# Publicly Auditable Garbled Circuit

San Ling[1], Chan Nam Ngo[2], Khai Hanh Tang[1](✉), and Huaxiong Wang[1]

[1] Nanyang Technological University
{lingsan,khaihanh.tang,hxwang}@ntu.edu.sg
[2] Privacy + Scaling Explorations
namncc@pse.dev

**Abstract.** Generic Secure Multiparty Computation (Generic MPC) recently received much attraction in the blockchain realm as it allows mutually distrustful parties to jointly compute a global function using their private inputs while keeping them private; and more so; the expression of the function can be done in a programmable manner (hence 'generic'); as opposed to the first rising star cryptographic technique Zero-Knowledge Proof (ZKP) which only allows computation on private input of a single party (via the 'commit-and-prove' approach). While ZKP, by nature, allows public verifiability, Generic MPC is not so: Generic MPC mostly focuses on Malicious Security in which the computing result is verifiable only among the computing parties. Yet, in the blockchain realm, public verifiability is important, as the consensus protocol is not just among the computing parties but also external servers. A few works were done to bridge this gap (albeit not in the blockchain realm), i.e., Public Auditable MPC. Public Audtitability is a stronger property than Public Verifiability: the first one certifies the computation done in the MPC, while the latter certifies only the relation between the outputs and the inputs. However, they are non-constant round protocols and only for Secret-Sharing-based MPC, i.e., round complexity scales linearly with the circuit multiplicative depth, while round latency is an important cost metric in the blockchain domain. We address this problem by providing a Public Auditable Garbled Circuit protocol that is maliciously secure, publicly auditable, and constant-round. Our protocol is efficient, with only minimal overhead in terms of round, communication, and public transcript size.

**Keywords:** Garbled circuit · VOLE · public verifier · MPC

## 1 Introduction

Secure Multiparty Computation (MPC) allows $n$ parties $P_1, \ldots, P_n$ to securely evaluate a joint function $f(x_1, \ldots, x_n)$ while keeping the secret input $x_i$ private to its owner $P_i$. MPC comes with two main flavors of security [20], namely *Semi-honest* and *Active Security*: **Semi-Honest** (SEM) only guarantees security if the adversary acts according to the protocol, i.e., one will not deviate from the protocol but only tries to break privacy from the MPC transcript that is available to it; while **Active Security** (ACT) is a stronger security setting than SEM that guarantees the security of the MPC protocol even if the adversary acts arbitrarily.

Generic MPC is enabled through two approaches: Secret-Sharing (SS) and Garbled-Circuit [20] (GC). We focus on the latter due to its attractive property of constant round complexity.[3]

**Efficiency of Actively Secure Garbled Circuit.** ACT for GC is trivially available through the usage of Zero-Knowledge-Proof [25], yet even with optimizations, it is necessary to prove a statement with 19 clauses conjunction per gate. Cut-and-Choose [20] is another approach for ACT in which the Garbler prepares $k$ circuits in total, and the Evaluator can choose $t$ to test and only accepts if all $t$ circuits are valid. Yet the preparation of many circuits and the testing for validity yield concrete inefficiency and become unreasonable when the circuit size is large. Another approach is through Authenticated Garbling [31], which will be our focus due to its efficiency compared with the previous two.

Yet, these classical flavors of security only take into consideration the parties that participate in the MPC protocol itself, i.e., only $P_1, \ldots, P_n$ can verify the security of the MPC protocol.

**Verifiable MPC.** We consider a somewhat "added-on" setting, in which an external party, let us call it Verifier, can verify the security of the MPC protocol, and we call protocols in this setting Verifiable MPC (V-MPC). The security verification can come in 4 flavors:

**Designated Verifier** (DV-MPC): A single Verifier who potentially has some private state can verify the protocol execution to be secure.

[3] Round complexity is an important factor for blockchain model due to bottleneck in block generation time.

**Collaborative Verifiable** (CV-MPC) A set of Verifiers who potentially have some private states can jointly (according to some access structure) verify the security of the protocol.[4]
**Multi Verifier** (MV-MPC) A set of Verifiers who potentially have some private states can individually verify the protocol's security.[5]
**Publicly Verifiable** (PV-MPC) Any external Verifier without any secret state can verify the security of the protocol (possibly with aid from some Common Random String CRS).

We focus on the last setting, which is also the strongest one: Publicly Verifiable MPC. Its motivation is to run an MPC protocol among $n$ parties, and the output will be acknowledged by the vast external verifiers of a blockchain consensus network.
**Verifiability vs Auditability.** Disregard the verifiability flavor, to make an MPC verifiable, there are two approaches: **Verifiable Output** (VO) where the output of the MPC is made verifiable, while **Auditable Transcript** (AT) where the transcript of the MPC is made verifiable. We distinguish that the VO flavor is independent while the VT one depends on the MPC protocol.

**Example.**[*Verifiability vs Auditability:*] Let us consider the Shuffle functionality.

The Shuffle functionality runs between $n$ parties $P_1, \ldots, P_n$ each with a secret input $x_1, \ldots, x_n$ starting as an ordered list of $O_0 = \{x_1, \ldots, x_n\}$. The functionality runs in $n$ rounds, in each round $R_i$, party $P_i$ sends a permutation $\pi_i$ to Shuffle in which the functionality will apply to obtain $O_i = \pi_i(O_{i-1})$. Finally, the functionality outputs $O_n$.
**Verifiability** VO only asks for the quality of the output, i.e., that $O_n$ is a permutation of $O_0$. In contrast, **Auditability** VT asks for the correctness of the process, i.e., that $O_n$ is a permutation of $O_0$ through a series of permutation $\pi_1, \ldots, \pi_n$.

**Verifiable MPC.** For VO-MPC, we can leverage Collaborative Zero-Knowledge Proof (co-ZKP, between $P_1, \ldots, P_n$) to produce a ZK proof that the output $y = f(x_1, \ldots, x_n)$ while keeping the $x_1, \ldots, x_n$ private to $P_1, \ldots, P_n$. In the literature, co-ZKP can be achieved by thresholdizing any existing ZKP, demonstrated by Collaborative Zero-Knowledge Succinct Argument of Knowledge (co-zkSNARKs) (c.f. [28]). Yet, there are potential Co-ZKP based on other ZKP such as: MPC-In-The-Head (MPCitH: seminal work IKOS07 [23], Other MPCitH techniques [15,26,10,30,21,14,1,2,16]), VOLE-IZK (DVZK with fast prover time and small memory: [32,19,9,35,33,5,18,34,6]), or VOLE-In-The-Head [7].[6] DV-MPC can leverage any private coin Verifier (Co-) ZKP, CV-MPC is achievable through thresholdizing the Verifier state in the private coin Verifier (Co-) ZKP, MV-MPC can be obtained by running multiple DV-MPC instances, and PV-MP can utilize any public coin verifier ZKP. Fortunately, all existing Co-ZKP is constant rounds protocols. Thus, the round complexity of the VO-MPC only depends on the underlying MPC protocol, i.e., a Secret-Sharing-based [20] (SS) VO-MPC will yield linear round complexity while a Garbled-Circuit-based [20] (GC) VO-MPC will have constant-round complexity.
**Auditable Transcript MPC.** For AT-MPC, we need the transcript auditable while keeping the privacy of the protocol execution. In general, for both SS-based and GC-based AT-MPC, the transcript can be made auditable through Additively-Homomorphic Commitment (COM, such as Pedersen, [8]) or Verifiable MAC (V-MAC, [31]) on each computation step. If the COM or V-MAC in the AT-MPC is publicly verifiable [7], then we have PV-MPC. COM-based has worse concrete efficiency than V-MAC-based AT-MPC due to the first operating on groups while the latter only operates on rings. One can consider ACT a special case of DV/MV AT-MPC where the Verifier is a party in the MPC protocol. Hence, if the ACT mechanism can be publicly verifiable, we will also obtain PV-MPC. As in VO-MPC, an SS-based AT-MPC will yield linear round complexity, while a GC-based VT-MPC can potentially yield a constant round complexity. This last one is also our main research question:

**Research Question.** *Can we pick a GC-based protocol that is actively secure (such as WRK17 [26], KRRW18 [27], DILO22 [18], or CWYY23 [17]) and make such active security public verifiable and obtain constant round publicly auditable MPC?*

We answer with definitive. We picked the basic blueprint of the actively secure Garbled Circuit protocol, namely, WRK17 [26], which relies on V-MAC. We then adopted VOLEitH [7] (with SoftSpokenOT [29])

[4] CV-MPC can be considered as a thresholdized version of DV-MPC.
[5] Such verification flavor is also called Crowd-Verifiable MPC [4].
[6] VOLE-In-The-Head is the public coin Verifier version of VOLE-IZK

to make the utilized V-MAC public auditable. We obtained a Constant Round Publicly Auditable Garbled Circuit Protocol with inherited security and efficiency from both WRK17 and VOLEitH.[7]

We implement and report the performance of our protocol at

`https://github.com/Crazy-Cryptographic-Buddies/pa-gc-rs`.

### 1.1 Structure of the Paper

Section 2 is that of the necessary preliminaries. In Section 3, we present a technical overview for constructing a public auditable garbled circuit. In Section 4, we present the functionality $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ and its secure realization, namely, protocol $\mathit{\Pi}_{\mathsf{sVOLE\text{-}2PC}}$, as the intermediate components supporting protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC}}$ enabling publicly auditable garbled circuits. Based on the technical overview, in Section 5, we present the construction of a public auditable garbled circuit by presenting the functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$ and its secure realization, namely, protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC}}$.

## 2 Preliminaries

**Notations.** For a value $k \in [0, 3]$, we denote by $(k_0, k_1) = \mathsf{bin}_2(k)$ to indicate that $(k_0, k_1) \in \{0, 1\}^2$ satisfying $k = k_0 + 2 \cdot k_1$.

$\mathbb{Z}$ is the ring of integers. We denote by $[a, b]$ to indicate the set $\{a, a+1, \ldots, b\}$ for any $a, b \in \mathbb{Z}$ satisfying $a \leq b$. If $a = 1$ and $a \leq b$, we simply denote by $[b]$ to indicate the set $\{1, \ldots, b\}$.

**Convention for VOLE correlations.** We use both $\mathit{\Delta}$ and $\nabla$ for VOLE correltions (deriving from WRK [31]) and for public verifiability via VOLEitH. We call those w.r.t. $\nabla$ the *VOLEitH correlations*.

### 2.1 Authenticated Bits via VOLE Correlations

Let $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ be two communicating parties. Assume that $\mathcal{P}_{\mathsf{A}}$ holds a secret bit $a \in \mathbb{F}_2$ while $\mathcal{P}_{\mathsf{B}}$ holds a global value $\mathit{\Delta}_{\mathsf{B}}$. Then, a VOLE correlation for authenticating $a$ w.r.t. $\mathit{\Delta}_{\mathsf{B}}$ is a correlation where $\mathcal{P}_{\mathsf{A}}$ additionally holds $\mathsf{M}[a]$ while $\mathcal{P}_{\mathsf{B}}$ additionally holds $\mathsf{K}[a]$ satisfying $\mathsf{M}[a] = \mathsf{K}[a] \oplus a \cdot \mathit{\Delta}_{\mathsf{B}}$.

Symmetrically, if $b$ is a secret value of $\mathcal{P}_{\mathsf{B}}$ while $\mathit{\Delta}_{\mathsf{A}}$ is a global value for $\mathcal{P}_{\mathsf{A}}$, then a VOLE correlation for authenticating $b$ w.r.t. $\mathit{\Delta}_{\mathsf{A}}$ is a correlation where $\mathcal{P}_{\mathsf{B}}$ additionally holds $\mathsf{M}[b]$ while $\mathcal{P}_{\mathsf{B}}$ additionally holds $\mathsf{K}[a]$ such that $\mathsf{M}[b] = \mathsf{K}[b] \oplus b \cdot \mathit{\Delta}_{\mathsf{A}}$.

### 2.2 VOLEitH from SoftSpokenOT

SoftSpokenOT [29] allows two parties to generate an authenticated bit, e.g., $[b]_{\mathsf{A}}$, in a way that each party only locally interacts with the random oracle (RO). This hence leads to publicly verifiable zero-knowledge proofs [7,16,12] which recently attracted a lot of attention from the research community. We recall the functionality $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ and the technique from SoftSpokenOT for realizing $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ for binary computations, over $\mathbb{F}_{2^k}$ for some $k \in \mathbb{N}$.

**Functionality $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$.** We recall the subspace VOLE functionality $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ [7], adapted from [16], in the following Figure 1.

**VOLEitH from SoftSpokenOT.** According to [29], an all-but-one OT correlation is equivalent to a subfield VOLE correlation over $\mathbb{F}_{2^\tau}$ for some $\tau \in \mathbb{N}$. Specifically, let $t_x \in \{0, 1\}$ for all $x \in \mathbb{F}_{2^\tau}$. Based on such an equivalence, SoftSpokenOT works by the observation that

$$\begin{aligned}\bigoplus_{x \in \mathbb{F}_{2^\tau} \setminus \{\nabla\}} t_x \cdot (\nabla \oplus x) &= \bigoplus_{x \in \mathbb{F}_{2^\tau}} t_x \cdot (\nabla \oplus x) \\ &= \bigoplus_{x \in \mathbb{F}_{2^\tau}} t_x \cdot x \oplus \nabla \cdot \bigoplus_{x \in \mathbb{F}_{2^\tau}} t_x.\end{aligned} \tag{1}$$

By the above observation, we can set $b = \bigoplus_{x \in \mathbb{F}_{2^\tau}} t_x$, $\mathsf{M}'[b] = \bigoplus_{x \in \mathbb{F}_{2^\tau}} t_x \cdot x$ and $\mathsf{K}'[b] = \bigoplus_{x \in \mathbb{F}_{2^\tau} \setminus \{\nabla\}} t_x \cdot (\nabla \oplus x)$. Hence, $\mathsf{M}'[b] = \mathsf{K}'[b] \oplus b \cdot \nabla$. We have the following discussions. We see that computing $\mathsf{K}'[b]$ does not involve $\nabla$ according to (1).

[7] We disregard the improvements in KRRW18 [27], DILO22 [18], or CWYY23 [17], due to non-trivial composition of such improvements and VOLEitH, for example they may require Learning-Parity-With-Noise assumption while SoftSpokenOT is incompatible with LPN. We leave such composition to future work.

This functionality is parameterized by $\mathbb{F}_2$ and its extension $\mathbb{F}_{2^\tau}$ for some parameter $\tau \in \mathbb{N}$. We denote by $\kappa$ the repetition parameter and $N$ the number of random sVOLE correlations.

1. Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$, work as follows.
   (a) $\nabla \xleftarrow{\$} \mathbb{F}_{2^\tau}$, $\mathbf{u} \xleftarrow{\$} \mathbb{F}_2^N$, $\mathbf{v} \xleftarrow{\$} \mathbb{F}_{2^\tau}^N$, and $\mathbf{w} := \mathbf{v} \oplus \mathbf{u} \cdot \nabla$.
   (b) If $\mathcal{P}$ is corrupted, receive $\mathbf{u}, \mathbf{v}$ from $\mathcal{P}$; and $\mathbf{w} := \mathbf{v} \oplus \mathbf{u} \cdot \nabla$.
   (c) If $\mathcal{V}$ is corrupted, receive $\nabla, \mathbf{w}$ from $\mathcal{V}$ and compute $\mathbf{v} := \mathbf{w} \oplus \mathbf{u} \cdot \nabla$.
   (d) Send $(\mathbf{u}, \mathbf{v})$ to $\mathcal{P}$.
2. Upon receiving (get) from $\mathcal{P}$ and $\mathcal{V}$, it sends $(\nabla, \mathbf{w})$ to both parties.

**Fig. 1.** The subspace VOLE functionality $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$.

Hence, we can run VOLEitH (from SoftSpokenOT) to generate an authenticated random string $\mathbf{u} = (u_i)_{i\in[N]} \in \{0,1\}^N$ for some $N \in \mathbb{N}$ such that it generates $\mathbf{v} = (\mathsf{M}'[u_i])_{i\in[N]} \in \mathbb{F}_{2^\tau}^N$ and $\mathbf{w} = (\mathsf{K}'[u_i])_{i\in[N]} \in \mathbb{F}_{2^\tau}^N$ satisfying $\mathbf{v} = \mathbf{w} \oplus \mathbf{u} \cdot \nabla$ as expected in functionality $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$. This is placed in Figure 2 as protocol $\Pi_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ that securely realizes $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$.

This functionality runs between $\mathcal{P}$ and $\mathcal{V}$ with $\tau, N \in \mathbb{N}$:

1. $\mathcal{P}$ employs a pseudo-random number PRG to generate $\mathbf{t}_x \in \{0,1\}^N$ $\forall x \in \mathbb{F}_{2^\tau}$. Here, $\mathcal{P}$ can compute $\mathbf{u} = \bigoplus_{x\in\mathbb{F}_{2^\tau}} \mathbf{t}_x$ and $\mathbf{v} = (\mathsf{M}'[u_i])_{i\in[N]} = \bigoplus_{x\in\mathbb{F}_{2^\tau}} \mathbf{t}_x \cdot x$. Then, $\mathcal{P}$ uses a vector commitment scheme (realizing $\mathcal{F}_{\mathsf{VC}}^{\tau,N}$ in Figure 3) to commit to all $\mathbf{t}_x$ $\forall x \in \mathbb{F}_{2^\tau}$ and sends the commitment to $\mathcal{V}$.
2. $\mathcal{V}$ samples and sends $\nabla \xleftarrow{\$} \mathbb{F}_{2^\tau}$ to $\mathcal{P}$.
3. $\mathcal{P}$ then opens to $\mathcal{V}$ all $\mathbf{t}_x$ for $x \in \mathbb{F}_{2^\tau} \setminus \{\nabla\}$.
4. $\mathcal{P}$, $\mathcal{V}$ computes $\mathbf{w} = (\mathsf{K}'[u_i])_{i\in[N]} = \bigoplus_{x\in\mathbb{F}_{2^\tau}\setminus\{\nabla\}} \mathbf{t}_x \cdot (\nabla \oplus x)$.
5. $\mathcal{V}$ accepts if the openings by $\mathcal{P}$ are correct and $(\mathsf{M}'[u_i])_{i\in[N]} = (\mathsf{K}'[u_i])_{i\in[N]} \oplus \mathbf{u} \cdot \nabla$.

**Fig. 2.** Protocol $\Pi_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ for VOLEitH.

This functionality runs between $\mathcal{P}$ and $\mathcal{V}$ with $\tau, N \in \mathbb{N}$:

1. Upon receiving (commit) from $\mathcal{P}$ and $\mathcal{V}$, sample $\mathbf{t}_x \xleftarrow{\$} \{0,1\}^N$ $\forall x \in \mathbb{F}_{2^\tau}$. If $\mathcal{P}$ is corrupted, receive $\mathbf{t}_x$ $\forall x \in \mathbb{F}_{2^\tau}$ from $\mathcal{P}$. Then, send (done) to both $\mathcal{P}$ and $\mathcal{V}$.
2. Upon receiving (get, $\nabla$) from $\mathcal{V}$, send $(\mathbf{t}_x)_{x\in\mathbb{F}_{2^\tau}\setminus\{\nabla\}}$ to both $\mathcal{P}$, $\mathcal{V}$.

**Fig. 3.** Functionality $\mathcal{F}_{\mathsf{VC}}^{\tau,N}$.

**Amplifying Soundness.** As in [16], to amplify soundness, one may need to repeat to repeat the protocol $\kappa$ times for some $\kappa \in \mathbb{N}$.
**Instantiations of VOLEitH.** As in [7], to instantiate VOLEitH, we can realize $\mathcal{F}_{\mathsf{VC}}^{\tau,N}$ by a GGM tree (previously used in [26,11,13]). Here, we skip the details of such realization and refer readers to [7]. Regarding the cost of vector commitments and their all-but-one openings, we denote by $\mathsf{costvc}_{\mathsf{com}}^{\tau,N}$ to indicate the size of the vector commitment to a vector of length $2^\tau$ bits and by $\mathsf{costvc}_{\mathsf{open}}^{\tau,N}$ to indicate the size of its all-but-one opening. We also denote by $\mathsf{costvc}_{\mathsf{total}}^{\tau,N}$ to indicate the total cost $\mathsf{costvc}_{\mathsf{com}}^{\tau,N} + \mathsf{costvc}_{\mathsf{open}}^{\tau,N}$. Hence,when repeating $\kappa$ times, the total cost is $\kappa \cdot (\mathsf{costvc}_{\mathsf{com}}^{\tau,N} + \mathsf{costvc}_{\mathsf{open}}^{\tau,N} + \tau) = \kappa \cdot (\mathsf{costvc}_{\mathsf{total}}^{\tau,N} + \tau)$ bits of communication where the extra $\kappa \cdot \tau$ bits is for global challenges.
**Removing Interactions.** As VOLEitH is public-coin, i.e., $\mathcal{V}$ has no secrets and $\nabla$ is sampled and made public by $\mathcal{V}$. Hence, VOLEitH can be made non-interactive by using Fiat-Shamir transformation [22] to

generate $\nabla$ from the RO. When repeating $\kappa$ times, the RO generates $\kappa$ global values (say, e.g., $(\nabla_j)_{j\in[\kappa]}$) instead.

### 2.3 Sub-Protocol $\Pi_{\mathsf{fix}}^{\mathcal{P},\mathcal{V}}$

Let $\mathcal{P}$ and $\mathcal{V}$ be two parties who maintain a VOLEitH correlation for a random bit $a$, i.e., $\mathcal{P}$ knows $a$ and $\mathsf{M}'[a]$ while both parties can only know $\mathsf{K}'[a]$ and $\nabla$ after invoking $\mathcal{F}_{\mathsf{sVOLE}}^{\mathcal{P},\mathcal{V},\tau,N}$ (c.f. Figure 1) by sending $(\mathsf{get})$ to it.

Assuming $\mathcal{P}$ has an additional secret bit $b \in \mathbb{F}_2$, we recall the protocol $\Pi_{\mathsf{fix}}^{\mathcal{P},\mathcal{V}}$ [9] in the following Figure 4 for obtaining a VOLEitH correlation for $b$ given the VOLEitH correlation as said above. The idea for doing so is according to the following implication.

$$\mathsf{M}'[a] = \mathsf{K}'[a] \oplus b\cdot\nabla \implies \underbrace{\mathsf{M}'[a]}_{\mathsf{M}'[b]} = \underbrace{(\mathsf{K}'[a] \oplus (a\oplus b)\cdot\nabla)}_{\mathsf{K}'[b]} \oplus b\cdot\nabla.$$

After running $\Pi_{\mathsf{fix}}^{\mathcal{P},\mathcal{V}}$, $\mathcal{P}$ holds $\mathsf{M}'[b]$ (unknown to $\mathcal{V}$) while both parties know $\mathsf{K}'[b]$ and $\nabla$ such that $\mathsf{M}'[b] = \mathsf{K}'[b] \oplus b\cdot\nabla$.

**Inputs:** $\mathcal{P}$ keeps $a$ and its corresponding IT-MAC tag for the VOLEitH correlation. $\mathcal{P}$ additionally keeps $b$.
**Output:** $\mathcal{P}$ and $\mathcal{V}$ hold a VOLEitH correlation for $b$.

1. *When* $\nabla$ *and* $\mathsf{K}'[a]$ *are unknown,* $\mathcal{P}$ sends $d := a \oplus b$ to $\mathcal{V}$. Then, $\mathcal{P}$ sets $\mathsf{M}'[b] := \mathsf{M}'[a]$.
2. *When* $\nabla$ *and* $\mathsf{K}'[a]$ *are known,* Both $\mathcal{P}$ and $\mathcal{V}$ locally set $\mathsf{K}'[b] := \mathsf{K}'[a] \oplus d\cdot\nabla$.

**Fig. 4.** Sub-Protocol $\Pi_{\mathsf{fix}}^{\mathcal{P},\mathcal{V}}$.

**Communication Cost of $\Pi_{\mathsf{fix}}^{\mathcal{P},\mathcal{V}}$.** The cost is only 1 bit since the only thing published is bit $d \in \{0,1\}$.

## 3 Technical Overview

We first recall the authenticated garbling WRK protocol [31] in Section 3.1. Then, we discuss our technique for making WRK protocol publicly auditable in Section 3.2.

### 3.1 Authenticated Garbling from WRK Protocol

We recall the authenticated garbling technique from WRK protocol [31] (recalled in [27]) without clearly discussing it in detail. Consider a circuit containing AND $(\cdot)$ and XOR $(\oplus)$ gates. We denote each gate by the tuple $(\alpha,\beta,\gamma,\mathsf{op})$ where $\mathsf{op} \in \{\cdot,\oplus\}$ is the operation, and $\alpha$, $\beta$, and $\gamma$ are the indices of the left, right, and output wires, respectively. As remarked in [31,27], XOR gates can be handled for free. Regarding AND gates, consider the tuple $(\alpha,\beta,\gamma,\cdot)$. Let $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ respectively play the roles of garbler and authenticator. We first recall a few notations before discussing the garbled circuits.

The value at each each wire $w$ is denoted by $z_w \in \{0,1\}$. To mask $z_w$, we use a mask $\lambda_w \in \{0,1\}$ and compute the masked value $\hat{z}_w = z_w \oplus \lambda_w$. For each $w$, $\lambda_w$ is split into shared random masks $r_w$ and $s_w$ held by $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, respectively, such that $\lambda_w = r_w \oplus s_w$.

The input wires are split into two separate sets $\mathcal{I}_{\mathsf{A}}$ and $\mathcal{I}_{\mathsf{B}}$ such that $\mathcal{P}_{\mathsf{A}}$ (resp., $\mathcal{P}_{\mathsf{B}}$) knows each $z_w$ for $w \in \mathcal{I}_{\mathsf{A}}$ (resp., $w \in \mathcal{I}_{\mathsf{B}}$). Therefore, in the beginning (at the phase **Input Preprocessing**), $\mathcal{P}_{\mathsf{B}}$ (resp., $\mathcal{P}_{\mathsf{A}}$) must send $s_w$ (resp., $r_w$) to $\mathcal{P}_{\mathsf{A}}$ (resp., $\mathcal{P}_{\mathsf{B}}$) for computing $\hat{z}_w$ for $w \in \mathcal{I}_{\mathsf{A}}$ (resp., $w \in \mathcal{I}_{\mathsf{B}}$). $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ then broadcast all $\hat{z}_w$ for $w \in \mathcal{I}_{\mathsf{A}} \cup \mathcal{I}_{\mathsf{B}}$. Then, the evaluation of the circuit, following a topological ordering, considers two types of gates:

– For each XOR gate $(\alpha,\beta,\gamma,\oplus)$, the computation is free. Having $\hat{z}_\alpha$ and $\hat{z}_\beta$, each party ($\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$) can locally compute $\hat{z}_\gamma := \hat{z}_\alpha \oplus \hat{z}_\beta = (z_\alpha \oplus z_\beta) \oplus (\lambda_\alpha \oplus \lambda_\beta)$. This is hence understood that $z_\gamma = z_\alpha \oplus z_\beta$ and $\lambda_\gamma = \lambda_\alpha \oplus \lambda_\beta$.

