Universally Composable Adaptive Oblivious Transfer[⋆]

Matthew Green and Susan Hohenberger

The Johns Hopkins University
Information Security Institute
3400 N. Charles Street; Baltimore, MD 21218, USA
{mgreen,susan}@cs.jhu.edu

Abstract. In an oblivious transfer (OT) protocol, a Sender with messages $M_1, \ldots, M_N$ and a Receiver with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the Receiver obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages and the Sender does not learn anything about $\sigma_1, \ldots, \sigma_k$. In an *adaptive* protocol, the Receiver may obtain $M_{\sigma_{i-1}}$ before deciding on σ_i. Efficient adaptive OT protocols are interesting as a building block for secure multiparty computation and for enabling oblivious searches on medical and patent databases.

Historically, adaptive OT protocols were analyzed with respect to a "half-simulation" definition which Naor and Pinkas showed to be flawed. In 2007, Camenisch, Neven, and shelat, and subsequent other works, demonstrated efficient adaptive protocols in the full-simulation model. These protocols, however, all use standard rewinding techniques in their proofs of security and thus are not universally composable. Recently, Peikert, Vaikuntanathan and Waters presented universally composable (UC) *non-adaptive* OT protocols for the 1-out-of-2 variant, in the static corruption model using certain trusted setup assumptions. However, it is not clear how to preserve UC security while extending these protocols to the adaptive k-out-of-N setting. Further, any such attempt would seem to require $O(N)$ computation per transfer for a database of size N. In this work, we present an efficient and UC-secure *adaptive* k-out-of-N OT protocol in the same model as Peikert *et al.*, where after an initial commitment to the database, the cost of each transfer is *constant.* Our construction is secure under bilinear assumptions in the standard model.

1 Introduction

Oblivious transfer (OT) was introduced by Rabin [31] and generalized by Even, Goldreich and Lempel [19] and Brassard, Crépeau and Robert [8]. It is a two-party protocol, where a Sender with messages $M_1, \ldots, M_N$ and a Receiver with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the Receiver obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages and

⋆ This work was supported by the NSF under grant CT-0716142 and Hohenberger's Microsoft New Faculty Fellowship.

the Sender does not learn anything about $\sigma_1, \ldots, \sigma_k$. Naor and Pinkas were the first to consider an *adaptive* setting, $\mathsf{OT}^N_{k \times 1}$, where the Receiver may obtain $M_{\sigma_{i-1}}$ before deciding on σ_i [28]. Efficient OT schemes are very important. OT^4_1 is a key building block for secure multi-party computation [21, 25, 34]. $\mathsf{OT}^N_{k \times 1}$ is a useful and interesting tool in its own right, enabling oblivious databases for applications such as medical record storage and patent searches [29].

Developing efficient *adaptive* protocols appears to be a more difficult and involved process than the non-adaptive protocols. Indeed, even finding the right security definition has proven challenging. Historically, many OT constructions were analyzed under a "half-simulation" definition, where the Sender and Receiver's security are described by a combination of simulation and game-based definitions. Naor and Pinkas [28] showed that schemes analyzed under this definition may admit practical attacks on the Receiver's privacy. To address this, Camenisch, Neven and shelat [10] and subsequently Green and Hohenberger [22] proposed efficient and fully-simulatable $\mathsf{OT}^N_{k \times 1}$ protocols under bilinear assumptions. Each of these protocols achieve the optimal total communication cost of $O(N+k)$ with reasonable constants. Unfortunately, their security proofs use adversarial rewinding, and thus do not imply security under concurrent execution.

Recently, Lindell [26] showed how to achieve efficient and fully-simulatable *non-adaptive* OT^2_1 under the DDH, Nth residuosity and quadratic residuosity assumptions, as well as the assumption that homomorphic encryption exists. Simultaneously, Peikert, Vaikuntanathan and Waters [30] proposed several non-adaptive, but universally composable OT^2_1 protocols based on DDH, quadratic residuosity and lattice assumptions. While both of these works add to our collective knowledge for non-adaptive OT, they do not shed much light on how to achieve efficient *adaptive* protocols. Indeed, Lindell points out that the adaptive case is considerably harder [26].

