$\{\mathsf{Enc}(\mathsf{CRS}, pk, \neg b, m_0)\} \overset{s}{\approx} \{\mathsf{Enc}(\mathsf{CRS}, pk, \neg b, m_1)\}$$
- ***Trapdoor Generation:*** $\forall(\mathsf{CRS}, t) \leftarrow \mathsf{Setup}(1^\kappa), \forall\sigma \in \{0,1\}, (pk', sk') \leftarrow$ $\mathsf{Gen}(\mathsf{CRS}, \sigma), \forall(pk, sk_0, sk_1) \leftarrow \mathsf{TrapGen}(\mathsf{CRS}, t)$,
$$\{pk', sk'\} \overset{s}{\approx} \{pk, sk_0\} \overset{s}{\approx} \{pk, sk_1\}$$
$$\Pr[m = \mathsf{Dec}(sk_0, \mathsf{Enc}(\mathsf{CRS}, pk, 0, m))] = 1$$
$$\Pr[m = \mathsf{Dec}(sk_1, \mathsf{Enc}(\mathsf{CRS}, pk, 1, m))] = 1$$

[PVW08] gives a construction of this cryptosystem that is based on the DDH hardness assumption. The details of the construction are as follows:

Construction 2 *Let $\mathbb{G}$ be a group with order p. Consider the following encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Branch}, \mathsf{TrapGen})$:*

1. $(\mathsf{CRS}, t) \leftarrow \mathsf{Setup}(1^\kappa)$
   - *sample* $g_0, g_1 \leftarrow \mathbb{G}$
   - *sample* $x_0, x_1 \leftarrow \mathbb{Z}_p$
   - *set* $h_0 = g_0^{x_0}$ *and* $h_1 = g_1^{x_1}$
   - $\mathsf{CRS} = (g_0, h_0, g_1, h_1)$
   - $t = (x_0, x_1)$
2. $(pk, sk) \leftarrow \mathsf{Gen}(\mathsf{CRS}, \sigma)$
   - *sample* $r \leftarrow \mathbb{Z}_p$
   - *set* $g = g_\sigma^r$ *and* $h = h_\sigma^r$
   - $pk = (g, h)$
   - $sk = r$
3. $c \leftarrow \mathsf{Enc}(\mathsf{CRS}, pk, b, m)$
   - *sample* $s, t \leftarrow \mathbb{Z}_p$
   - $pk_b = (g_b, h_b, g, h)$
   - *set* $u = g_b^s \cdot h_b^t$ *and* $v = g^s \cdot h^t$
   - *set* $c_1 = u$ *and* $c_2 = v \cdot m$
   - $c = c_1, c_2$
4. $m \leftarrow \mathsf{Dec}(sk, c)$
   - $m = \frac{c_2}{c_1^{sk}}$
5. $\sigma \leftarrow \mathsf{Branch}(\mathsf{CRS}, t, pk)$
   - *parse* $\mathsf{CRS} = (g_0, h_0, g_1, h_1)$
   - *parse* $t = (x_0, x_1)$
   - *parse* $pk = g, h$
   - *if* $g^{x_0} == h$, *set* $\sigma = 0$
   - *if* $g^{x_1} == h$, *set* $\sigma = 1$
6. $(pk, sk_0, sk_1) \leftarrow \mathsf{TrapGen}(\mathsf{CRS}, t)$
   - *parse* $\mathsf{CRS} = (g_0, h_0, g_1, h_1)$
   - *parse* $t = (x_0, x_1)$
   - *sample* $r \leftarrow \mathbb{Z}_p$
   - *set* $g = g_0^r$ *and* $h = h_0^r$
   - $pk = g, h$
   - $sk_0 = r$
   - $sk_1 = \frac{x_0 r}{x_1}$

For the above scheme, consider the following ciphertext rerandomizing function $y' \leftarrow \mathsf{Rerand}(\mathsf{CRS}, pk, b, y)$:

- parse $y = (c_1, c_2)$ and $\mathsf{CRS} = (g_0, h_0, g_1, h_1)$
- sample $s_b, t_b \leftarrow \mathbb{Z}_p$ and compute $c_1' = c_1 \cdot g_b^{s_b} \cdot h_b^{t_b}$ and $c_2' = c_2 \cdot g^{s_b} \cdot h^{t_b}$
- return $y' = (c_1', c_2')$

Now the details of the OT protocol are as follows:

Construction 3 *Let $\mathbb{G}$ be a group with order p. Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Branch}, \mathsf{TrapGen}, \mathsf{Rerand})$ be a dual-mode cryptosystem as in Construction 2. Consider the following OT protocol $\mathbf{OT} = (\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}, \mathsf{OT}_3)$:*

1. $\mathsf{CRS} \leftarrow \mathsf{CRSgen}(1^\kappa)$
   - *compute* $(\mathsf{CRS}, t) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)$
   - *sender S and receiver R gets* CRS
2. $(m_1, \mathsf{Aux}) \leftarrow \mathsf{OT}_1(\mathsf{CRS}; b)$
   - *R computes* $(pk, sk) \leftarrow \mathcal{E}.\mathsf{Gen}(\mathsf{CRS}, b)$
   - *set* $\mathsf{Aux} = sk$
   - *set* $m_1 = pk$ *and send* m_1 *to* S
3. $m_2 \leftarrow \mathsf{OT}_2(\mathsf{CRS}; m_1, s_0, s_1)$
   - *S parses* $m_1 = pk$
   - *sample* r_0 *and compute* $y_0 = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, 0, s_0; r_0)$
   - *sample* r_1 *and compute* $y_1 = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, 1, s_1; r_1)$

