Generalizing Efficient Multiparty Computation

Bernardo David[1]*, Ryo Nishimaki[2], Samuel Ranellucci[1], and Alain Tapp[3]

[1] Department of Computer Science
Aarhus University, Denmark
{bernardo,samuel}@cs.au.dk
[2] Secure Platform Laboratories
NTT, Japan
nishimaki.ryo@lab.ntt.co.jp
[3] DIRO
Université de Montréal, Canada
tappa@iro.umontreal.ca

Abstract. We focus on generalizing constructions of Batch Single-Choice Cut-And-Choose Oblivious Transfer and Multi-sender k-out-of-n Oblivious Transfer, which are at the core of efficient secure computation constructions proposed by Lindell *et al.* and the IPS compiler. Our approach consists in showing that such primitives can be based on a much weaker and simpler primitive called Verifiable Oblivious Transfer (VOT) with low overhead. As an intermediate step we construct Generalized Oblivious Transfer from VOT. Finally, we show that Verifiable Oblivious Transfer can be obtained from a structure preserving oblivious transfer protocol (SPOT) through an efficient transformation that uses Groth-Sahai proofs and structure preserving commitments.
Keywords: Oblivious Transfer, Structure Preserving Cryptography, Secure Computation, Universal Composability, Groth-Sahai Proof System, IPS compiler.

1 Introduction

Secure multiparty computation (MPC) allows mutually distrustful parties to compute functions on private data that they hold, without revealing their data to each other. Obtaining efficient multiparty computation is a highly sought after goal of cryptography since it can be employed in a multitude of practical applications, such as auctions, electronic voting and privacy preserving data analysis. Notably, it is known that secure two-party computation can be achieved from the garbled circuits technique first proposed by Yao [Yao86] and that general MPC can be obtained from a basic primitive called oblivious transfer (OT), which was introduced in [Rab81,EGL85]. The basic one-out-of-two oblivious transfer (OT_1^2) is a two-party primitive where a *sender* inputs two messages m_0, m_1 and a *receiver* inputs a bit c, referred to as the *choice bit*. The receiver learns m_c but not m_{1-c} and the sender learns nothing about the receiver's choice (*i.e.* c). This primitive was proven to be sufficient for achieving MPC in [Kil88,GMW87,CvdGT95].

Even though many approaches for constructing MPC exist, only recently methods that can be efficiently instantiated have been proposed. Among these methods, the IPS compiler [IPS08] stands out as an important construction, achieving MPC without honest majority in the OT-hybrid model. In this work, we will focus on the cut-and-choose OT based construction and the improvement of the IPS compiler introduced by Lindell et al. [LP11,LP12,LOP11,Lin13].

In the approaches for obtaining efficient MPC presented in [LP11,Lin13], the authors employ cut-and-choose OT, where the sender inputs s pairs of messages and the receiver can choose to learn both messages b_0, b_1 from $\frac{s}{2}$ input pairs, while he only learns one of the messages in the remaining pairs. A batch version of this primitive is then combined with Yao's protocol to achieve efficient MPC. In the improvement of the IPS compiler, the authors employ Multi-sender k-out-of-n OT, where j senders input a set of n messages out of which a receiver can choose to receive k messages. These complex primitives are usually constructed from

* Bernardo David and Samuel Ranellucci were supported by European Research Council Starting Grant 279447. The authors acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research centre (supported by the Danish Strategic Research Council) within which part of this work was performed.

specific number-theoretic and algebraic assumptions yielding little insight to their relationship with other generic and potentially simpler primitives.

In parallel to the efforts for obtaining efficient MPC, research has been devoted to obtaining constructions of basic primitives that can be efficiently combined between themselves in order to obtain more complex primitives and protocols. One of the main approach taken towards this goal has been called *structure preserving cryptography*, which aims at constructing primitives where basically all the public information (*e.g.* signatures, public keys, ciphertexts and commitments) are solely composed of bilinear group elements. This allows for the application of efficient Groth-Sahai non-interactive zero knowledge (NIZK) proof systems [GS08] (GS-Proofs) and efficient composition of primitives. Until now, the main results in this area have been structure preserving signature and commitment schemes [AFG$^+$10,AGHO11] and encryption [CHK$^+$11].

Our Contributions: The central goal of this paper is to present general constructions of the primitives used as the main building blocks in the frameworks of [LP11,LOP11,LP12,Lin13] in the universal composability model [Can01]. In contrast to previous works, we present *general* reductions from such complex primitives to simpler variants of OT without relying on specific number theoretic assumptions. We present three main results:

– **General constructions of Multi-sender k-out-of-n OT (MSOT) and Batch Single Choice Cut-and-Choose OT (CACOT) from Generalized OT (GOT):** We show that MSOT and CACOT can be obtained GOT [IK97] combined with proper access structures. Differently from the original constructions of [LP11,LP12,LOP11,Lin13], our constructions are based on a simple generic primitive, not requiring Committed OT or specific computational assumptions. These constructions can be readily used to instantiate the MPC frameworks presented in [LP11,LP12,LOP11,Lin13].
– **Generalized Oblivious Transfer based on Verifiable Oblivious Transfer:** Verifiable Oblivious Transfer (VOT) [CC00,JS07,KSV07] is a flavor of 1-out-of-2 OT where the sender can reveal one of his messages at any point during the protocol execution allowing the receiver to verify that this message is indeed one of the original sender's inputs. We show that GOT can be obtained from VOT, generalizing even more the constructions described before. Our generic construction of GOT may be of independent interest.
– **Structure Preserving Oblivious Transfer (SPOT) and a *generic* composable constructions of Verifiable Oblivious Transfer:** We introduce SPOT, which is basically a 1-out-of-2 OT compatible with GS-Proofs. We then build on this characteristic to provide a generic (non black-box) construction of VOT from any SPOT protocol combined with structure preserving extractable or equivocable commitments and Groth-Sahai NIZKs. Differently from the VOT protocols of [CC00,JS07,KSV07], our constructions are modular and independent of specific assumptions. Moreover, we provide a concrete round optimal SPOT protocol based on a framework by Peikert et al. [PVW08] and observe that the protocols in [CKWZ13] fit our definitions. This notion is also of independent interest in other scenarios besides general MPC.

Our contributions are two-fold, showing that efficient MPC can be based on a weaker simpler primitive (*i.e.* VOT) and providing a generic method for constructing such a primitive. Our results generalize previous approaches for efficient MPC by providing constructions ultimately based on SPOT rather than the original Multi Sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT. Such general constructions help understand the relationship between these complex variants of oblivious transfer and simpler primitives.

1.1 Efficiency

Our constructions are as efficient as the underlying NIZK proof system, structure preserving commitment and SPOT. Hence, they can easily take advantage of more efficient constructions of these primitives. In Table 1, we present an estimate of the concrete complexity of our protocols when instantiated with GS-Proofs and commitments [GS08] and our structure preserving variant of the DDH based UC secure OT of [PVW08]. Our general constructions achieve essentially the same round complexity as the previous DDH based constructions of the same funtionalities. Our constructions incur higher communication and computational overheads, which is expected since we do not optimize our protocols for an specific number theoretic assumptions as in previous works. We remark that independently of their concrete efficiency, our protocols are the first to realize MSOT and CACOT from generic primitives without relying on specific number theoretic assumptions.

Protocol		VOTs	Rounds	Computational Complexity	Communication Complexity
GOT	Sec. 4	n	6	$23n$ Exp. $+28$ Pair.	$24n+k+4$
CACOT	Sec. 5	$2ns$	6	$46ns$ Exp. $+56ns$ Pair.	$48.5ns$ $+n+4$
	[LP11]	-	6	$11.5ns+19n$ $+9s+5$ Exp.	$5ns+11n$ $+5s+5$
Modified CACOT	Sec. 6	$4ns+s$	6	$92ns+23s$ Exp. $112ns+28s$ Pair.	$16ns+16s$ $+k+4$
	[Lin13]	-	21	$10.5ns+20.5ns$ $+n+26$ Exp.	$5ns+n$ $+11s+15$
MSOT	Sec. 7	pn	8	$23pn$ Exp. $+28pn$ Pair.	$24pn+4p+k$ $p!/(p-3)!+4$
	[LOP11]	-	7	$4n+11(p-1)n$ $+k(p-1)$ Exp.	$12pn+1$

Table 1. Efficiency of our protocols compared to previous constructions based on DDH. The column VOTs shows the number of VOTs needed in our general constructions, "-" marks the previous protocols that do not enjoy general constructions. Exp. stands for exponentiations and Pair. stands for bilinear pairings. n and s express the number of inputs according to each protocol (explained in the respective sections), p is the number of senders in MSOT and k is the number of messages transferred to the receiver. Communication complexity is stated in terms of number of group elements exchanged.

2 Preliminaries

Notations and Conventions. For any $n \in \mathbb{N} \setminus \{0\}$, let $[n]$ be the set $\{1, \ldots, n\}$. When D is a random variable or distribution, $y \overset{\mathsf{R}}{\leftarrow} D$ denotes that y is randomly selected from D according to its distribution. If S is a set, then $x \overset{\mathsf{U}}{\leftarrow} S$ denotes that x is uniformly selected from S. $y := z$ denotes that y is set, defined or substituted by z. When b is a fixed value, $A(x) \rightarrow b$ (e.g., $A(x) \rightarrow 1$) denotes the event that machine (or algorithm) A outputs b on input x. We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible in $\lambda \in \mathbb{N}$ if for every constant $c \in \mathbb{N}$ there exists $k_c \in \mathbb{N}$ such that $f(\lambda) < \lambda^{-c}$ for any $\lambda > k_c$. Hereafter, we use $f < \mathsf{negl}(\lambda)$ to mean that f is negligible in λ. We write $\mathcal{X} \overset{\mathsf{c}}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.

Bilinear Groups. Let $\mathcal{G}$ be a bilinear group generator that takes security parameter 1^λ as input and outputs a description of bilinear groups $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g})$ where $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ are groups of prime order p, g and $\hat{g}$ are generators in $\mathbb{G}$ and $\mathbb{H}$, respectively, e is an efficient and non-degenerate map $e : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$. If $\mathbb{G} = \mathbb{H}$, then we call it the symmetric setting. If $\mathbb{G} \neq \mathbb{H}$ and there is no efficient mapping between the groups, then we call it the asymmetric setting.

Symmetric External Decisional Diffie-Hellman Assumption. Intuitively, SXDH is the assumption that the DDH assumption holds for both groups $\mathbb{G}$ and $\mathbb{H}$ in a bilinear group Λ. Let $\mathcal{G}^{\mathsf{DDH1}}(1^\lambda)$ be an algorithm that on input security parameter λ, generates parameters $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^\lambda)$ (where $\mathcal{G}$ is the bilinear group generator introduced in the previous paragraph.), chooses exponents $x, y, z \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, and outputs $\boldsymbol{I} := (\Lambda, g^x, g^y)$ and (x, y, z). When an adversary is given $\boldsymbol{I} \overset{\mathsf{R}}{\leftarrow} \mathcal{G}^{\mathsf{DDH1}}(1^\lambda)$ and $T \in \mathbb{G}$, it attempts to distinguish whether $T = g^{xy}$ or $T = g^z$. This is called the DDH1 problem. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda)$ is defined as follows:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda) := \Big| &\Pr\Big[\mathcal{A}(\boldsymbol{I}, g^{xy}) \rightarrow 1 \Big| (\boldsymbol{I}, x, y, z) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}^{\mathsf{DDH1}}(1^\lambda);\Big] \\ &- \Pr\Big[\mathcal{A}(\boldsymbol{I}, g^{z}) \rightarrow 1 \Big| (\boldsymbol{I}, x, y, z) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}^{\mathsf{DDH1}}(1^\lambda);\Big]\Big|\end{aligned}$$

Definition 1 (DDH1 assumption). *We say that the DDH1 assumption holds if for all PPT (Probabilistic Polynomial Time) adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda) < \mathsf{negl}(\lambda)$.*

The DDH2 assumption is similarly defined in terms of group $\mathbb{H}$. If both DDH1 and DDH2 assumptions hold simultaneously, then we say that the symmetric external Diffie-Hellman (SXDH) assumption holds.