– For each AND gate $(\alpha, \beta, \gamma, \cdot)$, the computation is not that simple. In fact, having $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$ and $\hat{z}_\beta = z_\beta \oplus \lambda_\beta$, the parties need to obtain $\hat{z}_\gamma = z_\alpha \cdot z_\beta \oplus \lambda_\gamma$ where $\lambda_\gamma = r_\gamma \oplus s_\gamma$ with $r_\gamma$ and $s_\gamma$ independently held by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively. To do this, we first observe that

$$\begin{aligned}\hat{z}_\alpha \cdot \hat{z}_\beta &= (z_\alpha \oplus \lambda_\alpha) \cdot (z_\beta \oplus \lambda_\beta)\\ &= z_\alpha \cdot z_\beta \oplus \lambda_\alpha \cdot z_\beta \oplus \lambda_\beta \cdot z_\alpha \oplus \lambda_\alpha \cdot \lambda_\beta\\ &= z_\alpha \cdot z_\beta \oplus \lambda_\alpha \cdot \underbrace{(z_\beta \oplus \lambda_\beta)}_{\hat{z}_\beta} \oplus \lambda_\beta \cdot \underbrace{(z_\alpha \oplus \lambda_\alpha)}_{\hat{z}_\alpha} \oplus \lambda_\alpha \cdot \lambda_\beta.\end{aligned}$$

Hence, $\lambda_\alpha \cdot \lambda_\beta \oplus \lambda_\gamma \oplus \lambda_\alpha \cdot \hat{z}_\beta \oplus \lambda_\beta \cdot \hat{z}_\alpha \oplus \hat{z}_\alpha \cdot \hat{z}_\beta = z_\alpha \cdot z_\beta \oplus \lambda_\gamma$. Notice that, $\lambda_\alpha \cdot \hat{z}_\beta = r_\alpha \cdot \hat{z}_\beta \oplus s_\alpha \cdot \hat{z}_\beta$, $\lambda_\beta \cdot \hat{z}_\alpha = r_\beta \cdot \hat{z}_\alpha \oplus s_\beta \cdot \hat{z}_\alpha$, and $\lambda_\gamma = r_\gamma \oplus s_\gamma$ such that $\mathcal{P}_\mathsf{A}$ can compute $r_\alpha \cdot \hat{z}_\beta$ and $r_\beta \cdot \hat{z}_\alpha$ while $\mathcal{P}_\mathsf{B}$ can compute $s_\alpha \cdot \hat{z}_\beta$ and $s_\beta \cdot \hat{z}_\alpha$. However, regarding $\lambda_\alpha \cdot \lambda_\beta$, [31] suggests to use additional shared random masks $r'_\gamma$ and $s'_\gamma$ respectively held by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ such that $r'_\gamma \oplus s'_\gamma = \lambda_\alpha \cdot \lambda_\beta$. Then, $\mathcal{P}_\mathsf{A}$ can construct the garbled table corresponding to the four possibilities of $(\hat{z}_\alpha, \hat{z}_\beta)$, i.e., by encrypting to the following four rows where the $k$-th row is

$$r_{\gamma,k} = r'_\gamma \oplus r_\gamma \oplus k_0 \cdot r_\beta \oplus k_1 \cdot r_\alpha \tag{2}$$

for $k \in [0, 3]$ with $(k_0, k_1) = \mathsf{bin}_2(k)$. Here, the $k$-th row assumes that $(\hat{z}_\alpha, \hat{z}_\beta) = (k_0, k_1) = \mathsf{bin}_2(k)$. Regarding $\mathcal{P}_\mathsf{B}$, if $k \in [0, 3]$, i.e., $k = \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$, is the selected row, then, with $(k_0, k_1) = \mathsf{bin}_2(k)$, $\mathcal{P}_\mathsf{B}$ can compute

$$s_{\gamma,k} = s'_\gamma \oplus s_\gamma \oplus k_0 \cdot s_\beta \oplus k_1 \cdot s_\alpha \oplus k_0 \cdot k_1 \tag{3}$$

and decrypt the corresponding row to obtain $r'_\gamma \oplus r_\gamma \oplus k_0 \cdot r_\beta \oplus k_1 \cdot r_\alpha$. Taking sum achieves $\hat{z}_\gamma = z_\alpha \cdot z_\beta \oplus \lambda_\gamma$ as desired.

We also note that encrypting requires the associated labels $\mathsf{L}_{w,0}$ and $\mathsf{L}_{w,1}$ corresponding to the assumptions $\hat{z}_w = 0$ and $\hat{z}_w = 1$. To be compatible with free XORs (i.e., $\hat{z}_\gamma = \hat{z}_\alpha \oplus \hat{z}_\beta$ for any XOR gate $(\alpha, \beta, \gamma, \oplus)$), [31] enforces $\mathsf{L}_{w,0} \oplus \varDelta_\mathsf{A} = \mathsf{L}_{w,1}$ where $\varDelta_\mathsf{A}$ is only known by $\mathcal{P}_\mathsf{A}$. This means that, for each wire $w$, $\mathcal{P}_\mathsf{B}$ can achieve either $\mathsf{L}_{w,0}$ or $\mathsf{L}_{w,1}$ and cannot achieve both. Hence, for each XOR gate $(\alpha, \beta, \gamma, \oplus)$, by enforcing $\mathsf{L}_{\gamma,0} = \mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0}$, we can see that

$$\begin{aligned}\mathsf{L}_{\gamma,\hat{z}_\gamma} &= \mathsf{L}_{\gamma,0} \oplus \hat{z}_\gamma \cdot \varDelta_\mathsf{A} = \mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0} \oplus (\hat{z}_\alpha \oplus \hat{z}_\beta) \cdot \varDelta_\mathsf{A}\\ &= (\mathsf{L}_{\alpha,0} \oplus \hat{z}_\alpha \cdot \varDelta_\mathsf{A}) \oplus (\mathsf{L}_{\beta,0} \oplus \hat{z}_\beta \cdot \varDelta_\mathsf{A}) = \mathsf{L}_{\alpha,\hat{z}_\alpha} \oplus \mathsf{L}_{\beta,\hat{z}_\beta}.\end{aligned}$$

However, for each AND gate $(\alpha, \beta, \gamma, \cdot)$, we cannot have $\mathsf{L}_{\gamma,0}$ related to $\mathsf{L}_{\alpha,0}$ and $\mathsf{L}_{\alpha,1}$. Hence, $\mathsf{L}_{\gamma,0}$ is hence independently sampled by $\mathcal{P}_\mathsf{A}$. We also would like to maintain that $\mathsf{L}_{\gamma,\hat{z}_\gamma} = \mathsf{L}_{\gamma,0} \oplus \hat{z}_\gamma \cdot \varDelta_\mathsf{A}$. At this point, it seems impossible to $\mathcal{P}_\mathsf{B}$ to obtain $\mathsf{L}_{\gamma,\hat{z}_\gamma}$ without the knowledge of $\varDelta_\mathsf{A}$. We will discuss how to deal with this after discussing authenticating shared random masks below.

**Authenticating Shared Random Masks.** As noted by [31], this design's privacy is guaranteed against malicious $\mathcal{P}_\mathsf{A}$. Nevertheless, $\mathcal{P}_\mathsf{A}$ can violate the correctness of the garbled table by changing the orders of rows in garbled tables or encrypting incorrect rows. To cope with this issue, [31] enforces every shared random mask from each party to be authenticated by using subfield VOLE. In particular, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ keep $\varDelta_\mathsf{A}$ and $\varDelta_\mathsf{B}$, respectively, such that, for each $r_w$ and $s_w$ held by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively, it should hold that

$$\mathsf{M}[r_w] = \mathsf{K}[r_w] \oplus r_w \cdot \varDelta_\mathsf{B} \text{ and } \mathsf{M}[s_w] = \mathsf{K}[s_w] \oplus s_w \cdot \varDelta_\mathsf{A}$$

where $(r_w, \mathsf{M}[r_w], \mathsf{K}[s_w])$ is held by $\mathcal{P}_\mathsf{A}$ while $(s_w, \mathsf{M}[s_w], \mathsf{K}[r_w])$ is held by $\mathcal{P}_\mathsf{B}$. By applying protocol $\varPi_\mathsf{fix}$ in Figure 4, for each garbled table corresponding to each AND gate $(\alpha, \beta, \gamma, \cdot)$, for $k \in [0, 3]$, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ can respectively achieve

$$(r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], \mathsf{K}[s_{\gamma,k}]) \text{ and } (s_{\gamma,k}, \mathsf{M}[s_{\gamma,k}], \mathsf{K}[r_{\gamma,k}]).$$

With the above employment of subfield VOLE to authenticate shared random masks, we are guaranteed that a malicious $\mathcal{P}_\mathsf{A}$ cannot take advantage in causing the protocol to deviate from its proper purpose. We now turn back to discussing how to make $\mathcal{P}_\mathsf{B}$ obtain $\mathsf{L}_{\gamma,\hat{z}_\gamma} = \mathsf{L}_{\gamma,0} \oplus \hat{z}_\gamma \cdot \varDelta_\mathsf{A}$ by leveraging the authenticated shared random masks. Hence, in each row of the garbled table for each AND gate $(\alpha, \beta, \gamma, \cdot)$, $\mathcal{P}_\mathsf{A}$ additionally encrypts $\mathsf{M}[r_{\gamma,k}]$ for $\mathcal{P}_\mathsf{B}$ to subsequently verify the authenticity of the decrypted $r_{\gamma,k}$.

**Computing Labels from Authenticated Shared Random Masks.** Since $\mathsf{L}_{\gamma,\hat{z}_\gamma} = \mathsf{L}_{\gamma,0} \oplus \hat{z}_\gamma \cdot \Delta_\mathsf{A}$, $\mathcal{P}_\mathsf{A}$ holds $r_{\gamma,k}$ and $\mathsf{K}[s_{\gamma,k}]$ while $\mathcal{P}_\mathsf{B}$ holds $\mathsf{M}[s_{\gamma,k}]$, $\mathcal{P}_\mathsf{A}$ can compute $\mathsf{L}_{\gamma,0} \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A} \oplus \mathsf{K}[s_{\gamma,k}]$. Since $\mathsf{M}[s_{\gamma,k}] = \mathsf{K}[s_{\gamma,k}] \oplus s_{\gamma,k} \cdot \Delta_\mathsf{A}$, we have

$$\begin{aligned}&\underbrace{\mathsf{L}_{\gamma,0} \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A} \oplus \mathsf{K}[s_{\gamma,k}]}_{\text{computed by } \mathcal{P}_\mathsf{A} \text{ above}} \oplus \mathsf{M}[s_{\gamma,k}] \\ &= \mathsf{L}_{\gamma,0} \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A} \oplus s_{\gamma,k} \cdot \Delta_\mathsf{A} = \mathsf{L}_{\gamma,0} \oplus \hat{z}_{\gamma,k} \cdot \Delta_\mathsf{A}.\end{aligned} \tag{4}$$

Hence, in each row of the garbled table for each AND gate $(\alpha, \beta, \gamma, \cdot)$, $\mathcal{P}_\mathsf{A}$ additionally encrypts $\mathsf{L}_{\gamma,0} \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A} \oplus \mathsf{K}[s_{\gamma,k}]$ for $\mathcal{P}_\mathsf{B}$ to subsequently compute $\mathsf{L}_{\gamma,0} \oplus \hat{z}_{\gamma,k} \cdot \Delta_\mathsf{A}$ according to (4). Notice that $\mathcal{P}_\mathsf{B}$ does not know $\mathsf{L}_{\gamma,0}$, $\Delta_\mathsf{A}$, and $\mathsf{K}[s_{\gamma,k}]$. Therefore, even $\mathcal{P}_\mathsf{B}$ can decrypt to obtain $\mathsf{L}_{\gamma,\hat{z}_\gamma}$, $\mathcal{P}_\mathsf{B}$ only knows either $\mathsf{L}_{\gamma,0}$ or $\mathsf{L}_{\gamma,1}$ (depending on the value of $\hat{z}_\gamma$) and cannot know both. Knowing both helps $\mathcal{P}_\mathsf{B}$ to decrypt all four rows of the garbled table, which is strictly prohibited since it compromises the privacy of $\mathcal{P}_\mathsf{A}$.
**Putting All Together.** For each garbled table corresponding to each AND gate $(\alpha, \beta, \gamma, \cdot)$, each encrypted row must include $r_{\gamma,k}$, $\mathsf{M}[r_{\gamma,k}]$, and $\mathsf{L}_{\gamma,0} \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A} \oplus \mathsf{K}[s_{\gamma,k}]$. The discussion is realized in Figure 13, namely, WRK protocol, in Appendix B. This protocol employs functionality $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ in Figure 5 as a subroutine for generating authenticated shared random masks and obtaining shared authenticated AND triples, i.e., authenticated $r_1, r_2, r_3$ (held by $\mathcal{P}_\mathsf{A}$) and $s_1, s_2, s_3$ (held by $\mathcal{P}_\mathsf{A}$) satisfying $(r_1 \oplus s_1) \cdot (r_2 \oplus s_2) = r_3 \oplus s_3$.

This functionality runs between $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ as follows:

1. Assuming no $\Delta_\mathsf{A}, \Delta_\mathsf{B}$ stored, upon receiving $\Delta_\mathsf{A}$ from $\mathcal{P}_\mathsf{A}$ and (init) from $\mathcal{P}_\mathsf{B}$, sample $\Delta_\mathsf{B} \xleftarrow{\$} \{0,1\}^\rho$, where $\rho$ is the statistical security parameter, and store $\Delta_\mathsf{A}$ and $\Delta_\mathsf{B}$. Then, send $\Delta_\mathsf{B}$ to $\mathcal{P}_\mathsf{B}$.
2. Upon receiving $(\mathsf{random}, r, \mathsf{M}[r], \mathsf{K}[s])$ from $\mathcal{P}_\mathsf{A}$ and $(\mathsf{random})$ from $\mathcal{P}_\mathsf{B}$, sample $s \xleftarrow{\$} \{0,1\}$, and compute $\mathsf{M}[s] := \mathsf{K}[s] \oplus s \cdot \Delta_\mathsf{A}$ and $\mathsf{K}[r] := \mathsf{M}[r] \oplus r \cdot \Delta_\mathsf{B}$. Then, send $(s, \mathsf{M}[s], \mathsf{K}[r])$ to $\mathcal{P}_\mathsf{B}$.
3. Upon receiving $(\mathsf{and}, (r_1, \mathsf{M}[r_1], \mathsf{K}[s_1]), (r_2, \mathsf{M}[r_2], \mathsf{K}[s_2]), (r_3, \mathsf{M}[r_3], \mathsf{K}[s_3]))$ from $\mathcal{P}_\mathsf{A}$ and $(\mathsf{and}, (s_1, \mathsf{M}[s_1], \mathsf{K}[r_1]), (s_2, \mathsf{M}[s_2], \mathsf{K}[r_2]))$ from $\mathcal{P}_\mathsf{B}$, send $(\mathsf{cheat})$ to $\mathcal{P}_\mathsf{B}$ if it does not hold that $\mathsf{M}[r_i] = \mathsf{K}[r_i] \oplus r_i \cdot \Delta_\mathsf{B}$ and $\mathsf{M}[s_i] = \mathsf{K}[s_i] \oplus s_i \cdot \Delta_\mathsf{A}$ for $i \in \{1, 2\}$. Otherwise, compute $s_3 = ((r_1 \oplus s_1) \cdot (r_2 \oplus s_2)) \oplus r_3$ and $\mathsf{M}[s_3] := \mathsf{K}[s_3] \oplus s_3 \cdot \Delta_\mathsf{A}$. Send $(s_3, \mathsf{M}[s_3], \mathsf{K}[r_3])$ to $\mathcal{P}_\mathsf{B}$.

**Fig. 5.** Functionality $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ [31, Figure 1].

**Theorem 1 (Security of $\Pi_{\mathsf{2PC}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ (Informal)).** *If $H$ is modeled as a random oracle, the protocol $\Pi_{\mathsf{2PC}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ in Figure 13, securely computes $f$ against malicious adversaries with statistical security $2^{-\rho}$ in the $\mathcal{F}_{\mathsf{pre}}$-hybrid model where $\rho$ is the bit size of $\Delta_\mathsf{A}$ (as well as $\Delta_\mathsf{B}$).*

## 3.2 Making WRK Publicly Auditable

Our purpose is to make WRK protocol publicly auditable. That is, a public verifier can trust that the output from $\mathcal{P}_\mathsf{B}$ obtained after the garbling, by $\mathcal{P}_\mathsf{A}$, and evaluating, by $\mathcal{P}_\mathsf{B}$, are correct. We follow the recent VOLEitH approach [7,16,12] by applying SoftSpokenOT [29] for making VOLE-based protocols (e.g., Wolverine [32] and Quicksilver [35]) publicly verifiable.
**Verifier $\mathcal{V}$.** To make the WRK protocol publicly auditable, we modify the protocol in Figure 13 to include an additional party called verifier $\mathcal{V}$. This party will play the role of verifying the transcript communicated between $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$.
**Issues.** We now discuss the employment of SoftSpokenOT [29] into WRK protocol. Assume that we transform all authenticated randomness of $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ into the forms specified by SoftSpokenOT (see Section 2.2). The transformation requires making $\Delta_\mathsf{A}$ and $\Delta_\mathsf{B}$ to be public. However, doing so incurs the following issues.

– For each wire $w$, $\mathcal{P}_\mathsf{B}$ is allowed to know only either $\mathsf{L}_{w,0}$ or $\mathsf{L}_{w,1}$. However, if $\Delta_\mathsf{A}$ is public and $\mathsf{L}_{w,1} = \mathsf{L}_{w,0} \oplus \Delta_\mathsf{A}$, knowing either $\mathsf{L}_{w,0}$ or $\mathsf{L}_{w,1}$ implies knowledge of both labels. This, hence, violates the security of the original WRK protocol.
– Using only $\Delta_\mathsf{A}$ and $\Delta_\mathsf{B}$ for public verification is not convenient for amplifying soundness. Notice that VOLEitH systems [7,16,12] usually require repeating the protocol with a few distinct global IT-MAC keys to reduce the soundness error.

– For verifying each shared AND triple, e.g., $(\alpha, \beta, \gamma, \cdot)$, $\mathcal{V}$ should believe that $\underbrace{(r_\alpha \oplus s_\alpha)}_{\lambda_\alpha} \cdot \underbrace{(r_\beta \oplus s_\beta)}_{\lambda_\beta} = \underbrace{r'_\gamma \oplus s'_\gamma}_{\lambda_\alpha \cdot \lambda_\beta}$. Here, in WRK protocol, $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ rely on the outputs of $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$, guaranteeing the above identity holds. However, when making public auditability, since any public verifier does not receive those outputs from $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$, she cannot verify such multiplications hold.

**Our Approach.** We briefly sketch our approach for coping with the above-mentioned issues as follows.
*Authenticating with SoftSpokenOT.* We first recall that $\Delta_{\mathsf{A}}$ employed in WRK protocol is for two purposes: (i) authenticating $\mathcal{P}_{\mathsf{B}}$'s shared random mask and (ii) allowing $\mathcal{P}_{\mathsf{B}}$ to compute $\mathsf{L}_{\gamma,\hat{z}_\gamma}$ for any AND gate $(\alpha, \beta, \gamma, \cdot)$. As discussed above, making $\Delta_{\mathsf{A}}$ public violates $\mathcal{P}_{\mathsf{A}}$'s privacy, and using SoftSpokenOT requires repeating multiple times to amplify soundness. Hence, we denote by $\kappa$ as the number of such repetitions. In our design, we still keep $\Delta_{\mathsf{A}}$ private to protect $\mathcal{P}_{\mathsf{A}}$'s privacy. To authenticate $\mathcal{P}_{\mathsf{A}}$'s random secrets and enable public auditability, we apply VOLEitH (c.f. Section 2.2) to make VOLEitH correlations for them as follows.

For each wire $w$, assume that $r_w$ is authenticated in a way that $\mathcal{P}_{\mathsf{A}}$ keeps $r_w$ and $(\mathsf{M}'_j[r_w])_{j\in[\kappa]}$. This can be done by first generating a VOLEitH correlation for a random bit $\overline{r}_w$. Then, by applying $\Pi_{\mathsf{fix}}^{\mathcal{P}_{\mathsf{A}},\mathcal{V}}$ (c.f. Section 2.3), we can transform from VOLEitH correltion for $\overline{r}_w$ to the one for $r_w$. Notice that the process must be done in two steps: before and after knowing $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$. When $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$ is known, the IT-MAC key $\mathsf{K}'_j[r_w]$ is also publicly known.

Symmetrically, we can make VOLEitH correlations for $\mathcal{P}_{\mathsf{B}}$'s secret by using another sequence of global values $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$.

Since $\nabla_{\mathsf{A},j}$ is not employed to determine $\mathsf{L}_{\gamma,\hat{z}_\gamma}$ for any AND gate $(\alpha, \beta, \gamma, \cdot)$, $\mathcal{P}_{\mathsf{A}}$'s privacy is hence guaranteed as long as $\Delta_{\mathsf{A}}$ is not compromised. On the other hand, any attempt from $\mathcal{P}_{\mathsf{A}}$ to violate the correctness of the protocol is hence captured by the authenticated shared random masks via $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$, instead of $\Delta_{\mathsf{A}}$, to make the protocol publicly auditable.
*Verifying Shared AND Triples.* For an AND gate $(\alpha, \beta, \gamma, \cdot)$, as discussed above, we additionally use $r'_\gamma$ and $s'_\gamma$ such that

$$\underbrace{(r_\alpha \oplus s_\alpha)}_{\lambda_\alpha} \cdot \underbrace{(r_\beta \oplus s_\beta)}_{\lambda_\beta} = \underbrace{r'_\gamma \oplus s'_\gamma}_{\lambda_\alpha \cdot \lambda_\beta}. \tag{5}$$

This can be done by running protocol $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}$ (c.f. Figure 6). Specifically, $(x_{\mathsf{A}}^{(i)}, y_{\mathsf{A}}^{(i)}, z_{\mathsf{A}}^{(i)})_{i\in[W]}$ and $(x_{\mathsf{B}}^{(i)}, y_{\mathsf{B}}^{(i)}, z_{\mathsf{B}}^{(i)})_{i\in[W]}$ are held by $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, respectively, and are authenticated via $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$, respectively. Our purpose is to show that, via $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}$, $(x_{\mathsf{A}}^{(i)} \oplus x_{\mathsf{B}}^{(i)}) \cdot (y_{\mathsf{A}}^{(i)} \oplus y_{\mathsf{B}}^{(i)}) = z_{\mathsf{A}}^{(i)} \oplus z_{\mathsf{B}}^{(i)}$ $\forall i \in [W]$. The intuition follows the multiplication trick [3,32]: We assume that $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ additionally keep the triples $(a_{\mathsf{A},j}^{(i)}, b_{\mathsf{A},j}^{(i)}, c_{\mathsf{A},j}^{(i)})_{j\in[\kappa],i\in[W]}$ and $(a_{\mathsf{B},j}^{(i)}, b_{\mathsf{B},j}^{(i)}, c_{\mathsf{B},j}^{(i)})_{j\in[\kappa],i\in[W]}$, respectively, authenticated via $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$, respectively, s.t.
$(a_{\mathsf{A},j}^{(i)} \oplus a_{\mathsf{B},j}^{(i)}) \cdot (b_{\mathsf{A},j}^{(i)} \oplus b_{\mathsf{B},j}^{(i)}) = c_{\mathsf{A},j}^{(i)} \oplus c_{\mathsf{B},j}^{(i)}$ $\forall j \in [\kappa], \forall i \in [W]$.

We denote by $x^{(i)} = x_{\mathsf{A}}^{(i)} \oplus x_{\mathsf{B}}^{(i)}$, $y^{(i)} = y_{\mathsf{A}}^{(i)} \oplus y_{\mathsf{B}}^{(i)}$, $z^{(i)} = z_{\mathsf{A}}^{(i)} \oplus z_{\mathsf{B}}^{(i)}$, $a_j^{(i)} = a_{\mathsf{A},j}^{(i)} \oplus a_{\mathsf{B},j}^{(i)}$, $b_j^{(i)} = b_{\mathsf{A},j}^{(i)} \oplus b_{\mathsf{B},j}^{(i)}$, and $c_j^{(i)} = c_{\mathsf{A},j}^{(i)} \oplus c_{\mathsf{B},j}^{(i)}$. Hence, $a_j^{(i)} \cdot b_j^{(i)} = c_j^{(i)}$ $\forall j \in [\kappa], i \in [W]$ accordingly.