The general framework used in [26, 30] (where the Receiver chooses the encryption keys) seems inherently at odds with allowing efficient adaptive schemes. Each transfer requires $O(N)$ work for the Sender, whereas this can be *constant* in our protocols. Even more alarming, it isn't clear how (without killing the efficiency and perhaps the UC security of [30]) a Sender could convince the Receiver that he is not changing the database values with each request. This problem of ensuring a *consistent* database gets even worse when multiple Receivers are considered, as we do in Section 5.

Our Results. In this work, we take a different approach to constructing OT protocols, which allows them to be simultaneously efficient, adaptive, universally composable and globally consistent. We summarize what is known about $\mathsf{OT}^N_{k \times 1}$ protocols in Figure 1. Let us describe some highlights.

1. *Universal Composability:* The Universal Composability framework [13] allows for the design of concurrent and composable cryptographic protocols, which are important properties in any practical deployment of an oblivious database. Canetti and Fischlin showed that OT cannot be UC-realized without trusted setup assumptions such as the existence of a Common Reference

Protocol	Rounds	Communication	Assumption
Half Simulation:			
NP99 [28]	$\ell k \log N + 1/2$	–	Sum Consistent Synthesizers + ℓ-round OT_1^2
CT05 [18]	$O(k) + 1/2$	$O(N)$	Decisional DH (in ROM)
Full Simulation:			
CNS07 [10]	$4k + 1/2$	$O(N)$	y-Power Decisional DH + q-Strong DH
CNS07 [10]	$O(k) + 1/2$	$O(N)$	Unique blind signature (in ROM)
GH07 [22]	$k + 1/2$	$O(N)$	Decisional Bilinear DH (in ROM)
UC ($\mathcal{F}_{CRS}$-hybrid):			
This work (§4)	$k + 1/2$	$O(N)$	SXDH + DLIN + q-Hidden LRSW

Fig. 1. Survey of efficient, adaptive k-out-of-N Oblivious Transfer protocols

String (CRS) [15]. This is formally referred to as the $\mathcal{F}_{CRS}$-hybrid model, and is assumed by the constructions of Peikert *et al.* [30] as well as those in this work. As in [30], we work in a static corruption model.

2. *Efficiency:* Our protocol is practical. For a database of N objects, the initialization phase requires $O(N)$ communication cost, and each transfer phase requires only constant cost, for reasonable constants. In contrast, simply repeating a OT_1^N scheme (such as [30]) k times would require $O(N)$ communication cost for *each* transfer plus the additional work required for the Sender to convince the Receiver that he isn't changing the database values dynamically. Moreover, the message space of our protocol is a group element (so at least 160 bits), whereas the quadratic residuosity and lattice-based schemes of [30] have *one*-bit message spaces. We note, however, that the DDH-based scheme of [30] allows for multiple bit messages.
3. *Model and Assumptions:* We focus on protocols secure in the standard model. Our construction can be implemented assuming SXDH [2, 5, 24, 32], Decision Linear [5], and q-Hidden LRSW (a non-interactive variant of the LRSW assumption [27], for which we give a generic group proof in the full version of this work [23].) We note that our decisional assumptions, SXDH and Decision Linear, are much more simple than the q-Power Decisional Diffie-Hellman assumption used in the (non-UC) adaptive OT of Camenisch *et al.* [10]. In the full version, we also provide a second construction that is secure in symmetric groups (i.e., where SXDH does not hold) under an alternative set of hardness assumptions. See Figure 1 for more.

Intuition behind the Construction. Oblivious Transfer protocols can be roughly divided into two categories. Let's restrict our attention to non-adaptive OT_1^N for the moment. In approach (1), which is used by [19, 26, 30, 31], the Receiver transmits a collection of specially-formed encryption keys to the Sender, who encrypts each message and returns the N ciphertexts to the Receiver. The protocol is secure provided that the encryption keys are formed such that a Receiver is able to decrypt at most *one* of the resulting ciphertexts. In approach (2), which is used by [10, 18, 22] and this work, the Sender encrypts the message collection under keys of her own choosing, and— in some interactive protocol with the Receiver— helps to decrypt *one* ciphertext.

While both approaches can be used to implement adaptive OT in theory, the first approach requires that the Sender generate a new set of ciphertexts at *each* transfer stage (for *each* receiver), requiring at least $O(N \cdot k)$ cost. Even worse, the Sender might be able to maliciously change the database between transfers and present different versions of the database to different receivers.