– *set* $m_2 = (y_0, y_1)$ *and send it to* R

4. $s_b \leftarrow \mathsf{OT}_{\mathsf{fin}}(\mathsf{CRS}; m_2, \mathsf{Aux})$
   – R *parses* $m_2 = (y_0, y_1)$ *and* $\mathsf{Aux} = sk$
   – *compute* $s_b \leftarrow \mathcal{E}.\mathsf{Dec}(sk, y_b)$
5. $m_2' \leftarrow \mathsf{OT}_3(\mathsf{CRS}; m_1, m_2)$
   – *parse* $m_2 = (y_0, y_1)$ *and* $m_1 = pk$
   – *sample* r_0 *and compute* $y_0' = \mathcal{E}.\mathsf{Rerand}(\mathsf{CRS}, pk, 0, y_0; r_0)$
   – *sample* r_1 *and compute* $y_1' = \mathcal{E}.\mathsf{Rerand}(\mathsf{CRS}, pk, 1, y_1; r_1)$
   – *set* $m_2' = (y_0', y_1')$

Lemma 9. *Assuming that Construction 2 is a secure dual-mode cryptosystem (Definition 18), the bit-OT protocol in [PVW08] as given in Construction 3 satisfies Definition 4 in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model.*

Proof Outline. We prove the security of the protocol by reducing it to the security of the dual-mode cryptosystem. The protocol in Construction 3 needs to securely satisfy three properties:

1. Maliciously Secure Bit-OT: We define a simulator that can simulate the view of a malicious PPT adversary that can statically corrupt the sender S and adaptively corrupt the receiver R. Then, without loss of generality, we consider the adversary's view in the case that only R is adaptively corrupted after the first message is sent. In this case, both OT messages are generated by the simulator with the help of a CRS and trapdoor that is honestly generated using Setup. This view can be shown as indistinguishable from the real view using a sequence of hybrids. First, the distribution of the honest sender's simulated message can be shown as indistinguishable from the real message due to the 'Messy Branch Indistinguishability' property of a dual-mode cryptosystem (Definition 18). Then the distribution of the honest receiver's simulated message can be shown as indistinguishable from the real message due to 'Trapdoor Generation' property.
2. Reusable first message: We define a similar simulator as above that can simulate the view of a malicious PPT adversary that can statically corrupt any subset of the senders $\{S_i\}_{i\in[\ell]}$ and adaptively corrupt the receiver R in an execution of the functionality $\mathcal{F}_{\mathsf{multiOT}}$ (Figure 8). We also use similar reductions to the 'Messy Branch Indistinguishability' and 'Trapdoor Generation' property of a dual-mode cryptosystem (Definition 18) to prove the indistinguishablility of the distributions of the real and simulated views of the adversary.
3. Rerandomizable second message: For the function $\mathbf{OT}.\mathsf{OT}_3$ it is immediate for the construction that the randomness sampled for this function is drawn from the same distribution as that used in $\mathbf{OT}.\mathsf{OT}_2$. We use this to argue that the rerandomizable second message property is satisfied.

Proof. Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Branch}, \mathsf{TrapGen})$ be a dual-mode cryptosystem (Definition 18 – Construction 2). The proof follows in three parts

corresponding to the 3 properties in Definition 4 that Construction 3 needs to satisfy.

First, we need to show that $(\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}) \in \mathbf{OT}$ is a maliciously secure bit-OT protocol in the CRS model that is secure in the presence of a PPT adversary $\mathcal{A}$ corrupting the sender S statically and the receiver R adaptively. Let us denote by $b \in \{0,1\}$ the input of R and by $a_0, a_1 \in \{0,1\}$ the input bits of S. Let $\mathsf{Sim}_{\mathsf{OT}}$ be a PPT adversary that simulates the view of $\mathcal{A}$ in an ideal execution of $\mathcal{F}_{\mathsf{OT}}$, working as follows:

1. $\mathsf{Sim}_{\mathsf{OT}}$ first invokes $(\mathsf{CRS}, t) \leftarrow \mathbf{OT}.\mathsf{CRSgen}(1^\kappa)$ and, emulating $\mathcal{F}_{\mathsf{CRS}}$, it gives CRS to $\mathcal{A}$.
2. If R is initially corrupted,
   - $\mathsf{Sim}_{\mathsf{OT}}$ accepts m_1 from $\mathcal{A}$ and parses $m_1 = pk = (g, h)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ computes $b' \leftarrow \mathcal{E}.\mathsf{Branch}(\mathsf{CRS}, t, pk)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ passes b' to $\mathcal{F}_{\mathsf{OT}}$.

   Otherwise, if R is initially honest,
   - $\mathsf{Sim}_{\mathsf{OT}}$ computes $(pk, sk_0, sk_1) \leftarrow \mathcal{E}.\mathsf{TrapGen}(\mathsf{CRS}, t)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ sends $m_1' = pk$ to $\mathcal{A}$.
3. If S is corrupted and R is initially honest,
   - $\mathsf{Sim}_{\mathsf{OT}}$ has (sk_0, sk_1).
   - $\mathsf{Sim}_{\mathsf{OT}}$ accepts m_2 from $\mathcal{A}$ and parses $m_2 = (y_0, y_1)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ computes $a_0' = \mathcal{E}.\mathsf{Dec}(sk_0, y_0)$ and $a_1' = \mathcal{E}.\mathsf{Dec}(sk_1, y_1)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ passes (a_0', a_1') to $\mathcal{F}_{\mathsf{OT}}$.