In brief, both parties obtain the authenticated additive shares of $d_j^{(i)} = x^{(i)} \oplus a_j^{(i)}$ and $e_j^{(i)} = y^{(i)} \oplus b_j^{(i)}$. By publishing the mentioned shares, the parties, including $\mathcal{V}$, can determine the values $d_j^{(i)}$ and $e_j^{(i)}$. Then, $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ compute the authenticated additive shares of $\tilde{z}_j^{(i)} = z^{(i)} \oplus c_j^{(i)} \oplus d_j^{(i)} \cdot b_j^{(i)} \oplus e_j^{(i)} \cdot a_j^{(i)}$. By publishing both shares of $\tilde{z}_j^{(i)}$ and checking that $\tilde{z}_j^{(i)} \oplus d_j^{(i)} \cdot e_j^{(i)} = 0$ for $j \in [\kappa]$ and $i \in [W]$, we are guaranteed that $x^{(i)} \cdot y^{(i)} = z^{(i)}$, for $i \in [W]$, with overwhelming probability. This can be explained as follows. Notice that

$$\begin{aligned}
&\tilde{z}_j^{(i)} \oplus d_j^{(i)} \cdot e_j^{(i)} \\
&= z^{(i)} \oplus c_j^{(i)} \oplus d_j^{(i)} \cdot b_j^{(i)} \oplus e_j^{(i)} \cdot a_j^{(i)} \oplus d_j^{(i)} \cdot e_j^{(i)} \\
&= z^{(i)} \oplus c_j^{(i)} \oplus (x^{(i)} \oplus a_j^{(i)}) \cdot b_j^{(i)} \oplus (y^{(i)} \oplus b_j^{(i)}) \cdot a_j^{(i)} \\
&\quad \oplus (x^{(i)} \oplus a_j^{(i)}) \cdot (y^{(i)} \oplus b_j^{(i)}) \\
&= z^{(i)} \oplus c_j^{(i)} \oplus x^{(i)} \cdot y^{(i)} \oplus a_j^{(i)} \cdot b_j^{(i)}.
\end{aligned}$$

This protocol runs between $\mathcal{P}_{\mathsf{A}}$, $\mathcal{P}_{\mathsf{B}}$, and $\mathcal{V}$. We assume that authenticated $\mathcal{P}_{\mathsf{A}}$'s values (respectively, $\mathcal{P}_{\mathsf{B}}$'s values) are w.r.t. $\nabla_{\mathsf{B},j}$ (respectively, $\nabla_{\mathsf{A},j}$) for $j \in [\kappa]$ and $\kappa, W \in \mathbb{N}$. When denoting $j$ and $i$ as subscripts of sets/sequences or after symbol $\forall$ without specifying the ranges, we understand that $j \in [\kappa]$ and $i \in [W]$, respectively.

1. $\mathcal{P}_{\mathsf{A}}$ keeps $\left((x_{\mathsf{A}}^{(i)}, y_{\mathsf{A}}^{(i)}, z_{\mathsf{A}}^{(i)})_i, (\mathsf{M}'_j[x_{\mathsf{A}}^{(i)}], \mathsf{M}'_j[y_{\mathsf{A}}^{(i)}], \mathsf{M}'_j[z_{\mathsf{A}}^{(i)}])_{j,i}, (a_{\mathsf{A},j}^{(i)}, b_{\mathsf{A},j}^{(i)}, c_{\mathsf{A},j}^{(i)}, \mathsf{M}'_j[a_{\mathsf{A},j}^{(i)}], \mathsf{M}'_j[b_{\mathsf{A},j}^{(i)}], \mathsf{M}'_j[c_{\mathsf{A},j}^{(i)}])_{j,i}\right)$ and $\mathcal{P}_{\mathsf{B}}$ keeps $\left((x_{\mathsf{B}}^{(i)}, y_{\mathsf{B}}^{(i)}, z_{\mathsf{B}}^{(i)})_i, (\mathsf{M}'_j[x_{\mathsf{B}}^{(i)}], \mathsf{M}'_j[y_{\mathsf{B}}^{(i)}], \mathsf{M}'_j[z_{\mathsf{B}}^{(i)}])_{j,i}, (a_{\mathsf{B},j}^{(i)}, b_{\mathsf{B},j}^{(i)}, c_{\mathsf{B},j}^{(i)}, \mathsf{M}'_j[a_{\mathsf{B},j}^{(i)}], \mathsf{M}'_j[b_{\mathsf{B},j}^{(i)}], \mathsf{M}'_j[c_{\mathsf{B},j}^{(i)}])_{j,i}\right)$. $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ work as follows.
   (a) $\mathcal{P}_{\mathsf{A}}$ publishes $d_{\mathsf{A},j}^{(i)} := x_{\mathsf{A}}^{(i)} \oplus a_{\mathsf{A},j}^{(i)}$, $e_{\mathsf{A},j}^{(i)} := y_{\mathsf{A}}^{(i)} \oplus b_{\mathsf{A},j}^{(i)}$, $\mathsf{M}'_j[d_{\mathsf{A},j}^{(i)}] := \mathsf{M}'_j[x_{\mathsf{A}}^{(i)}] \oplus \mathsf{M}'_j[a_{\mathsf{A},j}^{(i)}]$, $\mathsf{M}'_j[e_{\mathsf{A},j}^{(i)}] := \mathsf{M}'_j[y_{\mathsf{A}}^{(i)}] \oplus \mathsf{M}'_j[b_{\mathsf{A},j}^{(i)}]$ $\forall j$, $\forall i$.
   (b) $\mathcal{P}_{\mathsf{B}}$ publishes $d_{\mathsf{B},j}^{(i)} := x_{\mathsf{B}}^{(i)} \oplus a_{\mathsf{B},j}^{(i)}$, $e_{\mathsf{B},j}^{(i)} := y_{\mathsf{B}}^{(i)} \oplus b_{\mathsf{B},j}^{(i)}$, $\mathsf{M}'_j[d_{\mathsf{B},j}^{(i)}] := \mathsf{M}'_j[x_{\mathsf{B}}^{(i)}] \oplus \mathsf{M}'_j[a_{\mathsf{B},j}^{(i)}]$, $\mathsf{M}'_j[e_{\mathsf{B},j}^{(i)}] := \mathsf{M}'_j[y_{\mathsf{B}}^{(i)}] \oplus \mathsf{M}'_j[b_{\mathsf{B},j}^{(i)}]$ $\forall j$, $\forall i$.
   (c) Each party locally computes $d_j^{(i)} := d_{\mathsf{A},j}^{(i)} \oplus d_{\mathsf{B},j}^{(i)}$ and $e_j^{(i)} := e_{\mathsf{A},j}^{(i)} \oplus e_{\mathsf{B},j}^{(i)}$ $\forall j$, $\forall i$.
   (d) $\mathcal{P}_{\mathsf{A}}$ publishes $\tilde{z}_{\mathsf{A},j}^{(i)} := z_{\mathsf{A}}^{(i)} \oplus c_{\mathsf{A},j}^{(i)} \oplus d_j^{(i)} \cdot b_{\mathsf{A},j}^{(i)} \oplus e_j^{(i)} \cdot a_{\mathsf{A},j}^{(i)}$,
   $\mathsf{M}'_j[\tilde{z}_{\mathsf{A},j}^{(i)}] := \mathsf{M}'_j[z_{\mathsf{A}}^{(i)}] \oplus \mathsf{M}'_j[c_{\mathsf{A},j}^{(i)}] \oplus d_j^{(i)} \cdot \mathsf{M}'_j[b_{\mathsf{A},j}^{(i)}] \oplus e_j^{(i)} \cdot \mathsf{M}'_j[a_{\mathsf{A},j}^{(i)}]$ $\forall j$, $\forall i$.
   (e) $\mathcal{P}_{\mathsf{B}}$ publishes $\tilde{z}_{\mathsf{B},j}^{(i)} := z_{\mathsf{B}}^{(i)} \oplus c_{\mathsf{B},j}^{(i)} \oplus d_j^{(i)} \cdot b_{\mathsf{B},j}^{(i)} \oplus e_j^{(i)} \cdot a_{\mathsf{B},j}^{(i)}$,
   $\mathsf{M}'_j[\tilde{z}_{\mathsf{B},j}^{(i)}] := \mathsf{M}'_j[z_{\mathsf{B}}^{(i)}] \oplus \mathsf{M}'_j[c_{\mathsf{B},j}^{(i)}] \oplus d_j^{(i)} \cdot \mathsf{M}'_j[b_{\mathsf{B},j}^{(i)}] \oplus e_j^{(i)} \cdot \mathsf{M}'_j[a_{\mathsf{B},j}^{(i)}]$ $\forall j$, $\forall i$.
   (f) Each party checks whether $\tilde{z}_{\mathsf{A},j}^{(i)} \oplus \tilde{z}_{\mathsf{B},j}^{(i)} \oplus d_j^{(i)} \cdot e_j^{(i)} = 0$ $\forall j, \forall i$.
2. On common inputs $(\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j})_j$, $(\mathsf{K}'_j[x_{\mathsf{A}}^{(i)}], \mathsf{K}'_j[y_{\mathsf{A}}^{(i)}], \mathsf{K}'_j[z_{\mathsf{A}}^{(i)}])_{j,i}$, $(\mathsf{K}'_j[a_{\mathsf{A},j}^{(i)}], \mathsf{K}'_j[b_{\mathsf{A},j}^{(i)}], \mathsf{K}'_j[c_{\mathsf{A},j}^{(i)}])_{j,i}$,$(\mathsf{K}'_j[x_{\mathsf{B}}^{(i)}], \mathsf{K}'_j[y_{\mathsf{B}}^{(i)}], \mathsf{K}'_j[z_{\mathsf{B}}^{(i)}])_{j,i}$, and $(\mathsf{K}'_j[a_{\mathsf{B},j}^{(i)}], \mathsf{K}'_j[b_{\mathsf{B},j}^{(i)}], \mathsf{K}'_j[c_{\mathsf{B},j}^{(i)}])_{j,i}$, each party ($\mathcal{P}_{\mathsf{A}}$, $\mathcal{P}_{\mathsf{B}}$, or $\mathcal{V}$) can compute and check as follows.
   (a) Each party computes $\mathsf{K}'_j[d_{\mathsf{A},j}^{(i)}] := \mathsf{K}'_j[x_{\mathsf{A}}^{(i)}] \oplus \mathsf{K}'_j[a_{\mathsf{A},j}^{(i)}]$, $\mathsf{K}'_j[e_{\mathsf{A},j}^{(i)}] := \mathsf{K}'_j[y_{\mathsf{A}}^{(i)}] \oplus \mathsf{K}'_j[b_{\mathsf{A},j}^{(i)}]$ and checks whether $\mathsf{M}'_j[d_{\mathsf{A},j}^{(i)}] = \mathsf{K}'_j[d_{\mathsf{A},j}^{(i)}] \oplus d_{\mathsf{A},j}^{(i)} \cdot \nabla_{\mathsf{B},j}$, $\mathsf{M}'_j[e_{\mathsf{A},j}^{(i)}] = \mathsf{K}'_j[e_{\mathsf{A},j}^{(i)}] \oplus e_{\mathsf{A},j}^{(i)} \cdot \nabla_{\mathsf{B},j}$ $\forall j, \forall i$.
   (b) Each party computes $\mathsf{K}'_j[d_{\mathsf{B},j}^{(i)}] := \mathsf{K}'_j[x_{\mathsf{B}}^{(i)}] \oplus \mathsf{K}'_j[a_{\mathsf{B},j}^{(i)}]$, $\mathsf{K}'_j[e_{\mathsf{B},j}^{(i)}] := \mathsf{K}'_j[y_{\mathsf{B}}^{(i)}] \oplus \mathsf{K}'_j[b_{\mathsf{B},j}^{(i)}]$ and checks whether $\mathsf{M}'_j[d_{\mathsf{B},j}^{(i)}] = \mathsf{K}'_j[d_{\mathsf{B},j}^{(i)}] \oplus d_{\mathsf{B},j}^{(i)} \cdot \nabla_{\mathsf{A},j}$, $\mathsf{M}'_j[e_{\mathsf{B},j}^{(i)}] = \mathsf{K}'_j[e_{\mathsf{B},j}^{(i)}] \oplus e_{\mathsf{B},j}^{(i)} \cdot \nabla_{\mathsf{A},j}$ $\forall j, \forall i$.
   (c) Each party computes $\mathsf{K}'_j[\tilde{z}_{\mathsf{A},j}^{(i)}] := \mathsf{K}'_j[z_{\mathsf{A}}^{(i)}] \oplus \mathsf{K}'_j[c_{\mathsf{A},j}^{(i)}] \oplus d_j^{(i)} \cdot \mathsf{K}'_j[b_{\mathsf{A},j}^{(i)}] \oplus e_j^{(i)} \cdot \mathsf{K}'_j[a_{\mathsf{A},j}^{(i)}]$ and checks whether $\mathsf{M}'_j[\tilde{z}_{\mathsf{A},j}^{(i)}] = \mathsf{K}'_j[\tilde{z}_{\mathsf{A},j}^{(i)}] \oplus \tilde{z}_{\mathsf{A},j}^{(i)} \cdot \nabla_{\mathsf{B},j}$ $\forall j, \forall i$.
   (d) Each party computes $\mathsf{K}'_j[\tilde{z}_{\mathsf{B},j}^{(i)}] := \mathsf{K}'_j[z_{\mathsf{B}}^{(i)}] \oplus \mathsf{K}'_j[c_{\mathsf{B},j}^{(i)}] \oplus d_j^{(i)} \cdot \mathsf{K}'_j[b_{\mathsf{B},j}^{(i)}] \oplus e_j^{(i)} \cdot \mathsf{K}'_j[a_{\mathsf{B},j}^{(i)}]$ and checks whether $\mathsf{M}'_j[\tilde{z}_{\mathsf{B},j}^{(i)}] = \mathsf{K}'_j[\tilde{z}_{\mathsf{B},j}^{(i)}] \oplus \tilde{z}_{\mathsf{B},j}^{(i)} \cdot \nabla_{\mathsf{A},j}$ $\forall j, \forall i$.

**Fig. 6.** Sub-Protocol $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}$. Highlighted texts are those publicly revealed for public auditability purpose.

Since we assumed $c_j^{(i)} = a_j^{(i)} \cdot b_j^{(i)}$ above, it holds that $z^{(i)} = x^{(i)} \cdot y^{(i)}$ for $j \in [\kappa]$ and $i \in [W]$.

Above we already mentioned the shares of public values $d_j^{(i)}$, $e_j^{(i)}$, and $\tilde{z}_j^{(i)}$ without specifying in details how to compute those shares. In particular, we parse $d_j^{(i)} = d_{\mathsf{A},j}^{(i)} \oplus d_{\mathsf{B},j}^{(i)}$, $e_j^{(i)} = e_{\mathsf{A},j}^{(i)} \oplus e_{\mathsf{B},j}^{(i)}$, and $\tilde{z}_j^{(i)} = \tilde{z}_{\mathsf{A},j}^{(i)} \oplus \tilde{z}_{\mathsf{B},j}^{(i)}$ which are computed as in the description of protocol $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}$ (c.f. Figure 6).

*Using Balls-and-Bins Method for Generating Authenticated Shared AND Triples.* In our protocol, there are $W$ such AND gates. To check all $W$ AND gates, we generate $L$ pairs of shared random AND triples. Then, we apply the balls-and-bins method [3,32] to check all $W$ AND gates. We first recall the balls-and-bins method in Lemma 1, and by appropriately setting parameters (say, $L$, $\mathsf{bs}$, and $\mathsf{rm}$), the probability stated at the end of Lemma 1 can become negligible. Therefore, assume that such a probability is negligible.

**Lemma 1 (Balls-and-Bins [3,32]).** *There are $L = W \cdot \mathsf{bs} + \mathsf{rm}$ balls, where $W, \mathsf{bs}, \mathsf{rm} \in \mathbb{N}$, labeled either* good *or* bad. *(1)* $\mathsf{rm}$ *random balls are chosen. (2) The remaining $W \cdot \mathsf{bs}$ are randomly partitioned into $W$ bins where each bin contains exactly* $\mathsf{bs}$ *balls. We say that a bin is fully good (respectively, fully bad) if all balls in this bin are good (respectively, bad). Then, the probability that (i) the first randomly chosen* $\mathsf{rm}$ *random balls are labeled* good, *and (ii) at least one bin is fully bad and all other bins are either fully good or fully bad is at most $\binom{W \cdot \mathsf{bs} + \mathsf{rm}}{\mathsf{bs}}^{-1}$.*

The idea for using the balls-and-bins method to check all $W$ AND gates is as follows. We assume that $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ respectively keep

$$(\tilde{a}_{\mathsf{A},j}^{(\ell)}, \tilde{b}_{\mathsf{A},j}^{(\ell)}, \tilde{c}_{\mathsf{A},j}^{(\ell)})_{j \in [\kappa], \ell \in [L]}, \text{ and } (\tilde{a}_{\mathsf{B},j}^{(\ell)}, \tilde{b}_{\mathsf{B},j}^{(\ell)}, \tilde{c}_{\mathsf{B},j}^{(\ell)})_{j \in [\kappa], \ell \in [L]}$$

such that $(\tilde{a}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{a}^{(\ell)}_{\mathsf{B},j}) \cdot (\tilde{b}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{b}^{(\ell)}_{\mathsf{B},j}) = \tilde{c}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{c}^{(\ell)}_{\mathsf{B},j}$. These values are also authenticated by $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$ such that $\mathcal{P}_p$, for $p \in \{\mathsf{A},\mathsf{B}\}$, additionally keeps

$$(\mathsf{M}'_j[\tilde{a}^{(\ell)}_{p,j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{p,j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{p,j}])_{j\in[\kappa],\ell\in[L]} \text{ and}$$
$$(\mathsf{K}'_j[\tilde{a}^{(\ell)}_{p',j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{p',j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{p',j}])_{j\in[\kappa],\ell\in[L]}$$

where $p' = \mathsf{B}$, if $p = \mathsf{A}$, and $p' = \mathsf{A}$, otherwise.

To check whether the secrets $(x^{(i)}_{\mathsf{A}}, y^{(i)}_{\mathsf{A}}, z^{(i)}_{A})$ and $(x^{(i)}_{\mathsf{B}}, y^{(i)}_{\mathsf{B}}, z^{(i)}_{\mathsf{B}})$ of $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, respectively, at $W$ AND gates satisfy $(x^{(i)}_{\mathsf{A}} \oplus x^{(i)}_{\mathsf{B}}) \cdot (y^{(i)}_{\mathsf{A}} \oplus y^{(i)}_{\mathsf{B}}) = z^{(i)}_{\mathsf{A}} \oplus z^{(i)}_{\mathsf{B}}$, we employs the balls-and-bins method (c.f. Lemma 1) as follows.

Although we introduced $(\tilde{a}^{(\ell)}_{p,j}, \tilde{b}^{(\ell)}_{p,j}, \tilde{c}^{(\ell)}_{p,j})_{j\in[\kappa],\ell\in[L]}$, for $p \in \{\mathsf{A},\mathsf{B}\}$, authenticated via $(\nabla_{p',j})_{j\in[\kappa]}$ where $p' = \mathsf{B}$, if $p = \mathsf{A}$, and $p' = \mathsf{A}$, otherwise, $(\tilde{a}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{a}^{(\ell)}_{\mathsf{B},j}) \cdot (\tilde{b}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{b}^{(\ell)}_{\mathsf{B},j}) = \tilde{c}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{c}^{(\ell)}_{\mathsf{B},j}$ is not guaranteed.

Looking back at Lemma 1, we have balls labeled either $\mathsf{good}$ or $\mathsf{bad}$. When considering in our context, in the repetition $j \in [\kappa]$, the ball $\ell \in [L]$ is understood to be $\mathsf{good}$ if it holds that $(\tilde{a}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{a}^{(\ell)}_{\mathsf{B},j}) \cdot (\tilde{b}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{b}^{(\ell)}_{\mathsf{B},j}) = \tilde{c}^{(\ell)}_{\mathsf{A},j} \oplus \tilde{c}^{(\ell)}_{\mathsf{B},j}$. Hence, to Lemma 1, we first randomly pick $\mathsf{rm}$ random balls to check whether they are all labeled $\mathsf{good}$ by forcing both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ to reveal all values as well as the IT-MAC tags corresponding to the chosen $\mathsf{rm}$ balls that we just mentioned, i.e., if the ball $\ell$ is chosen among $\mathsf{rm}$ balls, $\tilde{a}^{(\ell)}_{p,j}$, $\tilde{b}^{(\ell)}_{p,j}$, and $\tilde{c}^{(\ell)}_{p,j}$ for $p \in \{\mathsf{A},\mathsf{B}\}$ as well as their corresponding IT-MAC tags w.r.t. $\nabla_{p',j}$ where $p' = \mathsf{B}$, if $p = \mathsf{A}$, and $p' = \mathsf{A}$, otherwise. For the remaining $W \cdot \mathsf{bs}$ balls, we randomly partition them into $W$ bins, corresponding to $W$ AND gates, such that each bin contains exactly $\mathsf{bs}$ balls. Then, we run $\mathsf{bs}$ times such that, for $k \in [\mathsf{bs}]$, $j \in [\kappa]$, and $i \in [W]$, in the $j$-th repetition, we use that $k$-th ball (w.r.t. the $j$-th repetition) from bin $i$ (w.r.t. the $j$-th repetition) to test the $i$-th AND gate in the by employing protocol $\Pi^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$ in Figure 6. Notice that if secret values of $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ do not hold at some AND gate, then, to pass the test by $\Pi^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$, all balls in the corresponding bin to such AND must be labeled $\mathsf{bad}$, since any good ball will make the test failed. Hence, it follows from Lemma 1 that such a case can happen with negligible probability.

With the above strategy, we follow [32] to choose and partition the balls as follows. Recall that we have a verifier $\mathcal{V}$ to verify the garbling process. $\mathcal{V}$ generates $\kappa$ permutations $\pi_j : [L] \to [L]$ for $j \in [\kappa]$ and send them to both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$.

Hence, for $p \in \{\mathsf{A},\mathsf{B}\}$, $\mathcal{P}_p$ shuffles $(\tilde{a}^{(\ell)}_{p,j}, \tilde{b}^{(\ell)}_{p,j}, \tilde{c}^{(\ell)}_{p,j})_{j\in[\kappa],\ell\in[L]}$ by assigning $(a^{(\ell)}_{p,j}, b^{(\ell)}_{p,j}, c^{(\ell)}_{p,j}) := (\tilde{a}^{(\pi_j(\ell))}_{p,j}, \tilde{b}^{(\pi_j(\ell))}_{p,j}, \tilde{c}^{(\pi_j(\ell))}_{p,j})$ for $p \in \{\mathsf{A},\mathsf{B}\}$, $j \in [\kappa]$ and $\ell \in [L]$ as well as the corresponding IT-MAC tags w.r.t. $(\nabla_{p',j})_{j\in[\kappa]}$. Notice that we do not need to shuffle the keys at the moment as we are employing SoftSpokenOT. Then, we have both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ revealing

$$(a^{(i)}_{p,j}, b^{(i)}_{p,j}, c^{(i)}_{p,j})_{p\in\{\mathsf{A},\mathsf{B}\},i\in[W\cdot\mathsf{bs}+1,W\cdot\mathsf{bs}+\mathsf{rm}]},$$

where $\mathsf{bs}$ and $\mathsf{rm}$ are parameters introduced in Lemma 1 that constitute $L = W \cdot \mathsf{bs} + \mathsf{rm}$, and their corresponding IT-MAC tags w.r.t. $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$ or $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$. Then, for $k \in [\mathsf{bs}]$, we will use

$$(a^{(i)}_{p,j}, b^{(i)}_{p,j}, c^{(i)}_{p,j})_{p\in\{\mathsf{A},\mathsf{B}\},j\in[\kappa],i\in[(k-1)\cdot\mathsf{bs}+1,k\cdot\mathsf{bs}+\mathsf{rm}]}$$

to test the $W$ AND gates by running $\Pi^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$.

*Handling Opened Authenticated Shares at Output Wires.* In the phase **Output Determination** in Figure 13, $\mathcal{P}_{\mathsf{B}}$ need to reveal $s_w$ for $w \in \mathcal{I}_{\mathsf{O}}$ and the respective IT-MAC tags (for VOLEitH correlations) for checking authenticity. When $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$ and $(\mathsf{K}'_j[s_w])_{j\in[\kappa]}$ are known, a public verifier can verify the validity of $s_w$.

## 4 $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ and $\Pi_{\mathsf{sVOLE\text{-}2PC}}$

In this section, we define the functionality $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ in Section 4.1 supporting our construction of $\Pi_{\mathsf{pa\text{-}2PC}}$ (c.f. Section 5). Then, we present $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ in Section 4.2.

### 4.1 Functionality $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$

We define $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ in Figure 7 as a subroutine for generating suitable global values, IT-MAC tags, and corresponding keys.

$\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ runs between $\mathcal{P}_\mathsf{A}$ $\mathcal{P}_\mathsf{B}$, and a verifier $\mathcal{V}$ with The sets $\mathcal{I}_\mathsf{A}, \mathcal{I}_\mathsf{B}$, and $\mathcal{I}_\mathsf{W}$, parameters $\kappa$, $\tau$, and $L$. When denoting $j$, $i$, and $\ell$ as subscripts of sets/sequences or after symbol $\forall$ without specifying the ranges, we understand that $j \in [\kappa]$, $i \in [W]$, and $\ell \in [L]$, respectively.