The latter approach is much better suited for the adaptive case. A single database can be committed to and then each decryption can be performed in constant computational and communication cost, for a total $O(N + k)$ cost. This approach is taken by the fully-simulatable protocols of [10], which both use rewinding in their simulations to (1) simulate proofs and (2) extract knowledge.[1]

An appealing naive approach to realizing UC-secure adaptive OT would be to modify the efficient standard-model protocol of Camenisch *et al.* [10] by simply replacing rewinding-based proofs with the non-interactive proof techniques of Groth and Sahai [24]. Unfortunately, this is non-trivial for two reasons. First, the Groth-Sahai techniques provide broad support for non-interactive, *witness indistinguishable* proofs of algebraic assertions in bilinear groups, but only provide non-interactive, *zero-knowledge* proofs for a restricted class of algebraic assertions. Unfortunately, the proof statements required by [10] fall outside of this class, and it does not seem easy to rectify this problem. Secondly, the protocol of [10] requires some form of extraction (e.g., extracting the chosen index from the adversarial Receiver or extracting the secret encryption keys from the adversarial Sender) for proofs containing elements of $\mathbb{Z}_p$; unfortunately, Groth-Sahai proofs of knowledge are f-extractable (but not fully extractable), where only some one-way function of the witness, $f(w)$, can be extracted (e.g., g^w) and not the witness w itself. Dealing with this limitation would necessitate substantial changes to the CNS protocol.

Instead, our construction starts from scratch. While we follow the "assisted decryption" framework of the CNS protocol, we are able to do so without the need for strong q-based decisional assumptions. We instead base the security of the ciphertexts in our scheme on the Decision Linear assumption [5]. Finally, since the Groth-Sahai proofs have not yet been shown to be either simulation-sound or UC in general, we develop techniques that permit UC simulation (even in the advanced case where multiple receivers interact with a single sender).

2 Definitions

Notation. By OT_k^N (resp., $\mathsf{OT}_{k\times 1}^N$), we denote a non-adaptive (resp., adaptive) k-out-of-N oblivious transfer protocol. Let $\overset{c}{\approx}$ denote computational indistinguishability, as defined in [13].

Adaptive k-out-of-N Oblivious Transfer. $\mathsf{OT}_{k\times 1}^N$ protocols consist of two phases: Initialization and Transfer. In the Initialization phase, the Sender commits to the input database $M_1, \ldots, M_N$. Subsequently, the Sender and Receiver

[1] Along the same lines, the half-simulation protocols of [20, 28] use a form of oblivious pseudorandom function evaluation (OPRF) to encrypt and obliviously decrypt the message database. Unfortunately, the evaluation protocols described in those works appear vulnerable to selective-failure attacks, and the modifications necessary to achieve UC security (or full simulation) seem substantial.

engage in up to k Transfers. During the i^{th} Transfer, the Receiver adaptively selects a message index $\sigma_i \in [1, N]$ and engages in a protocol such that it obtains M_{σ_i} (or $\perp$ if the protocol fails) and nothing else, while the Sender learns nothing about σ_i. The simulation-based nature of the security definition we use ensures that protocol failures must occur independently of the message index σ_i chosen by the Receiver (capturing the strong selective-failure blindness property [10].)

Universally Composable Security. As in [30], we work in the standard UC framework with static corruptions, where all parties are modeled as p.p.t. interactive Turing machines. Security of protocols is defined by comparing the protocol execution to an *ideal process* for carrying out the desired task. More formally, there is an *environment* $\mathcal{Z}$ whose task is to distinguish between two worlds: ideal and real. In the ideal world, "dummy parties" (some of whom may be corrupted by the *ideal adversary* $\mathcal{S}$) interact with an *ideal functionality* $\mathcal{F}$. In the real world, parties (some of whom may be corrupted by the *real world adversary* $\mathcal{A}$) interact with each other according to some protocol π. We refer to Canetti [13, 14] for a fuller description, as well as a definition of the ideal world ensemble $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and the real world ensemble $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. We use the established notion of a protocol π *securely realizing* an ideal functionality $\mathcal{F}$ as:

Definition 1. *Let $\mathcal{F}$ be a functionality. A protocol π* UC-realizes *$\mathcal{F}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$,*

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

Canetti and Fischlin showed that OT cannot be UC-realized without a trusted setup assumption [15]. Thus, as in [16, 30], we assume the existence of an honestly-generated Common Reference String (crs), and work in the so-called $\mathcal{F}_{CRS}$-hybrid model. The functionality is parameterized by a distribution D and a set $\mathcal{P}$ of recipients. For our purposes, $\mathcal{P}$ will include the OT Sender and Receiver only. Here the environment learns about the reference string from the adversary, and thus the simulator can set up a string with "trapdoor information", etc.