2.1 Universal Composability

The Universal Composability framework was introduced by Canetti in [Can01] to analyse the security of cryptographic protocols and primitives under arbitrary composition. In this framework, protocol security is analysed by comparing an ideal world execution and a real world execution under the supervision of an *environment* $\mathcal{Z}$, which is represented by a *PPT* machine and has access to all communication between individual parties. In the ideal world execution, dummy parties (possibly controlled by a *PPT simulator*) interact directly with the ideal functionality $\mathcal{F}$, which works as a fully secure third party that computes the desired function or primitive. In the real world execution, several *PPT* parties (possibly corrupted by a real world adversary $\mathcal{A}$) interact with each other by means of a protocol π that realizes the ideal functionality. The real world execution is represented by the ensemble $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, while the ideal execution is represented by the $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ is not able to efficiently distinguish between $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, thus implying that the real world protocol is as secure as the ideal functionality. It is known that a setup assumption is needed for UC realizing oblivious transfer as well as most "interesting" ideal functionalities [CF01]. In this work we consider security against static adversaries, *i.e.* the adversary can only corrupt parties before the protocol execution starts. We consider malicious adversaries that may deviate from the protocol in any arbitrary way. See [Can01] for further details.

Definition 2. *A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ if, for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that, for every environment $\mathcal{Z}$, the following holds:*

$$\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$$

We present oblivious transfer ($\mathcal{F}_{OT}$), commitment ($\mathcal{F}_{COM}$), and common reference string ($\mathcal{F}_{CRS}^{\mathcal{D}}$) ideal functionalities in Appendix A.

3 Generic Construction of Verifiable OT from Structure Preserving OT

In this section, we introduce Structure Preserving Oblivious Transfer (SPOT) and use it to construct verifiable oblivious transfer (VOT). SPOT will be the basic building block of our efficient general constructions of secure multiparty computation. It is first used in a general transformation that yields VOT, which will be used to obtain generalized oblivious transfer (GOT) and more complex primitives later on.

Structure Preserving Oblivious Transfer

Basically we require all the SPOT protocol messages (*i.e.* the protocol transcript) and inputs to be composed solely of group elements and the transcript to be generated from the inputs by pairing product equations or multi exponentiation equations, which allows us to apply GS proofs to prove relations between the parties' inputs and the protocol transcript. Further on, our general transformation will rely on GS proofs to show that a given sender input is associated with a specific protocol transcript.

Definition 3 (Structure Preserving Oblivious Transfer). *A structure preserving oblivious transfer protocol taking inputs m_0, m_1 from the sender and c from the receiver defined over a bilinear group $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g})$ must have the following properties:*

1. *Each of the sender's input messages m_0, m_1 consists of elements of $\mathbb{G}$ or $\mathbb{H}$.*
2. *All the messages exchanged between* **S** *and* **R** *(i.e. the protocol transcript) consist of elements of $\mathbb{G}$ and $\mathbb{H}$.*
3. *The relation between the protocol inputs m_0, m_1, c and a given protocol transcript is expressed by a set of pairing product equations or multi exponentiation equations.*

Notice that our general transformations can be applied to any OT protocol in a bit by bit approach, by mapping the binary representation of each element in a given protocol to specific group elements representing 0 and 1 and applying GS proofs individually to each of those elements. However, this trivial approach is extremely inefficient. The number of GS proofs and group elements exchanged between parties would grow polynomially. The first OT protocol to fit this definition was proposed in [GH08], but it relies simultaneously on the SXDH, the DLIN and the q-hidden LSRW assumptions. A recent result by Choi *et. al.* [CKWZ13] also introduced OT protocols based on DLIN and SXDH that match out definition of SPOT. However, these protocols already require a GS proof themselves, introducing extra overhead in applications that combine SPOT with GS proofs.

Obtaining SPOT from Dual-Mode Cryptosystems

The starting point for constructing SPOT is the general framework for universally composable oblivious transfer protocols proposed by Peikert *et al.* [PVW08] (hanceforth called PVW). The PVW framework provides a black-box construction of UC secure OT from dual-mode cryptosystems, which were initially instantiated under the DDH, QR and LWE assumptions. Essentially, this framework relies on an information theoretical reduction from UC secure OT to dual-mode cryptosystems in the CRS model, such that the resulting OT protocol inherits the characteristics of the underlying dual-mode cryptosystem. In order to obtain an OT protocol compatible with GS-proofs, we convert the DDH based dual-mode cryptosystem construction of [PVW08] into a scheme secure under the SXDH assumption (which can also be used to instantiate GS proofs). This scheme is then plugged in the PVW framework to obtain a UC secure OT protocol. Note that, in the resulting protocol, the CRS, all protocol messages and inputs are composed solely by group elements. Moreover, all the protocol messages are generated by pairing product equations. Therefore, we obtain a SPOT protocol whose security follows from the PVW framework. Our SXDH dual-mode cryptosystem is constructed as follows:

- $\mathsf{SetupMessy}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{\mathsf{R}} \mathcal{G}(1^\lambda)$, $g_0, g_1 \xleftarrow{\mathsf{U}} \mathbb{G}$, $x_0, x_1 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ where $x_0 \neq x_1$. Let $h_b := g_b^{x_b}$ for $b \in \{0,1\}$, $\mathsf{crs} := (g_0, h_0, g_1, h_1)$, and $t := (x_0, x_1)$. It outputs (crs, t).
- $\mathsf{SetupDec}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{\mathsf{R}} \mathcal{G}(1^\lambda)$, $g_0 \xleftarrow{\mathsf{U}} \mathbb{G}$, $y \xleftarrow{\mathsf{U}} \mathbb{Z}_p^*$, $g_1 := g_0^y$, $x \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, $h_b := g_b^x$ for $b \in \{0,1\}$, $\mathsf{crs} := (g_0, h_0, g_1, h_1)$, and $t := y$. It outputs (crs, t).
- $\mathsf{Gen}(\sigma)$ $r \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, $g := g_\sigma^r$, $h := h_\sigma^r$, $pk := (g, h) \in \mathbb{G}^2$, $sk := r$. It outputs (pk, sk).
- $\mathsf{Enc}(pk, b, m)$ For $pk = (g, h)$ and message $m \in \mathbb{G}$, reads (g_b, h_b) from $\mathsf{crs} = (g_0, h_0, g_1, h_1)$, chooses $s, t \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, and computes $u = g_b^s h_b^t$, $v = g^s h^t$. It outputs ciphertext $(u, v \cdot m) \in \mathbb{G}^2$.
- $\mathsf{Dec}(sk, c)$ $c = (c_0, c_1)$, It outputs c_1 / c_0^{sk}.
- $\mathsf{FindMessy}(t, pk)$ For input $t = (x_0, x_1)$ where $x_0 \neq x_1$, $pk = (g, h)$, if $h \neq g^{x_0}$, then it outputs $b = 0$ as a messy branch. Otherwise, we have $h = g^{x_0} \neq g^{x_1}$, so it outputs $b = 1$ as a messy branch.
- $\mathsf{TrapGen}(t)$ For input $t = y$, it chooses $r \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, computes $pk := (g_0^r, h_0^r)$ and outputs $(pk, sk := r, sk_1 := r/y)$.

Theorem 1. *The cryptosystem described above is a Dual-Mode Cryptosystem according to Definition 9 under the SXDH Assumption.*

The proof of this theorem and details of the PVW framework can be found in Appendix C, where we also describe how to use GS-proofs to prove relations between protocol inputs and transcripts.

Obtaining VOT

Verifiable oblivious transfer is basically a 1-out-of-2 oblivious transfer where the sender may choose to open one of its input messages m_b where $b \in \{0,1\}$ at any time, in such a way that the receiver is able to verify that this message had indeed been provided as input. This notion is formalized by the following ideal functionality:

Functionality $\mathcal{F}_{VOT}$

$\mathcal{F}_{VOT}$ interacts with a sender **S** a receiver **R** and an adversary $\mathcal{S}$.

- Upon receiving $(\mathsf{Send}, sid, ssid, x_0, x_1)$ from the **S**, if the pair $sid, ssid$ has not been used, store ($sid, ssid, x_0, x_1)$ and send $(\mathsf{Receipt}, sid, ssid)$ to **S**,**R** and $\mathcal{S}$.
- Upon receiving $(\mathsf{Transfer}, sid, ssid, c)$ from **R**, check if a $(\mathsf{Transfer}, sid, ssid)$ message has already been sent, if not, send $(\mathsf{transferred}, sid, ssid, x_c)$ to the receiver and $(\mathsf{transferred}, sid, ssid,)$ to $\mathcal{S}$, otherwise ignore the message.
- Upon receiving $(\mathsf{Open}, sid, ssid, b)$ from the sender, send $(\mathsf{reveal}, sid, ssid, b, x_b)$ to the receiver.

We will construct a general protocol π_{VOT} that realizes $\mathcal{F}_{VOT}$ from any universally composable SPOT protocol π_{SPOT} by combining it with a structure preserving commitment π_{COM} (such as the schemes in [GS08][AFG⁺10]) and Groth-Sahai NIZK proofs. An interesting property of this generic protocol is that even though it was designed for an underlying structure preserving protocol that realizes the 1-out-of-2 OT functionality $\mathcal{F}_{OT}$, it can be applied multiple times to the individual transfers of an adaptive OT protocol in order to obtain verifiable adaptive OT. In this case, the same CRS can be reused for all the individual transfers. Notice that this is the first generic construction of universally composable VOT.

We assume that both parties are running the underlying universally composable structure preserving oblivious transfer protocol $SPOT$ and describe the extra steps needed to obtain VOT. In the context of π_{COM}, we denote commitment to a message m by $\mathsf{Com}(m)$ and the opening of such a commitment by $\mathsf{Open(m)}$.

Protocol $\boldsymbol{\pi_{VOT}}$: **S** inputs two messages m_0, m_1 and **R** inputs a choice bit c.

- **Setup:** A common reference string is generated containing the following information:
  - The description of a bilinear group $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g})$.
  - The public parameters for an instance of a Groth-Sahai non-interactive zero knowledge proof system.
  - The CRS for the underlying structure preserving commitment scheme π_{COM}.
  - The CRS for the underlying UC structure preserving OT π_{SPOT}.
- **Commitment phase:** Before starting π_{SPOT}, **S** commits to m_0 and m_1 by sending $(sid, ssid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ to **R**, where $m_0, m_1 \in \{0,1\}^n$ (Notice that it is possible to efficiently map the messages into corresponding group elements that will serve as inputs to π_{SPOT} [GH08]).
- π_{SPOT} **protocol execution**: **S** and **R** run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} with **S**'s input (m_0, m_1) and **R**'s input c or until **S** decides to reveal one of its messages.
- **Reveal phase:** If **S** decides to reveal one of its messages m_b where $b \in \{0,1\}$ at any point of the protocol execution it sends a decommitment to m_b and a GS-proof ψ that the messages exchanged up to that point of the execution contain a valid transfer of message m_b, sending $(sid, ssid, b, \mathsf{Open(m_b)}, \psi)$ to **R**.
- **Verification phase:** After receiving the decommitment and the GS-proof, **R** verifies ψ and the decommitment validity. If both are valid, it accepts the revealed bit, otherwise it detects that **S** is cheating. If the protocol π_{SPOT} did not reach its end yet, **S** and **R** continue by executing the next steps, otherwise they halt.

Theorem 2. *For every universally composable structure preserving oblivious transfer protocol π_{SPOT} and every universally composable structure preserving commitment scheme π_{Com}, Protocol π_{VOT} securely realizes the functionality F_{VOT} in the $\mathcal{F}_{CRS}$ hybrid model under the assumption that Groth-Sahai proof systems are Zero Knowledge Proofs of Knowledge.*

Before proceeding to the security proof we show that the protocol works correctly. First of all, notice that since π_{SPOT} is a structure preserving oblivious transfer protocol it is possible to prove statements about the sender's input messages and the protocol transcript using Groth-Sahai NIZK proof systems. Correctness of Protocol π_{VOT} in the case that no Reveal phase happens follows from the correctness of protocol π_{SPOT}. The correctness of the Reveal phase follows from the commitment scheme's security and the GS-proof completeness and soundness. When **S** opens the commitment, **R** is able to check whether the revealed message is indeed one of the messages that **S** used as input in the beginning of the protocol and by verifying the GS-proof, **R** is able to check that the input message m_b is contained in the messages exchanged by both parties meaning that this message is indeed used in the protocol execution. The full proof is presented in Appendix D.

4 Generalized Oblivious Transfer

Generalized Oblivious Transfer is an interesting application of Verifiable Oblivious Transfer. An interesting way of describing an OT is by describing the groups of messages that the receiver can get as sets in a collection. In the case of a simple OT, he can learn the values indexed by one of the sets in the collection $\{\{1\},\{2\}\}$. The k-out-of-n OT is an OT with a collection that contains all the sets of index of k or less elements. This mindset allows us to present a very general form of oblivious transfer. There is an important link between generalized oblivious transfer and general access structures. The notation $\mathcal{F}_{GOT(\mathcal{I})}$ denotes the instance of generalized oblivious transfer associated with the enclosed[4] collection $\mathcal{I}$.