1. On inputs $\left(\mathsf{init}, (r_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (r'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{A},j}, \tilde{b}^{(\ell)}_{\mathsf{A},j}, \tilde{c}^{(\ell)}_{\mathsf{A},j})_{j,\ell}\right)$ and $\left(\mathsf{init}, (s_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (s'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{B},j}, \tilde{b}^{(\ell)}_{\mathsf{B},j}, \tilde{c}^{(\ell)}_{\mathsf{B},j})_{j,i}\right)$ from $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively:
   (a) Sample $\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j} \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall j$. If $\mathcal{V}$ is corrupted, receive $(\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j})_j$ from $\mathcal{V}$.
   (b) If $\mathcal{P}_\mathsf{A}$ is not corrupted, $\forall j$, sample $\mathsf{M}'_j[r_w] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$, $\mathsf{M}'_j[r'_w] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall w \in \mathcal{I}_\mathsf{W}$, and $\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall \ell$. If $\mathcal{P}_\mathsf{A}$ is corrupted, receive $\left((\mathsf{M}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$ from $\mathcal{P}_\mathsf{A}$. Then, compute $\left((\mathsf{K}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$ such that
$$\begin{cases} \mathsf{M}'_j[r_w] = \mathsf{K}'_j[r_w] \oplus r_w \cdot \nabla_{\mathsf{B},j} & \forall j, \forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}, \\ \mathsf{M}'_j[r'_w] = \mathsf{K}'_j[r'_w] \oplus r'_w \cdot \nabla_{\mathsf{B},j} & \forall j, \forall w \in \mathcal{I}_\mathsf{W}, \\ \mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}] = \mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}] \oplus \tilde{a}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j},\ \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}] = \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}] \oplus \tilde{b}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j},\ \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}] = \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}] \oplus \tilde{c}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j} & \forall j, \forall \ell. \end{cases} \tag{6}$$
   (c) If $\mathcal{P}_\mathsf{B}$ is not corrupted, $\forall j$, sample $\mathsf{M}'_j[s_w] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$, $\mathsf{M}'_j[s'_w] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall w \in \mathcal{I}_\mathsf{W}$, and $\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}] \xleftarrow{\$} \mathbb{F}_{2^\tau}$ $\forall \ell$. If $\mathcal{P}_\mathsf{B}$ is corrupted, receive $\left((\mathsf{M}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$ from $\mathcal{P}_\mathsf{B}$. Then, compute $\left((\mathsf{K}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$ such that
$$\begin{cases} \mathsf{M}'_j[s_w] = \mathsf{K}'_j[s_w] \oplus s_w \cdot \nabla_{\mathsf{A},j} & \forall j, \forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}, \\ \mathsf{M}'_j[s'_w] = \mathsf{K}'_j[s'_w] \oplus s'_w \cdot \nabla_{\mathsf{A},j} & \forall j, \forall w \in \mathcal{I}_\mathsf{W}, \\ \mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}] = \mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}] \oplus \tilde{a}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j},\ \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}] = \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}] \oplus \tilde{b}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j},\ \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}] = \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}] \oplus \tilde{c}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j} & \forall j, \forall \ell. \end{cases} \tag{7}$$
   (d) If $\mathcal{V}$ is corrupted, receive $\left((\mathsf{K}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$ and $\left((\mathsf{K}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$ from $\mathcal{V}$, and compute $\left((\mathsf{M}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$ and $\left((\mathsf{M}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$ such that both (6) and (7) hold.
   (e) Send $\left((\mathsf{M}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$ to $\mathcal{P}_\mathsf{A}$.
   (f) Send $\left((\mathsf{M}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$ to $\mathcal{P}_\mathsf{B}$.
2. On input $(\mathsf{get})$ from $\mathcal{P}_\mathsf{A}$, $\mathcal{P}_\mathsf{B}$, and $\mathcal{V}$, $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ publishes $(\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j})_{j \in [\kappa]}$,
$$\left((\mathsf{K}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j, \text{ and}$$
$$\left((\mathsf{K}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j.$$

**Fig. 7.** Functionality $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$. Highlighted texts are those publicly revealed for public auditability purpose.

Since $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ is for public auditability purpose, as discussed in Section 3.2, we have an additional party called verifier $\mathcal{V}$. This verifier plays the role of a public party (without possessing any secrets) to generate the public inputs $(\nabla_{\mathsf{A},j})_{j \in [\kappa]}$ and $(\nabla_{\mathsf{B},j})_{j \in [\kappa]}$ where $\kappa$ is the parameter for soundness amplification purpose.

This functionality is suitable with VOLEitH [7]. Specifically, we need the tuples

$$((r_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (r'_w)_{w \in \mathcal{I}_\mathsf{W}}) \text{ and } ((s_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (s'_w)_{w \in \mathcal{I}_\mathsf{W}})$$

kept by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively, and authenticated via $(\nabla_{\mathsf{B},j})_{j \in [\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j \in [\kappa]}$, respectively. Moreover, for verifying AND gates by using balls-and-bins method (c.f. Lemma 1), we additionally need $(\tilde{a}^{(\ell)}_{\mathsf{A},j}, \tilde{b}^{(\ell)}_{\mathsf{A},j}, \tilde{c}^{(\ell)}_{\mathsf{A},j})_{j \in [\kappa], \ell \in [L]}$ and $(\tilde{a}^{(\ell)}_{\mathsf{B},j}, \tilde{b}^{(\ell)}_{\mathsf{B},j}, \tilde{c}^{(\ell)}_{\mathsf{B},j})_{j \in [\kappa], \ell \in [L]}$ kept by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively, and authenticated via $(\nabla_{\mathsf{B},j})_{j \in [\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j \in [\kappa]}$, respectively.

Therefore, we define $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ for generating the global values $\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j}$ for $j \in [\kappa]$. Then, it generates the IT-MAC tags and keys for authenticating by the above values, using VOLEitH w.r.t. $(\nabla_{\mathsf{B},j})_{j \in [\kappa]}$ and $(\nabla_{\mathsf{A},j})_{j \in [\kappa]}$ as specified in Figure 7. Those correlations achieved must satisfy (6) and (7).

Protocol $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ runs between $\mathcal{P}_\mathsf{A}$ $\mathcal{P}_\mathsf{B}$, and a verifier $\mathcal{V}$ with The sets $\mathcal{I}_\mathsf{A}, \mathcal{I}_\mathsf{B}$, and $\mathcal{I}_\mathsf{W}$, parameters $\kappa$, $\tau$, $L$, and $N = |\mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}| + W + 3L$ where $W = |\mathcal{I}_\mathsf{W}|$. When denoting $j$, $i$, and $\ell$ as subscripts of sets/sequences or after symbol $\forall$ without specifying the ranges, we understand that $j \in [\kappa]$, $i \in [W]$, and $\ell \in [L]$, respectively.

1. On inputs $\left(\mathsf{init}, (r_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (r'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{A},j}, \tilde{b}^{(\ell)}_{\mathsf{A},j}, \tilde{c}^{(\ell)}_{\mathsf{A},j})_{j,\ell}\right)$ and $\left(\mathsf{init}, (s_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (s'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{B},j}, \tilde{b}^{(\ell)}_{\mathsf{B},j}, \tilde{c}^{(\ell)}_{\mathsf{B},j})_{j,\ell}\right)$ from $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively:
   (a) $\mathcal{P}_\mathsf{A}$ and $\mathcal{V}$ send (init) to $\kappa$ instances of $\mathcal{F}^{\mathcal{P}_\mathsf{A},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$. Besides, $\mathcal{P}_\mathsf{B}$ and $\mathcal{V}$ send (init) to $\kappa$ instances of $\mathcal{F}^{\mathcal{P}_\mathsf{B},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$.
   (b) $\forall j$, $\mathcal{P}_\mathsf{A}$ receives outputs $\mathbf{u}_j = \left((\overline{r}_{w,j})_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}} \| (\overline{r}'_{w,j})_{w \in \mathcal{I}_\mathsf{W}} \| (\overline{a}^{(\ell)}_{\mathsf{A},j}, \overline{b}^{(\ell)}_{\mathsf{A},j}, \overline{c}^{(\ell)}_{\mathsf{A},j})_\ell\right)$ and $\mathbf{v}_j = \left((\mathsf{M}'_j[\overline{r}_{w,j}])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}} \| (\mathsf{M}'_j[\overline{r}'_{w,j}])_{w \in \mathcal{I}_\mathsf{W}} \| (\mathsf{M}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)$ from the $j$-th instance of $\mathcal{F}^{\mathcal{P}_\mathsf{A},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ which can be arranged as $(\overline{r}_{w,j}, \mathsf{M}'_j[\overline{r}_{w,j}])_{j, w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}$, $(\overline{r}'_{w,j}, \mathsf{M}'_j[\overline{r}'_{w,j}])_{j, w \in \mathcal{I}_\mathsf{W}}$, and $(\overline{a}^{(\ell)}_{\mathsf{A},j}, \mathsf{M}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}], \overline{b}^{(\ell)}_{\mathsf{A},j}, \mathsf{M}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}], \overline{c}^{(\ell)}_{\mathsf{A},j}, \mathsf{M}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}])_{j,\ell}$.
   Similarly, $\mathcal{P}_\mathsf{B}$ receives outputs from $\kappa$ instances of $\mathcal{F}^{\mathcal{P}_\mathsf{B},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ which can be arranged as $(\overline{s}_{w,j}, \mathsf{M}'_j[\overline{s}_{w,j}])_{j, w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}$, $(\overline{s}'_{w,j}, \mathsf{M}'_j[\overline{s}'_{w,j}])_{j, w \in \mathcal{I}_\mathsf{W}}$, and $(\overline{a}^{(\ell)}_{\mathsf{B},j}, \mathsf{M}'_j[\overline{a}^{(\ell)}_{\mathsf{B},j}], \overline{b}^{(\ell)}_{\mathsf{B},j}, \mathsf{M}'_j[\overline{b}^{(\ell)}_{\mathsf{B},j}], \overline{c}^{(\ell)}_{\mathsf{B},j}, \mathsf{M}'_j[\overline{c}^{(\ell)}_{\mathsf{B},j}])_{j,\ell}$.
   (c) $\forall j \in [\kappa]$ and $w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ use $\Pi_{\mathsf{fix}}$ (c.f. Figure 4) to respectively obtain $(r_w, \mathsf{M}'_j[r_w])$, from inputs $r_w$ and $(\overline{r}_{w,j}, \mathsf{M}'_j[\overline{r}_{w,j}])$, and $(s_w, \mathsf{M}'_j[s_w])$, from inputs $s_w$ and $(\overline{s}_{w,j}, \mathsf{M}'_j[\overline{s}_{w,j}])$. Doing this requires each $\hat{r}_{w,j} := \overline{r}_{w,j} \oplus r_w \ \forall j, \forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$ and $\hat{s}_{w,j} := \overline{s}_{w,j} \oplus s_w \ \forall j, \forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$ to be published.
   (d) $\forall j \in [\kappa]$ and $w \in \mathcal{I}_\mathsf{W}$, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ use $\Pi^{\mathcal{P},\mathcal{V}}_{\mathsf{fix}}$, for $\mathcal{P} \in \{\mathcal{P}_\mathsf{A}, \mathcal{P}_\mathsf{B}\}$, to respectively obtain $(r'_w, \mathsf{M}'_j[r'_w])$, from inputs $r'_w$ and $(\overline{r}'_{w,j}, \mathsf{M}'_j[\overline{r}'_{w,j}])$, and $(s'_w, \mathsf{M}'_j[s'_w])$, from inputs $s'_w$ and $(\overline{s}'_{w,j}, \mathsf{M}'_j[\overline{s}'_{w,j}])$. This requires publishing each $\hat{r}'_{w,j} := \overline{r}_{w,j} \oplus r_w \ \forall j, \forall w \in \mathcal{I}_\mathsf{W}$ and $\hat{s}'_{w,j} := \overline{s}_{w,j} \oplus s_w \ \forall j, \forall w \in \mathcal{I}_\mathsf{W}$.
   (e) $\forall j$, $\forall \ell$, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ use $\Pi^{\mathcal{P},\mathcal{V}}_{\mathsf{fix}}$, for $\mathcal{P} \in \{\mathcal{P}_\mathsf{A}, \mathcal{P}_\mathsf{B}\}$, to achieve $(\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])$ and $(\mathsf{M}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{M}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])$, respectively. Doing this requires $\hat{a}^{(\ell)}_{p,j} := \overline{a}^{(\ell)}_{p,j} \oplus \tilde{a}^{(\ell)}_{p,j}$, $\hat{b}^{(\ell)}_{p,j} := \overline{b}^{(\ell)}_{p,j} \oplus \tilde{b}^{(\ell)}_{p,j}$, and $\hat{c}^{(\ell)}_{p,j} := \overline{c}^{(\ell)}_{p,j} \oplus \tilde{c}^{(\ell)}_{p,j} \ \forall p \in \{\mathsf{A}, \mathsf{B}\}, \forall j, \forall \ell$ to be published.
2. On input (get) from $\mathcal{P}_\mathsf{A}$, $\mathcal{P}_\mathsf{B}$, and $\mathcal{V}$, $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ works as follows:
   (a) $\mathcal{P}_\mathsf{A}$ and $\mathcal{V}$ send (get) to $\kappa$ instances of $\mathcal{F}^{\mathcal{P}_\mathsf{A},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ to receive outputs which can be parsed as $(\nabla_{\mathsf{B},j})_j$ and $\left((\mathsf{K}'_j[\overline{r}_{w,j}])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\overline{r}'_{w,j}])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right)_j$.
   Similarly, $\mathcal{P}_\mathsf{A}$ and $\mathcal{V}$ send (get) to $\kappa$ instances of $\mathcal{F}^{\mathcal{P}_\mathsf{B},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ to receive outputs which can be parsed as $(\nabla_{\mathsf{A},j})_j$ and $\left((\mathsf{K}'_j[\overline{s}_{w,j}])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\overline{s}'_{w,j}])_{w \in \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\overline{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\overline{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\overline{c}^{(\ell)}_{\mathsf{B},j}])_\ell\right)_j$.
   (b) Any party can compute locally, by applying protocol $\Pi_{\mathsf{fix}}$ for all $\forall j$, as follows:

$$\begin{aligned}
\forall w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}, \quad & \mathsf{K}'_j[r_{w,j}] := \mathsf{K}'_j[\overline{r}_{w,j}] \oplus \hat{r}_{w,j} \cdot \nabla_{\mathsf{B},j}, \quad && \mathsf{K}'_j[s_{w,j}] := \mathsf{K}'_j[\overline{s}_{w,j}] \oplus \hat{s}_{w,j} \cdot \nabla_{\mathsf{A},j}; \\
\forall w \in \mathcal{I}_\mathsf{W}, \quad & \mathsf{K}'_j[r'_{w,j}] := \mathsf{K}'_j[\overline{r}'_{w,j}] \oplus \hat{r}'_{w,j} \cdot \nabla_{\mathsf{B},j}, \quad && \mathsf{K}'_j[s'_{w,j}] := \mathsf{K}'_j[\overline{s}'_{w,j}] \oplus \hat{s}'_{w,j} \cdot \nabla_{\mathsf{A},j}; \\
\forall \ell, \quad & \mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}] := \mathsf{K}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}] \oplus \hat{a}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j}, \quad && \mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}] := \mathsf{K}'_j[\overline{a}^{(\ell)}_{\mathsf{B},j}] \oplus \hat{a}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j}; \\
\forall \ell, \quad & \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}] := \mathsf{K}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}] \oplus \hat{b}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j}, \quad && \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}] := \mathsf{K}'_j[\overline{b}^{(\ell)}_{\mathsf{B},j}] \oplus \hat{b}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j}; \\
\forall \ell, \quad & \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}] := \mathsf{K}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}] \oplus \hat{c}^{(\ell)}_{\mathsf{A},j} \cdot \nabla_{\mathsf{B},j}, \quad && \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}] := \mathsf{K}'_j[\overline{c}^{(\ell)}_{\mathsf{B},j}] \oplus \hat{c}^{(\ell)}_{\mathsf{B},j} \cdot \nabla_{\mathsf{A},j}.
\end{aligned}$$

**Fig. 8.** Protocol $\Pi_{\mathsf{sVOLE\text{-}2PC}}$. Highlighted texts are those publicly revealed for public auditability purpose.

## 4.2 Protocol $\Pi_{\mathsf{sVOLE\text{-}2PC}}$

To realize $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$, we use $\mathcal{F}_{\mathsf{sVOLE}}$ (Section 2.2 and Figure 1). For a circuit that $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ interact to compute, recall that $\mathcal{I}_\mathsf{B}$ and $\mathcal{I}_\mathsf{B}$ are the sets of input wires that $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively, know the actual values. $\mathcal{I}_\mathsf{W}$ is the set of output wires of the AND gates.

For authenticating $\mathcal{P}_\mathsf{A}$'s secrets, we let $\mathcal{P}_\mathsf{A}$ and $\mathcal{V}$ access to $\mathcal{F}^{\mathcal{P}_\mathsf{A},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$, by sending (init), where $\tau$ is the bit size for IT-MAC tags and keys of VOLEitH correlations, $N = |\mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}| + W + 3L$ and $W = |\mathcal{I}_\mathsf{W}|$. Then, in the $j$-th invocation for $j \in [\kappa]$, $\mathcal{P}_\mathsf{A}$ receives as secret inputs random vectors $\mathbf{u}_j \in \mathbb{F}_2^N$ and $\mathbf{v}_j \in \mathbb{F}_{2^\tau}^N$ s.t.

$$\begin{aligned}
\mathbf{u}_j &= \left((\overline{r}_{w,j})_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}} \| (\overline{r}'_{w,j})_{w \in \mathcal{I}_\mathsf{W}} \| (\overline{a}^{(\ell)}_{\mathsf{A},j}, \overline{b}^{(\ell)}_{\mathsf{A},j}, \overline{c}^{(\ell)}_{\mathsf{A},j})_{\ell \in [L]}\right) \text{ and} \\
\mathbf{v}_j &= \left((\mathsf{M}'_j[\overline{r}_{w,j}])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}} \| (\mathsf{M}'_j[\overline{r}'_{w,j}])_{w \in \mathcal{I}_\mathsf{W}} \right. \\
&\qquad \left. \| (\mathsf{M}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{M}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}])_\ell\right),
\end{aligned}$$

respectively. In other words, $\mathbf{v}_j$ contains IT-MAC tags corresponding to those in $\mathbf{u}_j$. Then, later on, $\mathcal{P}_\mathsf{A}$ and $\mathcal{V}$ can send (get) to $\mathcal{F}^{\mathcal{P}_\mathsf{A},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ to achieve the corresponding global value $\nabla_{\mathsf{B},j}$, for $j \in [\kappa]$, and the corresponding public IT-MAC keys

$$\mathbf{w}_j = \Big( (\mathsf{K}'_j[\overline{r}_{w,j}])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[\overline{r}'_{w,j}])_{w \in \mathcal{I}_\mathsf{W}}, \\ (\mathsf{K}'_j[\overline{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\overline{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\overline{c}^{(\ell)}_{\mathsf{A},j}])_\ell \Big)_j .$$

Hence, the random secret values and IT-MAC tags (kept by $\mathcal{P}_\mathsf{A}$) and the public global value $\nabla_{\mathsf{B},j}$ and IT-MAC keys together form the VOLEitH correlations for the $N$ bits of $\mathbf{u}_j$.

By using $\Pi^{\mathcal{P}_\mathsf{A},\mathcal{V}}_{\mathsf{fix}}$ we can adapt the VOLEitH correlations, w.r.t. $\nabla_{\mathsf{B},j}$, for random values in vector $\mathbf{u}_j$ into those for $(r_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}$, $(r'_w)_{w \in \mathcal{I}_\mathsf{W}}$, and $(\tilde{a}^{(\ell)}_{\mathsf{A},j}, \tilde{b}^{(\ell)}_{\mathsf{A},j}, \tilde{c}^{(\ell)}_{\mathsf{A},j})_{\ell \in [L]}$ which are kept by $\mathcal{P}_\mathsf{A}$.

Symmetrically, we can imitate the above process for forming VOLEitH correlations for secrets of $\mathcal{P}_\mathsf{B}$. Note that, in this case, they are authenticated w.r.t. $(\nabla_{\mathsf{A},j})_{j \in [\kappa]}$.

Eventually, we describe $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ in Figure 8 in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model to realize $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ following the above discussion.

**Security.** We discuss the security of $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ in Theorem 2.

**Theorem 2 (Security of $\Pi_{\mathsf{sVOLE\text{-}2PC}}$).** *The protocol $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ in Figure 8 securely realizes $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ in Figure 7 against malicious adversaries with statistical security $2^{-\kappa \cdot \tau}$ in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model.*

**Communication Cost of $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ (c.f. Figure 8).** Notice that we need to run in total $2 \cdot \kappa$ times the functionality $\mathcal{F}^{\mathcal{P},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ where $\mathcal{P}$ is either $\mathcal{P}_\mathsf{A}$ or $\mathcal{P}_\mathsf{B}$. When realizing this functionality with $\Pi^{p,\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$ (c.f. Figure 2) and vector commitments (discussed in Section 2.2), it requires $2 \cdot \kappa$ vector commitments and openings, subsequently determining the IT-MAC keys of authenticated values. Recall the notation $\mathsf{costvc}^{\tau,N}_{\mathsf{total}}$ from Section 2.2. Hence, in total, it requires $2 \cdot \kappa \cdot (\mathsf{costvc}^{\tau,N}_{\mathsf{total}} + \tau)$ for the $2 \cdot \kappa$ invocations to $\Pi^{\mathcal{P},\mathcal{V},\tau,N}_{\mathsf{sVOLE}}$. Using protocol $\Pi^{\mathcal{P},\mathcal{V}}_{\mathsf{fix}}$ (c.f. Section 2.3), where $\mathcal{P}$ is either $\mathcal{P}_\mathsf{A}$ or $\mathcal{P}_\mathsf{B}$, for $2 \cdot \kappa \cdot N$ times costs additionally $2 \cdot \kappa \cdot N$ bits. Hence, the total bits of communication of $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ is $2 \cdot \kappa \cdot (\mathsf{costvc}^{\tau,N}_{\mathsf{total}} + \tau + N)$.

$\mathcal{F}_{\mathsf{pa\text{-}2pc}}$ runs between $\mathcal{P}_\mathsf{A}$, $\mathcal{P}_\mathsf{B}$, and a verifier $\mathcal{V}$. The common input is a circuit for a function $f : \{0,1\}^{|\mathcal{I}_\mathsf{A}|} \times \{0,1\}^{|\mathcal{I}_\mathsf{B}|} \to \{0,1\}^{|\mathcal{I}_\mathsf{O}|}$ with inputs sets $\mathcal{I}_\mathsf{A}$ and $\mathcal{I}_\mathsf{B}$, the set $\mathcal{I}_\mathsf{W}$ of output wires of AND gates, and the output set $\mathcal{I}_\mathsf{O}$.

1. Receive as inputs $\big(\mathsf{input}, (z_w)_{w \in \mathcal{I}_\mathsf{A}}, (z_w)_{w \in \mathcal{I}_\mathsf{O}}\big)$ and $\big(\mathsf{input}, (z_w)_{w \in \mathcal{I}_\mathsf{B}}, (z_w)_{w \in O}\big)$ from $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively.
2. Upon receiving $(\mathsf{prove})$ from all three parties $\mathcal{P}_\mathsf{B}$, $\mathcal{P}_\mathsf{B}$, and $\mathcal{V}$, check:
   - Matching outputs: $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ share the same $(z_w)_{w \in \mathcal{I}_\mathsf{O}}$ from the inputs of both parties.
   - Correct computation of $f((z_w)_{w \in \mathcal{I}_\mathsf{A}} \| (z_w)_{w \in \mathcal{I}_\mathsf{B}}) = (z_w)_{w \in \mathcal{I}_\mathsf{O}}$.

   Returns $\mathsf{true}$ to $\mathcal{V}$ if all constraints hold and $\mathsf{false}$ otherwise.

**Fig. 9.** Functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$.

## 5 Publicly Auditable 2PC

We now present the public auditable 2PC protocol $\Pi_{\mathsf{pa\text{-}2PC}}$, in Section 5.2 following the technical overview in Section 3.2. This protocol realizes the functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$ in Fig. 9 (c.f. Section 5.1).

### 5.1 Functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$

In the functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$, there are two involved parties, namely, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, for evaluating a garbled circuit. Then, we have an additional party, namely, verifier $\mathcal{V}$, for auditing the private evaluation, conducted by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, of a function $f$. This function $f$ can be realized by a Boolean circuit whose input wires are captured by the set $\mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B}$ such that $\mathcal{I}_\mathsf{A}$ and $\mathcal{I}_\mathsf{B}$ are the sets of input wires of the circuit, whose actual values are known by $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively. The set $\mathcal{I}_\mathsf{O}$ contains all output wires of the circuit.

We assume the output of $f$, captured by the wires in $\mathcal{I}_\mathsf{O}$, from the evaluations of $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ are public. Therefore, both parties keep the same set of actual values of those in the output set $\mathcal{I}_\mathsf{O}$. The verifier $\mathcal{V}$ accepts if she believes that the evaluation, by both $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, are correct w.r.t. $f$.

**Parties:** $\mathcal{P}_\mathsf{A}$, $\mathcal{P}_\mathsf{B}$, and a verifier $\mathcal{V}$.
**Inputs:** Both parties $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ agree on a common circuit for function $f : \{0,1\}^{|\mathcal{I}_\mathsf{A}|} \times \{0,1\}^{|\mathcal{I}_\mathsf{B}|} \to \{0,1\}^{|\mathcal{I}_\mathsf{O}|}$ specified by the sets $\mathcal{I}_\mathsf{A}$, $\mathcal{I}_\mathsf{B}$, $\mathcal{I}_\mathsf{W}$, and $\mathcal{I}_\mathsf{O}$. $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ also keep common parameters $\rho, \kappa, \tau, \mathsf{bs}, \mathsf{rm} \in \mathbb{N}$ and $L = W \cdot \mathsf{bs} + \mathsf{rm}$, and agree to a commitment scheme $\mathcal{COM}$ (c.f. Appendix A).
**Remark:** Denoting $j$, $i$, and $\ell$ as subscripts of sets/sequences or after $\forall$ without specifying the ranges means $j \in [\kappa]$, $i \in [W]$, and $\ell \in [L]$, respectively.
**Function-Independent Preprocessing:**

1. $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ send $(\mathsf{init})$ to $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ to receives $\Delta_\mathsf{A} \in \mathbb{F}_{2^\rho}$ and $\Delta_\mathsf{B} \in \mathbb{F}_{2^\rho}$, respectively.
2. For each $w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}$, both $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ send $(\mathsf{random})$ to $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$. In return, $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ sends $(r_w, \mathsf{M}[r_w], \mathsf{K}[s_w]) \in \mathbb{F}_2 \times \mathbb{F}_{2^\rho} \times \mathbb{F}_{2^\rho}$ and $(s_w, \mathsf{M}[s_w], \mathsf{K}[r_w]) \in \mathbb{F}_2 \times \mathbb{F}_{2^\rho} \times \mathbb{F}_{2^\rho}$ to $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, respectively. Define $\lambda_w := r_w \oplus s_w$. $\mathcal{P}_\mathsf{A}$ picks a $\mathsf{L}_{w,0} \xleftarrow{\$} \mathbb{F}_{2^\rho}$.
3. $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ invoke $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ $L \cdot \kappa$ times to obtain $(\tilde{a}_{\mathsf{A},j}^{(\ell)}, \tilde{b}_{\mathsf{A},j}^{(\ell)}, \tilde{c}_{\mathsf{A},j}^{(\ell)})$ and $(\tilde{a}_{\mathsf{B},j}^{(\ell)}, \tilde{b}_{\mathsf{B},j}^{(\ell)}, \tilde{c}_{\mathsf{B},j}^{(\ell)})$, respectively, s.t. $(\tilde{a}_{\mathsf{A},j}^{(\ell)} \oplus \tilde{a}_{\mathsf{B},j}^{(\ell)}) \cdot (\tilde{b}_{\mathsf{A},j}^{(\ell)} \oplus \tilde{b}_{\mathsf{B},j}^{(\ell)}) = \tilde{c}_{\mathsf{A},j}^{(\ell)} \oplus \tilde{c}_{\mathsf{B},j}^{(\ell)}$ $\forall \ell, \forall j$.