   If S is corrupted and R is initially corrupted,
   - $\mathsf{Sim}_{\mathsf{OT}}$ accepts m_2 from $\mathcal{A}$ and sends $\perp$ to $\mathcal{F}_{\mathsf{OT}}$.

   Otherwise, if S is honest,
   - $\mathsf{Sim}_{\mathsf{OT}}$ receives the output $a \in \{0,1\}$ from $\mathcal{F}_{\mathsf{OT}}$.
   - If R is corrupted, $\mathsf{Sim}_{\mathsf{OT}}$ has b, otherwise it picks $b \leftarrow \{0,1\}$ at random.
   - $\mathsf{Sim}_{\mathsf{OT}}$ sets $a_b = a$ and $a_{\neg b} = 0$ and computes $m_2' \leftarrow \mathbf{OT}.\mathsf{OT}_2(\mathsf{CRS}; pk, a_0, a_1)$.
   - $\mathsf{Sim}_{\mathsf{OT}}$ gives m_2' to $\mathcal{A}$.
4. If R is adaptively corrupted after the first message,
   - $\mathsf{Sim}_{\mathsf{OT}}$ receives b from $\mathcal{F}_{\mathsf{OT}}$ and sends (b, sk_b) to $\mathcal{A}$.

Let κ be the computational security parameter, $\mathcal{R}$ denote the space of randomness of all the parties participating in the protocol and $\vec{r}$ denote the contents of the random tape of all the parties and the simulator. Let us, without loss of generality, consider the case that S is honest and R is adaptively corrupted after the first message is sent. This is the case for which the simulator needs to output all the messages in the transcript. The output of this simulator is distributed as,

$$\left\{b, (a_0, a_1), m_1', m_2', sk_b\right\}_{\kappa \in \mathbb{N}, \vec{r} \in \mathcal{R}}$$

This distribution can be shown as indistinguishable from the real view using $\mathbf{OT}$ by reducing to the 'Messy Branch Indistinguishability' and 'Trapdoor Generation' property of a dual-mode cryptosystem (Definition 18). Consider the following set of hybrids:

– Hybrid H_0. This is the distribution of the output of $\mathsf{Sim}_{\mathsf{OT}}$. It differs from the view of the real execution in that all the messages in the transcript are simulated messages.

$$\mathsf{H}_0 = \left\{b, (a_0, a_1), m_1', m_2', sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_1. This is the distribution output by an intermediate hybrid experiment in which the sender S's message m_2 is generated as in the real execution but everything else is generated as in $\mathsf{Sim}_{\mathsf{OT}}$.

$$\mathsf{H}_1 = \left\{b, (a_0, a_1), m_1', m_2, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_2. This is the distribution of the view of the real execution of the protocol. It differs from the previous hybrid in that the simulated receiver's message is replaced by m_1 as produced in the real execution of **OT**.

$$\mathsf{H}_1 = \left\{b, (a_0, a_1), m_1, m_2, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

Claim 13 *Assuming that the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Messy Branch Indistinguishability', the views in hybrid distributions H_0 and H_1 are statistically close.*

Proof. The hybrids H_0 and H_1 differ only in the way that the honest sender's message is generated. In H_1 it is m_2, generated as in the real execution of the protocol. This can be parsed as $m_2 = (y_0, y_1)$ where $y_0 = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, 0, a_0)$ and $y_1 = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, 1, a_1)$. In H_0 it is the simulated message m_2'. Let b be the branch corresponding to pk. Then, $m_2' = (y_0', y_1')$ where $y_b' = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, b, a_b)$, same as in H_1, but $y_{\neg b}' = \mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, \neg b, 0)$. So the only difference between the two hybrid experiments is that the ciphertext encrypted using the messy branch $\neg b$ is different. However, since the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Messy Branch Indistinguishability', it holds that,

$$\left\{\mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, \neg b, 0)\right\} \overset{\mathrm{s}}{\approx} \left\{\mathcal{E}.\mathsf{Enc}(\mathsf{CRS}, pk, \neg b, a_{\neg b})\right\}$$

It follows from the triangle inequality of statistical differences that the difference between H_0 and H_1 can be no more than that of the above distribution and so these distributions are also statistically close.

Claim 14 *Assuming that the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Trapdoor Generation', the views in distributions H_1 and H_2 are statistically close.*

Proof. The hybrids H_1 and H_2 differ only in the way that the honest receiver's message is generated. In H_2 it is m_1, generated as in the real execution of the protocol. This can be parsed as $m_1 = pk$ where $(pk, sk) \leftarrow \mathcal{E}.\mathsf{PKgen}(\mathsf{CRS}, b)$. In H_1 it is the simulated message m_1'. This is generated as $m_1' = pk'$ where $(pk', sk_0, sk_1) \leftarrow \mathcal{E}.\mathsf{TrapGen}(\mathsf{CRS}, t)$. However, since the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Trapdoor Generation', it holds that,

$$\left\{pk, sk\right\} \overset{\mathrm{s}}{\approx} \left\{pk', sk_b\right\}$$

It follows from the triangle inequality of statistical differences that the difference between H_2 and H_1 can be no more than that of the above distribution and so these distributions are also statistically close.

Since none of the listed adjacent hybrids are distinguishable, it follows that the distributions of the real view of **OT** and the simulated view are also indistinguishable. Hence, this protocol $(\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}) \in \mathbf{OT}$ is a maliciously secure bit-OT protocol in the CRS model that is secure in the presence of a PPT adversary $\mathcal{A}$ corrupting the sender S statically and the receiver R adaptively.

