Generalizing Efficient Multiparty Computation

Bernardo M. 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.

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].

⋆ Bernardo David and Samuel Ranellucci were supported by European Research Council Starting Grant 279447. The authors acknowledge 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.

A. Lehmann and S. Wolf (Eds.): ICITS 2015, LNCS 9063, pp. 15–32, 2015.
DOI: 10.1007/978-3-319-17470-9_2

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 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.

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.

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, $y \stackrel{\mathsf{R}}{\leftarrow} D$ denotes that y is randomly selected from D according to its distribution. If S is a set, then $x \stackrel{\mathsf{U}}{\leftarrow} S$ 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 $\mathcal{X} \stackrel{\mathsf{c}}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.

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.

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.	$12pn + 1$

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 $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. 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:

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

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

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.

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 $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, while the ideal execution is represented by the $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The rationale behind this framework lies in showing that the environment $\mathcal{Z}$ is not able to efficiently distinguish between $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, 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:*

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

We present oblivious transfer ($\mathcal{F}_{OT}$), commitment ($\mathcal{F}_{COM}$), and common reference string ($\mathcal{F}^{\mathcal{D}}_{CRS}$) ideal functionalities in the full version of this paper.

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).

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:

- $\mathsf{SetupMessy}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \stackrel{\mathsf{R}}{\leftarrow} \mathcal{G}(1^\lambda)$, $g_0, g_1 \stackrel{\mathsf{U}}{\leftarrow} \mathbb{G}$, $x_0, x_1 \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$ 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).
- $\mathsf{SetupDec}(1^\lambda)$ $\Lambda := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, \hat{g}) \stackrel{\mathsf{R}}{\leftarrow} \mathcal{G}(1^\lambda)$, $g_0 \stackrel{\mathsf{U}}{\leftarrow} \mathbb{G}$, $y \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p^*$, $g_1 := g_0^y$, $x \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, $h_b := g_b^x$ for $b \in \{0,1\}$, $\mathsf{crs} := (g_0, h_0, g_1, h_1)$, and $t := y$. It outputs (crs, t).
- $\mathsf{Gen}(\sigma)$ $r \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, $g := g_\sigma^r$, $h := h_\sigma^r$, $pk := (g, h) \in \mathbb{G}^2$, $sk := r$. 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 $s, t \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, 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 $r \stackrel{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$, 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 the definition of [PVW08] under the SXDH Assumption.*

The proof of this theorem and details of the PVW framework can be found in the full version of this paper, 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 π_{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 $(sid, ssid, b, \mathsf{Open}(\mathsf{m_b}), \psi)$ 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 the full version of this paper.

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[1] 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 $\mathcal{S}$.

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

1. The sender selects $k_1, ..., k_n \stackrel{\mathsf{U}}{\leftarrow} \{0,1\}^l$ (one-time pads)
2. Let $\mathcal{A} = \mathcal{I}^*$, the sender selects $s \stackrel{\mathsf{U}}{\leftarrow} \mathcal{M}$ and $(s_1, ..., s_n) = Share_{\mathcal{A}}(s)$.
3. The sender selects a set of n unused ssids, denote these ids as $(\text{ssid}_1, \ldots, \text{ssid}_n)$ and sends $(\mathsf{Ids}, sid, ssid, \text{ssid}_1, \ldots, \text{ssid}_n)$ 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 $(\mathsf{Ids}, sid, ssid, \text{ssid}_1, \ldots, \text{ssid}_n)$ 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 $\text{Check}_{\mathcal{A}}(s_1, ..., s_n) \neq 1$.
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.*

The proof of this theorem is presented in the full version of this paper.

4.2 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 $\mathcal{C} =$* **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.*

The proof of this theorem is presented in the full version of this paper.

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.

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

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 $P_1, \dots, P_p$ 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, $S'_{bc} = \sum\limits_a s_{abc}$.

     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 $\mathcal{F}_{MSOT}$.*

The proof of this theorem is presented in the full version of this paper.

References

[AFG+10] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)

[AGHO11] Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011)

[BCKL08] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)

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

[CC00] Cachin, C., Camenisch, J.L.: Optimistic fair secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 93–111. Springer, Heidelberg (2000)

[CF01] Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

[CHK+11] Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure preserving CCA secure encryption and applications. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 89–106. Springer, Heidelberg (2011)

[CKWZ13] Choi, S.G., Katz, J., Wee, H., Zhou, H.-S.: Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 73–88. Springer, Heidelberg (2013)

[CS97] Camenisch, J.L., Stadler, M.A.: Efficient group signature schemes for large groups (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)

[CvdGT95] Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) Advances in Cryptology - CRYPT0 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)

[DHLW10] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS, pp. 511–520. IEEE Computer Society (2010)

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

[GH08] Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC 1987, pp. 218–229. ACM (1987)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[GSW10] Ghadafi, E., Smart, N.P., Warinschi, B.: Groth-Sahai proofs revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010)

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

[IPS08] Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)

[JS07] Jarecki, S.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)

[Kil88] Kilian, J.: Founding crytpography on oblivious transfer. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 20–31. ACM (1988)

[KSV07] Kiraz, M.S., Schoenmakers, B., Villegas, J.: Efficient committed oblivious transfer of bit strings. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 130–144. Springer, Heidelberg (2007)

[Lin13] Lindell, Y.: Fast cut-and-choose based protocols for malicious and covert adversaries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 1–17. Springer, Heidelberg (2013)

[LOP11] Lindell, Y., Oxman, E., Pinkas, B.: The IPS compiler: Optimizations, variants and concrete efficiency. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 259–276. Springer, Heidelberg (2011)

[LP11] Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (2011)

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

[PVW08] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)

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

[SSR08] Shankar, B., Srinathan, K., Rangan, C.P.: Alternative protocols for generalized oblivious transfer. In: Rao, S., Chatterjee, M., Jayanti, P., Murthy, C.S.R., Saha, S.K. (eds.) ICDCN 2008. LNCS, vol. 4904, pp. 304–309. Springer, Heidelberg (2008)

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

[Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS 1986, pp. 162–167. IEEE (1986)

Springer

http://www.springer.com/978-3-319-17469-3

Information Theoretic Security
8th International Conference, ICITS 2015, Lugano, Switzerland, May 2-5, 2015. Proceedings
Lehmann, A.; Wolf, S. (Eds.)
2015, XIV, 297 p. 29 illus., Softcover
ISBN: 978-3-319-17469-3