Definition 4. *We define the following basic facts about enclosed collections:*

- Let $I = \{1, 2, ..., n\}$ be a set of indices. A collection $\mathcal{A} \subseteq \mathcal{P}(I)$ is **monotone** if the fact that $\mathcal{B} \in \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{C}$ implies that $\mathcal{C} \in \mathcal{A}$.
- An **access structure** is a monotone collection $\mathcal{A}$ of non-empty sets of I. A set S is **authorized** if $S \in \mathcal{A}$ and a set S' is **minimal** if there exists no strict subset S'' of S' such that $S'' \in \mathcal{A}$.
- The **complement** of a collection $\mathcal{C}$ is defined as $\mathcal{C}^* = \{B \subseteq I \mid \exists\, C \in \mathcal{C},\ B = I - C\}$.
- We define **Closure**$(\mathcal{C}) = \{C \subseteq C' \mid C' \in \mathcal{C}\}$.
- A collection $\mathcal{C}$ is **enclosed** if $\mathcal{C} =$ **Closure**$(\mathcal{C})$.
- An element $C \in \mathcal{C}$ is **maximal** if there exists no $C' \in \mathcal{C}$ such that $C \subseteq C'$ and $C \neq C'$.

Theorem 3. *For every enclosed collection C, there exists a unique access structure $\mathcal{A}$ such that $\mathcal{C}^* = \mathcal{A}$*

See [SSR08] for a full proof.

Definition 5. *A* **secret sharing scheme** *is a triplet of randomized algorithms (Share, Reconstruct, Check) over a message space $\mathcal{M}$ with an access structure $\mathcal{A}$. Share$_{\mathcal{A}}(s)$ always output shares $(s_1, \ldots, s_n)$ such that:*
(1) for all $A \in \mathcal{A}$, Reconstruct$_{\mathcal{A}}(\{(i, s_i) \mid i \in A\}) = s$,
(2) for any $A' \notin \mathcal{A}$, $\{(i, s_i) \mid i \in A'\}$ gives no information about s.
Check$_{\mathcal{A}}(s_1, \ldots, s_n) = 1$ if and only if for all $A \in \mathcal{A}$, Reconstruct$_{\mathcal{A}}(\{(i, s_i) \mid i \in A\}) = s$.

Definition 6. *We say that shares $(s_1, \ldots, s_n)$ are* **consistent** *if*
Check$_{\mathcal{A}}(s_1, \ldots, s_n) = 1$.

Functionality $\mathcal{F}_{GOT}(\mathcal{I})$

$\mathcal{F}_{GOT}(\mathcal{I})$ interacts with a sender **S**, a receiver **R** and an adversary $\mathcal{S}$ and is parametrized by an enclosed collection $\mathcal{I}$.

- Upon receiving $(\mathsf{Send}, sid, ssid, m_1, \ldots, m_n)$ from the **S**, if the pair $sid, ssid$ has not already been used, store $(sid, ssid, m_1, \ldots, m_n)$ and send $(\mathsf{receipt}, sid, ssid)$ to **S** and **R**.
- Upon receiving $(\mathsf{Choice}, sid, ssid, I)$ where I is a set of indices, if no $(\mathsf{Choice}, sid, ssid)$ message was previously sent and I is in $\mathcal{I}$, then for each $i \in I$, send $(\mathsf{Reveal}, sid, ssid, i, m_i)$ to **R** and $(\mathsf{Reveal}, sid, ssid)$ to the adversary $\mathcal{S}$.

[4] See definition 1

4.1 Protocol

In this section, we will present a protocol that implements $\mathcal{F}_{GOT}$ in the
$\mathcal{F}_{VOT}, \mathcal{F}_{COM} - hybrid$ model with the aid of secret sharing. The protocol is inspired by [SSR08] but is secure against a stronger adversary. The fact that every enclosed collection is the complement of an access structure will be key to this construction. The protocol requires n instances of $\mathcal{F}_{VOT}$. The selection of the secret sharing scheme is dictated by the security parameter. Namely, for security parameter s, we require that the message space of the secret sharing scheme must have cardinality greater or equal to 2^s. The size of the elements transferred in the $\mathcal{F}_{VOT}$ is the maximum between the length of the messages and the size of the shares which depends on the underlying access structure. Let $\mathcal{I}$ be the enclosed collection that defines the subsets of messages that are accessible to the receiver.

Protocol: $\boldsymbol{\pi_{GOT(\mathcal{I})}}$ (The sender has input $(m_1, \ldots, m_n)$ and the receiver has input $I \in \mathcal{I}$.)

1. The sender selects $k_1, ..., k_n \overset{\cup}{\leftarrow} \{0,1\}^l$ (one-time pads)
2. Let $\mathcal{A} = \mathcal{I}^*$, the sender selects $s \overset{\cup}{\leftarrow} \mathcal{M}$ and $(s_1, ..., s_n) = Share_{\mathcal{A}}(s)$.
3. The sender selects a set of n unused ssids, denote these ids as $(\mathrm{ssid}_1, \ldots, \mathrm{ssid}_n)$ and sends $(\mathsf{Ids}, sid, ssid, \mathrm{ssid}_1, \ldots, \mathrm{ssid}_n)$ to the receiver. For each $i \in [n]$, the sender sends $(\mathsf{send}, k_i, s_i, sid, ssid_i)$ to $\mathcal{F}_{VOT}$.
4. The receiver awaits $(\mathsf{Ids}, sid, ssid, \mathrm{ssid}_1, \ldots, \mathrm{ssid}_n)$ from the sender. He aborts if any of the ssid are not unused. Let $I \in \mathcal{I}$ be the set of messages that the receiver wishes to receive. He sets $b_i = 0$ when $i \in I$ otherwise he sets $b_i = 1$. For each $i \in [n]$, the receiver sends $(\mathsf{Transfer}, b_i, sid, ssid_i)$ to $\mathcal{F}_{VOT}$ and records the result.
5. The receiver executes the recover algorithm with the shares he received and obtains S. If the reconstruction failed, he chooses an arbitrary value for S instead. The receiver sends $(\mathsf{commit}, sid, ssid, S)$ to $\mathcal{F}_{COM}$.
6. The sender awaits $(\mathsf{committed}, sid, ssid)$ from $\mathcal{F}_{COM}$. Then, for each $i \in [n]$, the sender sends $(\mathsf{open}, 1, sid, ssid_i)$ to $\mathcal{F}_{VOT}$.
7. The receiver awaits for each $i \in [n]$, the message $(\mathsf{reveal}, 1, s_i, sid, ssid_i)$ from $\mathcal{F}_{VOT}$. The receiver aborts if $\mathrm{Check}_{\mathcal{A}}(s_1, ..., s_n) \neq 1$.
8. The receiver sends $(\mathsf{open}, sid, ssid)$ to $\mathcal{F}_{COM}$. The sender on receipt of $(\mathsf{reveal}, sid, ssid, S)$ verifies that $S = s$ and if not, he aborts the protocol.
9. The sender sends $z_i = m_i \oplus k_i$ to the receiver. ($\{m_i \mid i \in [n]\}$ is the set of messages)
10. The receiver for each $i \in I$, outputs (i, m_i) where $m_i = z_i \oplus k_i$.

Theorem 4. *π_{GOT} securely realizes $\mathcal{F}_{GOT}$ in the $\mathcal{F}_{VOT}$, $\mathcal{F}_{COM}$ hybrid model.*

Proof. First, we note that the protocol is correct by construction. In our simulation, the simulator runs the adversary internally. Whenever the simulator receives a message from the environment, it forwards it to the adversary. Whenever the adversary would send a message to the environment, it sends the message to the environment. The simulator only uses and records the messages that the adversary would send to the honest parties.

(Corrupt static sender) The simulator awaits that $\mathcal{A}$ sends the message $(\mathsf{Ids}, sid, ssid, \mathrm{ssid}_1, \ldots, \mathrm{ssid}_n)$, he aborts if one of the ids is not unused. For each $i \in [n]$, the simulator awaits that $\mathcal{A}$ send the command $(\mathsf{Send}, k_i, s_i, sid, ssid_i)$ for the oblivious transfer functionality. The simulator then forwards the message $(\mathsf{committed}, sid, ssid)$ to $\mathcal{A}$. For each $i \in [n]$, the simulator awaits that $\mathcal{A}$ sends $(\mathsf{open}, 1, sid, ssid_i)$. The simulator checks if the shares are consistent, if not, the simulator aborts. The simulator extracts S and sends $(\mathsf{reveal}, sid, ssid, S)$ to $\mathcal{A}$. For each $i \in [n]$, the simulator awaits z_i from $\mathcal{A}$. For each $i \in [n]$, the simulator sets $m_i = z_i \oplus k_i$, he then sends $(\mathsf{Send}, sid, ssid, m_1, ..., m_n)$ to $\mathcal{F}_{GOT}$.

The simulator in the ideal world and the receiver in the real world both check the validity of shares. Therefore, if the shares are not consistent, both scenarios result in an abort and in this case, the resulting views are indistinguishable. To complete the proof, we only need to show that if $\mathcal{A}$ sends consistent shares then the environment cannot distinguish between the real and ideal world. First, we note that the ideal world transcript produced by the simulator is indistinguishable from the real world transcript between the environment and a receiver. Therefore, if we show that the environment receives the same output in both worlds then the ideal world and real world are indistinguishable. This holds since the simulator is able to extract the messages that a receiver could receive and sends them to the ideal functionality. Therefore, in this case, the real and ideal world are perfectly indistinguishable.

(Corrupt static receiver) Simulator generates a random secret s, a set of random k_i and shares $s_i = Share(s)$ as well as unused $(\mathsf{Ids}, sid, ssid, \mathrm{ssid}_1, \ldots, \mathrm{ssid}_n)$. The simulator awaits for each $i \in [n]$, the command $(\mathsf{transfer}, b_i, sid, ssid_i)$ from $\mathcal{A}$. Simulator sets $d_i = k_i$ when $b_i = 0$ and $d_i = s_i$ when $b_i = 1$ and then forwards to $\mathcal{A}$, $(\mathsf{transferred}, b_i, d_i, sid, ssid_i)$. The simulator records $I = \{i \in [n] \mid b_i = 0\}$. The simulator awaits for the command $(\mathsf{Commit}, sid, ssid, S)$ from $\mathcal{A}$. The simulator then sends the message $(\mathsf{Reveal}, sid, ssid, 1, s_i)$ to $\mathcal{A}$. On reception of $(\mathsf{open}, sid, ssid, ,)$ checks if $S = S'$ and $I \in \mathcal{I}$, if it is not the case abort. The simulator selects random strings r_i. The simulator sends $(\mathsf{Choice}, sid, ssid, I)$ to $\mathcal{F}_{GOT}$ and awaits for each $i \in [n]$, the message $(\mathsf{reveal}, sid, ssid, i, m_i)$ from $\mathcal{F}_{GOT}$. For all $i \in I$, the simulator sends $(m_i \oplus k_i)$ and for all $i \notin I$, he sends r_i to $\mathcal{A}$.

First, we note that the shares and the keys are identically distributed in the real world and the ideal world. Therefore, the message received from the transfer messages and the reveal message are also identically distributed. Therefore, only the z_i can give the ability to the environment to distinguish between the real and ideal world. Note that as long as the set I is within $\mathcal{I}$ then for each $i \notin I$, he only gets either a message encrypted using a one-time pad chosen uniformly at random or a message chosen uniformly at random. Therefore, in that case the real world and ideal world are indistinguishable. Thus, the only way that the environment can distinguish between the real and ideal world is if the adversary chooses a set of keys associated to a set of indices which is not in the enclosed collection and then correctly guesses the secret. If he doesn't guess the secret correctly, then in both worlds the result is an abort which is indistinguishable. Therefore, the adversary can only allow the environment to distinguish between the real and ideal world by guessing the secret. We thus have that the probability of guessing between the real and ideal world is $|\mathcal{M}|^{-1} \leq 2^{-s}$ where $|M|$ denotes the cardinality of the message space of the secret sharing scheme. We can therefore deduce that the real and ideal world are indistinguishable.

4.2 Basic Applications

The GOT protocol can be used in numerous applications. For example, it can be used to instantiate k-out-of-n OT using a (n-k)-out-of-n secret sharing. Priced Oblivious Transfer can be instantiated using weighted secret sharing. The price of an object m_i is the weight of the share s_i. Another more complex application is multivariate oblivious polynomial evaluation presented in [Tas11]. Although the GOT protocol in that paper is different, the techniques we presented earlier could also be used to secure their protocol.