Next, we need to show that $(\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}) \in \mathbf{OT}$ is a maliciously secure protocol in the CRS model realizing $\mathcal{F}_{\mathsf{multiOT}}$ (Figure 8) that is secure in the presence of a PPT adversary $\mathcal{A}$ corrupting the receiver R adaptively and any subset of the senders $\{S_i\}_{i\in[\ell]}$ statically. Let us denote by $b \in \{0,1\}$ the input of R and by $a_0^i, a_1^i \in \{0,1\}$ the input bits of each S_i. Let $\mathsf{Sim}_{\mathsf{multiOT}}$ be a PPT adversary that simulates the view of $\mathcal{A}$ in an ideal execution of $\mathcal{F}_{\mathsf{multiOT}}$, working as follows:

1. $\mathsf{Sim}_{\mathsf{multiOT}}$ first invokes $(\mathsf{CRS}, t) \leftarrow \mathbf{OT}.\mathsf{CRSgen}(1^\kappa)$ and, emulating $\mathcal{F}_{\mathsf{CRS}}$, it gives CRS to $\mathcal{A}$.
2. If R is initially corrupted,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ accepts m_1 from $\mathcal{A}$ and parses $m_1 = pk = (g, h)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ computes $b' \leftarrow \mathcal{E}.\mathsf{Branch}(\mathsf{CRS}, t, pk)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ passes b' to $\mathcal{F}_{\mathsf{multiOT}}$.

   Otherwise, if R is initially honest,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ computes $(pk, sk_0, sk_1) \leftarrow \mathcal{E}.\mathsf{TrapGen}(\mathsf{CRS}, t)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ sends $m_1' = pk$ to $\mathcal{A}$.
3. For each $i \in [\ell]$, if S_i is corrupted and R is initially honest,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ has (sk_0, sk_1).
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ accepts m_2^i from $\mathcal{A}$ and parses $m_2^i = (y_0^i, y_1^i)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ computes $a_0'^i = \mathcal{E}.\mathsf{Dec}(sk_0, y_0^i)$ and $a_1'^i = \mathcal{E}.\mathsf{Dec}(sk_1, y_1^i)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ passes $(a_0'^i, a_1'^i)$ to $\mathcal{F}_{\mathsf{multiOT}}$.

   If S_i is corrupted and R is initially corrupted,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ accepts m_2^i from $\mathcal{A}$ and sends $\perp$ to $\mathcal{F}_{\mathsf{multiOT}}$.

   Otherwise, if S_i is honest,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ receives the output $a^i \in \{0,1\}$ from $\mathcal{F}_{\mathsf{multiOT}}$.
   - If R is corrupted, $\mathsf{Sim}_{\mathsf{multiOT}}$ has b, otherwise it picks $b \leftarrow \{0,1\}$ at random.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ sets $a_b^i = a^i$ and $a_{\neg b}^i = 0$ and $m_2'^i \leftarrow \mathbf{OT}.\mathsf{OT}_2(\mathsf{CRS}; pk, a_0^i, a_1^i)$.
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ gives $m_2'^i$ to $\mathcal{A}$.
4. If R is adaptively corrupted after the first message,
   - $\mathsf{Sim}_{\mathsf{multiOT}}$ receives b from $\mathcal{F}_{\mathsf{multiOT}}$ and sends (b, sk_b) to $\mathcal{A}$.

Let κ be the computational security parameter, $\mathcal{R}$ denote the space of randomness of all the parties participating in the protocol and $\vec{r}$ denote the contents of the random tape of all the parties and the simulator. Let us, without loss of generality, consider the case that all $\{S_i\}_{i\in[\ell]}$ is honest and R is adaptively corrupted after

the first message is sent. This is the case for which the simulator needs to output all the messages in the transcript. The output of this simulator is distributed as,

$$\left\{b, \{a_0^i, a_1^i\}_{i\in[\ell]}, m_1', \{m_2'^i\}_{i\in[\ell]}, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

This distribution can be shown as indistinguishable from the real view using **OT** by reducing to the 'Messy Branch Indistinguishability' and 'Trapdoor Generation' property of a dual-mode cryptosystem (Definition 18). Consider the following set of hybrids:

- Hybrid H_0. This is the distribution of the output of $\mathsf{Sim}_{\mathsf{multiOT}}$. It differs from the view of the real execution in that all the messages in the transcript are simulated messages.

$$\mathsf{H}_0 = \left\{b, \{a_0^i, a_1^i\}_{i\in[\ell]}, m_1', \{m_2'^i\}_{i\in[\ell]}, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

- Hybrid H_i. For each $i \in [\ell]$, this is the distribution output by an intermediate hybrid experiment in which the messages of senders in $\{S_{i'}\}_{i'\leq i}$ are generated as in the real execution but everything else is generated as in $\mathsf{Sim}_{\mathsf{multiOT}}$.

$$\mathsf{H}_i = \left\{b, \{a_0^{i'}, a_1^{i'}\}_{i'\in[\ell]}, m_1', \{m_2^{i'}\}_{i'\leq i}, \{m_2'^{i'}\}_{i'>i\in[\ell]}, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

- Hybrid $\mathsf{H}_{\ell+1}$. This is the distribution of the view of the real execution of the protocol. It differs from the previous hybrid H_ℓ in that the simulated receiver's message is replaced by m_1 as produced in the real execution of **OT**.

$$\mathsf{H}_{\ell+1} = \left\{b, \{a_0^i, a_1^i\}_{i\in[\ell]}, m_1, \{m_2^i\}_{i\in[\ell]}, sk_b\right\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

Claim 15 *Assuming that the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Messy Branch Indistinguishability', then for each $i \in [\ell]$ views in hybrid distributions H_i and H_{i-1} are statistically close.*

The proof for this closely follows that in Claim 13.