Figure 2 describes the $\mathcal{F}_{CRS}$ functionality and Figure 3 describes the $\mathcal{F}_{OT}^{N\times 1}$ functionality.

We briefly mention that there are techniques for designing and analyzing multiple OT protocols which use a single reference string; i.e., a multi-session extension. One might worry that if multiple protocols now share some joint state, then they can no longer be analyzed separately and then composed later. Fortunately, this is addressed by *universal composition with joint state* (JUC) [17] and could be done in our case. A second issue with sharing the reference string is that we make no guarantee about the security of protocols which use the same reference string in ways other than those specified by the OT protocol, and here we explicitly assume that the crs is only available to certain parties. This is at odds with the notion that the crs is a "global" entity, however, there are strong impossibility results for UC-realizing OT in a setting where the crs is available to everyone (including the environment) and can no longer be crafted by the simulator. There are models, such as the *augmented CRS* functionality

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

Upon receiving input $(\mathsf{sid}, \mathsf{crs})$ from party P, first verify that $p \in \mathcal{P}$; else ignore the input. If there is no value r recorded, then choose and record $r \leftarrow D$. Finally send output $(\mathsf{sid}, \mathsf{crs}, r)$ to P.

Fig. 2. Ideal functionality for the common reference string [14]

Functionality $\mathcal{F}_{OT}^{N\times 1}$

$\mathcal{F}_{OT}^{N\times 1}$ proceeds as follows, parameterized with integers N, ℓ and running with an oblivious transfer Sender **S**, a receiver **R** and an adversary $\mathcal{S}$.

- Upon receiving a message $(\mathsf{sid}, \mathsf{sender}, m_1, \ldots, m_N)$ from **S**, where each $m_i \in \{0,1\}^\ell$, store $(m_1, \ldots, m_N)$.
- Upon receiving a message $(\mathsf{sid}, \mathsf{receiver}, \sigma)$ from **R**, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously received. If no such message was received, send nothing to **R**. Otherwise, send $(\mathsf{sid}, \mathsf{request})$ to **S** and receive the tuple $(\mathsf{sid}, b \in \{0,1\})$ in response. Pass (sid, b) to the adversary, and: If $b = 0$, send $(\mathsf{sid}, \perp)$ to **R**. If $b = 1$, send (sid, m_σ) to **R**.

Fig. 3. Functionality for adaptive Oblivious Transfer, based on the OT_1^2 definition from [16]

$\mathcal{F}_{\text{ACRS}}$ [12], which overcome these impossibility results, but we do not explore these advanced UC issues with respect to our OT construction in this work.

3 Preliminaries

Bilinear Groups. Let $\mathsf{BMsetup}$ be an algorithm that, on input 1^κ, outputs the parameters for a bilinear mapping as $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2)$, where g generates $\mathbb{G}_1$ and $\tilde{g}$ generates $\mathbb{G}_2$, the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ each have prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

Symmetric External Diffie-Hellman Assumption (SXDH) [2, 5, 24, 32]**:** Let $\mathsf{BMsetup}(1^\kappa) \rightarrow \gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$. The SXDH assumption states that the Decisional Diffie-Hellman problem is hard within both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Groups where SXDH holds is one of the three settings for Groth-Sahai proofs [24].