4.3 Insecurity of previously published GOT protocols

The GOT protocol presented in this article improves on the one from [SSR08] and [Tas11] significantly. We believe that their protocols are secure against semi-honest adversaries but unfortunately, a malicious sender can easily break the privacy of both schemes.

The protocol of [SSR08] works as follows: first the dealer generates shares for a randomly chosen secret, then the sender and receiver execute n instances of oblivious transfer where the receiver can learn either a share or a key chosen uniformly at random. The receiver then reconstructs the secret and sends it back to the sender. On receipt of a value, the sender checks that it is indeed the secret that he generated shares for. The sender can thus use the keys to encrypt messages and he is guaranteed that the receiver cannot learn a set of messages that is not within the enclosed collection.

However, it is possible for a malicious sender to determine if a specific message was chosen by the receiver. We will now proceed to demonstrate an attack on [SSR08]. An adversary wishes to learn if a receiver learns the message m_c. He selects a secret s and executes the share algorithm resulting in shares $\{s_i\}$. He replaces s_c by s'_c and executes the GOT protocol with those shares. As a result, if the receiver chooses to learn m_c, he will reconstruct s correctly otherwise he will reconstruct an $s' \neq s$. The attack breaks the privacy of the receiver. The same idea can be applied to attack the protocol from [Tas11].

5 Batch Single-Choice Cut-and-Choose OT

The Batch Single-Choice Cut-and-Choose OT ($\mathcal{F}_{CACOT}$) is an an instantiation of $\mathcal{F}_{GOT}$ for a specific enclosed collection. The procedure was introduced in [LP11] and it was used to implement constant round secure function evaluation.

Definition 7 makes formal the enclose collection used $\mathcal{F}_{CACOT}$. Informally, the data that will be transferred has a three dimensional structure; a table of pairs. Each row is composed of s pairs and each column is composed of n pairs. The receiver can learn two categories of element of the table. First he can learn exactly all the pairs for a subset of half the columns. In addition to that, independently for each line, he can either learn the first element of every pair or the second element of every pair.

Definition 7. *Let $T_{i,j,k}$, where $i \in [n], j \in [s]$ and $k \in \{0,1\}$. Let $A(J,\sigma)$ where $J \subseteq [s], \sigma \in \{0,1\}^n$ be the following subset of T: for all i and for all j if $j \in J$ both $T_{i,j,0}$ and $T_{i,j,1}$ are in the set otherwise only $T_{i,j,\sigma(i)}$ belongs to the set. Let $\mathcal{C}' = \bigcup_{|J|=s/2,\sigma} A(J,\sigma)$ then we define $\mathcal{C} = \mathbf{Closure}(\mathcal{C}')$. Furthermore any maximal element of $\mathcal{C}$ can be uniquely specified by some J and σ as defined previously.*

We can now formally define the Batch Single-Choice Cut-and-Choose OT.

Definition 8. $\mathcal{F}_{CACOT} = \mathcal{F}_{GOT(\mathcal{C})}$.

Theorem 5. *Any $\mathcal{F}_{CACOT}$ can be implemented with $2ns$ calls to $\mathcal{F}_{VOT}$ where the elements transferred by $\mathcal{F}_{VOT}$ are the maximum between twice the size of the secret and the value of the messages transferred.*

Proof. We will now show that there exists an efficient secret sharing scheme with an access structure $\mathcal{C}^*$ such that its complement is the enclosed collection $\mathcal{C}$. We will use a combination of linear secret sharing (mod p) to share $s = \sigma\text{-}S + J\text{-}S \mod p$, the sum of two secret. Thus, to reconstruct s, the participant will need to reconstruct both $\sigma\text{-}S$ and $J\text{-}S$. Let s_{ijk},where $i \in [n], j \in [s], k \in \{0,1\}$, be the shares of the secret sharing implementing the access structure $\mathcal{C}^*$. We construct s_{ijk} by the concatenation of $\sigma\text{-}S_{ijk}$ and $J\text{-}S_{ij}$ as defined below.

– σ-Share: The sender selects a random $\sigma\text{-}S$. He shares $\sigma\text{-}S$ using a n-out-of-n Secret Sharing resulting in shares $\sigma\text{-}S_i$ and then takes each S_i and distributes each of them twice using an $s/2$-out-of-s secret sharing resulting in shares $\sigma\text{-}S_{ij0}$, $\sigma\text{-}S_{ij1}$.
– J-Share: The sender selects a random $J\text{-}S$ and shares it using a $s/2$-out-of-s secret sharing resulting in shares $J\text{-}S_j$. He then shares each $J\text{-}S_j$ using a n-out-of-n secret sharing resulting in shares $J\text{-}S_{ij}$ where $J\text{-}S_{ij}$ is a share of $J\text{-}S_j$.

6 Modified Cut-and-Choose from [Lin13]

The Cut-and-Choose OT from [Lin13] is very similar to the one in [LP11] but there are two important differences. First, the set of indices in J is no longer size restricted (instead of size $s/2$). In addition, for each $j \notin J$, the receiver receives a special string v_j which will allow the receiver to prove that $j \notin J$. Although, we could still use the protocol for generalized oblivious transfer defined above, the complement access structure is very complicated. Instead, we will present a hybrid of the protocols from [Tas11] and [SSR08] to realize this functionality.

The protocol follow the same basic structure as the previous protocol: (1) sharing of a secret, (2) verifiable oblivious transfer, (3) commitment, (4) proof of share validity and finally (5) the message encryption and transmission. Note that the input selection for each row i is still denoted as σ_i.

Construction

Essentially, by reconstructing the secret which has been shared with the secret sharing scheme below, the prover will be able to prove two statements. First, it will show that, for each column, the receiver either didn't learn the verification string or one element from each pair. Second, it demonstrates that for each row, the receiver either learned the first element of all pairs, or the second element of all pairs. The first statement which can be thought of as a proof of ignorance reflects the approach of [SSR08], while the second one, which can be thought as a proof of knowledge, reflects the approach of [Tas11]. The protocol that follows is thus a hybrid of [Tas11] and [SSR08]. Since the protocol is very similar to the GOT protocol, we will only describe how shares are constructed and what is transferred by the verifiable oblivious transfer.

Sharing

This part describes how a sender will generate shares of a secret. The reconstruct procedure of this secret sharing naturally follows from its description. This secret will then be used as in the previous protocols to ensure that the receiver does not learn keys for a set of indices which is not within the enclosed collection. The sharing will first split the secret into two shares, sc and sr. The receiver will be able to extract sc only if for each column, he either did not learn the verification string, or he did not learn one element from each pair. The purpose of sr is to ensure that for each row, for all pairs within that row he learned the first element, or he learned for all pairs the second element. The notation k-n is used as shorthand for $\{S \subset \{1, \ldots, n\} \mid |S| \leq k\}$. In particular, the notation $\mathrm{Share}_{k\text{-}n}$ denotes the sharing of a secret using a k-out-of-n secret sharing.

$(\mathrm{sc}, \mathrm{sr}) = \mathrm{share}_{2\text{-}2}(s)$
$(\mathrm{sr}_1, \ldots, \mathrm{sr}_n) = \mathit{share}_{n\text{-}n}(\mathrm{sr})$
$(\mathrm{sr}_{i10}, \ldots, \mathrm{sr}_{in0}) = \mathrm{share}_{s\text{-}s}(\mathrm{sr}_i)$
$(\mathrm{sr}_{i11}, \ldots, \mathrm{sr}_{in1}) = \mathrm{share}_{s\text{-}s}(\mathrm{sr}_i)$
$(\mathrm{sc}_1, \ldots, \mathrm{sc}_s) = \mathrm{share}_{s\text{-}s}(\mathrm{sc})$
$(\mathrm{sc}_{1j}, \ldots, \mathrm{sc}_{nj}) = \mathrm{share}_{n\text{-}n}(\mathrm{sc}_j)$
$(\mathrm{sc}^0_{ij0}, \mathrm{sc}^1_{ij0}) = \mathrm{share}_{2\text{-}2}(\mathrm{sc}_{ij})$
$(\mathrm{sc}^0_{ij1}, \mathrm{sc}^1_{ij1}) = \mathrm{share}_{2\text{-}2}(\mathrm{sc}_{ij})$

Sender's input to VOT

This part describes which messages will be sent by the sender to $\mathcal{F}_{VOT}$. We will use $\mathrm{vid}_j, \mathrm{kid}_{i,j,k}, \mathrm{srid}_{i,j,k}$ indexed by variable i, j, k to denote distinct ssids.

$(\mathsf{Send}, \mathrm{sid}, \mathrm{vid}_j, v_j, \mathrm{sc}_j)$
$(\mathsf{Send}, \mathrm{sid}, \mathrm{kid}_{i,j,k}, k_{ijk}, \mathrm{sc}^0_{ijk})$
$(\mathsf{Send}, \mathrm{sid}, \mathrm{srid}_{i,j,k}, \mathrm{sr}_{ijk}, \mathrm{sc}^1_{ijk})$

Receiver's input to VOT

These are the messages that the receiver will send to $\mathcal{F}_{VOT}$. We also add next to them a description of the values learned by the receiver. Note that these values allow the sender to reconstruct both sc and sr as well as get the keys for a set of indices within the enclosed collection.

For each $j \in J$, the receiver sends $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{vid}_j, 1)$ to $\mathcal{F}_{VOT}$, he learns $\{\mathrm{sc}_j \mid j \in J\}$.
For each $j \notin J$, the receiver sends $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{vid}_j, 0)$ to $\mathcal{F}_{VOT}$, he learns $\{v_j \mid j \notin J\}$.
For each $j \in J, i \in [n], k \in \{0, 1\}$, the receiver sends to $\mathcal{F}_{VOT}$
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{kid}_{i,j,k}, 0)$, he learns $\{(k_{ijk}) \mid j \in J, i \in [n], k \in \{0, 1\}\}$.
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{srid}_{i,j,k}, 0)$, he learns $\{(\mathrm{sr}_{ijk}) \mid j \in J, i \in [n], k \in \{0, 1\}\}$.
For each $j \notin J, i \in [n]$, the receiver sends to $\mathcal{F}_{VOT}$
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{kid}_{i,j,\sigma_i}, 0)$, he learns $\{(k_{ij\sigma_i}) \mid j \notin J, i \in [n]\}$.
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{srid}_{i,j,\sigma_i}, 0)$, he learns $\{(sr_{ij\sigma_i}) \mid j \notin J, i \in [n]\}$.
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{kid}_{i,j,1-\sigma_i}, 1)$, he learns $\{\mathrm{sc}^0_{ij(1-\sigma_i)} \mid j \notin J, i \in [n]\}$
    $(\mathsf{Transfer}, \mathrm{sid}, \mathrm{srid}_{i,j,1-\sigma_i}, 1)$, he learns $\{\mathrm{sc}^1_{ij(1-\sigma_i)} \mid j \notin J, i \in [n]\}$

Share reconstruction and commitment

In this phase, the receiver reconstructs a secret using the reconstruction algorithm for the secret sharing described in 6. He then commits to that value.

Proof of share validity

The sender sends the messages described below to $\mathcal{F}_{VOT}$. This allows the receiver to check that the shares are consistent relative to the secret sharing defined in 6. If the shares are not consistent, the receiver aborts.

- for each $j \in J$,
  $(\mathsf{Reveal}, \mathrm{sid}, \mathrm{vid}_j, 1)$, the receiver learns sc_j.
- for each $i \in [n], j \in J, k \in \{0,1\}$
  $(\mathsf{Reveal}, \mathrm{sid}, \mathrm{kid}_{i,j,k}, 1)$, the receiver learns $\mathrm{sc}^0_{i,j,k}$.
  $(\mathsf{Reveal}, \mathrm{sid}, \mathrm{srid}_{i,j,k}, 0)$, the receiver learns $\mathrm{sr}_{i,j,k}$.
  $(\mathsf{Reveal}, \mathrm{sid}, \mathrm{srid}_{i,j,k}, 1)$ the receiver learns $\mathrm{sc}^1_{i,j,k}$.