Claim 16 *Assuming that the dual-mode cryptosystem $\mathcal{E}$ (Construction 2) satisfies 'Trapdoor Generation', the views in distributions H_ℓ and $\mathsf{H}_{\ell+1}$ are statistically close.*

The proof for this closely follows that in Claim 14.

Since none of the listed adjacent hybrids are distinguishable, it follows that the distributions of the real view of **OT** and the simulated view are also indistinguishable. Hence, this protocol $(\mathsf{CRSgen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_{\mathsf{fin}}) \in \mathbf{OT}$ is a maliciously secure protocol realizing $\mathcal{F}_{\mathsf{multiOT}}$ in the CRS model that is secure in the presence of a PPT adversary $\mathcal{A}$ corrupting the receiver R adaptively and any subset of the senders $\{S_i\}_{i\in[\ell]}$ statically.

Lastly, it remains to show that the OT second message is rerandomizable. For this, note that the function $\mathbf{OT}.\mathsf{OT}_3$ makes 2 calls to $\mathcal{E}.\mathsf{Rerand}$ and $\mathbf{OT}.\mathsf{OT}_2$ makes two calls to $\mathcal{E}.\mathsf{Enc}$ of the dual-mode cryptosystem. When this cryptosystem

Functionality $\mathcal{F}^f$ in the Covert Setting

Let $f : \{0,1\}^m \to \{0,1\}^l$ be a function. Let $\mathcal{A}$ be a PPT ideal-world adversary. Let $\mathcal{C} = \{C_i\}_{i\in[m]}$ be the clients. Let $\epsilon = \frac{1}{2}$ be the probability of cheating detection. The functionality $\mathcal{F}^f$ works as follows:

- $\mathcal{F}^f$ accepts an input bit x_i from each $C_i \in \mathcal{C}$
- If, for any corrupted C_i, $\mathcal{F}^f$ receives (cheat, i) from $\mathcal{A}$,
  - with probability ϵ, $\mathcal{F}^f$ sends $(\mathsf{corrupt}, i)$ to all parties
  - otherwise, $\mathcal{F}^f$ sends all inputs $(\mathsf{undetected}, \{x_i\}_{C_i\in\mathcal{C}})$ to $\mathcal{A}$; then it accepts an output τ from $\mathcal{A}$ and this is given to all honest parties as output.
- Otherwise, $\mathcal{F}^f$ computes $f(x)$ and gives $f(x)$ to $\mathcal{A}$
- If $\mathcal{A}$ sends OK to $\mathcal{F}^f$, it sends $f(x)$ to $\mathcal{C}$

Fig. 27 Functionality in the Covert Setting

is instantiated as in Construction 2, both Rerand and Enc samples as randomness two elements $s, t \leftarrow \mathbb{Z}_p$. Hence the randomness used in $\mathbf{OT}.\mathsf{OT}_3$ and $\mathbf{OT}.\mathsf{OT}_2$ is identically distributed. Further, let $(s,t) \diamond (s',t') = (s+s', t+t')$. Then it holds by construction that:

$$\mathsf{OT}_2(\mathbf{m_1}, (a_0,a_1); r \diamond r') = \mathsf{OT}_3(\mathsf{OT}_2(\mathbf{m_1}, (a_0,a_1); r'); r)$$

This completes the proof.

C A Covertly Secure SCALES Protocol

Consider an execution of functionality $\mathcal{F}_f$ that proceeds as in Figure 27.

Definition 19. *A SCALES protocol (Definition 1) Π for a function f is said to be a* ***covertly secure SCALES protocol*** *if it is a UC-secure protocol for the functionality $\mathcal{F}_f$ (Figure 27) in the $(\mathcal{F}_{\mathcal{B}}, \mathcal{F}_{\mathsf{CRS}})$-hybrid model against a covert PPT adversary that is allowed all-but-one adaptive corruption of clients, and, for each phase, all-but-one adaptive corruption with erasures of participating servers.*

A Covertly Secure SCALES Protocol. The malicious-secure protocol given in Figure 26 uses, for each encoder, s instances of parallel runs of the Σ-protocol inside the ephemeral prover zero-knowledge protocol (Figure 22) to realize the functionality $\mathcal{F}_{\mathsf{EPZK}}$. This helps to give privacy error that is 2^{-s}. Note that the same protocol can be weakened to provide covert security instead.

Covert security, or security in the ϵ-deterrent model, where $\epsilon = \frac{1}{2}$ can be provided with only *one* execution of the Σ-protocol inside the Ephemeral Prover Zero-Knowledge protocol per encoding server. Let us denote as $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ the ephemeral

prover zero-knowledge functionality that achieves $\mathcal{F}_{\mathsf{EPZK}}$ in the covert setting (Figure 27).