Decision Linear Assumption (DLIN) [5]**:** Let $\mathsf{BMsetup}(1^\kappa) \rightarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$. For all p.p.t. adversaries Adv, the following probability is strictly less than $1/2 + 1/\text{poly}(\kappa)$:

$$\Pr[a, b, c, d \xleftarrow{\$} \mathbb{Z}_p; f \leftarrow g^c; \tilde{f} \leftarrow \tilde{g}^c; h \leftarrow g^d; \tilde{h} \leftarrow \tilde{g}^d;$$
$$z_0 \leftarrow h^{a+b}; z_1 \xleftarrow{\$} \mathbb{G}_1; d \leftarrow \{0,1\} : \mathsf{Adv}(\gamma, g, \tilde{g}, f, \tilde{f}, h, \tilde{h}, g^a, f^b, z_d) = d].$$

Note that this is a weaker asymmetric version of the original DLIN assumption of Boneh, Boyen and Shacham [5], which was set in symmetric groups.

q-Hidden LRSW Assumption: Let $\mathsf{BMsetup}(1^\kappa) \rightarrow \gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$. For all p.p.t. adversaries Adv, the following probability is strictly less than $1/\mathrm{poly}(\kappa)$:

$$\begin{aligned}&\Pr[s,t \xleftarrow{\$} \mathbb{Z}_p; \tilde{S} \leftarrow \tilde{g}^s, \tilde{T} \leftarrow \tilde{g}^t; \forall i \in [1 \ldots q], x_i, y_i \xleftarrow{\$} \mathbb{Z}_p, b_i \leftarrow g^{y_i}, \tilde{b}_i \leftarrow \tilde{g}^{y_i};\\ &A \leftarrow \mathsf{Adv}(\gamma, \tilde{S}, \tilde{T}, \{b_1, b_1^{s+x_1st}, b_1^{x_1}, b_1^{x_1t}, g^{x_1}, \tilde{b}_1\}, \ldots, \{b_q, b_q^{s+x_qst}, b_q^{x_q}, b_q^{x_qt}, g^{x_q}, \tilde{b}_q\}) :\\ &\quad A = (a_1, a_2, a_3, a_4, a_5, a_6) \wedge x \notin \{x_1, \ldots, x_q\} \wedge x \in \mathbb{Z}_p^* \wedge a_1 \in \mathbb{G}_1 \wedge\\ &\qquad a_2 = a_1^{s+xst} \wedge a_3 = a_1^x \wedge a_4 = a_1^{xt} \wedge a_5 = g^x \wedge e(a_1, \tilde{g}) = e(g, a_6)].\end{aligned}$$

Related formulations of the above assumption in an oracle-setting, where the x_i values are chosen dynamically by Adv, are the LRSW assumption which was introduced by Lysyanskaya *et al.* [27] and the Strong LRSW assumption of Ateniese *et al.* [1]. We eliminate the oracle and instead give q random tuples, which are also slightly changed. In the full version of this work [23], we show that the above assumption admits a proof in Shoup's generic group model [33].

3.1 Groth-Sahai Proofs

The Groth-Sahai proof system [24] permits a variety of efficient non-interactive proofs of the satisfiability of one or more pairing product equations. For variables $\{\mathcal{X}\}_{1 \ldots m} \in \mathbb{G}_1, \{\mathcal{Y}\}_{1 \ldots n} \in \mathbb{G}_2$ and constants $\{\mathcal{A}\}_{1 \ldots n} \in \mathbb{G}_1, \{\mathcal{B}\}_{1 \ldots m} \in \mathbb{G}_2, a_{i,j} \in \mathbb{Z}_p$, and $t_T \in \mathbb{G}_T$, these equations have the form:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{a_{i,j}} = t_T$$

Groth and Sahai show how to construct Witness Indistinguishable proof-of-knowledge of a satisfying witness to such an equation, in prime-order groups where the SXDH or Decision Linear assumptions hold. The proof system they describe can be composed over multiple equations involving the same variables. They point out that in some special cases, their techniques can be strengthened to provide Zero Knowledge. Unlike the interactive proofs used in [10, 22], the Groth-Sahai proofs do not use adversarial rewinding in their security analysis.

Groth-Sahai Commitments [24]. At the core of the Groth-Sahai system is a homomorphic commitment scheme to elements of $\mathbb{G}_1$ or $\mathbb{G}_2$.[2] The public parameters for the commitment scheme can be generated in two ways. Method (1) leads to a perfectly-binding commitment scheme, while method (2) leads to a perfectly-*hiding* scheme. Note that the two parameter distributions are computationally indistinguishable under the SXDH assumption. When the GS commitment parameters are configured according to method (1), they are equivalent

[2] As noted in [3, 24] commitment scheme can also be used to commit to elements of $\mathbb{Z}_p$, though we use this only in the context of simulating proofs.

to an Elgamal encryption of a group element, and can be decrypted by a party that knows a trapdoor to the commitment parameters. When commitments are configured according to method (2), a "simulation" trapdoor can be used on random commitments to open them to any value g^x (or $\tilde{g}^x$) for known x.