Message encryption and transmission

For each $i \in [n], j \in [s], k \in \{0,1\}$, the sender encrypts the message m_{ijk} using k_{ijk} resulting in $z_{i,j,k}$. He then sends $z_{i,j,k}$ to the receiver. For each $i \in [n], j \notin J$, the receiver can decrypt m_{i,j,σ_i} since he knows k_{i,k,σ_i}. For each $i \in [m], j \in J$, the receiver can decrypt $m_{i,j,0}, m_{i,j,1}$ since he knows $k_{i,j,0}$ and $k_{i,j,1}$

7 Multi-sender k-Out-of-n OT

The Multi-sender k-out-of-n OT functionality was defined in [LOP11] where it was used to optimize the IPS compiler. The functionality involves p senders and one receiver. It is essentially many k-out-of-n OT executed in parallel with the same choice made by the receiver in each execution. This OT primitive can be implemented using ideas similar to the ones we presented to implement GOT in conjunction with the appropriate use of linear secret sharing.

The protocol is divided in four phases. In the first phases, the senders will construct/distribute the shares of a special secret sharing with value S. They must commit to this information. In the VOT phase, each sender will transfer a key for each message along with the associated share. The receiver will read the key associated with the messages he wishes to learn and otherwise he will obtain a share. The next phase is a verification phase, the receiver will commit to S which he could only obtain if he was requesting the same k messages from each sender. The senders will open all their commitment so that the shares are validated by the receiver. If the verification phase succeeds, the receiver opens S which proved he only read a legal set of key. In the last phase, the senders will transmit all the messages encrypted with the appropriate key.

The following functionality and protocol involves p senders with n messages of length r each and one receiver. We denote the shares of a a-out-of-b linear secret sharing as $\{B\}_{a\text{-}b}$.

Functionality $\mathcal{F}_{MSOT}$

$\mathcal{F}_{MSOT}$ interacts with senders $P_1, \ldots, P_p$ and receiver P_r

- **Inputs:** For $j = 1, \ldots, p$, upon receiving message $(\mathsf{Send}, sid, ssid, x_{1j}, \ldots, x_{nj})$ from a sender P_j, record all x_{ij}.
- **Outputs:** Upon receiving message $(\mathsf{Transfer}, sid, ssid, I \subset [n])$, check if $|I| = k$, if not abort. Send to receiver P_r, for each $j = 1, \ldots, p$ and $i \in I$, the message $(\mathsf{Receipt}, sid, ssid, i, j, x_{ij})$.

Protocol: (π_{MSOT})

- **Preparation**
  1. Each sender a selects a random secret S_a and broadcasts a non-interactive commitment to S_a. We define $S = \sum_a S_a$.
  2. Each sender a reshares S_a to obtain $\{S_{ab}\}_{(n-k)\text{-}n}$.
  3. Each sender a reshares each S_{ab} to obtain $\{S_{abc}\}_{p\text{-}p}$.
  4. For each j, b and c, sender j sends share S_{jbc} to sender c.

5. Each sender c computes for each b, $S'_{bc} = \sum_a s_{abc}$.

   We have that $S''_b = \{S'_{bc}\}_{p\text{-}p}$ and $\sum S''_b = S$.

– **VOT's**
  1. Each sender j selects uniformly at random a set of n keys k_{ij} of length r (one-time pads). He also selects n unused ids denoted by $ssid_{ij}$ and sends them to the receiver.
  2. Each sender j, for each $i \in [n]$ sends $\mathcal{F}_{VOT}$ the message $(\mathsf{Send}, \text{sid}, ssid_{ij}, k_{ij}, S'_{ij})$.
  3. Let $I \in \mathcal{I}$ be the set of messages that the receiver wishes to receive, he sets $b_i = 0$ if $i \in I$ otherwise he sets $b_i = 1$. For each i, for each sender, the receiver sends $\mathcal{F}_{VOT}$ the message $(\mathsf{Transfer}, \text{sid}, ssid_{ij}, b_i)$ and records the result.

– **Verification**
  1. Receiver computes $S''_b = \{S'_{bc}\}_{p\text{-}p}$ then $S = \sum S''_b$ and broadcasts a non-interactive commitment to S. The receiver commits to a random S if he cannot reconstruct S.
  2. Each sender j, for each i, player j sends $(\mathsf{open}, \text{sid}, ssid_{ij}, 1)$ to $\mathcal{F}_{VOT}$, thus revealing his shares to the receiver.
  3. Receiver verifies that the shares are consistent with a legal preparation phase and aborts otherwise.
  4. Receiver reveals S and if the secret is invalid, the senders abort the protocol.

– **Transfer**
  1. Each sender sends $m_{ij} \oplus k_{ij}$ to the receiver who can now calculate m_{ij} for all $i \in I$.

Theorem 6. *π_{MSOT} securely realizes $\mathcal{F}_{MSOT}$.*

Proof. Due to the restriction of space, we will provide a proof sketch for security of multi-sender k-out-of-n OT.

(Corrupt senders) The senders are controlled by the environment. It can be shown that the situation is analogous to a single corrupt sender in the GOT simulator. As such, the protocol for multi-sender k-out-of-n OT against corrupt senders follows the same simulation as a single corrupt sender in GOT. Therefore, the real model with corrupt sender can be simulated using the ideal functionality.

(Corrupt receiver) If the senders are controlled by many trusted third parties, this would be indistinguishable from a single trusted third party that controls all the honest parties. It can be shown that for the receiver the situation is analogous to the GOT, As such the real model with corrupt sender can be perfectly simulated using the ideal functionality.

(Corrupt senders and corrupt receiver) The simulator selects random-shares for each honest participant and forwards any sharing to $\mathcal{A}$. After receiving shares from the $\mathcal{A}$, the simulator simulates commitment to the secrets for all honest player. The simulator selects random keys and simulates a $\mathcal{F}_{VOT}$for each honest party. The simulator awaits that the $\mathcal{A}$(acting as the receiver) sends a commitment command for a secret S . The simulator reveals the share of honest parties to the $\mathcal{A}$in a fashion that simulates the open command of VOT. The simulator reveals the commited shares and awaits the open command from the corrupt senders. The simulator awaits the open command from receiver and checks if secret matches the opened shares otherwise abort. The simulator extracts the set of messages that corrupt receiver chose by looking at honest sender's VOT and seeing what value the receiver chose from them. The simulator sends F_{MSOT} the receiver's choice and forwards either the appropriate messages or a random message to the $\mathcal{A}$.

8 Conclusion

In this paper, we presented the first generic constructions of Multi Sender k-out-of-n OT and Batch Single Choice Cut-and-Choose OT. These constructions are based on Generalized OT, which we show how to build from Verifiable OT and proper access structures. Moreover, we formalize structure preserving OT, instantiating a practical protocol and show how to use it to obtain VOT. This sequence of results provides a novel view to the MPC frameworks of [LP11,LP12,LOP11,Lin13], shedding light on the general relations between the primitives used as their main building blocks. As future works we suggest the construction of more efficient general constructions and further investigation of the relations between different flavors of oblivious transfer. An interesting problem in this realm is analyzing the trade-off between share size and number of oblivious transfers in GOT protocols. Moreover, we suggest obtaining versions of our protocols secure against adaptive adversaries building on the recent results of [CKWZ13].

References

[AFG$^+$10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer Berlin / Heidelberg, 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *Proceedings of the 31st annual conference on Advances in cryptology CRYPTO'11*, Lecture Notes in Computer Science, pages 649–666, Berlin, Heidelberg, 2011. Springer-Verlag.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC'08*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptograpic protocols. In *FOCS'01*, 2001. Current Full Version Available at Cryptology ePrint Archive, Report 2000/067.

[CC00] C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Advances in Cryptology CRYPTO 2000*, pages 93–111. Springer, 2000.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 19–40, London, UK, 2001. Springer-Verlag.

[CHK$^+$11] Jan Camenisch, Kristiyan Haralambiev, Markulf Kohlweiss, Jorn Lapon, and Vincent Naessens. Structure preserving CCA secure encryption and applications. In Dong Lee and Xiaoyun Wang, editors, *Advances in Cryptology ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 89–106. Springer Berlin / Heidelberg, 2011.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2013.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[CvdGT95] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. *Advances in Cryptology CRYPTO'95*, pages 110–123, 1995.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520. IEEE Computer Society, 2010.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[GH08] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology ASIACRYPT'08*, Lecture Notes in Computer Science, pages 179–197, Berlin, Heidelberg, 2008. Springer-Verlag.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC'87*, pages 218–229. ACM, 1987.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT'08*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-Sahai proofs revisited. In *PKC'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2010.

[IK97] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium*, pages 174–183. IEEE, 1997.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *Proceedings of the 26th annual international conference on Advances in Cryptology EUROCRYPT'07*, Lecture Notes in Computer Science, pages 97–114, Berlin, Heidelberg, 2007. Springer-Verlag.

[Kil88] J. Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[KSV07] M. Kiraz, B. Schoenmakers, and J. Villegas. Efficient committed oblivious transfer of bit strings. *Information Security*, pages 130–144, 2007.

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[LOP11] Y. Lindell, E. Oxman, and B. Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. *Advances in Cryptology–CRYPTO 2011*, pages 259–276, 2011.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 329–346. Springer, 2011.

[LP12] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptology*, 25(4):680–722, 2012.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Aiken Compuation Laboratory, Harvard University, 1981. TR-81.

[SSR08] B. Shankar, K. Srinathan, and C.P. Rangan. Alternative protocols for generalized oblivious transfer. In *Proceedings of the 9th international conference on Distributed computing and networking*, pages 304–309. Springer-Verlag, 2008.

[Tas11] T. Tassa. Generalized oblivious transfer by secret sharing. *Designs, Codes and Cryptography*, 58(1):11–21, 2011.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS'86*, pages 162–167. IEEE, 1986.

A Ideal Functionalities

In this section we present the ideal functionalities $\mathcal{F}_{CRS}^{\mathcal{D}}$and $\mathcal{F}_{OT}$as defined in [PVW08] and $\mathcal{F}_{COM}$as defined in [CF01]. Functionality $\mathcal{F}_{OT}$ only deviates from the definition of [PVW08] in that it takes inputs x_0, x_1 an arbitrary domain D rather than $\{0,1\}^\ell$, which is necessary since in SPOT those inputs must be elements from either $\mathbb{G}$ or $\mathbb{H}$.

Functionality $\mathcal{F}_{OT}$

$\mathcal{F}_{OT}$ interacts with a sender $\mathbf{S}$, a receiver $\mathbf{R}$ and an adversary $\mathcal{S}$ and is parametrized by a domain D.

- Upon receiving a message $(\mathsf{sender}, sid, ssid, x_0, x_1)$ from $\mathbf{S}$, where $x_0, x_1 \in D$, store $(sid, ssid, x_0, x_1)$.
- Upon receiving a message $(\mathsf{receiver}, sid, ssid, \sigma)$ from $\mathbf{R}$, check if a $(\mathsf{sender}, sid, ssid, \ldots)$ message has been sent. If yes, send $(\mathsf{transfer}, sid, ssid, x_\sigma)$ to $\mathbf{R}$ and $(\mathsf{transfer}, sid, ssid)$ to $\mathcal{S}$ and halt. If not, send nothing to $\mathbf{R}$ (but continue running).

Functionality $\mathcal{F}_{CRS}^{\mathcal{D}}$

$\mathcal{F}_{CRS}^{\mathcal{D}}$ interacts with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$ and is parametrized by an algorithm $\mathcal{D}$.

- When receiving a message $(sid, ssid, P_i, P_j)$ from P_i, let $\mathsf{crs} \leftarrow \mathcal{D}(1^n)$, send $(sid, ssid, \mathsf{crs})$ to P_i and send $(sid, ssid, \mathsf{crs}, P_i, P_j)$ to the adversary. Next, when receiving $(sid, ssid, P_i, P_j)$ from P_j (and only P_j), send $(sid, ssid, \mathsf{crs})$ to P_j and to $\mathcal{S}$, and halt.

Functionality $\mathcal{F}_{COM}$

$\mathcal{F}_{COM}$ interacts with a sender $\mathbf{S}$, a receiver $\mathbf{R}$ and an adversary $\mathcal{S}$.

- **Commit Phase**: Upon receiving a message $(\mathsf{commit}, sid, ssid, \mathbf{S}, \mathbf{R}, m)$ from $\mathbf{S}$, where $m \in \{0,1\}^\ell$, record the tuple $(ssid, \mathbf{S}, \mathbf{R}, m)$ and send the message $(\mathsf{committed}, sid, ssid, \mathbf{S}, \mathbf{R})$ to $\mathbf{R}$ and $\mathcal{S}$. (The lengths of the strings ℓ is fixed and known to all parties). Ignore any future commit messages with the same $ssid$ from $\mathbf{S}$ to $\mathbf{R}$.
- **Open Phase**: Upon receiving a message $(\mathsf{open}, sid, ssid)$ from $\mathbf{S}$: If a tuple $(ssid, \mathbf{S}, \mathbf{R}, m)$ was previously recorded, then send the message $(\mathsf{reveal}, sid, ssid, \mathbf{S}, \mathbf{R}, m)$ to $\mathbf{R}$ and $\mathcal{S}$. Otherwise, ignore.