**Function-Dependent Preprocessing:**

4. Following the topological ordering of the circuit, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ work as follows.
   - For each XOR gate $(\alpha, \beta, \gamma, \oplus)$, $\mathcal{P}_\mathsf{A}$ computes $(r_\gamma, \mathsf{M}[r_\gamma], \mathsf{K}[s_\gamma]) := (r_\alpha \oplus r_\beta, \mathsf{M}[r_\alpha] \oplus \mathsf{M}[r_\beta], \mathsf{K}[s_\alpha] \oplus \mathsf{K}[s_\beta])$ and $\mathsf{L}_{\gamma,0} := \mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0}$. $\mathcal{P}_\mathsf{B}$ computes $(s_\gamma, \mathsf{M}[s_\gamma], \mathsf{K}[r_\gamma]) := (s_\alpha \oplus s_\beta, \mathsf{M}[s_\alpha] \oplus \mathsf{M}[s_\beta], \mathsf{K}[r_\alpha] \oplus \mathsf{K}[r_\beta])$.
   - For each AND gate $(\alpha, \beta, \gamma, \cdot)$:
     (a) $\mathcal{P}_\mathsf{A}$ (resp., $\mathcal{P}_\mathsf{B}$) sends $(\mathsf{and}, (r_\alpha, \mathsf{M}[r_\alpha], \mathsf{K}[s_\alpha]), (r_\beta, \mathsf{M}[r_\beta], \mathsf{K}[s_\beta]))$ (resp., $(\mathsf{and}, (s_\alpha, \mathsf{M}[s_\alpha], \mathsf{K}[r_\alpha]), (s_\beta, \mathsf{M}[s_\beta], \mathsf{K}[r_\beta]))$) to $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$. In return, $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ sends $(r'_\gamma, \mathsf{M}[r'_\gamma], \mathsf{K}[s'_\gamma])$ and $(s'_\gamma, \mathsf{M}[s'_\gamma], \mathsf{K}[r'_\gamma])$ respectively to $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$. Here, $r'_\gamma \oplus s'_\gamma = \lambda_\alpha \cdot \lambda_\beta$.
     (b) $\forall k \in [0,3]$, let $(k_0, k_1) = \mathsf{bin}_2(k)$, and $\mathcal{P}_\mathsf{A}$ computes $r_{\gamma,k} := r'_\gamma \oplus r_\gamma \oplus k_0 \cdot r_\beta \oplus k_1 \cdot r_\alpha$,
     $$\mathsf{M}[r_{\gamma,k}] := \mathsf{M}[r'_\gamma] \oplus \mathsf{M}[r_\gamma] \oplus k_0 \cdot \mathsf{M}[r_\beta] \oplus k_1 \cdot \mathsf{M}[r_\alpha], \quad \mathsf{K}[s_{\gamma,k}] := \mathsf{K}[s'_\gamma] \oplus \mathsf{K}[s_\gamma] \oplus k_0 \cdot \mathsf{K}[s_\beta] \oplus k_1 \cdot \mathsf{K}[s_\alpha] \oplus k_0 \cdot k_1 \cdot \Delta_\mathsf{A}.$$
     (c) $\forall k \in [0,3]$, let $(k_0, k_1) = \mathsf{bin}_2(k)$, and $\mathcal{P}_\mathsf{B}$ computes $s_{\gamma,k} := s'_\gamma \oplus s_\gamma \oplus k_0 \cdot s_\beta \oplus k_1 \cdot s_\alpha \oplus k_0 \cdot k_1$,
     $$\mathsf{M}[s_{\gamma,k}] := \mathsf{M}[s'_\gamma] \oplus \mathsf{M}[s_\gamma] \oplus k_0 \cdot \mathsf{M}[s_\beta] \oplus k_1 \cdot \mathsf{M}[s_\alpha], \quad \mathsf{K}[r_{\gamma,k}] := \mathsf{K}[r'_\gamma] \oplus \mathsf{K}[r_\gamma] \oplus k_0 \cdot \mathsf{K}[r_\beta] \oplus k_1 \cdot \mathsf{K}[r_\alpha].$$
5. *Obtaining IT-MAC Tags for VOLEitH:* $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ respectively send $\left(\mathsf{init}, (r_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (r'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}_{\mathsf{A},j}^{(\ell)}, \tilde{b}_{\mathsf{A},j}^{(\ell)}, \tilde{c}_{\mathsf{A},j}^{(\ell)})_{j \in [\kappa], \ell \in [L]}\right)$ and $\left(\mathsf{init}, (s_w)_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (s'_w)_{w \in \mathcal{I}_\mathsf{W}}, (\tilde{a}_{\mathsf{B},j}^{(\ell)}, \tilde{b}_{\mathsf{B},j}^{(\ell)}, \tilde{c}_{\mathsf{B},j}^{(\ell)})_{j \in [\kappa], \ell \in [L]}\right)$ to $\mathcal{F}_\mathsf{sVOLE\text{-}2PC}$ (c.f. Figure 7). Then, $\mathcal{P}_\mathsf{A}$ (resp., $\mathcal{P}_\mathsf{B}$) receives $(\mathsf{M}'_j[r_w])_{j \in [\kappa], w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}$, $(\mathsf{M}'_j[r'_w])_{j \in [\kappa], w \in \mathcal{I}_\mathsf{W}}$, $(\mathsf{M}'_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}], \mathsf{M}'_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}], \mathsf{M}'_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}])_{j \in [\kappa], \ell \in [L]}$ (resp., $(\mathsf{M}'_j[s_w])_{j \in [\kappa], w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}$, $(\mathsf{M}'_j[s'_w])_{j \in [\kappa], w \in \mathcal{I}_\mathsf{W}}$, $(\mathsf{M}'_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}], \mathsf{M}'_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}], \mathsf{M}'_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}])_{j \in [\kappa], \ell \in [L]}$).
6. *Making VOLEitH Proof and Components for Garbled Tables:*
   - For each XOR gate $(\alpha, \beta, \gamma, \oplus)$, for $j \in [\kappa]$, $\mathcal{P}_\mathsf{A}$ (resp., $\mathcal{P}_\mathsf{B}$) computes $\mathsf{M}'_j[r_\gamma] := \mathsf{M}'_j[r_\alpha] \oplus \mathsf{M}'_j[r_\beta]$ (resp., $\mathsf{M}'_j[s_\gamma] := \mathsf{M}'_j[s_\alpha] \oplus \mathsf{M}'_j[s_\beta]$).
   - For each AND gate $(\alpha, \beta, \gamma, \cdot)$:
     (a) $\forall k \in [0,3]$, $\forall j \in [\kappa]$, let $(k_0, k_1) = \mathsf{bin}_2(k)$, and $\mathcal{P}_\mathsf{A}$ computes $\mathsf{M}'_j[r_{\gamma,k}] := \mathsf{M}'_j[r'_\gamma] \oplus \mathsf{M}'_j[r_\gamma] \oplus k_0 \cdot \mathsf{M}'_j[r_\beta] \oplus k_1 \cdot \mathsf{M}'_j[r_\alpha]$.
     (b) $\forall k \in [0,3]$, $\forall j \in [\kappa]$, let $(k_0, k_1) = \mathsf{bin}_2(k)$, and $\mathcal{P}_\mathsf{B}$ computes $\mathsf{M}'_j[s_{\gamma,k}] := \mathsf{M}'_j[s'_\gamma] \oplus \mathsf{M}'_j[s_\gamma] \oplus k_0 \cdot \mathsf{M}'_j[s_\beta] \oplus k_1 \cdot \mathsf{M}'_j[s_\alpha]$.
7. *Encrypting and Committing:* For each AND gate $(\alpha, \beta, \gamma, \cdot)$,
   - $\mathcal{P}_\mathsf{A}$ computes $\mathsf{L}_{\alpha,1} := \mathsf{L}_{\alpha,0} \oplus \Delta_\mathsf{A}$ and $\mathsf{L}_{\beta,1} := \mathsf{L}_{\beta,0} \oplus \Delta_\mathsf{A}$. $\mathcal{P}_\mathsf{A}$ publishes the following, for all $k \in [0,3]$ with $(k_0, k_1) = \mathsf{bin}_2(k)$:
   $$G_{\gamma,k} := H(\mathsf{L}_{\alpha,k_0}, \mathsf{L}_{\beta,k_1}, \gamma, k) \oplus (r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], (\mathsf{M}'_j[r_{\gamma,k}])_{j \in [\kappa]}, \mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A}).$$
   - $\forall k \in [0,3]$, $\mathcal{P}_\mathsf{B}$ samples randomness $\mathsf{rand}_{\gamma,k}$ and commits to $(s_{\gamma,k}, (\mathsf{M}'_j[s_{\gamma,k}])_{j \in [\kappa]})$ via $\mathcal{COM}$, by using $\mathsf{rand}_{\gamma,k}$, to obtain and publish commitment $\mathsf{cm}_{\mathsf{B},\gamma,k}$.

**Fig. 10.** Protocol $\Pi_\mathsf{pa\text{-}2PC\text{-}pre}$ for Preprocessing for Publicly Auditable 2PC.

## 5.2 Protocol $\Pi_\mathsf{pa\text{-}2PC}$

We present the sub-protocols $\Pi_\mathsf{pa\text{-}2PC\text{-}pre}$ (for preprocessing) and $\Pi_\mathsf{pa\text{-}2PC\text{-}eval}$ (for evaluating) in Figures 10 and 11, respectively. They employ $\Pi_\mathsf{check\text{-}AND}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}$ (c.f. Figure 6) as a subroutine for executing and in the $\mathcal{F}_\mathsf{pre}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$- and $\mathcal{F}_\mathsf{sVOLE\text{-}2PC}$-hybrid model. We denote by $\Pi_\mathsf{pa\text{-}2PC}$ the concatenation of $\Pi_\mathsf{pa\text{-}2PC\text{-}pre}$ and $\Pi_\mathsf{pa\text{-}2PC\text{-}eval}$, i.e.,

$$\Pi_\mathsf{pa\text{-}2PC} = (\Pi_\mathsf{pa\text{-}2PC\text{-}pre}, \Pi_\mathsf{pa\text{-}2PC\text{-}eval}).$$

$\mathit{\Pi}_{\mathsf{pa\text{-}2PC}}$ hence realizes functionality $\mathcal{F}_{\mathsf{pa\text{-}2pc}}$ described in Section 5.1. Assume that the function $f$ is specified by the sets $\mathcal{I}_\mathsf{A}$, $\mathcal{I}_\mathsf{B}$, $\mathcal{I}_\mathsf{O}$, and $\mathcal{I}_\mathsf{W}$ as specified in Section 4. The description of $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}pre}}$ and $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}eval}}$ follows those discussed in Section 3.2.

**Sub-Protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}pre}}$ (c.f. Figure 10).** This sub-protocol is designed for both $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ to communicate in advance before knowing the actual values of the input wires.

Initially, similarly to WRK protocol [31], $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ have access to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ (recalled in Figure 5) to obtain the random values as well as the VOLE correlation. Specifically, as in steps 1, 2, and 3, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ access to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ to obtain $\mathit{\Delta}_\mathsf{A}$ and $\mathit{\Delta}_\mathsf{B}$, respectively, the randomness $(r_w)_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}$ and $(s_w)_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}$, respectively, as well as their corresponding VOLE correlations w.r.t. $\mathit{\Delta}_\mathsf{B}$ and $\mathit{\Delta}_\mathsf{A}$, respective. Here, $\mathcal{P}_\mathsf{A}$ prepares the labels at all $w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}$ to prepare for garbling. Both $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ also obtain the random shared AND triples $(\tilde{a}_{\mathsf{A},j}^{(i)},\tilde{b}_{\mathsf{A},j}^{(i)},\tilde{c}_{\mathsf{A},j}^{(i)})_{j\in[\kappa],i\in[W]}$ and $(\tilde{a}_{\mathsf{B},j}^{(i)},\tilde{b}_{\mathsf{B},j}^{(i)},\tilde{c}_{\mathsf{B},j}^{(i)})_{j\in[\kappa],i\in[W]}$, respectively, where $W=|\mathcal{I}_\mathsf{W}|$.

Then, similarly to WRK protocol [31], $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ prepare for garbling as specfied in step 4. Notice that $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ additionally access to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}}$ to obtain $(r'_\gamma)_{(\alpha,\beta,\gamma,\cdot)}$ and $(s'_\gamma)_{(\alpha,\beta,\gamma,\cdot)}$, respectively, and their corresponding VOLE correlations w.r.t. $\mathit{\Delta}_\mathsf{B}$ and $\mathit{\Delta}_\mathsf{A}$, respectively, such that $r'_\gamma\oplus s'_\gamma=(r_\alpha\oplus s_\alpha)\cdot(r_\beta\oplus s_\beta)$. Then, the parties employ $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ to generate IT-MAC tags for VOLEitH correlations for those secrets mentioned above as in step 5. Additionally, in step 6, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ respectively compute IT-MAC tags for those values computed in step 4.

In step 7, $\mathcal{P}_\mathsf{A}$ then follows WRK protocol [31] to encrypt the authenticated garbled table. However, in this case, $\mathcal{P}_\mathsf{A}$ must additionally commit to $(\mathsf{M}'_j[r_{\gamma,k}])_{j\in[\kappa]}$ as shown in step 7. The purpose is to enable public auditability via VOLEitH the correct use of $r_{\gamma,k}$. On the other hand, to allow public auditability by VOLEitH, for each AND gate, $\mathcal{P}_\mathsf{B}$ must commit to every $(s_{\gamma,k},(\mathsf{M}'_j[s_{\gamma,k}])_{j\in[\kappa]})$ by using randomness $\mathsf{rand}_{\gamma,k}$ for $k\in[0,3]$. We can do so as follows: (i) In protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}eval}}$, $\mathcal{P}_\mathsf{B}$ needs to use one among the four values (w.r.t. $k\in[0,3]$) to evaluate the garbled table, $\mathcal{P}_\mathsf{B}$ hence reveals the selected value as well as the IT-MAC tag for auditing purpose by VOLEitH. One way is to publish all those mentioned values (without committing to them). However, revealing all those values may help $\mathcal{P}_\mathsf{B}$ to recover every row in the garbled table. Therefore, committing to them does help to protect $\mathcal{P}_\mathsf{B}$'s secrets. (We defer to Appendix A for a preliminary of commitment schemes.) By the hiding property of commitment schemes, $\mathcal{P}_\mathsf{A}$ is unable to recover $\mathcal{P}_\mathsf{B}$'s secrets; (ii) When evaluating, at each AND gate, $\mathcal{P}_\mathsf{B}$ only needs to open (by revealing the corresponding randomness) the commitment to the value that helps evaluate the garbled table while skipping the other three commitments. The binding property of the commitment scheme guarantees that $\mathcal{P}_\mathsf{B}$ cannot change the value behind the selected commitment.

**Sub-Protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}eval}}$ (c.f. Figure 11).** When both $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ learn their corresponding values w.r.t. wires in the sets $\mathcal{I}_\mathsf{A}$ and $\mathcal{I}_\mathsf{B}$, respectively, they, together with $\mathcal{V}$, run protocol $\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}eval}}$ to evaluate and allow $\mathcal{V}$ to audit the evaluation without leaking secrets of any party except the output of the circuit realizing $f$.

In **proof preparation**, the parties follow the strategy in technical overview (c.f. Section 3.2) to use balls-and-bins method for checking AND gates. However, in this step, the parties have not had $(\nabla_{p,j})_{p\in\{\mathsf{A},\mathsf{B}\},j\in[\kappa]}$ and the IT-MAC keys. Therefore, they only manipulate the values and the IT-MAC tags. In **input processing**, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ the actual values $(z_w)_{w\in\mathcal{I}_\mathsf{A}}$ and $(z_w)_{w\in\mathcal{I}_\mathsf{B}}$, repsectively, corresponding to both parties' input sets $\mathcal{I}_\mathsf{A}$ and $\mathcal{I}_\mathsf{B}$, respectively. In this phase, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ follow WRK protocol to send and check the authenticated IT-MAC tags w.r.t. $\mathit{\Delta}_\mathsf{A}$ and $\mathit{\Delta}_\mathsf{B}$. Moreover, they also need to send those IT-MAC tags for VOLEitH correlations to allow $\mathcal{V}$ to audit. In **cicruit evaluation**, $\mathcal{P}_\mathsf{B}$ follows WRK protocol [31] to decrypt the rows in the garbled tables. Notice that, at each AND gate, since previously $\mathcal{P}_\mathsf{A}$ also encrypts the IT-MAC tags for VOLEitH correlations at each garbled table corresponding to each AND gate, the decryption from $\mathcal{P}_\mathsf{B}$ hence achieves such IT-MAC tags (say, $\mathsf{M}'_j[r_{\gamma,k}]$ for $k=\hat{z}_\alpha+2\cdot\hat{z}_\beta$). On the other hand, in step 7, $\mathcal{P}_\mathsf{B}$ commits to $(s_{\gamma,k},(\mathsf{M}'_j[s_{\gamma,k}])_{j\in[\kappa]})$ for $k=\hat{z}_\alpha+2\cdot\hat{z}_\beta$. Hence, $\mathcal{P}_\mathsf{B}$ simply opens by revealing the corresponding randomness $\mathsf{rand}_{\gamma,k}$. In **output determination**, $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ determine the output values following WRK protocol [31]. In **proof proceeding**, the parties obtain $(\nabla_{\mathsf{A},j})_{k\in[\kappa]}$, $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$, and the IT-MAC keys for VOLEitH from $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ and proceed to check the published IT-MAC tags for VOLEitH obtained above. Notice that, in steps 11 and 13, the parties check the AND gates w.r.t. balls-and-bins method [3,32] as discussed in Section 3.2.

**Theorem 3 (Security of $\mathit{\Pi}_{\mathsf{pa\text{-}2PC}}$).** *If $H$ is modeled as a random oracle and $\mathcal{COM}$ is secure (see Appendix A for a definition of commitment schmes), the combined protocol*

$$\mathit{\Pi}_{\mathsf{pa\text{-}2PC}}=(\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}pre}},\mathit{\Pi}_{\mathsf{pa\text{-}2PC\text{-}eval}})$$

*(c.f. Figures 10 and 11), securely computes $f$ against malicious adversaries with statistical security $2^{-\rho}$ and soundness error*

$$2^{-\kappa\cdot\tau} + \binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa} + \mathsf{negl}(\kappa,\tau)$$

*in the $\mathcal{F}_{\mathsf{pre}}$- and $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$-hybrid models where $\mathsf{negl}(\kappa,\tau)$ denotes the negligible probability for breaking the security of components in the protocol, e.g., commitments.*

The proof of Theorem 3 is deferred to Appendix C.

**Removing $\mathcal{V}$.** Notice that, although $\Delta_{\mathsf{A}}$ and $\Delta_{\mathsf{B}}$ are not revealed, as explained in Section 3.2, we can verify authenticity of all shared masks via $\{\nabla_{\mathsf{A},j},\nabla_{\mathsf{B},j}\}_{j\in[\kappa]}$ by following the paradigm from [7]. Since $\{\nabla_{\mathsf{A},j},\nabla_{\mathsf{B},j}\}_{j\in[\kappa]}$ are generated via $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ and $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ securely realizes $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$, we can remove $\mathcal{V}$ by generating the global values $\{\nabla_{\mathsf{A},j},\nabla_{\mathsf{B},j}\}_{j\in[\kappa]}$ via some random oracle (Fiat-Shamir paradigm [22]) since $\Pi_{\mathsf{sVOLE\text{-}2PC}}$ is public-coin. Moreover, we also can generate $(\pi_j)_{j\in\kappa}$ via random oracles. Hence, we can transform the entire transcript of $\Pi_{\mathsf{pa\text{-}2PC}} = (\Pi_{\mathsf{pa\text{-}2PC\text{-}pre}},\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}})$ into a non-interactive ZKP which is the tuple $\Sigma_{\mathsf{FS}} = (\mathrm{CMT},\mathrm{RSP}_1,\mathrm{RSP}_2)$ in ROM as specified in Figure 12 while $\mathrm{CH}_1$ and $\mathrm{CH}_2$ are generated by the random oracles. Hence, we only require interaction between $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ for generating the NI-ZKP proof. Eventually, we denote by $\Pi^{\mathsf{FS}}_{\mathsf{pa\text{-}2PC}}$ the version of $\Pi_{\mathsf{pa\text{-}2PC}}$ obtained from replacing $\mathcal{V}$'s challenges by those generated from random oracles, following the above-discussed Fiat-Shamir transformation.

*Number of Rounds of $\Pi^{\mathsf{FS}}_{\mathsf{pa\text{-}2PC}}$.* Since we apply Fiat-Shamir, generating VOLEitH correlations can be done locally in each party. The interactions involved are those for running authenticated garbling w.r.t. $\Delta_{\mathsf{A}}$ and $\Delta_{\mathsf{B}}$ following the paradigm from WRK protocol [31]. Notice that the components (IT-MAC tags and keys) of VOLEitH correlations are usually published together with messages sent from the underlying WRK protocol except that we need an additional round for $\mathcal{P}_{\mathsf{B}}$ to publish $\mathsf{cm}_{\mathsf{B},\gamma,k}$ in step 7 of $\Pi_{\mathsf{pa\text{-}2PC\text{-}pre}}$. Hence, the total number of rounds of $\Pi^{\mathsf{FS}}_{\mathsf{pa\text{-}2PC}}$ is that of WRK protocol plus 1.

*Transcript Size.* We refer to Figures 10, 11, and 12 for notations. The size of CMT is $2\cdot\kappa\cdot\mathsf{costvc}^{\tau,N}_{\mathsf{com}} + 4\cdot W\cdot(1+\rho+\kappa\cdot\tau+|\mathsf{cm}|)$ where $\mathsf{costvc}^{\tau,N}_{\mathsf{com}}$ is introduced in Section 4.2 and $|\mathsf{cm}|$ is the size of $\mathsf{cm}_{\mathsf{B},w,k}$. The size in bits of $\mathrm{RSP}_1$ (where $|\mathsf{rand}|$ is the size of each $\mathsf{rand}_{\gamma,k}$) is

$$\begin{aligned}&2\cdot\kappa\cdot\mathsf{rm}\cdot(3+3\cdot\tau)+\mathsf{bs}\cdot\underbrace{6\cdot\kappa\cdot W\cdot(2\cdot\tau+1)}_{|\mathsf{msgand}_k|}\\&+|\mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}|\cdot(2+2\cdot\rho+\kappa\cdot\tau)+W\cdot(2+3\cdot\rho+2\cdot\kappa\cdot\tau+|\mathsf{rand}|)\\&+|\mathcal{I}_{\mathsf{O}}|\cdot(3+\rho+2\cdot\kappa\cdot\tau).\end{aligned}$$

The size of $\mathrm{RSP}_2$ is $2\cdot\kappa\cdot\mathsf{costvc}^{\tau,N}_{\mathsf{open}}$ bits where $\mathsf{costvc}^{\tau,N}_{\mathsf{open}}$ is introduced in Section 4.2. Hence, to simplify, the size of $\Sigma_{\mathsf{FS}}$ is bounded by

$$\begin{aligned}\mathcal{O}\Big(&\kappa\cdot\mathsf{costvc}^{\tau,N}_{\mathsf{total}}+|\mathsf{cm}|\cdot W+\kappa\cdot\tau\cdot(W\cdot\mathsf{bs}+\mathsf{rm})\\&+(\rho+\kappa\cdot\tau)\cdot(|\mathcal{I}_{\mathsf{A}}|+|\mathcal{I}_{\mathsf{B}}|+W+|\mathcal{I}_{\mathsf{O}}|)\Big)\end{aligned}$$

where $\mathsf{costvc}^{\tau,N}_{\mathsf{total}} = \mathsf{costvc}^{\tau,N}_{\mathsf{com}} + \mathsf{costvc}^{\tau,N}_{\mathsf{open}}$ as defined in Section 2.2.

## Acknowledgments

The work of San Ling, Khai Hanh Tang, and Huaxiong Wang was supported by Singapore Ministry of Education Academic Research Fund Tier 2 (Grant MOE-000623-00) and Ethereum Support Program (Grant FY24-1414-A).