Corollary 1. *Assuming the hardness of the DDH problem, the protocol in Figure 26 with $s = 1$ is a SCALES protocol for computing the function f that is secure in the ϵ-deterrent model for $\epsilon = \frac{1}{2}$ in the $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$-hybrid.*

C.1 Proof of Corollary 1

Proof. The protocol in Figure 26 satisfies the communication pattern given in Definition 1. The protocol has 3 phases: a computation phase, a verification phase, and a decoding phase. The clients post messages on to the bulletin-board at the beginning of each phase. Each of the 3 phases is composed of multiple rounds, after the clients' first message. In each round one stateless server reads the state of the bulletin-board, performs computation, erases its state and posts one message. Finally, note that in Figure 26, while the clients retain state, they read only information whose size is proportional to their individual input size, the security parameter κ, and, for some phases, the number of servers participating in the previous phase. Their computation is also of the same order. The servers are stateless. Hence Figure 26 is a SCALES protocol.

It now remains to show that Figure 26 is a *covert-secure SCALES protocol* (Definition 19) in the $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$-hybrid. Let $\mathcal{A}$ be a PPT covert adversary. Let $C_S \subset \mathcal{C}$ be the set of clients that $\mathcal{A}$ corrupts statically and let $C_A \subset \mathcal{C} - C_S$ denote the set of clients that it adaptively corrupts. Out of each of the 3 phases in the protocol, let it be the case that $\mathcal{A}$ corrupts all-but-one server in each phase. For phase $p \in [3]$, let $\mathcal{S}_p$ denote the set of servers that participated in that phase and let S^*_p be the one honest server for that phase. In the ideal world, let Sim be the PPT adversary interacting with $\mathcal{A}$. The protocol in Figure 26 assumes the existence of an Explanable Projective RGS (Definition 9), a 2-round Explanable UOT protocol (Definition 10), and ephemeral prover zero-knowledge protocols (Figure 22) for the relations $\mathcal{R}_{\mathsf{Gb}}$ (Equation 3) and $\mathcal{R}_{\mathsf{Rerand}}$ (Equation 4), in the covert security case, realizing the functionality $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$. We denote by $\mathsf{Sim}_{\mathsf{GS}}$ a PPT simulator that takes as input the function f and $f(x)$, and outputs a simulated garbling F' and a corresponding set of active input labels $\mathcal{L}'$. For our construction, we let $\mathsf{Sim}_{\mathsf{GS}}$ operate as described in [LP09]. It holds that the output $(F', \mathcal{L}')$ is computationally indistinguishable from a real garbled circuit and its active input label set. The simulator Sim for our protocol uses $\mathsf{Sim}_{\mathsf{GS}}$ in a black-box way. Let $\mathcal{F}$ be the ideal functionality computing f (Figure 27). The actions of the PPT simulator Sim in the ideal-world interaction for $\epsilon = \frac{1}{2}$ in the $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$-hybrid are as follows:

1. **Computation Phase:**
   - For each statically corrupted client $C_i \in C_S$,
     - Sim invokes the i^{th} instance of $\mathsf{Sim}_{\mathsf{UOT}}$ in a black-box way and extracts either inputs $x'_i \in \{0, 1\}$ or $\perp$ (abort_i,$\mathsf{corrupt}_i$).
     - Sim passes these to $\mathcal{F}$.

For each client C_i not yet corrupted, Sim invokes the i^{th} instance of $\mathsf{Sim}_{\mathsf{UOT}}$ in a black-box way and emulates the transcript.

– For each adaptively corrupted $C_i \in C_A$,
  - Sim passes the index i to $\mathcal{F}$ and receives x_i.
  - Sim continues the i^{th} execution of $\mathsf{Sim}_{\mathsf{UOT}}$, producing the randomness r_i to explain the transcript thus far.
  - Sim gives (x_i, r_i) to the adversary $\mathcal{A}$.

– $\forall j \in [c_1]$, if server E_j is corrupted:
  - $\forall i \in [m]$, Sim continues the execution of the i^{th} execution of $\mathsf{Sim}_{\mathsf{UOT}}$ and extracts the server's inputs from the incoming messages:
    If $j = 0$, these inputs are $(\mathsf{L}^0_{i,0}, \mathsf{L}^1_{i,0})$
    Otherwise, these are transformations σ^j_i
  - Sim emulates $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ and gets statement $\alpha_j = F_j$ and witness $w_j = r_j$.
  - Using all these inputs, if $j = 0$ and $\mathcal{R}_{\mathsf{Gb}}$ holds, or if $j \neq 0$ and $\mathcal{R}_{\mathsf{Rerand}}$ holds, then set $b_j = 1$ and continue the execution.
  - Otherwise, if the required relation does not hold and Sim, emulating $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ receives abort_j from $\mathcal{A}$, Sim sends abort_j to $\mathcal{F}$ and set $b_j = 0$.
  - Otherwise, if the required relation does not hold and Sim, emulating $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ receives cheat_j from $\mathcal{A}$, Sim sends cheat_j to $\mathcal{F}$:
    * If $\mathcal{F}$ returns $(\mathsf{undetected}, \{x_i\}_{i\in[m]})$, set $b_j = 1$ and continue.
    * Otherwise, if $\mathcal{F}$ returns $\mathsf{corrupt}_j$, set $b_j = 0$.
  - Sim additionally posts F_j on behalf of $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$.

– $\forall j \in [c_1]$, if server E_j is honest:
  - Sim gets $f(x')$ from $\mathcal{F}$ and invokes $(F', \mathcal{L}') \leftarrow \mathsf{Sim}_{\mathsf{GS}}(1^{\kappa}, f, f(x'))$.
  - Sim sets $b_j = 1$ within its emulation of $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ and posts F'.
  - $\forall i \in [m]$, Sim continues the i^{th} instance of $\mathsf{Sim}_{\mathsf{UOT}}$ with receiver's output as $\mathsf{L}_{i,j} \in \mathcal{L}'$.