B A Summary of Groth-Sahai Proof System

Groth and Sahai presented efficient non-interactive proof systems based on bilinear groups [GS08]. They are quite common tools to construct cryptographic primitives and protocols and often called GS-proofs. GS-proofs are either non-interactive witness indistinguishable (NIWI) or non-interactive zero-knowledge (NIZK) proofs for the satisfiability of equations such as pairing product equations, multi exponentiation equations. GS-proofs are not for general (NP) statements, but efficient and quite useful when witnesses for proofs consist of *group elements*. The GS-proof system can be instantiated under the subgroup decision, SXDH, or decisional linear (DLIN) assumption. See the journal version [**?**] for full details of the GS-proof system since some errors in [GS08] were corrected (The correction was done by Ghadafi, Smart, and Warinschi [GSW10]).

Before we explain the GS-proof system, we explain Groth-Sahai commitment (GS-commitment) schemes since they are central components of the GS-proof system. Thus, we explain the GS-commitment scheme.

Groth-Sahai Commitment Scheme. First, parameters $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \xleftarrow{\mathsf{R}} \mathcal{G}(1^\lambda)$ are generated. One important property of the GS-commitment is that we can directly commit *group elements*.

$\mathsf{GS.ComSetup}(\Lambda)$**:** It takes as input Λ and outputs CRS crs_{com}.

$\mathsf{GS.Commit}(\mathsf{crs}_{com}, m, d)$**:** In order to commit message $m \in \mathbb{G}$, it takes as input m, CRS crs_{com}, and decommitment value d and outputs commitment $c \xleftarrow{\mathsf{R}} \mathsf{GS.Commit}(\mathsf{crs}_{com}, m, d)$.

$\mathsf{GS.ExpCom}(\mathsf{crs}_{com}, m, b, d)$**:** It takes as input $m \in \mathbb{Z}_p$, base $b \in \mathbb{G}$, crs_{com}, and d and outputs (b, c) where $c \xleftarrow{\mathsf{R}} \mathsf{GS.Commit}(\mathsf{crs}_{com}, b^m, d)$.

$\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, m, d)$**:** If d is a valid decommitment for (m, c), then it outputs 1. Otherwise, it outputs 0. That is, $\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, m, d) \to 1/0$. In the case of verifying that (b, c) is a commitment to exponent m, $\mathsf{GS.Opening}(\mathsf{crs}_{com}, c, b^m, d) \to 1/0$.

The setup algorithm of the GS-commitment scheme can output two types of CRS. Under one type of CRS, called extractable CRS, the commitment is perfectly binding, computationally hiding, and extractable. Under the other type of CRS, called equivocal CRS, the commitment is perfectly hiding, computationally binding, and equivocal. These two CRSs are computationally indistinguishable.

The GS-commitment scheme based on the SXDH assumption is as follows (The opening algorithm is omitted).

$\mathsf{GS.ComSetup}(\Lambda)$ It chooses $\alpha, \hat{\alpha}, \beta, \hat{\beta}, \rho, \hat{\rho} \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets

Extractable CRS: $\boldsymbol{u} := (g, g^\alpha)$, $\boldsymbol{v} := (g^\rho, g^{\rho\alpha})$, $\mathsf{crs}_{com} := (\boldsymbol{u}, \boldsymbol{v})$ and extraction key $xk := \alpha$.

Equivocal CRS: $\boldsymbol{u} := (g, g^\alpha)$, $\boldsymbol{v} := (g^\rho, g^{\rho\alpha+\beta})$, $\mathsf{crs}_{com} := (\boldsymbol{u}, \boldsymbol{v})$ and equivocation key $ek := (\alpha, \beta)$.

$\mathsf{GS.Commit}(\mathsf{crs}_{com}, m, d)$**:** For $m \in \mathbb{G}$, it chooses $r_1, r_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and computes $c := (u_1^{r_1} v_1^{r_2}, m \cdot u_2^{r_1} v_2^{r_2})$

If the CRS is an extractable one and we have trapdoor $xk = \alpha$, we can extract m like the decryption algorithm of the ElGamal encryption scheme. If the CRS is an equivocal one and we have trapdoor $ek = (\alpha, \beta)$, m, and d, we can open c to any value $m' \neq m$.

Groth-Sahai Proof System. The proof system consists of the following algorithms.

$\mathsf{GS.Setup}(\Lambda)$**:** It takes as input Λ and outputs CRS $\mathsf{crs}_{\mathsf{gs}}$ for the proof system. Note that $\mathsf{crs}_{\mathsf{gs}}$ includes crs_{com} in the Groth-Sahai commitment scheme.

$\mathsf{GS.Prove}(\mathsf{crs}_{\mathsf{gs}}, \mathsf{Eq}_{\mathsf{gs}}, \omega)$**:** It takes as input $\mathsf{crs}_{\mathsf{gs}}$, statement $\mathsf{Eq}_{\mathsf{gs}}$ (some equation), and witness ω and outputs non-interactive (WI or ZK) proof of knowledge π.

$\mathsf{GS.Vrfy}(\mathsf{crs}_{\mathsf{gs}}, \mathsf{Eq}_{\mathsf{gs}}, \pi)$**:** It takes as input $\mathsf{crs}_{\mathsf{gs}}$, $\mathsf{Eq}_{\mathsf{gs}}$, and π and verifies the proof. If it is valid, then outputs 1, otherwise 0.

$\mathsf{GS.ExtSetup}(\Lambda)$**:** It outputs $\mathsf{crs}_{\mathsf{gs}}$ and trapdoor td_{ext}. Note that the distribution of $\mathsf{crs}_{\mathsf{gs}}$ is identical to a CRS generated by $\mathsf{GS.Setup}$. This trapdoor is used to extract valid witnesses from valid proofs.

$\mathsf{GS.Extract}(\mathsf{crs}_{\mathsf{gs}}, td_{\mathsf{ext}}, \pi)$**:** It takes as input $\mathsf{crs}_{\mathsf{gs}}$, td_{ext}, and π and extracts witness ω that satisfies the statement of π. This algorithm does not require any rewinding.

$\mathsf{GS.SimSetup}(\Lambda)$**:** It outputs simulated CRS $\widetilde{\mathsf{crs}}$ and trapdoor td_{sim}. $\widetilde{\mathsf{crs}}$ is computationally indistinguishable from $\mathsf{crs}_{\mathsf{gs}}$.

$\mathsf{GS.SimProve}(\widetilde{\mathsf{crs}}, td_{\mathsf{sim}}, \mathsf{Eq}_{\mathsf{gs}})$: It takes as input $\widetilde{\mathsf{crs}}$, td_{sim}, and $\mathsf{Eq}_{\mathsf{gs}}$ and outputs simulated proof $\widetilde{\pi}$ for $\mathsf{Eq}_{\mathsf{gs}}$. It holds that $\mathsf{GS.Vrfy}(\widetilde{\mathsf{crs}}, \mathsf{Eq}_{\mathsf{gs}}, \widetilde{\pi}) = 1$. This algorithm does not require any rewinding.

It is important that we can extract only witnesses that are *group elements* by using trapdoor td_{ext} in the GS-proof system. For example, if we use witness $w \in \mathbb{Z}_p$, then we compute $g^w \in \mathbb{G}$ and generate a proof with witness g^w. Thus, we can only extract g^w. This is not w itself but an output of function $F(w) := g^w$. Therefore, it is said that the GS-proof system has F-extractability [BCKL08].

In the GS-proof system, a statement $\mathsf{Eq}_{\mathsf{gs}}$ consists of the following values:

Variables: $\{X_i\}_i \in \mathbb{G}, \{Y_j\}_j \in \mathbb{H}$ where $i \in [m]$, $j \in [n]$, and $\{x_{i'}\}_{i'}, \{y_{j'}\}_{j'} \in \mathbb{Z}_p$ where $i' \in [m']$, $j' \in [n']$.
Constants: $t_T \in \mathbb{G}_T, T_1 \in \mathbb{G}, T_2 \in \mathbb{H}$, $\{A_j\}_j \in \mathbb{G}, \{B_i\}_i \in \mathbb{H}$, $\{a_j\}_j, \{b_i\}_i, \{\gamma_{i,j}\}_{i,j}, \in \mathbb{Z}_p$ where $j \in [n]$ or $j \in [n']$ and $i \in [m]$ or $i \in [m']$.

Groth and Sahai proposed how to construct non-interactive proofs of knowledge (NIPK) for various equations over bilinear groups such as pairing product equations and multiscalar multiplication equations as follows.

$$\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, Y_i)^{\gamma_{i,j}} = t_T,$$
$$\prod_{i=1}^{n'} A_i^{y_i} \cdot \prod_{i=1}^{m} X_i^{b_i} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n'} X_i^{\gamma_{i,j} y_j} = T_1,$$
$$\prod_{i=1}^{n} Y_i^{a_i} \cdot \prod_{i=1}^{m'} B_i^{x_i} \cdot \prod_{i=1}^{m'}\prod_{j=1}^{n} Y_i^{\gamma_{i,j} x_i} = T_2,$$

A proof includes GS-commitments $\{C_i\}_i, \{D_j\}_j$ to $\{X_i\}_i \in \mathbb{G}, \{Y_j\}_j \in \mathbb{H}$.

The proof system satisfies the following properties [GS08,**?**].

Correctness: For all $\pi \xleftarrow{\mathsf{R}} \mathsf{GS.Prove}(\mathsf{crs}_{\mathsf{gs}}, \mathsf{Eq}_{\mathsf{gs}}, \omega)$, it holds $\mathsf{GS.Vrfy}(\mathsf{crs}_{\mathsf{gs}}, \pi) \to 1$.
Extractability: For $(\mathsf{crs}_{\mathsf{gs}}, td_{\mathsf{ext}}) \xleftarrow{\mathsf{R}} \mathsf{GS.ExtSetup}(\Lambda)$ and proof π, if $\mathsf{GS.Vrfy}(\mathsf{crs}_{\mathsf{gs}}, \pi) \to 1$, then witness $\omega := \mathsf{GS.Extract}(\mathsf{crs}_{\mathsf{gs}}, td_{\mathsf{ext}}, \pi)$ satisfies statement $\mathsf{Eq}_{\mathsf{gs}}$ with probability 1.
CRS indistinguishability: For $\mathsf{crs}_{\mathsf{gs}} \xleftarrow{\mathsf{R}} \mathsf{GS.Setup}(\Lambda)$ and $\widetilde{\mathsf{crs}} \xleftarrow{\mathsf{R}} \mathsf{GS.SimSetup}(\Lambda)$, it holds $\mathsf{crs}_{\mathsf{gs}} \overset{\mathsf{c}}{\approx} \widetilde{\mathsf{crs}}$.
Composable Witness Indistinguishability: For all PPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{cpWI}}(\lambda) := 2 \cdot \Pr\left[b = b' \middle| \begin{array}{l} \widetilde{\mathsf{crs}} \xleftarrow{\mathsf{R}} \mathsf{GS.SimSetup}(\Lambda); (\mathsf{Eq}_{\mathsf{gs}}, \omega_0, \omega_1, s) \xleftarrow{\mathsf{R}} \mathcal{A}(\widetilde{\mathsf{crs}}); \\ b \xleftarrow{\mathsf{U}} \{0,1\}; \pi \xleftarrow{\mathsf{R}} \mathsf{GS.Prove}(\widetilde{\mathsf{crs}}, \mathsf{Eq}_{\mathsf{gs}}, \omega_b); b' \xleftarrow{\mathsf{R}} \mathcal{A}(s, \pi) \end{array} \right] - 1 = 0.$$

Note that the GS-proof system provides ZK proofs for more restricted class of statements than those of WI proofs.