## References

1. Aguilar-Melchor, C., Gama, N., Howe, J., Hülsing, A., Joseph, D., Yue, D.: The Return of the SDitH. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. Lecture Notes in Computer Science, vol. 14008, pp. 564–596. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-30589-4_20
2. Aguilar-Melchor, C., Hülsing, A., Joseph, D., Majenz, C., Ronen, E., Yue, D.: SDitH in the QROM. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. Lecture Notes in Computer Science, vol. 14444, pp. 317–350. Springer Nature Singapore (2023). https://doi.org/10.1007/978-981-99-8739-9_11

3. Araki, T., Barak, A., Furukawa, J., Lichter, T., Lindell, Y., Nof, A., Ohara, K., Watzman, A., Weinstein, O.: Optimized Honest-Majority MPC for Malicious Adversaries — Breaking the 1 Billion-Gate Per Second Barrier. In: 2017 IEEE Symposium on Security and Privacy – S&P 2017. pp. 843–862. IEEE (2017). https://doi.org/10.1109/SP.2017.15
4. Baldimtsi, F., Kiayias, A., Zacharias, T., Zhang, B.: Crowd Verifiable Zero-Knowledge and End-to-End Verifiable Multiparty Computation. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. Lecture Notes in Computer Science, vol. 12493, pp. 717–748. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-64840-4_24
5. Baum, C., Braun, L., Munch-Hansen, A., Razet, B., Scholl, P.: Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security – CCS 2021. p. 192–211. Association for Computing Machinery (2021). https://doi.org/10.1145/3460120.3484812
6. Baum, C., Braun, L., Munch-Hansen, A., Scholl, P.: Moz$\mathbb{Z}_{2^k}$arella: Efficient Vector-OLE and Zero-Knowledge Proofs over $\mathbb{Z}_{2^k}$. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13510, pp. 329–358. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-15985-5_12
7. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14085, pp. 581–615. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-38554-4_19
8. Baum, C., Damgård, I., Orlandi, C.: Publicly Auditable Secure Multi-Party Computation. In: Abdalla, M., De Prisco, R. (eds.) Security and Cryptography for Networks – SCN 2014. Lecture Notes in Computer Science, vol. 8642, pp. 175–196. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-10879-7_11
9. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac′n′Cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. Lecture Notes in Computer Science, vol. 12828, pp. 92–122. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-84259-8_4
10. Baum, C., Nof, A.: Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography – PKC 2020. Lecture Notes in Computer Science, vol. 12110, pp. 495–526. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-45374-9_17
11. Baum, C., de Saint Guilhem, C.D., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: Short and Fast Signatures from AES. In: Garay, J.A. (ed.) Public-Key Cryptography – PKC 2021. Lecture Notes in Computer Science, vol. 12710, pp. 266–297. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-75245-3_11
12. Bui, D.: Shorter VOLEitH Signature from Multivariate Quadratic. Cryptology ePrint Archive, Paper 2024/465 (2024), https://eprint.iacr.org/2024/465
13. Bui, D., Carozza, E., Couteau, G., Goudarzi, D., Joux, A.: Faster Signatures from MPC-in-the-Head. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024. Lecture Notes in Computer Science, vol. 15484, pp. 396–428. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-0875-1_13
14. Carozza, E., Couteau, G., Joux, A.: Short Signatures from Regular Syndrome Decoding in the Head. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. Lecture Notes in Computer Science, vol. 14008, pp. 532–563. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-30589-4_19
15. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS 2017. p. 1825–1842. Association for Computing Machinery (2017). https://doi.org/10.1145/3133956.3133997
16. Cui, H., Liu, H., Yan, D., Yang, K., Yu, Y., Zhang, K.: ReSolveD: Shorter Signatures from Regular Syndrome Decoding and VOLE-in-the-Head. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. Lecture Notes in Computer Science, vol. 14601, pp. 229–258. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-57718-5_8
17. Cui, H., Wang, X., Yang, K., Yu, Y.: Actively Secure Half-Gates with Minimum Overhead Under Duplex Networks. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. Lecture Notes in Computer Science, vol. 14005, pp. 35–67. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-30617-4_2
18. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Improving Line-Point Zero Knowledge: Two Multiplications for the Price of One. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security – CCS 2022. p. 829–841. Association for Computing Machinery (2022). https://doi.org/10.1145/3548606.3559385

19. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-Point Zero Knowledge and Its Applications. In: Tessaro, S. (ed.) 2nd Conference on Information-Theoretic Cryptography – ITC 2021. Leibniz International Proceedings in Informatics (LIPIcs), vol. 199. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.ITC.2021.5
20. Evans, D., Kolesnikov, V., Rosulek, M.: A Pragmatic Introduction to Secure Multi-Party Computation. Found. Trends Priv. Secur. **2**(2-3), 70–246 (2018). https://doi.org/10.1561/3300000019
21. Feneuil, T., Joux, A., Rivain, M.: Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13508, pp. 541–572. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-15979-4_19
22. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer Berlin Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
23. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing – STOC 2007. p. 21–30. Association for Computing Machinery (2007). https://doi.org/10.1145/1250790.1250794
24. Jain, A., Krenn, S., Pietrzak, K., Tentes, A.: Commitments and Efficient Zero-Knowledge Proofs from Learning Parity with Noise. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 663–680. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_40
25. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) Advances in Cryptology - EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 97–114. Springer Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_6
26. Katz, J., Kolesnikov, V., Wang, X.: Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security – CCS 2018. p. 525–537. Association for Computing Machinery (2018). https://doi.org/10.1145/3243734.3243805
27. Katz, J., Ranellucci, S., Rosulek, M., Wang, X.: Optimizing authenticated garbling for faster secure two-party computation. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. Lecture Notes in Computer Science, vol. 10993, pp. 365–391. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-96878-0_13
28. Ozdemir, A., Boneh, D.: Experimenting with Collaborative zk-SNARKs: Zero-Knowledge Proofs for Distributed Secrets. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium – USENIX Security 2022. pp. 4291–4308. USENIX Association (2022), https://www.usenix.org/conference/usenixsecurity22/presentation/ozdemir
29. Roy, L.: SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13507, pp. 657–687. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-15802-5_23
30. Delpech de Saint Guilhem, C., Orsini, E., Tanguy, T.: Limbo: Efficient Zero-knowledge MPCitH-based Arguments. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security – CCS 2021. p. 3022–3036. Association for Computing Machinery (2021). https://doi.org/10.1145/3460120.3484595
31. Wang, X., Ranellucci, S., Katz, J.: Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS 2017. p. 21–37. Association for Computing Machinery (2017). https://doi.org/10.1145/3133956.3134053
32. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits. In: 2021 IEEE Symposium on Security and Privacy – S&P 2021. pp. 1074–1091. IEEE (2021). https://doi.org/10.1109/SP40001.2021.00056
33. Weng, C., Yang, K., Xie, X., Katz, J., Wang, X.: Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning. In: Bailey, M.D., Greenstadt, R. (eds.) 30th USENIX Security Symposium – USENIX Security 2021. USENIX Association (2021), https://www.usenix.org/conference/usenixsecurity21/presentation/weng
34. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: AntMan: Interactive Zero-Knowledge Proofs with Sublinear Communication. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security – CCS 2022. p. 2901–2914. Association for Computing Machinery (2022). https://doi.org/10.1145/3548606.3560667
35. Yang, K., Sarkar, P., Weng, C., Wang, X.: QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security – CCS 2021. p. 2986–3001. Association for Computing Machinery (2021). https://doi.org/10.1145/3460120.3484556

## A Commitment Schemes (Preliminary)

We recall the definition of commitment schemes (adapted from [24]) in the following Definition 1.

**Definition 1.** *A commitment scheme $\mathcal{COM}$ is a triple of algorithms $\mathcal{COM} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Verify})$ defined as follows:*

$\mathsf{KeyGen}(1^\lambda) \to \mathsf{ck}$: *On input the security parameter $1^\lambda$, this randomized algorithm returns a commitment key* $\mathsf{ck}$.

$\mathsf{Com}(\mathsf{ck}, M) \to (\mathsf{cm}, \mathsf{rand})$: *On inputs a commitment key and a message $M$, this algorithm returns a commitment* $\mathsf{cm}$ *to $M$ and a randomness* $\mathsf{rand}$.

$\mathsf{Verify}(\mathsf{ck}, M, \mathsf{cm}, \mathsf{rand}) \to \{0, 1\}$: *This is the verification algorithm to verify whether* $\mathsf{cm}$ *is a valid commitment to $M$ w.r.t.* $\mathsf{rand}$. *If so, this algorithm returns 1; otherwise, it returns 0. In case that* $\mathsf{cm}$ *opens to $M$.*

*$\mathcal{COM}$ is secure if it satisfies the following properties:*

– ***Correctness.*** *For any* $\mathsf{ck}$ *output by* $\mathsf{KeyGen}$, *any* $(\mathsf{cm}, \mathsf{rm})$ *output by* $\mathsf{Com}(\mathsf{ck}, M)$, *it always holds that*

$$\mathsf{Verify}(\mathsf{ck}, M, \mathsf{cm}, \mathsf{rand}) = 1.$$

– ***Computational Binding.*** *For any PPT adversary $\mathcal{A}$, any* $\mathsf{ck}$ *output by* $\mathsf{KeyGen}$, *$\mathcal{A}$ has negligible probability to succeed in finding*

$$(\mathsf{cm}, M, \mathsf{rand}, M', \mathsf{rand}')$$

*satisfying*

$$\begin{cases} \mathsf{Verify}(\mathsf{ck}, M, \mathsf{cm}, \mathsf{rand}) = 1, \\ \mathsf{Verify}(\mathsf{ck}, M', \mathsf{cm}, \mathsf{rand}') = 1, \\ M \neq M'. \end{cases}$$

– ***Statistical Hiding.*** *For any adversary $\mathcal{A}$, the following two distributions are statistically indistinguishable.*

$$\left\{ (\mathsf{cm}, \mathsf{rand}) \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ (M_0, M_1, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{ck}), \\ (\mathsf{cm}, \mathsf{rand}) \leftarrow \mathsf{Com}(\mathsf{ck}, M_0) \end{array} \right\} \text{ and}$$

$$\left\{ (\mathsf{cm}, \mathsf{rand}) \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ (M_0, M_1, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{ck}), \\ (\mathsf{cm}, \mathsf{rand}) \leftarrow \mathsf{Com}(\mathsf{ck}, M_1) \end{array} \right\}.$$

## B WRK Protocol

We recall WRK protocol $\Pi_{\mathsf{2PC}}^{\mathcal{P}_\mathsf{A}, \mathcal{P}_\mathsf{B}}$ [31] in Figure 13.

## C Proof of Theorem 3

*Proof (Proof of Theorem 3).*

Correctness is straightforward. We consider the following cases.

For the cases of malicious $\tilde{\mathcal{P}}_\mathsf{A}$ and $\tilde{\mathcal{P}}_\mathsf{B}$, the proofs follow the paradigm from [31]. Here, we focus on constructing a simulator $\mathcal{S}_\mathcal{V}$ for simulating the transcript from the view of a public verifier and an extractor $\mathcal{E}$ for extracting the witnesses of $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$, hence implying the knowledge soundness of our combined protocol $\Pi_{\mathsf{pa\text{-}2PC}} = (\Pi_{\mathsf{pa\text{-}2PC\text{-}pre}}, \Pi_{\mathsf{pa\text{-}2PC\text{-}eval}})$ (c.f. Figures 10 and 11).

**Semi-Honest $\mathcal{V}$.** We construct the simulator $\mathcal{S}_\mathcal{V}$ as follows. $\mathcal{S}_\mathcal{V}$ receives the inputs as specified for the case that $\mathcal{V}$ is corrupted in Figure 7. In particular, it receives

$$\begin{array}{l} (\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j})_{j \in [\kappa]}, \\ (\mathsf{K}'_j[r_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[r'_w])_{w \in \mathcal{I}_\mathsf{W}}, \\ (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{A},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{A},j}])_{\ell \in [L], j \in [\kappa]}, \\ (\mathsf{K}'_j[s_w])_{w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B} \cup \mathcal{I}_\mathsf{W}}, (\mathsf{K}'_j[s'_w])_{w \in \mathcal{I}_\mathsf{W}}, \text{ and} \\ (\mathsf{K}'_j[\tilde{a}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{b}^{(\ell)}_{\mathsf{B},j}], \mathsf{K}'_j[\tilde{c}^{(\ell)}_{\mathsf{B},j}])_{\ell \in [L], j \in [\kappa]}. \end{array}$$

Additionally, $\mathcal{S}_{\mathcal{V}}$ also receives permutations $(\pi_j)_{j\in[\kappa]}$ for proving satisfaction of AND gates. We now separate the strategy for $\mathcal{S}_{\mathcal{V}}$ into the following two cases, namely, of garbling and of AND gates.

*The Case of Garbling.* Notice that $\mathcal{V}$ cannot posses $\Delta_{\mathsf{A}}$ and $\Delta_{\mathsf{B}}$ which are secrets of $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, respectively. Therefore, $\mathcal{V}$ cannot proceed the checks of valid VOLE tuples, w.r.t. $\Delta_{\mathsf{A}}$ and $\Delta_{\mathsf{B}}$, when decrypting the garbled tables. Moreover, we do not care whether $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) = r'_\gamma \oplus s'_\gamma$ for any AND gate $(\alpha,\beta,\gamma,\cdot)$ as $\mathcal{S}_{\mathcal{V}}$ can simulate the checks for the AND gates, to be discussed below. On the other hand, all selected and decrypted rows of the garbled tables and the labels determined by $\mathcal{P}_{\mathsf{A}}$ during evaluation can be published without compromising the privacy of both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$.

By the above observations, we construct $\mathcal{S}_{\mathcal{V}}$ as follows.

1. Sample $\Delta_{\mathsf{A}}$ and $\Delta_{\mathsf{B}}$ uniformly.
2. Find a random $(\mathfrak{z}_w)_{w\in\mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}}$ such that, (i) for any XOR gate $(\alpha,\beta,\gamma,\oplus)$, it holds that $\mathfrak{z}_\gamma = \mathfrak{z}_\alpha \oplus \mathfrak{z}_\beta$, and, (ii) for any $w \in \mathcal{I}_{\mathsf{O}}$, $\mathfrak{z}_w = z_w$. This sequence $(\mathfrak{z}_w)_{w\in\mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}}$ is the simulated values that $\mathcal{S}_{\mathcal{V}}$ generates as $\mathcal{S}_{\mathcal{V}}$ does not know the actual witnesses of both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$. Notice that we do not care whether $\mathfrak{z}_\gamma = \mathfrak{z}_\alpha \cdot \mathfrak{z}_\beta$ for any AND gate $(\alpha,\beta,\gamma,\cdot)$.
3. For each $w \in \mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}$, $\mathcal{S}_{\mathcal{V}}$ works as follows.
   (a) Sample $r_w, s_w \xleftarrow{\$} \{0,1\}$. Let $\lambda_w = r_w \oplus s_w$ and $\hat{\mathfrak{z}}_w = \mathfrak{z}_w \oplus \lambda_w = \mathfrak{z}_w \oplus r_w \oplus s_w$ be the simulated masked value at wire $w$.
   (b) Sample $\mathsf{M}[r_w]$, $\mathsf{K}[r_w]$, $\mathsf{M}[s_w]$, and $\mathsf{K}[s_w]$ s.t. $\mathsf{M}[r_w] = \mathsf{K}[r_w] \oplus r_w \cdot \Delta_{\mathsf{B}}$ and $\mathsf{M}[s_w] = \mathsf{K}[s_w] \oplus s_w \cdot \Delta_{\mathsf{A}}$.
   (c) For $j \in [\kappa]$, compute $\mathsf{M}'_j[r_w]$ and $\mathsf{M}'_j[s_w]$ s.t. $\mathsf{M}'_j[r_w] = \mathsf{K}'_j[r_w] \oplus r_w \cdot \nabla_{\mathsf{B},j}$ and $\mathsf{M}'_j[s_w] = \mathsf{K}'_j[s_w] \oplus s_w \cdot \nabla_{\mathsf{A},j}$.
4. For each AND gate $(\alpha,\beta,\gamma,\cdot)$, notice that, in the real protocol $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$, $\mathcal{P}_{\mathsf{B}}$ can determine the row $k = \hat{z}_\alpha + 2\cdot\hat{z}_\beta$ to decrypt the respective row of the garbled table from the real masked values $\hat{z}_\alpha$ and $\hat{z}_\beta$. Hence, $\mathcal{S}_{\mathcal{V}}$ can similarly determine $k = \hat{\mathfrak{z}}_\alpha + 2\cdot\hat{\mathfrak{z}}_\beta$ from the simulated masked values $\hat{\mathfrak{z}}_\alpha$ and $\hat{\mathfrak{z}}_\beta$. Therefore, when simulating the garbled tables, $\mathcal{S}_{\mathcal{V}}$ must guarantee that, from the decryption, one can obtain $\hat{\mathfrak{z}}_\gamma = \hat{z}_\gamma \oplus r_\gamma \oplus s_\gamma$. This is the crucial point for our sampling of $s'_\gamma$ and $r'_\gamma$. As said above, we do not care whether $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) = r'_\gamma \oplus s'_\gamma$. Hence, $\mathcal{S}_{\mathcal{V}}$ samples $r'_\gamma$ and $s'_\gamma$ such that
$$(r'_\gamma \oplus s'_\gamma) \oplus \lambda_\gamma \oplus \hat{\mathfrak{z}}_\alpha \cdot \lambda_\beta \oplus \hat{\mathfrak{z}}_\beta \cdot \lambda_\alpha = \hat{\mathfrak{z}}_\gamma$$
where $\lambda_\alpha = r_\alpha \oplus s_\alpha$ and $\lambda_\beta = r_\beta \oplus s_\beta$. With $r'_\gamma$ and $s'_\gamma$ in hand, $\mathcal{S}_{\mathcal{V}}$ computes $\mathsf{M}'[r'_\gamma]$, $\mathsf{M}'_j[r'_\gamma]$, $\mathsf{M}[s'_\gamma]$, and $\mathsf{M}'_j[s'_\gamma]$ s.t.
$$\begin{cases} \mathsf{M}[r'_\gamma] = \mathsf{K}[r'_\gamma] \oplus r'_\gamma \cdot \Delta_{\mathsf{B}}, \\ \mathsf{M}'_j[r'_\gamma] = \mathsf{K}'_j[r'_\gamma] \oplus r'_\gamma \cdot \nabla_{\mathsf{B},j} \ \forall j \in [\kappa], \\ \mathsf{M}[s'_\gamma] = \mathsf{K}[s'_\gamma] \oplus s'_\gamma \cdot \Delta_{\mathsf{A}}, \\ \mathsf{M}'_j[s'_\gamma] = \mathsf{K}'_j[s'_\gamma] \oplus s'_\gamma \cdot \nabla_{\mathsf{A},j} \ \forall j \in [\kappa]. \end{cases}$$
5. Eventually, we need the labels for encrypting and decrypting purposes. In particular, for $w \in \mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}$, $\mathcal{S}_{\mathcal{V}}$ computes $\mathsf{L}_{w,0} \xleftarrow{\$} \mathbb{F}_{2^\tau}$ as specified in $\Pi_{\mathsf{pa\text{-}2PC\text{-}pre}}$ (c.f. Figure 10).

*The Case of AND Gates.* As said above, we do not care whether $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) = r'_\gamma \oplus s'_\gamma$. Hence, the simulation, in this case, makes the simulated proofs for checking AND gates indistinguishable from the real proofs. Recall that $L = W \cdot \mathsf{bs} + \mathsf{rm}$. Having permutations $(\pi_j)_{j\in[\kappa]}$, $\mathcal{S}_{\mathcal{V}}$ works as follows.

For $j \in [\kappa]$ and $i \in [L]$, we consider the following two sub-cases.

1. If $i \in [(k-1)\cdot W + 1, k\cdot W]$ for $k \in [\mathsf{bs}]$, in the real protocol $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$ (c.f. Figure 11), $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ use authenticated values $(a^{(i)}_{\mathsf{A},j}, b^{(i)}_{\mathsf{A},j}, c^{(i)}_{\mathsf{A},j})$ and $(a^{(i)}_{\mathsf{B},j}, b^{(i)}_{\mathsf{B},j}, c^{(i)}_{\mathsf{B},j})$ for testing whether $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) = r'_\gamma \oplus s'_\gamma$ for the $(i-(k-1)\cdot W)$-th AND gate $(\alpha,\beta,\gamma,\cdot)$ by using protocol $\Pi^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$ (c.f. Figure 6). However, as said above, $\mathcal{S}_{\mathcal{V}}$ does not generate $(r_\alpha, r_\beta, r'_\gamma)$ and $(s_\alpha, s_\beta, s'_\gamma)$ satisfying such condition. Therefore, to encompass the checking in sub-protocol $\Pi^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$, $\mathcal{S}_{\mathcal{V}}$ first samples $a^{(i)}_{\mathsf{A},j}, b^{(i)}_{\mathsf{A},j}, a^{(i)}_{\mathsf{B},j}, b^{(i)}_{\mathsf{B},j} \xleftarrow{\$} \mathbb{F}_2$. Then, $\mathcal{S}_{\mathcal{V}}$ samples $(c^{(i)}_{\mathsf{A},j}, c^{(i)}_{\mathsf{B},j})$ satisfying
$$\begin{aligned} &(c^{(i)}_{\mathsf{A},j} \oplus r'_\gamma \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{A},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{A},j}) \\ &\oplus (c^{(i)}_{\mathsf{B},j} \oplus s'_\gamma \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{B},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{B},j}) \\ &\oplus d^{(i)}_j \cdot e^{(i)}_j = 0 \end{aligned}$$
where $d^{(i)}_j = r'_\alpha \oplus a^{(i)}_{\mathsf{A},j} \oplus s'_\alpha \oplus a^{(i)}_{\mathsf{B},j}$ and $e^{(i)}_j = r'_\beta \oplus b^{(i)}_{\mathsf{A},j} \oplus s'_\beta \oplus b^{(i)}_{\mathsf{B},j}$.

2. If $i \in [\mathsf{bs} \cdot W + 1, \mathsf{bs} \cdot W + \mathsf{rm}]$, $\mathcal{S}_{\mathcal{V}}$ simply first samples $a_{\mathsf{A},j}^{(i)}, b_{\mathsf{A},j}^{(i)}, a_{\mathsf{B},j}^{(i)}, b_{\mathsf{B},j}^{(i)} \xleftarrow{\$} \mathbb{F}_2$. Then, $\mathcal{S}_{\mathcal{V}}$ samples $(c_{\mathsf{A},j}^{(i)}, c_{\mathsf{B},j}^{(i)})$ satisfying
$$(a_{\mathsf{A},j}^{(i)} \oplus a_{\mathsf{B},j}^{(i)}) \cdot (b_{\mathsf{A},j}^{(i)} \oplus b_{\mathsf{B},j}^{(i)}) = c_{\mathsf{A},j}^{(i)} \oplus c_{\mathsf{B},j}^{(i)}.$$

Having $(a_{\mathsf{A},j}^{(\ell)}, b_{\mathsf{A},j}^{(\ell)}, c_{\mathsf{A},j}^{(\ell)}, a_{\mathsf{B},j}^{(\ell)}, b_{\mathsf{B},j}^{(\ell)}, c_{\mathsf{B},j}^{(\ell)})_{j\in[\kappa],\ell\in[L]}$ sampled and computed according to the above process, $\mathcal{S}_{\mathcal{V}}$ can easily compute $(\tilde{a}_{\mathsf{A},j}^{(\ell)}, \tilde{b}_{\mathsf{A},j}^{(\ell)}, \tilde{c}_{\mathsf{A},j}^{(\ell)}, \tilde{a}_{\mathsf{B},j}^{(\ell)}, \tilde{b}_{\mathsf{B},j}^{(\ell)}, \tilde{c}_{\mathsf{B},j}^{(\ell)})_{j\in[\kappa],\ell\in[L]}$ such that, for $j \in [\kappa]$ and $\ell \in [L]$, it holds that

$$\begin{aligned}&(a_{\mathsf{A},j}^{(\ell)}, b_{\mathsf{A},j}^{(\ell)}, c_{\mathsf{A},j}^{(\ell)}, a_{\mathsf{B},j}^{(\ell)}, b_{\mathsf{B},j}^{(\ell)}, c_{\mathsf{B},j}^{(\ell)})\\ &= (\tilde{a}_{\mathsf{A},j}^{(\pi_j(\ell))}, \tilde{b}_{\mathsf{A},j}^{(\pi_j(\ell))}, \tilde{c}_{\mathsf{A},j}^{(\pi_j(\ell))}, \tilde{a}_{\mathsf{B},j}^{(\pi_j(\ell))}, \tilde{b}_{\mathsf{B},j}^{(\pi_j(\ell))}, \tilde{c}_{\mathsf{B},j}^{(\pi_j(\ell))})\end{aligned}$$

by leveraging the bijections of permutations $(\pi_j)_{j\in[\kappa]}$.

Finally, $\mathcal{S}_{\mathcal{V}}$ computes the IT-MAC tags $\mathsf{M}'_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}]$, $\mathsf{M}'_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}]$, $\mathsf{M}'_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}]$, $\mathsf{M}'_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}]$, $\mathsf{M}'_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}]$, and $\mathsf{M}'_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}]$ s.t.

$$\begin{cases}\mathsf{M}'_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}] \oplus \tilde{a}_{\mathsf{A},j}^{(\ell)} \cdot \nabla_{\mathsf{B},j},\\ \mathsf{M}'_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}] \oplus \tilde{b}_{\mathsf{A},j}^{(\ell)} \cdot \nabla_{\mathsf{B},j},\\ \mathsf{M}'_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}] \oplus \tilde{c}_{\mathsf{A},j}^{(\ell)} \cdot \nabla_{\mathsf{B},j},\\ \mathsf{M}'_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}] \oplus \tilde{a}_{\mathsf{B},j}^{(\ell)} \cdot \nabla_{\mathsf{A},j},\\ \mathsf{M}'_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}] \oplus \tilde{b}_{\mathsf{B},j}^{(\ell)} \cdot \nabla_{\mathsf{A},j},\\ \mathsf{M}'_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}] = \mathsf{K}'_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}] \oplus \tilde{c}_{\mathsf{B},j}^{(\ell)} \cdot \nabla_{\mathsf{A},j}.\end{cases}$$

With the above strategy, we can see that the simulated proof is indistinguishable from the real one.

**Knowledge Soundness of $\Pi_{\mathsf{pa\text{-}2PC}}$.** To show that protocol $\Pi_{\mathsf{pa\text{-}2PC}}$ is knowledge-sound, we construct an extractor $\mathcal{E}$ acting as a verifier $\mathcal{V}$ and interacting with $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ to extract their witnesses. $\mathcal{E}$ additionally can rewind both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ to change the randomness as follows.

1. *Initial Execution.* Initially, $\mathcal{E}$ acts as an honest $\mathcal{V}$ to run $\Pi_{\mathsf{pa\text{-}2PC}}$ (by sequentially running $\Pi_{\mathsf{pa\text{-}2PC\text{-}pre}}$ and $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$).
2. *Rewound Execution.* Then, $\mathcal{E}$ rewinds $\Pi_{\mathsf{pa\text{-}2PC}}$ to step 9 of $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$ when both parties access $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ (c.f. Figure 7). At this time, $\mathcal{E}$ samples new IT-MAC global keys (for VOLEitH proofs) and interacts with both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ w.r.t. the newly sampled keys.

We assume that the IT-MAC global keys step 1 are denoted by $(\nabla_{\mathsf{A},j}^{(0)}, \nabla_{\mathsf{B},j}^{(0)})_{j\in[\kappa]}$. The ones for step 2 are $(\nabla_{\mathsf{A},j}^{(1)}, \nabla_{\mathsf{B},j}^{(1)})_{j\in[\kappa]}$. We now proceed the extraction as follows.