2. **Verification Phase:**
   – Sim emulates $\mathcal{F}^{\epsilon}_{\mathsf{EPZK}}$ and reveals $b = \bigcap_{j\in[c_1]} b_j$.
3. **Decoding Phase:**
   – If $b = 1$, then $\forall i \in [m]$, complete executing the i^{th} instance of $\mathsf{Sim}_{\mathsf{UOT}}$.
   – $\forall k \in [c_3]$, server D_k is corrupted,
     - Sim gets y' from $\mathcal{A}$.
     - If $y' \neq f(x')$ send abort_k to $\mathcal{F}$.
   – $\forall k \in [c_3]$, server D_k is honest, Sim posts $f(x')$ onto $\mathcal{B}$.

Let κ be a computational security parameter. Let $\mathcal{R}$ be the space of randomness of all participating parties and let $\vec{r}$ be the contents of the random tape of all the parties in the execution. For each $i \in [m]$, let $\tau_{\mathsf{Sim}^i_{\mathsf{UOT}}}$ denote the simulated view of the i^{th} instance of the UOT sub-protocol. The view produced by the above simulation is distributed as,

$$\Big\{\{x_i\}_{i\in[m]}, f(x'), F', \{F_j\}_{j\in[c_1]-S^*_1}, b, \{\tau_{\mathsf{Sim}^i_{\mathsf{UOT}}}\}_{i\in[m]},$$
$$\{\mathsf{L}_{i,j} \in \mathcal{L}'\}_{C_i\in C_A\cup C_S}, \{\mathsf{L}^{x'_i}_{i,j}\}_{C_i\in C_A\cup C_S, j\in[c_1]-S^*_1}, \{y'\}_{k\in[c_3]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

In order to show that the distribution is computationally indistinguishable from the distribution of the view of a real execution of the protocol, consider the following set of hybrids:

– Hybrid H_0. This is the distribution of the output of Sim. It differs from the view of in a real execution on two fronts. First, the garbling F' and the clients' active labels $\mathcal{L}'$ in the view corresponding to the honest encoder are output by a garbling simulator $\mathsf{Sim}_{\mathsf{GS}}$ in such a way that evaluating the garbling would always result in $f(x')$. Next, the transcripts of all the m instances of the UOT sub-protocols are generated using a simulator $\mathsf{Sim}_{\mathsf{UOT}}$.

$$\mathsf{H}_0 = \Big\{\{x_i\}_{i\in[m]}, f(x'), F', \{F_j\}_{j\in[c_1]-S_1^*}, b, \{\tau_{\mathsf{Sim}^i_{\mathsf{UOT}}}\}_{i\in[m]},$$
$$\{\mathsf{L}_{i,j} \in \mathcal{L}'\}_{C_i\in C_A\cup C_S}, \{\mathsf{L}^{x'_i}_{i,j}\}_{C_i\in C_A\cup C_S, j\in[c_1]-S_1^*}, \{y'\}_{k\in[c_3]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

– Hybrid H_i. For each $i \in [m]$, this is the distribution output by an intermediate hybrid experiment in which all the messages of the first i UOT protocol executions are generated as in the real protocol execution. The rest of the messages are generated as in the UOT simulation.

$$\mathsf{H}_i = \Big\{\{x_{i'}\}_{i'\in[m]}, f(x'), F', \{F_j\}_{j\in[c_1]-S_1^*}, b, \{\tau^{i'}_{\mathsf{UOT}}\}_{i'\le i}, \{\tau_{\mathsf{Sim}^{i'}_{\mathsf{UOT}}}\}_{i'>i\in[m]}$$
$$\{\mathsf{L}_{i',j} \in \mathcal{L}'\}_{C_{i'}\in C_A\cup C_S}, \{\mathsf{L}^{x'_{i'}}_{i',j}\}_{C_{i'}\in C_A\cup C_S, j\in[c_1]-S_1^*}, \{y'\}_{k\in[c_3]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

The last hybrid H_m is the distribution that only differs from that of the real execution in that the garbling of the honest encoder is produced by the garbling simulator.

– Hybrid H_{m+1}. This is the distribution of the view of a real execution of the protocol.

$$\mathsf{H}_{m+1} = \Big\{\{x_i\}_{i\in[m]}, f(x'), \{F_j\}_{j\in[c_1]}, b, \{\tau^i_{\mathsf{UOT}}\}_{i\in[m]},$$
$$\{\mathsf{L}^{x'_i}_{i,j}\}_{C_i\in C_A\cup C_S, j\in[c_1]}, \{y'\}_{k\in[c_3]}\Big\}_{\kappa\in\mathbb{N},\vec{r}\in\mathcal{R}}$$

Claim 17 *Assume that* UOT *is a UOT protocol that realizes the functionality* $\mathcal{F}_{\mathsf{UOT}}$ *(Definition 7) in the presence of a PPT malicious adversary corrupting all-but-one of the sender and updaters statically and the receiver adaptively. Then for each* $i \in [m]$*, the views in the hybrid distributions* H_i *and* H_{i-1} *are computationally indistinguishable.*

Proof. The hybrid distributions H_i and H_{i-1} differ only in the way that the i^{th} UOT sub-protocol transcript is produced. In H_i, this is produced as in the real execution of the functions in UOT. In H_{i-1}, this is produced as given in the output of the simulator $\mathsf{Sim}_{\mathsf{UOT}}$. Since UOT securely realizes $\mathcal{F}_{\mathsf{UOT}}$ (Definition 7) in the presence of a PPT malicious adversary corrupting all-but-one of the sender and updaters statically and the receiver adaptively, it follows that both these transcripts are computationally indistinguishable.