Composable Zero Knowledge: For all PPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{cpZK}}(\lambda) := 2 \cdot \Pr\left[b = b' \middle| \begin{array}{l} (\widetilde{\mathsf{crs}}, td_{\mathsf{sim}}) \xleftarrow{\mathsf{R}} \mathsf{GS.SimSetup}(\Lambda); \\ (\mathsf{Eq}_{\mathsf{gs}}, \omega, s) \xleftarrow{\mathsf{R}} \mathcal{A}(\widetilde{\mathsf{crs}}, td_{\mathsf{sim}}); \\ \pi_0 \xleftarrow{\mathsf{R}} \mathsf{GS.Prove}(\widetilde{\mathsf{crs}}, \mathsf{Eq}_{\mathsf{gs}}, \omega); \\ \pi_1 \xleftarrow{\mathsf{R}} \mathsf{GS.SimProve}(\widetilde{\mathsf{crs}}, td_{\mathsf{sim}}, \mathsf{Eq}_{\mathsf{gs}}); \\ b \xleftarrow{\mathsf{U}} \{0,1\}; b' \xleftarrow{\mathsf{R}} \mathcal{A}(s, \pi_b) \end{array} \right] - 1 = 0.$$

C Dual-Mode Encryption and Structure Preserving OT

Peikert *et al.* defined the notion of dual-mode encryption and showed that this primitive implies universally composable oblivious transfer [PVW08]. In this section, we explain the definition of dual-mode encryption and then present a structure preserving construction based on the SXDH assumption. We can easily verify that our construction is a structure preserving OT.

Definition 9 (Dual-Mode Encryption [PVW08]). *A dual-mode encryption scheme with message space* $\{0,1\}^\ell$ *consists of the following PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{FindMessy}, \mathsf{TrapGen})$*:*

- $\mathsf{Setup}(1^\lambda, \mu)$*: On input security parameter* λ *and mode* $\mu \in \{0,1\}$*, it outputs CRS* crs *and trapdoor* t*. We let* $\mathsf{SetupMessy}(\cdot) := \mathsf{Setup}(\cdot, 0)$ *and* $\mathsf{SetDec}(\cdot) := \mathsf{Setup}(\cdot, 1)$*.*
- $\mathsf{Gen}(\sigma)$*: On input branch value* $\sigma \in \{0,1\}$*, it outputs public key* pk *and secret key* sk *on branch* σ*.*
- $\mathsf{Enc}(pk, b, m)$*: On input* pk*, branch* $b \in \{0,1\}$*, and message* $m \in \{0,1\}^\ell$*, it outputs ciphertext* c *on branch* b*.*
- $\mathsf{Dec}(sk, c)$*: On input* sk *and* c*, it outputs message* m*.*
- $\mathsf{FindMessy}(t, pk)$*: On input* t *and* pk*, it outputs branch value* $b \in \{0,1\}$ *corresponding to a messy branch of* pk*.*
- $\mathsf{TrapGen}(t)$*: On input* t*, it outputs public key* pk *and corresponding secret keys for branches* 0 *and* 1*, respectively.*

These algorithms must satisfy the following properties.

- *Completeness for decryptable branch: For any* $\mu \in \{0,1\}$*, any* $(\mathsf{crs}, t) \overset{\mathsf{R}}{\leftarrow} \mathsf{Setup}(1^\lambda, \mu)$*, any* $\sigma \in \{0,1\}$*, any* $(pk, sk) \overset{\mathsf{R}}{\leftarrow} \mathsf{Gen}(\sigma)$*, and any* $m \in \{0,1\}^\ell$*,* $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \sigma, m)) \to m$*.*
- *Indistinguishability of modes:* $\mathsf{crs}_0 \overset{\mathsf{c}}{\approx} \mathsf{crs}_1$ *where* $(\mathsf{crs}_\mu, t_\mu) \overset{\mathsf{R}}{\leftarrow} \mathsf{Setup}(1^\lambda, \mu)$*.*
- *Trapdoor identification of a messy branch: For any* $(\mathsf{crs}, t) \overset{\mathsf{R}}{\leftarrow} \mathsf{SetupMessy}(1^\lambda)$ *and any* pk*,* $\mathsf{FindMessy}(t, pk) \to b$ *such that* $\mathsf{Enc}(pk, b, m_0) \overset{\mathsf{s}}{\approx} \mathsf{Enc}(pk, b, m_1)$ *for any* $m_0, m_1 \in \{0,1\}^\ell$*.*
- *Trapdoor generation of keys decryptable on both branches: For any* $(\mathsf{crs}, t) \overset{\mathsf{R}}{\leftarrow} \mathsf{SetupDec}(1^\lambda)$ *and any* $\sigma \in \{0,1\}$*,* $(pk, sk_\sigma) \overset{\mathsf{s}}{\approx} \mathsf{Gen}(\sigma)$ *where* $(k, sk_0, sk_1) \overset{\mathsf{R}}{\leftarrow} \mathsf{TrapGen}(t)$*.*

C.1 Obtaining Structure Preserving Dual-Mode Encryption Scheme from SXDH

We construct a dual-mode encryption scheme from the SXDH assumption by adapting the DDH based construction of [PVW08] in Section 3. It is almost the same as the PVW construction since we just rewrite it over asymmetric bilinear groups. However, we write the proof sketch for confirmation.

Proof Sketch In messy mode, CRS is $\mathsf{crs} = (g_0, g_0^{x_0}, g_1, g_1^{x_1})$ and this is non-DDH tuple. In decryption mode, CRS is $\mathsf{crs} = (g_0, g_0^{x}, g_1, g_1^{x})$ and this is DDH tuple. These CRSs are computationally indistinguishable by the SXDH assumption. In messy mode, public key is $pk = (g_\sigma^r, h_\sigma^r) = (g_\sigma^r, g_\sigma^{x_\sigma r})$ for $x_0 \neq x_1$, so if we have trapdoor $t = (x_0, x_1)$, we can test $h = g^{x_0}$ or not. If $\sigma = 1$, then it holds $h \neq g^{x_0}$ and $b = 0$ is the messy branch. Otherwise, $b = 1$ is the messy branch.

In decryption mode, public and secret keys are $(pk, sk_0, sk_1) = ((g_0^r, h_0^r), r, r/y)$. Thus, the distribution of (pk, sk_0) is completely the same as $\mathsf{Gen}(0)$. If we set $r := r'y$, then we can rewrite $(pk, sk_1) = (g_0^{r'y}, h_0^{r'y}, r') = (g_1^{r'}, h_1^{r'}, r')$ and this distribution is completely the same as $\mathsf{Gen}(1)$ since $r' = r/y \in \mathbb{Z}_p$ is uniformly random.

C.2 UC OT based on Dual-Mode Encryption [PVW08]

Let $\mathsf{mode} \in \{\mathsf{mes}, \mathsf{dec}\}$. Peikert, Water, and Vaikuntanathan proposed protocol $\mathsf{dm}^{\mathsf{mode}}$ as follows: Parties use the dual-mode encryption scheme described above. Sender $\mathbf{S}$ has input $m_0, m_1 \in \mathbb{G}$, receiver $\mathbf{R}$ has input $\sigma \in \{0,1\}$. $\mathbf{S}$ and $\mathbf{R}$ query $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$ with $(sid, \mathbf{S}, \mathbf{R})$ and gets (sid, crs). If mode is mes (resp. dec), then crs is generated by $\mathsf{SetupMessy}$ (resp. $\mathsf{SetupDec}$). First, $\mathbf{R}$ computes $(pk, sk) \overset{\mathsf{R}}{\leftarrow} \mathsf{Gen}(\mathsf{crs}, \sigma)$ and sends $(sid, ssid, pk)$ to $\mathbf{S}$. When $\mathbf{S}$ receives $(sid, ssid, pk)$, it computes $y_b \overset{\mathsf{R}}{\leftarrow} \mathsf{Enc}(pk, b, m_b)$ for each $b \in \{0,1\}$, and sends $(sid, ssid, y_0, y_1)$ to $\mathbf{R}$. When $\mathbf{R}$ receives $(sid, ssid, y_0, y_1)$, it outputs $(sid, ssid, \mathsf{Dec}(sk, y_\sigma))$.

Theorem 7 ([PVW08]). *Let* $\mathsf{mode} \in \{\mathsf{mes}, \mathsf{dec}\}$*. If the SXDH assumption holds, protocol* $\mathsf{dm}^{\mathsf{mode}}$ *securely realizes* $\hat{\mathcal{F}}_{OT}$ *in the* $\mathcal{F}_{CRS}^{\mathsf{mode}}$*-hybrid mode in the static corruption model.*

The messages exchanged in protocol $\mathsf{dm}^{\mathsf{mode}}$ are all group elements, so it is compatible with the GS-proof system. The following commitment scheme based on the SXDH assumption is used to construct GS-proofs based on the SXDH assumption. The CRS is $\mathsf{crs} := (\boldsymbol{u} := (u_1, u_2), \boldsymbol{v} := (v_1, v_2))$ where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{G}^2$, the witness is $X \in \mathbb{G}$, and the commitment is $\mathsf{Com} := (u_1^{r_1} v_1^{r_2}, X \cdot u_2^{r_1} v_2^{r_2})$ where $r_1, r_2 \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$.

In protocol $\mathsf{dm}^{\mathsf{mode}}$, if the sender generates a commitment of m_b by the above commitment scheme, then it can prove that the message inside encryption $y_b = \mathsf{Enc}(pk, b, m_b)$ is consistent with the message inside the commitment by the GS-proof system since the statement is a set of linear equations. That is, for $(c_0, c_1) = (u, v \cdot m_b) = (g_b^s h_b^t, g^s h^t \cdot m_b)$ and $\mathsf{Com} := (com_1, com_2) = (u_1^{r_1} v_1^{r_2}, u_2^{r_1} v_2^{r_2} \cdot m_b)$, it should hold $c_1/com_2 = g^s h^t u_2^{-r_1} v_2^{-r_2}$. Equivalently,

$$\begin{pmatrix} g_b & h_b & 1 & 1 \\ 1 & 1 & u_1 & v_1 \\ g & h & u_2^{-1} & v_2^{-1} \end{pmatrix} \cdot \begin{pmatrix} s \\ t \\ r_1 \\ r_2 \end{pmatrix} := \begin{pmatrix} g_b^s \cdot h_b^t \cdot 1^{r_1} \cdot 1^{r_2} \\ 1^s \cdot 1^t \cdot u_1^{r_1} \cdot v_1^{r_2} \\ g^s \cdot h^t \cdot (u_2^{-1})^{r_1} (v_2^{-1})^{r_2} \end{pmatrix} = \begin{pmatrix} g_b^s h_b^t \\ u_1^{r_1} v_1^{r_2} \\ g^s h^t u_2^{-r_1} v_2^{-r_2} \end{pmatrix}. \qquad (1)$$

This notation will be explained more precisely in the next section. Thus, we can use Groth-Sahai NIZK proofs for the linear equations above.

C.3 GS-Proofs for Relations Between Inputs and the Transcript

In order to prove that revealed m_b is consistent with the messages exchanged in the OT protocol execution, we use Groth-Sahai NIZK proofs. In the UC OT protocol described above, the sender proves that the message inside encryption $y_b = \mathsf{Enc}(pk, b, m_b)$ is the same m_b revealed by the sender. That is, for $(c_0, c_1) = (u, v \cdot m_b) = (g_b^s h_b^t, g^s h^t \cdot m_b)$ and $\mathsf{GS.Commit}(\mathsf{crs}_{com}, m_b, (r_1, r_2)) = (com_1, com_2) = (u_1^{r_1} v_1^{r_2}, u_2^{r_1} v_2^{r_2} \cdot m_b)$, it holds that $c_1/com_2 = g^s h^t u_2^{-r_1} v_2^{-r_2}$. This is equivalent to the equation (1) in the previous section. Thus, we use Groth-Sahai NIZK proofs for the equation above.