*Extracting Randomness for Masking.* In step 9 (c.f. Figure 11), for $t \in \{0,1\}$, we denote

$$\begin{aligned}&(\nabla_{\mathsf{A},j}^{(t)})_{j\in[\kappa]}, (\mathsf{K}'^{(t)}_j[r_w])_{w\in\mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}}, (\mathsf{K}'^{(t)}_j[r'_w])_{w\in\mathcal{I}_{\mathsf{W}}},\\ &(\mathsf{K}'^{(t)}_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}], \mathsf{K}'^{(t)}_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}], \mathsf{K}'^{(t)}_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}])_{j\in[\kappa],\ell\in[L]},\\ &(\nabla_{\mathsf{B},j}^{(t)})_{j\in[\kappa]}, (\mathsf{K}'^{(t)}_j[s_w])_{w\in\mathcal{I}_{\mathsf{A}}\cup\mathcal{I}_{\mathsf{B}}\cup\mathcal{I}_{\mathsf{W}}}, (\mathsf{K}'^{(t)}_j[s'_w])_{w\in\mathcal{I}_{\mathsf{W}}},\\ &(\mathsf{K}'^{(t)}_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}], \mathsf{K}'^{(t)}_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}], \mathsf{K}'^{(t)}_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}])_{j\in[\kappa],\ell\in[L]}\end{aligned} \tag{8}$$

such that

- $t = 0$ is for those of initial execution of $\mathcal{E}$, and
- $t = 1$ is for those of rewound execution of $\mathcal{E}$.

For any $j \in [\kappa]$, since $\nabla_{\mathsf{B},j}^{(0)}$ and $\nabla_{\mathsf{B},j}^{(1)}$ are sampled independently from $\mathbb{F}_{2^\tau}$ in two different executions, with overwhelming probability $\frac{2^\tau-1}{2^\tau}$, $\nabla_{\mathsf{B},j}^{(0)} \neq \nabla_{\mathsf{B},j}^{(1)}$. Hence, we can instruct $\mathcal{E}$ to witnesses with the extraction idea as follows. Assume that $r \in \{0,1\}$ is authenticated such that

$$\begin{aligned}&\begin{cases}\mathsf{M}'_j[r] = \mathsf{K}'^{(0)}_j[r] \oplus r \cdot \nabla_{\mathsf{B},j}^{(0)},\\ \mathsf{M}'_j[r] = \mathsf{K}'^{(1)}_j[r] \oplus r \cdot \nabla_{\mathsf{B},j}^{(1)}.\end{cases}\\ \iff\ &\mathsf{K}'^{(0)}_j[r] \oplus r \cdot \nabla_{\mathsf{B},j}^{(0)} = \mathsf{K}'^{(1)}_j[r] \oplus r \cdot \nabla_{\mathsf{B},j}^{(1)}.\end{aligned}$$

If $\nabla^{(0)}_{\mathsf{A},j} \neq \nabla^{(1)}_{\mathsf{A},j}$, then we can compute $r$ by simple algebra techniques. Hence, with this idea, $\mathcal{E}$ can extract the value mentioned above.

Hence, according to (8), we can extract

$$(r_w)_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}, (r'_w)_{w\in\mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{A},j}, \tilde{b}^{(\ell)}_{\mathsf{A},j}, \tilde{c}^{(\ell)}_{\mathsf{A},j})_{j\in[\kappa],\ell\in[L]},$$
$$(s_w)_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}, (s'_w)_{w\in\mathcal{I}_\mathsf{W}}, (\tilde{a}^{(\ell)}_{\mathsf{B},j}, \tilde{b}^{(\ell)}_{\mathsf{B},j}, \tilde{c}^{(\ell)}_{\mathsf{B},j})_{j\in[\kappa],\ell\in[L]}.$$

*Extracting Inputs.* In steps 5 and 6 (c.f. Figure 11), $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ published $\hat{z}_w$ for $w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B}$. Hence, $\mathcal{E}$ can compute inputs $z_w := \hat{z}_w \oplus r_w \oplus s_w$ for $w \in \mathcal{I}_\mathsf{A} \cup \mathcal{I}_\mathsf{B}$ where $r_w$ and $s_w$ are already extracted above.
*Implications for XOR Gates' Satisfiability.* As all parties compute $\hat{z}_\gamma := \hat{z}_\alpha \oplus \hat{z}_\beta$ for any XOR gate $(\alpha, \beta, \gamma, \oplus)$ (step 7 of Figure 11), we can imply the randomness for output of XOR gates as $r_\gamma = r_\alpha \oplus r_\beta$ and $s_\gamma = s_\alpha \oplus s_\beta$, and the actual value $z_\gamma = z_\alpha \oplus z_\beta$ once $\mathcal{E}$ knows $r_\alpha, r_\beta, s_\alpha, s_\beta, z_\alpha$, and $z_\beta$.
*Satisfiability w.r.t. Protocol $\Pi^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$ (c.f. Figure 6).* For simplicity, we first assume that, for $j \in [\kappa]$ and $\ell \in [L]$, it holds that

$$(a^{(\ell)}_{\mathsf{A},j} \oplus a^{(\ell)}_{\mathsf{B},j}) \cdot (b^{(\ell)}_{\mathsf{A},j} \oplus b^{(\ell)}_{\mathsf{B},j}) = c^{(\ell)}_{\mathsf{A},j} \oplus c^{(\ell)}_{\mathsf{B},j}.$$

With the computations and verifications in step 2 of $\Pi^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$, we can imply that, for $j \in [\kappa]$ and $i \in [W]$,

$$\begin{aligned}
d^{(i)}_{\mathsf{A},j} &= x^{(i)}_\mathsf{A} \oplus a^{(i)}_{\mathsf{A},j}, \quad e^{(i)}_{\mathsf{A},j} = y^{(i)}_\mathsf{A} \oplus b^{(i)}_{\mathsf{A},j},\\
d^{(i)}_{\mathsf{B},j} &= x^{(i)}_\mathsf{B} \oplus a^{(i)}_{\mathsf{B},j}, \quad e^{(i)}_{\mathsf{B},j} = y^{(i)}_\mathsf{B} \oplus b^{(i)}_{\mathsf{B},j},\\
\tilde{z}^{(i)}_{\mathsf{A},j} &= z^{(i)}_\mathsf{A} \oplus c^{(i)}_{\mathsf{A},j} \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{A},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{A},j},\\
\tilde{z}^{(i)}_{\mathsf{B},j} &= z^{(i)}_\mathsf{B} \oplus c^{(i)}_{\mathsf{B},j} \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{B},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{B},j}.
\end{aligned}$$

Notice that, in step 1 of $\Pi^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}_{\mathsf{check\text{-}AND}}$, every party computes $d^{(i)}_j := d^{(i)}_{\mathsf{A},j} \oplus d^{(i)}_{\mathsf{B},j}$ and $e^{(i)}_j := e^{(i)}_{\mathsf{A},j} \oplus e^{(i)}_{\mathsf{B},j}$, and checks whether $\tilde{z}^{(i)}_{\mathsf{A},j} \oplus \tilde{z}^{(i)}_{\mathsf{B},j} \oplus d^{(i)}_j \cdot e^{(i)}_j = 0$.

For simplicity, let $x^{(i)} := x^{(i)}_\mathsf{A} \oplus x^{(i)}_\mathsf{B}$, $y^{(i)} := y^{(i)}_\mathsf{A} \oplus y^{(i)}_\mathsf{B}$, $z^{(i)} := z^{(i)}_\mathsf{A} \oplus z^{(i)}_\mathsf{B}$, $a^{(i)}_j := a^{(i)}_{\mathsf{A},j} \oplus a^{(i)}_{\mathsf{B},j}$, $b^{(i)}_j := b^{(i)}_{\mathsf{A},j} \oplus b^{(i)}_{\mathsf{B},j}$, and $c^{(i)}_j := c^{(i)}_{\mathsf{A},j} \oplus c^{(i)}_{\mathsf{B},j}$. Hence, we see that

$$\begin{aligned}
d^{(i)}_j &= d^{(i)}_{\mathsf{A},j} \oplus d^{(i)}_{\mathsf{B},j} = x^{(i)}_\mathsf{A} \oplus a^{(i)}_{\mathsf{A},j} \oplus x^{(i)}_\mathsf{B} \oplus a^{(i)}_{\mathsf{B},j} = x^{(i)} \oplus a^{(i)}_j,\\
e^{(i)}_j &= e^{(i)}_{\mathsf{A},j} \oplus e^{(i)}_{\mathsf{B},j} = y^{(i)}_\mathsf{A} \oplus b^{(i)}_{\mathsf{A},j} \oplus y^{(i)}_\mathsf{B} \oplus b^{(i)}_{\mathsf{B},j} = y^{(i)} \oplus b^{(i)}_j.
\end{aligned}$$

Hence, it follows that

$$\begin{aligned}
\tilde{z}^{(i)}_j &= \tilde{z}^{(i)}_{\mathsf{A},j} \oplus \tilde{z}^{(i)}_{\mathsf{B},j}\\
&= z^{(i)}_\mathsf{A} \oplus c^{(i)}_{\mathsf{A},j} \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{A},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{A},j}\\
&\quad \oplus z^{(i)}_\mathsf{B} \oplus c^{(i)}_{\mathsf{B},j} \oplus d^{(i)}_j \cdot b^{(i)}_{\mathsf{B},j} \oplus e^{(i)}_j \cdot a^{(i)}_{\mathsf{B},j}\\
&= z^{(i)} \oplus c^{(i)}_j \oplus d^{(i)}_j \cdot b^{(i)}_j \oplus e^{(i)}_j \cdot a^{(i)}_j
\end{aligned}$$

Since $\tilde{z}^{(i)}_{\mathsf{A},j} \oplus \tilde{z}^{(i)}_{\mathsf{B},j} \oplus d^{(i)}_j \cdot e^{(i)}_j = 0$, it implies that

$$(z^{(i)} \oplus c^{(i)}_j \oplus d^{(i)}_j \cdot b^{(i)}_j \oplus e^{(i)}_j \cdot a^{(i)}_j) \oplus d^{(i)}_j \cdot e^{(i)}_j = 0.$$

By expressing $d^{(i)}_j = x^{(i)} \oplus a^{(i)}_j$ and $e^{(i)}_j = y^{(i)} \oplus b^{(i)}_j$, we see that the above equation is equivalent to

$$\begin{aligned}
z^{(i)} \oplus c^{(i)}_j \oplus (x^{(i)} \oplus a^{(i)}_j) \cdot b^{(i)}_j \oplus (y^{(i)} \oplus b^{(i)}_j) \cdot a^{(i)}_j&\\
\oplus (x^{(i)} \oplus a^{(i)}_j) \cdot (y^{(i)} \oplus b^{(i)}_j) = 0.&
\end{aligned}$$

It follows that

$$z^{(i)} \oplus c^{(i)}_j \oplus x^{(i)} \cdot y^{(i)} \oplus a^{(i)}_j \cdot b^{(i)}_j = 0.$$

As originally we assumed $(a^{(i)}_{\mathsf{A},j} \oplus a^{(i)}_{\mathsf{B},j}) \cdot (b^{(i)}_{\mathsf{A},j} \oplus b^{(i)}_{\mathsf{B},j}) = c^{(i)}_{\mathsf{A},j} \oplus c^{(i)}_{\mathsf{B},j}$. We see that $a^{(i)}_j \cdot b^{(i)}_j = c^{(i)}_j$. The above equation is equivalent to

$$z^{(i)} \oplus x^{(i)} \cdot y^{(i)} = 0.$$

Hence, we imply that

$$(x_{\mathsf{A}}^{(i)} \oplus x_{\mathsf{B}}^{(i)}) \cdot (y_{\mathsf{A}}^{(i)} \oplus y_{\mathsf{B}}^{(i)}) = z_{\mathsf{A}}^{(i)} \oplus z_{\mathsf{B}}^{(i)}.$$

*Extracting Randomness for Arguing AND Gates' Satisfiability.* From union bound, the probability that $\nabla_{\mathsf{B},j}^{(0)} \neq \nabla_{\mathsf{B},j}^{(1)}\ \forall j \in [\kappa]$ is at least $\frac{2^\tau - \kappa}{2^\tau}$ which is overwhelming by a careful setting of parameters $\tau$ and $\kappa$. In such a case, $\mathcal{E}$ can extract $(\tilde{a}_{\mathsf{A},j}^{(\ell)}, \tilde{b}_{\mathsf{A},j}^{(\ell)}, \tilde{c}_{\mathsf{A},j}^{(\ell)})_{j\in[\kappa],\ell\in[L]}$ by using the above way for each $j \in [\kappa]$. Similarly, $\mathcal{E}$ can also extract $(\tilde{a}_{\mathsf{B},j}^{(\ell)}, \tilde{b}_{\mathsf{B},j}^{(\ell)}, \tilde{c}_{\mathsf{B},j}^{(\ell)})_{j\in[\kappa],\ell\in[L]}$ with the same probability.

We now consider $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$ (c.f. Figure 11). Notice that, in steps 2 and 10, the random tuples are shuffled according to permutations $(\pi_j)_{j\in[\kappa]}$ provided by $\mathcal{V}$ in step 1. The actions of permuting those components following the "balls-and-bins" experiment [3,32] specified in Lemma 1 for testing the satisfaction $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) = r'_\gamma \oplus s'_\gamma$ for all AND gates $(\alpha,\beta,\gamma,\cdot)$. This is done by running steps 4, 11, and 13. Hence, if both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ cheat by having $(r_\alpha \oplus s_\alpha)\cdot(r_\beta \oplus s_\beta) \neq r'_\gamma \oplus s'_\gamma$ for some AND gate $(\alpha,\beta,\gamma,\cdot)$. This requires both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ to have enough luck to win the experiment in Lemma 1 for all $\kappa$ repetitions which has probability at most $\binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa}$. Otherwise, cheating $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ can be detected with probability at least $1-\binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa}$ according to $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}},\mathcal{V},\kappa,W}$, as analyzed above, and the testing in step 11 (c.f. Figure 11).

*Arguing AND Gates' Satisfiability.* For each AND gate $(\alpha,\beta,\gamma,\cdot)$ as in step 7 (c.f. Figure 11), let $k = \hat{z}_\alpha + 2\cdot\hat{z}_\beta$ be the index determined by $\mathcal{P}_{\mathsf{B}}$. From here $\mathcal{P}_{\mathsf{B}}$ publishes the decrypted tuple

$$(r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], (\mathsf{M}'[r_{\gamma,k}])_{j\in[\kappa]}, \mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k}\cdot \Delta_{\mathsf{A}}).$$

Notice that, from step 12, $\mathcal{E}$ already determined

$$\begin{cases} \mathsf{K}_j'^{(t)}[r_{\gamma,k}] := \mathsf{K}_j'^{(t)}[r'_\gamma] \oplus \mathsf{K}_j'^{(t)}[r_\gamma] \\ \qquad\qquad \oplus \hat{z}_\alpha \cdot \mathsf{K}_j'^{(t)}[r_\beta] \oplus \hat{z}_\beta \cdot \mathsf{K}_j'^{(t)}[r_\alpha]\ \forall t \in \{0,1\}, \\ \mathsf{K}_j'^{(t)}[s_{\gamma,k}] := \mathsf{K}_j'^{(t)}[s'_\gamma] \oplus \mathsf{K}_j'^{(t)}[s_\gamma] \oplus \hat{z}_\alpha \cdot \mathsf{K}_j'^{(t)}[s_\beta] \\ \qquad\qquad \oplus \hat{z}_\beta \cdot \mathsf{K}_j'^{(t)}[s_\alpha] \oplus \hat{z}_\alpha \cdot \hat{z}_\beta \cdot \nabla_{\mathsf{B},j}\ \forall t \in \{0,1\}. \end{cases}$$

With a similar argument, we can extract $r_{\gamma,k}$ and $s_{\gamma,k}$. Moreover, they also satisfy

$$\begin{cases} r_{\gamma,k} = r'_\gamma \oplus r_\gamma \oplus \hat{z}_\alpha \cdot r_\beta \oplus \hat{z}_\beta \cdot r_\alpha, \\ s_{\gamma,k} = s'_\gamma \oplus s_\gamma \oplus \hat{z}_\alpha \cdot s_\beta \oplus \hat{z}_\beta \cdot s_\alpha \oplus \hat{z}_\alpha \cdot \hat{z}_\beta \end{cases} \tag{9}$$

according to the above system. According to step 7, $\mathcal{E}$ can determine $\hat{z}_\gamma := r_{\gamma,k} \oplus s_{\gamma,k}$. Let $\lambda'_\gamma := r'_\gamma \oplus s'_\gamma$, $\lambda_\gamma := r_\gamma \oplus s_\gamma$, $\lambda_\alpha := r_\alpha \oplus s_\alpha$, and $\lambda_\beta := r_\beta \oplus s_\beta$. Hence, we can transform (9) to be

$$\hat{z}_\gamma = \lambda'_\gamma \oplus \lambda_\gamma \oplus \hat{z}_\alpha \cdot \lambda_\beta \oplus \hat{z}_\beta \cdot \lambda_\alpha \oplus \hat{z}_\alpha \cdot \hat{z}_\beta.$$

By definition, $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$ and $\hat{z}_\beta = z_\beta \oplus \lambda_\beta$. Hence, the above equation is equivalent to

$$\hat{z}_\gamma = \lambda'_\gamma \oplus \lambda_\gamma \oplus (z_\alpha \oplus \lambda_\alpha)\cdot\lambda_\beta \oplus (z_\beta \oplus \lambda_\beta)\cdot\lambda_\alpha \\ \oplus (z_\alpha \oplus \lambda_\alpha)\cdot(z_\beta \oplus \lambda_\beta).$$

It follows that

$$\hat{z}_\gamma = \lambda'_\gamma \oplus \lambda_\gamma \oplus z_\alpha \cdot z_\beta \oplus \lambda_\alpha \cdot \lambda_\beta.$$

As shown above, with probability at least $1-\binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa}$, $\lambda'_\gamma = \lambda_\alpha\cdot\lambda_\beta$. Hence, in such a case, $\hat{z}_\gamma = z_\alpha\cdot z_\beta \oplus \lambda_\gamma$. Since $\lambda_\gamma$ was extracted above, we can easily guarantee that $\hat{z}_\gamma$ is a masked value of $z_\alpha \cdot z_\beta$ by $\lambda_\gamma$.

Step 16 (c.f. Figure 11) implies that the output $(z_w)_{w\in\mathcal{I}_{\mathsf{W}}}$ is correct.

*Soundness Error.* As we need a single rewinding to extract all potential witnesses as well as the constraints assuming $(a_{\mathsf{A},j}^{(\ell)} \oplus a_{\mathsf{B},j}^{(\ell)}) \cdot (b_{\mathsf{A},j}^{(\ell)} \oplus b_{\mathsf{B},j}^{(\ell)}) = c_{\mathsf{A},j}^{(\ell)} \oplus c_{\mathsf{B},j}^{(\ell)}$ holds for all $j \in [\kappa]$ and $\ell \in [L]$. We see that the soundness error for cheating $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ to break the scheme is $2^{-\kappa\cdot\tau}$ for all $\kappa$ repetitions.

The use of balls-and-bins method (c.f. Lemma 1) incurs a soundness error $\binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa}$.

Hence, the total soundness error is

$$2^{-\kappa\cdot\tau} + \binom{W\cdot\mathsf{bs}+\mathsf{rm}}{\mathsf{bs}}^{-\kappa} + \mathsf{negl}(\kappa,\tau)$$

where $\mathsf{negl}(\kappa,\tau)$ denotes the negligible probability for breaking the security of components in the protocol, e.g., commitments.

**Proof Preparation:**

1. $\mathcal{V}$ uniformly samples and publishes random permutations $\pi_j : [L] \rightarrow [L]\ \forall j \in [\kappa]$.
2. For $j \in [\kappa]$ and $p \in \{\mathsf{A}, \mathsf{B}\}$, $\mathcal{P}_p$ permutes the random values and the IT-MAC tags as follows.

$$(a_{p,j}^{(\ell)}, b_{p,j}^{(\ell)}, c_{p,j}^{(\ell)}) := (\tilde{a}_{p,j}^{(\pi_j(\ell))}, \tilde{b}_{p,j}^{(\pi_j(\ell))}, \tilde{c}_{p,j}^{(\pi_j(\ell))}),\ (\mathsf{M}'_j[a_{p,j}^{(\ell)}], \mathsf{M}'_j[b_{p,j}^{(\ell)}], \mathsf{M}'_j[c_{p,j}^{(\ell)}]) := (\mathsf{M}'_j[\tilde{a}_{p,j}^{(\pi_j(\ell))}], \mathsf{M}'_j[\tilde{b}_{p,j}^{(\pi_j(\ell))}], \mathsf{M}'_j[\tilde{c}_{p,j}^{(\pi_j(\ell))}])\ \forall \ell \in [L].$$

3. For $p \in \{\mathsf{A}, \mathsf{B}\}$, $\mathcal{P}_p$ publishes $(a_{p,j}^{(i)}, b_{p,j}^{(i)}, c_{p,j}^{(i)}, \mathsf{M}'_j[a_{p,j}^{(i)}], \mathsf{M}'_j[b_{p,j}^{(i)}], \mathsf{M}'_j[c_{p,j}^{(i)}])_{j\in[\kappa], i \in [W\cdot\mathsf{bs}+1, W\cdot\mathsf{bs}+\mathsf{rm}]}$.
4. For $k \in [\mathsf{bs}]$, run step 1 of protocol $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}$ (c.f. Figure 6) with inputs

$$((r_\alpha, r_\beta, r'_\gamma, \mathsf{M}'_j[r_\alpha], \mathsf{M}'_j[r_\beta], \mathsf{M}'_j[r'_\gamma])_{j\in[\kappa],\ \text{all } (\alpha,\beta,\gamma,\cdot)}, (a_{\mathsf{A},j}^{(i)}, b_{\mathsf{A},j}^{(i)}, c_{\mathsf{A},j}^{(i)}, \mathsf{M}'_j[a_{\mathsf{A},j}^{(i)}], \mathsf{M}'_j[b_{\mathsf{A},j}^{(i)}], \mathsf{M}'_j[c_{\mathsf{A},j}^{(i)}])_{j\in[\kappa], i\in[(k-1)\cdot W+1, k\cdot W]}) \text{ from } \mathcal{P}_\mathsf{A},$$
$$((s_\alpha, s_\beta, s'_\gamma, \mathsf{M}'_j[s_\alpha], \mathsf{M}'_j[s_\beta], \mathsf{M}'_j[s'_\gamma])_{j\in[\kappa],\ \text{all } (\alpha,\beta,\gamma,\cdot)}, (a_{\mathsf{B},j}^{(i)}, b_{\mathsf{B},j}^{(i)}, c_{\mathsf{B},j}^{(i)}, \mathsf{M}'_j[a_{\mathsf{B},j}^{(i)}], \mathsf{M}'_j[b_{\mathsf{B},j}^{(i)}], \mathsf{M}'_j[c_{\mathsf{B},j}^{(i)}])_{j\in[\kappa], i\in[(k-1)\cdot W+1, k\cdot W]}) \text{ from } \mathcal{P}_\mathsf{B}.$$

**Input Processing:** When $\mathcal{P}_\mathsf{A}$ and $\mathcal{P}_\mathsf{B}$ have $(z_w)_{w\in\mathcal{I}_\mathsf{A}} \in \{0,1\}^{|\mathcal{I}_\mathsf{A}|}$ and $(z_w)_{w\in\mathcal{I}_\mathsf{B}} \in \{0,1\}^{|\mathcal{I}_\mathsf{B}|}$, respectively, the parties work as follows.

5. $\mathcal{P}_\mathsf{A}$ publishes $(r_w, \mathsf{M}[r_w], (\mathsf{M}'_j[r_w])_{j\in[\kappa]})_{w\in\mathcal{I}_\mathsf{A}}$. $\mathcal{P}_\mathsf{B}$ checks whether $\mathsf{M}[r_w] = \mathsf{K}[r_w] \oplus r_w \cdot \Delta_\mathsf{B}$ for $w \in \mathcal{I}_\mathsf{A}$. $\mathcal{P}_\mathsf{B}$ then publishes $\hat{z}_w = z_w \oplus \lambda_w = z_w \oplus r_w \oplus s_w\ \forall w \in \mathcal{I}_\mathsf{A}$. Finally, $\mathcal{P}_\mathsf{A}$ publishes $\mathsf{L}_{w,\hat{z}_w}\ \forall w \in \mathcal{I}_\mathsf{A}$.
6. $\mathcal{P}_\mathsf{B}$ publishes $(s_w, \mathsf{M}[s_w], (\mathsf{M}'_j[s_w])_{j\in[\kappa]})_{w\in\mathcal{I}_\mathsf{B}}$. $\mathcal{P}_\mathsf{A}$ checks whether $\mathsf{M}[s_w] = \mathsf{K}[s_w] \oplus s_w \cdot \Delta_\mathsf{A}\ \forall w \in \mathcal{I}_\mathsf{B}$. $\mathcal{P}_\mathsf{A}$ then publishes $\hat{z}_w = z_w \oplus \lambda_w = z_w \oplus r_w \oplus s_w$ and $\mathsf{L}_{w,\hat{z}_w}$ for $w \in \mathcal{I}_\mathsf{B}$.

**Circuit Evaluation:**

7. $\mathcal{P}_\mathsf{B}$ evaluates the circuit following a topological order. For each gate $(\alpha, \beta, \gamma, \mathsf{op})$ where $\mathsf{op} \in \{\cdot, \oplus\}$, $\mathcal{P}_\mathsf{B}$ holds $(\hat{z}_\alpha, \mathsf{L}_{\alpha,\hat{z}_\alpha})$ and $(\hat{z}_\beta, \mathsf{L}_{\beta,\hat{z}_\beta})$ where $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$ and $\hat{z}_\beta = z_\beta \oplus \lambda_\beta$.
   - If $\mathsf{op} = \oplus$, $\mathcal{P}_\mathsf{B}$ computes $\hat{z}_\gamma := \hat{z}_\alpha \oplus \hat{z}_\beta$ and $\mathsf{L}_{\gamma,\hat{z}_\gamma} := \mathsf{L}_{\alpha,\hat{z}_\alpha} \oplus \mathsf{L}_{\beta,\hat{z}_\beta}$.
   - If $\mathsf{op} = \cdot$, for $k := \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$, $\mathcal{P}_\mathsf{B}$ computes and publishes $(r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], (\mathsf{M}'[r_{\gamma,k}])_{j\in[\kappa]}, \mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A}) := G_{\gamma,k} \oplus H(\mathsf{L}_{\alpha,\hat{z}_\alpha}, \mathsf{L}_{\beta,\hat{z}_\beta}, \gamma, k)$. Then, $\mathcal{P}_\mathsf{B}$ checks whether $\mathsf{M}[r_{\gamma,k}] = \mathsf{K}[r_{\gamma,k}] \oplus r_{\gamma,k} \cdot \Delta_\mathsf{B}$. If so, $\mathcal{P}_\mathsf{B}$ computes and publishes $\hat{z}_\gamma = s_{\gamma,k} \oplus r_{\gamma,k}$ and $\mathsf{L}_{\gamma,\hat{z}_\gamma} := (\mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k} \cdot \Delta_\mathsf{A}) \oplus \mathsf{M}[s_{\gamma,k}]$. Eventually, $\mathcal{P}_\mathsf{B}$ publishes $(s_{\gamma,k}, (\mathsf{M}'_j[s_{\gamma,k}])_{j\in[\kappa]})$ and $\mathsf{rand}_{\gamma,k}$.