However, if there existed a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_i and H_{i-1} with non-negligible advantage ϵ, then $\mathcal{A}$ can be used in a black-box way as a subroutine by a PPT distinguisher D to distinguish between the output of $\mathsf{Sim}_{\mathsf{UOT}}$ and the protocol UOT. D works by first generating the view of the protocol up to the $i-1^{th}$ UOT protocol transcript as in the real

execution. It then gives the inputs x_i, $(\mathsf{L}^0_{i,0}, \mathsf{L}^0_{i,1})$ and $\{\sigma^j_i\}_{j\in[d]}$ to the challenger $\mathcal{C}$ that returns a message m. D then completes the rest of the view of the protocol as in the simulation $\mathsf{Sim}_{\mathsf{UOT}}$ and gives this to $\mathcal{A}$. Note that if $m = \tau_{\mathsf{Sim}^i_{\mathsf{UOT}}}$ as output by $\mathsf{Sim}_{\mathsf{UOT}}$ then the complete transcript belongs in the distribution H_{i-1}. Otherwise, if $m = \tau^i_{\mathsf{UOT}}$ as in the real execution of the algorithms in UOT, then the transcript belongs in the distribution H_i. Finally, D outputs whatever $\mathcal{A}$ outputs. In this experiment D's advantage is the same as that of $\mathcal{A}$, which is non-negligible. However since UOT is a secure protocol, no such D can exist and hence no such $\mathcal{A}$ can exist.

Claim 18 *Assuming that it holds that,*

$$\{F', \mathcal{L}'\}_{(F',\mathcal{L}')\leftarrow\mathsf{Sim}_{\mathsf{GS}}(1^\kappa, f, f(x'))} \overset{\mathrm{c}}{\approx} \{F, X\}_{(F,e)\leftarrow\mathsf{RGS.Gb}(f),\ X=\mathsf{RGS.En}(e,x')}$$

the hybrid distributions H_m *and* H_{m+1} *are computationally indistinguishable.*

Proof. The hybrid distributions H_m and H_{m+1} differ only in the way that the honest encoder's garbled circuit and active labels set is produced. In H_m, this is produced as $(F', \mathcal{L}') \leftarrow \mathsf{Sim}_{\mathsf{GS}}(1^\kappa, f, f(x'))$. In H_{m+1}, this is produced as given in the real protocol execution using functions in RGS. Since it holds that both these outputs are computationally indistinguishable, it follows that the transcripts in both hybrid distributions are computationally indistinguishable.

However, if there existed a PPT adversary $\mathcal{A}$ that can distinguish between hybrids H_m and H_{m+1} with non-negligible advantage ϵ, then $\mathcal{A}$ can be used in a black-box way as a subroutine by a PPT distinguisher D to distinguish between the output of $\mathsf{Sim}_{\mathsf{GS}}$ and the functions in RGS. D works by first generating the view of the protocol up to before the honest encoder's message as in the real execution. It then gives the inputs x', f, κ to the challenger $\mathcal{C}$ that returns a message m. D then completes the rest of the view of the protocol as in the real execution and gives this to $\mathcal{A}$. Note that if $m = (F', \mathcal{L}')$ as output by $\mathsf{Sim}_{\mathsf{GS}}$ then the complete transcript belongs in the distribution H_m. Otherwise, if $m = (F, X)$ as in the real execution of the algorithms in RGS, then the transcript belongs in the distribution H_{m+1}. Finally, D outputs whatever $\mathcal{A}$ outputs. In this experiment D's advantage is the same as that of $\mathcal{A}$, which is non-negligible. However since $\mathsf{Sim}_{\mathsf{GS}}$ is a secure simulation for the function in RGS, no such D can exist and hence no such $\mathcal{A}$ can exist.

Therefore, as none of the listed adjacent hybrids are distinguishable and there are polynomially many such hybrids, it follows that the distributions of the real and simulated transcripts are also computationally indistinguishable.

Lemma 10. *The ephemeral prover zero-knowledge protocol (Figure 22) when instantiated with 1 run of the Σ-protocols for each server, has soundness $\frac{1}{2}$, realizing $\mathcal{F}^\epsilon_{\mathsf{EPZK}}$ for $\epsilon = \frac{1}{2}$.*

Proof. The ephemeral prover zero-knowledge protocol (Figure 22) uses a DCOT protocol and Σ-protocols with soundness $\frac{1}{2}$ as its building blocks. Let this protocol

be run with no parallel repetitions and let $\mathcal{A}$ be a PPT adversary that corrupts all-but-one of the challengers either statically or adaptively, and all but one of the proving servers.

Note that since $\mathcal{A}$ does not corrupt all the clients, it can never know the value of b until the reveal phase of DCOT. That is, as long as one client C_i is honest, $b = \oplus_{i \in [m]} b_i$ remains a value chosen uniformly at random from $\{0, 1\}$, and is secret until the reveal phase. This corresponds to each Σ-protocol individually maintaining soundness error $\frac{1}{2}$ where either all provers open to the transcript $(a, 0, c_0)$ or they all open to $(a, 1, c_1)$.

A verifier accepts in an execution of the protocol only if for all the provers, the Σ-protocol transcript is accepting. If $\mathcal{A}$ chooses to cheat in any one of the Σ-protocol transcripts, it will get caught with probability $\epsilon = \frac{1}{2}$. If it cheats in the transcript of p different provers, it will be detected with probability $\geq \frac{1}{2}$.