We borrow the notation of Dodis, Haralambiev, López-Alt, and Wichs [DHLW10]. The matrix multiplication of $\boldsymbol{A} \in \mathbb{G}^{m \times n}$ and $X \in \mathbb{Z}^{n \times k}$ is defined as follows.

$$\boldsymbol{A} \cdot X := \langle \boldsymbol{a}_i, \boldsymbol{x}_j \rangle := \prod_{r=1}^{n} \boldsymbol{a}_{i,r}^{x_{r,j}}$$

where

$$\boldsymbol{A} = \{\boldsymbol{a}_{i,j}\}_{m \times n} = \begin{pmatrix} — \boldsymbol{a}_1^\top — \\ \cdots \\ — \boldsymbol{a}_m^\top — \end{pmatrix}, \ X = \{x_{i,j}\}_{n \times k} = \begin{pmatrix} | & & | \\ \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_k \\ | & & | \end{pmatrix}.$$

Let $\boldsymbol{b}_{i,j} := \langle \boldsymbol{a}_i, \boldsymbol{x}_j \rangle$. For $\boldsymbol{\Upsilon} = \{\boldsymbol{\gamma}_{i,j}\}_{n \times k} \in \mathbb{H}^{n \times k}$, let

$$\boldsymbol{t}_{i,j} = \prod_{r=1}^{n} e(\boldsymbol{a}_{i,r}, \boldsymbol{\gamma}_{r,j})$$

and set $\boldsymbol{A} \bullet \boldsymbol{\Upsilon} := \{\boldsymbol{t}_{i,j}\}_{n \times k} \in \mathbb{G}_T^{n \times k}$. Dodis, et al [DHLW10] expressed GS NIZK for linear equations in a general form as follows.

$$\begin{pmatrix} \boldsymbol{b}_{1,1} & \boldsymbol{b}_{1,2} & \cdots & \boldsymbol{b}_{1,N} \\ \boldsymbol{b}_{2,1} & \boldsymbol{b}_{2,2} & \cdots & \boldsymbol{b}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b}_{1,M} & \boldsymbol{b}_{2,M} & \cdots & \boldsymbol{b}_{M,N} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_M \end{pmatrix}$$

where witness $\omega = (w_1, \ldots, w_N)$. Group description is $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \gamma_0) \overset{\mathsf{R}}{\leftarrow} \mathcal{G}(1^\lambda)$. If we set $K = 1$, then the proof what we want is instantiated with the SXDH assumption in asymmetric groups. The following are expressions by Dodis et al. [DHLW10]. We introduce them here for concreteness and it will help readers to understand the proof system for linear equations.

GS.SimSetup: Outputs CRS $\mathsf{crs} := \boldsymbol{\Upsilon}$ and $tk := \boldsymbol{t}$ where $\gamma_1, \ldots, \gamma_K \overset{\mathsf{U}}{\leftarrow} \mathbb{H}$, $\boldsymbol{t} := (t_1, \ldots, t_K) \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p^K$, and

$$\boldsymbol{\Upsilon} = \begin{pmatrix} \gamma_0^{\sum_{i=1}^K t_i} & \gamma_1^{t_1} & \gamma_2^{t_2} & \cdots & \gamma_K^{t_K} \\ \gamma_0 & \gamma_1 & 1 & \cdots & 1 \\ \gamma_0 & 1 & \gamma_2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_0 & 1 & 1 & \cdots & \gamma_K \end{pmatrix} \in \mathbb{H}^{(K+1)\times(K+1)}.$$

For the normal setup, we use $\gamma_0^{t'}$ where $t' \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$ instead of $\gamma_0^{\sum_{i=1}^K t_i}$.

GS.Prove: For inputs $\mathsf{Eq}_{\mathsf{gs}} := (\boldsymbol{B} \in \mathbb{G}^{M\times N}, \boldsymbol{c} \in \mathbb{G}^M)$ and $\omega := \boldsymbol{w} \in \mathbb{Z}_p^N$, it chooses $R \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p^{N\times K}$ and outputs $\pi := (\boldsymbol{\Delta}, \boldsymbol{P})$ where

$$\boldsymbol{\Delta} := \left(\begin{array}{c|c} w_1 & \\ \vdots & R \\ w_N & \end{array}\right) \cdot \boldsymbol{\Upsilon} \text{ and } \boldsymbol{P} := \boldsymbol{B} \cdot R.$$

Here, $\boldsymbol{\Delta} \in \mathbb{H}^{N\times(K+1)}$, $\boldsymbol{P} \in \mathbb{G}^{M\times K}$.

GS.SimProve: For inputs $\mathsf{Eq}_{\mathsf{gs}} = (\boldsymbol{B} \in \mathbb{G}^{M\times N}, \boldsymbol{c} \in \mathbb{G}^M)$ and $tk = \boldsymbol{t}$, it chooses $R \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p^{N\times K}$ and outputs $\pi := (\boldsymbol{\Delta}, \boldsymbol{P})$ where

$$\boldsymbol{\Delta} := \left(\begin{array}{c|c} 0 & \\ \vdots & R \\ 0 & \end{array}\right) \cdot \boldsymbol{\Upsilon} \text{ and } \boldsymbol{P} := \left(\begin{array}{c|c} \boldsymbol{c}_1 & \\ \vdots & \boldsymbol{B} \\ \boldsymbol{c}_M & \end{array}\right) \cdot \left(\frac{-t_1 \ \cdots \ -t_N}{R}\right)$$

Here, $\boldsymbol{\Delta} \in \mathbb{H}^{N\times(K+1)}$ and $\boldsymbol{P} \in \mathbb{G}^{M\times K}$.

GS.Vrfy: For input $\pi = (\boldsymbol{\Delta}, \boldsymbol{P})$, it outputs 1 iff it holds that

$$\boldsymbol{B} \bullet \boldsymbol{\Delta} := \left(\begin{array}{c|c} \boldsymbol{c}_1 & \\ \vdots & \boldsymbol{P} \\ \boldsymbol{c}_M & \end{array}\right) \bullet \boldsymbol{\Upsilon}.$$

Thus, if we set $K = 1$ and instantiate this with the SXDH assumption, then we can obtain the GS NIZK proof for equation (1) and achieve a VOT protocol from our SPOT protocol.

D Security Proof of π_{VOT}

In order to prove the security of this protocol we construct simulator that interacts with an internal copy of the adversary $\mathcal{A}$ that may corrupt parties, the ideal functionality $\mathcal{F}_{VOT}$ and an environment $\mathcal{Z}$. For the sake of simplicity, we present the case of a corrupted **S** and that of a corrupted **R** separately. In the trivial case where both parties are corrupted, the simulator $\mathcal{S}$ simply forwards all the messages between $\mathcal{Z}$ and $\mathcal{A}$. Analogously, when both parties are honest, $\mathcal{S}$ forwards all the messages between the parties **S**, **R** and $\mathcal{Z}$.

In order to construct the simulators we will use the following setup:

Setup: Simulator $\mathcal{S}$ generates a common reference string by computing the following information:

- The "simulated" public parameters for an of a Groth-Sahai NIZK proof system.
- The "simulated" CRS for the underlying structure preserving commitment scheme π_{Com} that allows the simulator to generate an equivocal commitment using trapdoor t.
- The "simulated" CRS for the underlying UC structure preserving OT π_{SPOT} that allows the simulator to obtain the receiver's choice bit c or the sender's input messages m_0, m_1.

D.1 Simulator for the Case of a corrupted S:

Simulator $\mathcal{S}$ interacts with a corrupted sender $\mathcal{A}$, the ideal functionality $\mathcal{F}_{VOT}$ and the environment $\mathcal{Z}$.

- **Setup:** When $\mathcal{A}$ requests a CRS, $\mathcal{S}$ responds by sending the CRS described above.
- **Commitment Phase:** $\mathcal{S}$ waits for the commitments $(sid, ssid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ from $\mathcal{A}$ and then interacts with it using protocol π_{SPOT}.
- π_{SPOT} **protocol execution** $\mathcal{S}$ and $\mathcal{A}$ run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} or until $\mathcal{A}$ decides to reveal one of its messages. If $\mathcal{A}$ doesn't decide to reveal one of its messages before the protocol ends, $\mathcal{S}$ uses the same procedures of the original π_{SPOT} simulator to extract m_0, m_1 and sends $(\mathsf{Send}, sid, ssid, m_0, m_1)$ to $\mathcal{F}_{VOT}$.
- **Reveal phase:** If the $\mathcal{A}$ decides to reveal one of its messages m_v at any point of the protocol execution sending $\mathcal{S}$ the message $(sid, ssid, b, \mathsf{Open(m_b)}, \psi)$, $\mathcal{S}$ verifies whether the the revealed message is correct by using the same procedures of a honest receiver. If it is correct, $\mathcal{S}$ sends $(sid, ssid, \mathsf{Open}, b)$ to $\mathcal{F}_{VOT}$. Otherwise, it reveals an invalid bit by sending a corrupted message to $\mathcal{F}_{VOT}$. $\mathcal{S}$ then proceeds by simulating the execution with $\mathcal{A}$ using the same procedures of the original π_{SPOT} simulator.

Lemma 1 *When $\mathcal{A}$ corrupts only the sender, for any universally composable structure preserving oblivious transfer protocol π_{SPOT} and structure preserving commitment scheme π_{Com}, the following holds:*

$$EXEC_{\pi_{VOT},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\hat{\mathcal{F}}_{VOT},\mathcal{S},\mathcal{Z}}$$

Proof. Note that the simulator $\mathcal{S}$ generates all messages exactly as the real protocol and as the original simulator for protocol π_{SPOT}. Thus, the simulation is indistinguishable from the real execution with protocol π_{VOT}. The difference in the CRS is computational indistinguishable since the simulated common reference strings from the commitment protocol, since the GS-proof system and the structure preserving OT protocol are computationally indistinguishable from the real common reference string used for each of these protocols.

D.2 Simulator for the Case of a corrupted receiver:

Simulator $\mathcal{S}$ interacts with a corrupted sender $\mathcal{A}$, ideal functionality $\mathcal{F}_{VOT}$ and environment $\mathcal{Z}$.

- **Setup:** When $\mathcal{A}$ requests a CRS, $\mathcal{S}$ responds by sending the CRS described above.
- **Commitment phase:** Before starting π_{SPOT}, $\mathcal{S}$ commits to two random messages $m_0, m_1 \in_R \{0,1\}^n$ by sending $(\mathsf{Com(m_0)}, \mathsf{Com(m_1)}, sid, ssid,)$ to $\mathcal{A}$.
- π_{SPOT} **protocol execution:** $\mathcal{S}$ and $\mathcal{A}$ run π_{SPOT} storing all the messages exchanged during the protocol execution up to the end of π_{SPOT} or until $\mathcal{S}$ receives a message $(\mathsf{Reveal}, sid, ssid, b, m'_b)$ from $\mathcal{F}_{VOT}$. If not interrupted, $\mathcal{S}$follows the procedures of the original π_{SPOT} simulator, extracts the choice bit c and sends $(\mathsf{Transfer}, sid, ssid, c)$ to $\mathcal{F}_{VOT}$, obtains m_c and sends it to $\mathcal{A}$ to complete the protocol.
- **Reveal phase:** If $\mathcal{S}$ receives a message $(\mathsf{Reveal}, sid, ssid, (), sid, ssid, b, m'_b)$ from $\mathcal{F}_{VOT}$ (meaning that the honest sender revealed his message m_b) it reveals the same bit m_b to $\mathcal{A}$ by doing the following:
  1. $\mathcal{S}$generates an arbitrary opening to $\mathsf{Open(m'_b)}$ corresponding to the original commitment $\mathsf{Com}(m_b)$ using the trapdoor corresponding to the simulated CRS for π_{Com}.
  2. $\mathcal{S}$generates a proof ψ that the messages exchanged with $\mathcal{A}$contain a valid transfer of m'_b using the simulated CRS for the GS NIZK proof system.
  3. $\mathcal{S}$sends $(sid, ssid, b, \mathsf{Open(m'_b)}, \psi)$ to $\mathcal{A}$.
- If there are still steps of π_{SPOT} to be executed, $\mathcal{S}$ proceeds as the original π_{SPOT} simulator until the end of the protocol.

Lemma 2 *When $\mathcal{A}$ corrupts only the receiver, for any universally composable structure preserving oblivious transfer protocol π_{SPOT} and any universally composable structure preserving commitment scheme π_{Com} the following holds:*

$$\mathrm{EXEC}_{\pi_{VOT},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}_{VOT},\mathcal{S},\mathcal{Z}}$$

Proof. The only deviations from the real protocol π_{VOT} lie in the commitments $(sid, ssid, \mathsf{Com}(m_0), \mathsf{Com}(m_1))$ sent in the beginning of the protocol and in the case that the honest sender decides to reveal one of its messages. In contrast to the real protocol, $\mathcal{S}$ commits to random $m_0, m_1 \in_R \{0,1\}^n$, and computational indistinguishability from commitments to the real messages follows from the commitments hiding property.

When the honest sender decides to reveal one of the messages, $\mathcal{S}$deviates from the real protocol by opening one of the previously sent commitments to an arbitrary value and generates a GS-proof of a relation for which it does not know the witnesses. The underlying commitment protocol has been set up with a simulated CRS. This allows the simulator who knows the trapdoor t to issue decommitments to arbitrary values and this difference is computationally indistinguishable. The indistinguishability for the GS-proof that the messages exchanged between the parties contain a transfer where $m_b = m'_b$ follows from the zero knowledge property of GS-proof systems that are set up with the simulated CRS. The only remaining difference is the CRS, which is indistinguishable since the simulated common reference strings of the underlying protocols are themselves computationally indistinguishable from the real common reference strings.