**Output Determination:**

8. $\mathcal{P}_\mathsf{A}$ publishes $(r_w, \mathsf{M}[r_w], (\mathsf{M}'_j[r_w])_{j\in[\kappa]})_{w\in\mathcal{I}_\mathsf{O}}$. $\mathcal{P}_\mathsf{B}$ checks whether $\mathsf{M}[r_w] = \mathsf{K}[r_w] \oplus r_w \cdot \Delta_\mathsf{B}$. Then, $\mathcal{P}_\mathsf{B}$ computes $z_w := \hat{z}_w \oplus r_w \oplus s_w$ and publishes $(s_w, (\mathsf{M}'_j[s_w])_{j\in[\kappa]})_{w\in\mathcal{I}_\mathsf{O}}$ and $(z_w)_{w\in\mathcal{I}_\mathsf{O}}$.

**Proof Proceeding:**

9. Each party sends $(\mathsf{get})$ to $\mathcal{F}_{\mathsf{sVOLE\text{-}2PC}}$ (c.f. Figure 7) to obtain $(\nabla_{\mathsf{A},j})_{j\in[\kappa]}$, $(\mathsf{K}'_j[r_w])_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}$, $(\mathsf{K}'_j[r'_w])_{w\in\mathcal{I}_\mathsf{W}}$, $(\mathsf{K}'_j[\tilde{a}_{\mathsf{A},j}^{(\ell)}], \mathsf{K}'_j[\tilde{b}_{\mathsf{A},j}^{(\ell)}], \mathsf{K}'_j[\tilde{c}_{\mathsf{A},j}^{(\ell)}])_{j\in[\kappa],\ell\in[L]}$, $(\nabla_{\mathsf{B},j})_{j\in[\kappa]}$, $(\mathsf{K}'_j[s_w])_{w\in\mathcal{I}_\mathsf{A}\cup\mathcal{I}_\mathsf{B}\cup\mathcal{I}_\mathsf{W}}$, $(\mathsf{K}'_j[s'_w])_{w\in\mathcal{I}_\mathsf{W}}$, $(\mathsf{K}'_j[\tilde{a}_{\mathsf{B},j}^{(\ell)}], \mathsf{K}'_j[\tilde{b}_{\mathsf{B},j}^{(\ell)}], \mathsf{K}'_j[\tilde{c}_{\mathsf{B},j}^{(\ell)}])_{j\in[\kappa],\ell\in[L]}$.
10. Each party permutes $(\mathsf{K}'_j[a_{p,j}^{(\ell)}], \mathsf{K}'_j[b_{p,j}^{(\ell)}], \mathsf{K}'_j[c_{p,j}^{(\ell)}]) := (\mathsf{K}'_j[\tilde{a}_{p,j}^{(\pi_j(\ell))}], \mathsf{K}'_j[\tilde{b}_{p,j}^{(\pi_j(\ell))}], \mathsf{K}'_j[\tilde{c}_{p,j}^{(\pi_j(\ell))}])\ \forall j \in [\kappa], \forall p \in \{\mathsf{A}, \mathsf{B}\}, \forall \ell \in [L]$.
11. $\forall j$, $\forall i \in [W\cdot\mathsf{bs}+1, W\cdot\mathsf{bs}+\mathsf{rm}]$, $\forall p \in \{\mathsf{A}, \mathsf{B}\}$, each party checks whether $(a_{\mathsf{A},j}^{(i)} \oplus a_{\mathsf{B},j}^{(i)}) \cdot (b_{\mathsf{A},j}^{(i)} \oplus b_{\mathsf{B},j}^{(i)}) = c_{\mathsf{A},j}^{(i)} \oplus c_{\mathsf{B},j}^{(i)}$, $\mathsf{M}'_j[a_{p,j}^{(i)}] = \mathsf{K}'_j[a_{p,j}^{(i)}] \oplus a_{p,j}^{(i)} \cdot \nabla_{p',j}$, $\mathsf{M}'_j[b_{p,j}^{(i)}] = \mathsf{K}'_j[b_{p,j}^{(i)}] \oplus b_{p,j}^{(i)} \cdot \nabla_{p',j}$ and $\mathsf{M}'_j[c_{p,j}^{(i)}] = \mathsf{K}'_j[c_{p,j}^{(i)}] \oplus c_{p,j}^{(i)} \cdot \nabla_{p',j}$ where $p' = \mathsf{B}$, if $p = \mathsf{A}$, and $p' = \mathsf{A}$, otherwise.
12. Each party works as follows.
    (a) For each XOR gate $(\alpha, \beta, \gamma, \oplus)$, $\forall j \in [\kappa]$, each party computes $\mathsf{K}'_j[r_\gamma] := \mathsf{K}'_j[r_\alpha] \oplus \mathsf{K}'_j[r_\beta]$ (resp., $\mathsf{K}'_j[s_\gamma] := \mathsf{K}'_j[s_\alpha] \oplus \mathsf{K}'_j[s_\beta]$).
    (b) For each AND gate $(\alpha, \beta, \gamma, \cdot)$, $\forall k \in [0, 3]$, let $k := \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$, $\forall j \in [\kappa]$. Each party computes

$$\mathsf{K}'_j[r_{\gamma,k}] := \mathsf{K}'_j[r'_\gamma] \oplus \mathsf{K}'_j[r_\gamma] \oplus \hat{z}_\alpha \cdot \mathsf{K}'_j[r_\beta] \oplus \hat{z}_\beta \cdot \mathsf{K}'_j[r_\alpha] \text{ and } \mathsf{K}'_j[s_{\gamma,k}] := \mathsf{K}'_j[s'_\gamma] \oplus \mathsf{K}'_j[s_\gamma] \oplus \hat{z}_\alpha \cdot \mathsf{K}'_j[s_\beta] \oplus \hat{z}_\beta \cdot \mathsf{K}'_j[s_\alpha] + \hat{z}_\alpha \cdot \hat{z}_\beta \cdot \nabla_{\mathsf{A},j}.$$

13. For $k \in [\mathsf{bs}]$, run step 2 of protocol $\Pi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}$ (c.f. Figure 6) with inputs

$$((\mathsf{K}'_j[r_\alpha], \mathsf{K}'_j[r_\beta], \mathsf{K}'_j[r'_\gamma])_{j\in[\kappa],\ \text{all } (\alpha,\beta,\gamma,\cdot)}, (\mathsf{K}'_j[a_{\mathsf{A},j}^{(i)}], \mathsf{K}'_j[b_{\mathsf{A},j}^{(i)}], \mathsf{K}'_j[c_{\mathsf{A},j}^{(i)}])_{j\in[\kappa], i\in[(k-1)\cdot W+1, k\cdot W]}),$$
$$((\mathsf{K}'_j[s_\alpha], \mathsf{K}'_j[s_\beta], \mathsf{K}'_j[s'_\gamma])_{j\in[\kappa],\ \text{all } (\alpha,\beta,\gamma,\cdot)}, (\mathsf{K}'_j[a_{\mathsf{B},j}^{(i)}], \mathsf{K}'_j[b_{\mathsf{B},j}^{(i)}], \mathsf{K}'_j[c_{\mathsf{B},j}^{(i)}])_{j\in[\kappa], i\in[(k-1)\cdot W+1, k\cdot W]})$$

    for verifying the validity of authenticated values used for proving AND gates.
14. Each party checks whether all of the followings hold for all $j \in [\kappa]$.

$$\mathsf{M}'_j[r_w] = \mathsf{K}'_j[r_w] \oplus r_w \cdot \nabla_{\mathsf{B},j}\ \forall w \in \mathcal{I}_\mathsf{A}, \quad \mathsf{M}'_j[s_w] = \mathsf{K}'_j[s_w] \oplus s_w \cdot \nabla_{\mathsf{A},j}\ \forall w \in \mathcal{I}_\mathsf{B},$$
$$\mathsf{M}'_j[r_w] = \mathsf{K}'_j[r_w] \oplus r_w \cdot \nabla_{\mathsf{B},j}\ \forall w \in \mathcal{I}_\mathsf{O}, \quad \mathsf{M}'_j[s_w] = \mathsf{K}'_j[s_w] \oplus s_w \cdot \nabla_{\mathsf{A},j}\ \forall w \in \mathcal{I}_\mathsf{O},$$

    For each AND gate $(\alpha, \beta, \gamma, \cdot)$: $\mathsf{M}'_j[r_{\gamma,k}] = \mathsf{K}'_j[r_{\gamma,k}] \oplus r_{\gamma,k} \cdot \nabla_{\mathsf{B},j}$ and $\mathsf{M}'_j[s_{\gamma,k}] = \mathsf{K}'_j[s_{\gamma,k}] \oplus s_{\gamma,k} \cdot \nabla_{\mathsf{A},j}$ where $k = \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$.

15. For each AND gate $(\alpha, \beta, \gamma, \cdot)$: each party checks whether $\mathsf{cm}_{\mathsf{B},\gamma,k}$ opens to $(s_{\gamma,k}, (\mathsf{M}'_j[s_{\gamma,k}])_j)$ w.r.t. $\mathsf{rand}_{\gamma,k}$ where $k = \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$.
16. From the published $\hat{z}_w$ for all possible wire $w$, each party can check whether $z_w = \hat{z}_w \oplus r_w \oplus s_w$ for $w \in \mathcal{I}_\mathsf{O}$.

**Fig. 11.** Protocol $\Pi_{\mathsf{pa\text{-}2PC\text{-}eval}}$ for Evaluation for Publicly Auditable 2PC.

$$\mathrm{CMT} = \Big( (\mathsf{vc}_j^{\mathcal{P},\tau,N})_{\mathcal{P}\in\{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}\}, j\in[\kappa]}, (G_{w,k}, \mathsf{cm}_{\mathsf{B},w,k})_{w\in\mathcal{I}_\mathsf{W}, k\in[0,3]} \Big), \qquad \mathrm{CH}_1 = (\pi_j)_{j\in[\kappa]} \leftarrow h_{\mathsf{FS}}^{(1)}(\kappa, \rho, \tau, \mathsf{bs}, \mathsf{rm}, |\mathcal{I}_\mathsf{A}|, |\mathcal{I}_\mathsf{B}|, W, |\mathcal{I}_\mathsf{O}|, \mathrm{CMT}),$$

$$\begin{aligned}\mathrm{RSP}_1 = \Big( & (a_{p,j}^{(i)}, b_{p,j}^{(i)}, c_{p,j}^{(i)}, \mathsf{M}'_j[a_{p,j}^{(i)}], \mathsf{M}'_j[b_{p,j}^{(i)}], \mathsf{M}'_j[c_{p,j}^{(i)}])_{p\in\{\mathsf{A},\mathsf{B}\}, j\in[\kappa], i\in[W\cdot\mathsf{bs}+1, W\cdot\mathsf{bs}+\mathsf{rm}]}, (\mathsf{msgand}_k)_{k\in[\mathsf{bs}]}, \\ & (r_w, \mathsf{M}[r_w], (\mathsf{M}'_j[r_w])_{j\in[\kappa]}, \hat{z}_w, \mathsf{L}_{w,\hat{z}_w})_{w\in\mathcal{I}_\mathsf{A}}, (s_w, \mathsf{M}[s_w], (\mathsf{M}'_j[s_w])_{j\in[\kappa]}, \hat{z}_w, \mathsf{L}_{w,\hat{z}_w})_{w\in\mathcal{I}_\mathsf{B}}, \\ & \big(r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], (\mathsf{M}'[r_{\gamma,k}])_{j\in[\kappa]}, \mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k}\cdot \varDelta_\mathsf{A}, \hat{z}_\gamma, \mathsf{L}_{\gamma,\hat{z}_\gamma}, s_{\gamma,k}, (\mathsf{M}'_j[s_{\gamma,k}])_{j\in[\kappa]}, \mathsf{rand}_{\gamma,k}\big)_{\text{all } (\alpha,\beta,\gamma,\cdot) \text{ and } k=\hat{z}_\alpha+2\cdot\hat{z}_\beta}, \\ & (r_w, \mathsf{M}[r_w], (\mathsf{M}'[r_w])_{j\in[\kappa]}, s_w, (\mathsf{M}'_j[s_w])_{j\in[\kappa]}, z_w)_{w\in\mathcal{I}_\mathsf{O}} \Big),\end{aligned}$$

$$\mathrm{CH}_2 = (\nabla_{\mathsf{A},j}, \nabla_{\mathsf{B},j})_{j\in[\kappa]} \leftarrow h_{\mathsf{FS}}^{(2)}(\mathrm{CH}_1, \mathrm{RSP}_1), \qquad \mathrm{RSP}_2 = (\mathsf{vcopen}_j^{\mathcal{P},\tau,N})_{\mathcal{P}\in\{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B}\}, j\in[\kappa]}.$$

where

- $\mathrm{CH}_1$ and $\mathrm{CH}_2$ are $\mathcal{V}$'s messages generated from random oracles heuristically realized by hash functions $h_{\mathsf{FS}}^{(1)}$ and $h_{\mathsf{FS}}^{(2)}$;
- $\mathsf{vc}_j^{\mathcal{P},\tau,N}$ and $\mathsf{vcopen}_j^{\mathcal{P},\tau,N}$ are the vector commitment and its all-but-one opening for generating VOLEitH correlations in the $j$-th call to $\varPi_{\mathsf{sVOLE\text{-}2PC}}$ by $\mathcal{P} \in \{\mathcal{P}_\mathsf{A}, \mathcal{P}_\mathsf{B}\}$ and $\mathcal{V}$; and
- $\mathsf{msgand}_k$ is the message published from running step 1 in the $k$-th call to $\varPi_{\mathsf{check\text{-}AND}}^{\mathcal{P}_\mathsf{A},\mathcal{P}_\mathsf{B},\mathcal{V},\kappa,W}$.

**Fig. 12.** Transcript of $\varPi_{\mathsf{pa\text{-}2PC}} = (\varPi_{\mathsf{pa\text{-}2PC\text{-}pre}}, \varPi_{\mathsf{pa\text{-}2PC\text{-}eval}})$ in Figures 10 and 11.

**Inputs:** Both parties $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ agree on a common circuit for function $f : \{0,1\}^{|\mathcal{I}_{\mathsf{A}}|} \times \{0,1\}^{|\mathcal{I}_{\mathsf{B}}|} \to \{0,1\}^{|\mathcal{I}_{\mathsf{O}}|}$. In the input-preprocessing phase, $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ holds $\mathbf{x} = (z_i)_{i \in \mathcal{I}_{\mathsf{A}}} \in \{0,1\}^{|\mathcal{I}_{\mathsf{A}}|}$ and $\mathbf{y} = (z_i)_{i \in \mathcal{I}_{\mathsf{B}}} \in \{0,1\}^{|\mathcal{I}_{\mathsf{B}}|}$, respectively. Moreover, both parties also agree on a common parameter $\rho \in \mathbb{N}$.

**Function-Independent Preprocessing:**

- $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ send $(\mathsf{init})$ to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$ to receive $\Delta_{\mathsf{A}} \in \mathbb{F}_{2^\rho}$ and $\Delta_{\mathsf{B}} \in \mathbb{F}_{2^\rho}$, respectively.
- For each $w \in \mathcal{I}_{\mathsf{A}} \cup \mathcal{I}_{\mathsf{B}} \cup \mathcal{I}_{\mathsf{W}}$, both $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$ send $\mathsf{random}$ to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$. In return, $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$ sends $(r_w, \mathsf{M}[r_w], \mathsf{K}[s_w])$ and $(s_w, \mathsf{M}[s_w], \mathsf{K}[r_w])$ to $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$, respectively. Define $\lambda_w := r_w \oplus s_w$. $\mathcal{P}_{\mathsf{A}}$ samples $\mathsf{L}_{w,0} \xleftarrow{\$} \mathbb{F}_{2^\rho}$.

**Function-Dependent Preprocessing:**

- For each XOR gate $(\alpha, \beta, \gamma, \oplus)$, $\mathcal{P}_{\mathsf{A}}$ computes $(r_\gamma, \mathsf{M}[r_\gamma], \mathsf{K}[s_\gamma]) := (r_\alpha \oplus r_\beta, \mathsf{M}[r_\alpha] \oplus \mathsf{M}[r_\beta], \mathsf{K}[s_\alpha] \oplus \mathsf{K}[s_\beta])$ and $\mathsf{L}_{\gamma,0} := \mathsf{L}_{\alpha,0} \oplus \mathsf{L}_{\beta,0}$. $\mathcal{P}_{\mathsf{B}}$ computes $(s_\gamma, \mathsf{M}[s_\gamma], \mathsf{K}[r_\gamma]) := (s_\alpha \oplus s_\beta, \mathsf{M}[s_\alpha] \oplus \mathsf{M}[s_\beta], \mathsf{K}[r_\alpha] \oplus \mathsf{K}[r_\beta])$.
- For each AND gate $(\alpha, \beta, \gamma, \cdot)$:
  - $\mathcal{P}_{\mathsf{A}}$ (resp., $\mathcal{P}_{\mathsf{B}}$) sends $(\mathsf{and}, (r_\alpha, \mathsf{M}[r_\alpha], \mathsf{K}[s_\alpha]), (r_\beta, \mathsf{M}[r_\beta], \mathsf{K}[s_\beta]))$ (resp., $(\mathsf{and}, (s_\alpha, \mathsf{M}[s_\alpha], \mathsf{K}[r_\alpha]), (s_\beta, \mathsf{M}[s_\beta], \mathsf{K}[r_\beta])))$ to $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$. In return, $\mathcal{F}_{\mathsf{pre}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$ sends $(r'_\gamma, \mathsf{M}[r'_\gamma], \mathsf{K}[s'_\gamma])$ and $(s'_\gamma, \mathsf{M}[s'_\gamma], \mathsf{K}[r'_\gamma])$ respectively to $\mathcal{P}_{\mathsf{A}}$ and $\mathcal{P}_{\mathsf{B}}$. Here, $r'_\gamma \oplus s'_\gamma = \lambda_\alpha \cdot \lambda_\beta$.
  - $\forall k \in [0,3]$, let $(k_0, k_1) := \mathsf{bin}_2(k) \in \{0,1\}^2$, i.e., $k = k_0 + 2 \cdot k_1$, and $\mathcal{P}_{\mathsf{A}}$ computes

$$\begin{aligned} r_{\gamma,k} &:= r'_\gamma \oplus r_\gamma \oplus k_0 \cdot r_\beta \oplus k_1 \cdot r_\alpha, \\ \mathsf{M}[r_{\gamma,k}] &:= \mathsf{M}[r'_\gamma] \oplus \mathsf{M}[r_\gamma] \oplus k_0 \cdot \mathsf{M}[r_\beta] \oplus k_1 \cdot \mathsf{M}[r_\alpha], \\ \mathsf{K}[s_{\gamma,k}] &:= \mathsf{K}[s'_\gamma] \oplus \mathsf{K}[s_\gamma] \oplus k_0 \cdot \mathsf{K}[s_\beta] \oplus k_1 \cdot \mathsf{K}[s_\alpha] \oplus k_0 \cdot k_1 \cdot \Delta_{\mathsf{A}}. \end{aligned}$$

  - $\forall k \in [0,3]$, let $(k_0, k_1) := \mathsf{bin}_2(k) \in \{0,1\}^2$, i.e., $k = k_0 + 2 \cdot k_1$, and $\mathcal{P}_{\mathsf{B}}$ computes

$$\begin{aligned} s_{\gamma,k} &:= s'_\gamma \oplus s_\gamma \oplus k_0 \cdot s_\beta \oplus k_1 \cdot s_\alpha \oplus k_0 \cdot k_1, \\ \mathsf{M}[s_{\gamma,k}] &:= \mathsf{M}[s'_\gamma] \oplus \mathsf{M}[s_\gamma] \oplus k_0 \cdot \mathsf{M}[s_\beta] \oplus k_1 \cdot \mathsf{M}[s_\alpha], \\ \mathsf{K}[r_{\gamma,k}] &:= \mathsf{K}[r'_\gamma] \oplus \mathsf{K}[r_\gamma] \oplus k_0 \cdot \mathsf{K}[r_\beta] \oplus k_1 \cdot \mathsf{K}[r_\alpha]. \end{aligned}$$

  - $\mathcal{P}_{\mathsf{A}}$ computes $\mathsf{L}_{\alpha,1} := \mathsf{L}_{\alpha,0} \oplus \Delta_{\mathsf{A}}$ and $\mathsf{L}_{\beta,1} := \mathsf{L}_{\beta,0} \oplus \Delta_{\mathsf{A}}$. $\mathcal{P}_{\mathsf{A}}$ sends the following to $\mathcal{P}_{\mathsf{B}}$, for all $k \in [0,3]$ with $(k_0, k_1) := \mathsf{bin}_2(k) \in \{0,1\}^2$:

$$G_{\gamma,k} := H(\mathsf{L}_{\alpha,k_0}, \mathsf{L}_{\beta,k_1}, \gamma, k) \oplus (r_{\gamma,k}, \mathsf{M}[r_{\gamma,k}], \mathsf{L}_{\gamma,k} \oplus \mathsf{K}[s_{\gamma,k}] \oplus r_{\gamma,k} \cdot \Delta_{\mathsf{A}}).$$

**Input Processing:**

- For $w \in \mathcal{I}_{\mathsf{A}}$, $\mathcal{P}_{\mathsf{A}}$ sends $(r_w, \mathsf{M}[r_w])$ to $\mathcal{P}_{\mathsf{B}}$. $\mathcal{P}_{\mathsf{B}}$ checks whether $\mathsf{M}[r_w] = \mathsf{K}[r_w] \oplus r_w \cdot \Delta_{\mathsf{B}}$. $\mathcal{P}_{\mathsf{B}}$ then sends $\hat{z}_w = z_w \oplus \lambda_w = z_w \oplus r_w \oplus s_w$ to $\mathcal{P}_{\mathsf{A}}$. Finally, $\mathcal{P}_{\mathsf{A}}$ sends $\mathsf{L}_{w,\hat{z}_w}$ to $\mathcal{P}_{\mathsf{B}}$.
- For $w \in \mathcal{I}_{\mathsf{B}}$, $\mathcal{P}_{\mathsf{B}}$ sends $(s_w, \mathsf{M}[s_w])$ to $\mathcal{P}_{\mathsf{A}}$. $\mathcal{P}_{\mathsf{A}}$ checks whether $\mathsf{M}[s_w] = \mathsf{K}[s_w] \oplus s_w \cdot \Delta_{\mathsf{A}}$. $\mathcal{P}_{\mathsf{A}}$ then sends $\hat{z}_w = z_w \oplus \lambda_w = z_w \oplus r_w \oplus s_w$ and $\mathsf{L}_{w,\hat{z}_w}$ to $\mathcal{P}_{\mathsf{B}}$.

**Circuit Evaluation:**

- $\mathcal{P}_{\mathsf{B}}$ evaluates the circuit following a topological order. For each gate $(\alpha, \beta, \gamma, \mathsf{op})$ where $\mathsf{op} \in \{\cdot, \oplus\}$, $\mathcal{P}_{\mathsf{B}}$ holds $(\hat{z}_\alpha, \mathsf{L}_{\alpha,\hat{z}_\alpha})$ and $(\hat{z}_\beta, \mathsf{L}_{\beta,\hat{z}_\beta})$ where $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$ and $\hat{z}_\beta = z_\beta \oplus \lambda_\beta$.
  - If $\mathsf{op} = \oplus$, $\mathcal{P}_{\mathsf{B}}$ computes $\hat{z}_\gamma := \hat{z}_\alpha \oplus \hat{z}_\beta$ and $\mathsf{L}_{\gamma,\hat{z}_\gamma} := \mathsf{L}_{\alpha,\hat{z}_\alpha} \oplus \mathsf{L}_{\beta,\hat{z}_\beta}$.
  - If $\mathsf{op} = \cdot$, $\mathcal{P}_{\mathsf{B}}$ computes $i := \hat{z}_\alpha + 2 \cdot \hat{z}_\beta$. Then, $\mathcal{P}_{\mathsf{B}}$ computes $(r_{\gamma,i}, \mathsf{M}[r_{\gamma,i}], \mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,i}] \oplus r_{\gamma,i} \cdot \Delta_{\mathsf{A}}) := G_{\gamma,i} \oplus H(\mathsf{L}_{\alpha,\hat{z}_\alpha}, \mathsf{L}_{\beta,\hat{z}_\beta}, \gamma, i)$. Then, $\mathcal{P}_{\mathsf{B}}$ checks whether $\mathsf{M}[r_{\gamma,i}] = \mathsf{K}[r_{\gamma,i}] \oplus r_{\gamma,i} \cdot \Delta_{\mathsf{B}}$. If so, $\mathcal{P}_{\mathsf{B}}$ computes $\hat{z}_\gamma = s_{\gamma,i} \oplus r_{\gamma,i}$ and $\mathsf{L}_{\gamma,\hat{z}_\gamma} := (\mathsf{L}_{\gamma,0} \oplus \mathsf{K}[s_{\gamma,i}] \oplus r_{\gamma,i} \cdot \Delta_{\mathsf{A}}) \oplus \mathsf{M}[s_{\gamma,i}]$.

**Output Determination:**

- For $w \in \mathcal{I}_{\mathsf{O}}$, $\mathcal{P}_{\mathsf{A}}$ sends $(r_w, \mathsf{M}[r_w])$ to $\mathcal{P}_{\mathsf{B}}$ who checks whether $\mathsf{M}[r_w] = \mathsf{L}[r_w] \oplus r_w \cdot \Delta_{\mathsf{B}}$. If so, $\mathcal{P}_{\mathsf{B}}$ computes $z_w := \hat{z}_w \oplus r_w \oplus s_w$.

**Fig. 13.** Protocol $\Pi_{\mathsf{2PC}}^{\mathcal{P}_{\mathsf{A}},\mathcal{P}_{\mathsf{B}}}$ [31, Figure 2].