The Proof System. We now describe the proof system at a high level, adopting some notation and exposition from [3]. For this description we will conceal many of the underlying details, though the reader can refer to [3, 24] for a more detailed explanation. The proof system contains the following (possibly probabilistic) polynomial time algorithms:

$\mathsf{GSSetup}(\gamma)$. On input $\gamma \in \mathsf{BMsetup}(1^\kappa)$, outputs a string GS containing parameters for the proof system. This string embeds binding parameters for the G-S commitment scheme.

$\mathsf{GSProve}\,(GS, S, W)$. On input a statement S describing the equation, and a satisfying witness $W \in \langle\{\mathcal{X}\}_{1\ldots m}, \{\mathcal{Y}\}_{1\ldots n}\rangle$, outputs a proof π. To formulate this proof, a commitment $\hat{C}_i$ is generated for each element in W. The proof embeds openings to the commitments in such a way that a prover can ascertain that S is verifiably satisfied, and yet the elements of W remain hidden.

$\mathsf{GSVerify}(GS, \pi)$. Verifies the proof π (using the commitments and opening values) and outputs Accept if π is valid, Reject otherwise. (For compactness of notation, we will specify that π embeds the statement S).

Above we describe the proof system in normal operation. In our security proofs we will additionally use:

$\mathsf{GSExtractSetup}(\gamma)$. Outputs GS (distributed identically to the output of $\mathsf{GSSetup}(\gamma)$) and an extraction trapdoor td_{ext} containing a trapdoor for the commitment scheme. This trapdoor permits an extraction of a valid witness from the commitments embedded within a proof.

$\mathsf{GSExtract}(GS, td_{ext}, \pi)$. Given a proof π and the extraction trapdoor, extracts $\mathcal{X}_i$ or $\mathcal{Y}_i$ from each commitment $\hat{C}_i$, and outputs the witness $W = \langle\{\mathcal{X}\}_{1\ldots M}, \{\mathcal{Y}\}_{1\ldots N}\rangle$ that satisfies the equations.

$\mathsf{GSSimulateSetup}(\gamma)$. Outputs parameters GS' that are computationally indistinguishable from the output of $\mathsf{GSSetup}(\gamma)$, as well as a simulation trapdoor td_{sim} which consists of a simulation trapdoor for the commitment scheme.

$\mathsf{GSSimProve}(GS', td_{sim}, S)$. Given simulation parameters GS' and trapdoor td_{sim}, outputs a proof π of statement S that such that $\mathsf{GSVerify}(GS', \pi) =$ Accept. Note that this algorithm operates on certain restricted classes of statements (see below).

GS proofs can be defined over multiple pairing product equations. In this case, satisfiability implies knowledge of a witness for the full set of equations. In our constructions, we will denote a GS proof statement using the notation of Camenisch and Stadler [11]. For instance, $NIWI_{GS}\{(a_1, a_2) : e(a_1, a_2)e(g, h^{-1}) = 1 \wedge e(a_2, g_2)e(d_2^{-1}, a_3) = 1\}$ represents a non-interactive Witness Indistinguishable proof of knowledge, formed under parameters GS, of a witness $W = \langle a_1, a_2\rangle$

that simultaneously satisfies both listed equations. All values not in enclosed within the initial ()'s are assumed to be known to the verifier.

Witness Indistinguishability and Zero Knowledge. In general, Groth-Sahai proofs satisfy a strong definition of Witness Indistinguishability in groups where the SXDH assumption holds (complete security definitions can be found in the full version of this work [23]). However, for certain restricted classes of statements, the proof system can also be used to construct non-interactive Zero Knowledge (NIZK) proofs. For certain trivial statements, this is simply a matter of using a WI proof for which a witness can easily be found. E.g., in the special case where $t_T = 1$ for a pairing product equation, a simulator can always compute a satisfying witness by selecting each $\mathcal{X}_i$ or $\mathcal{Y}_i$ to be g^0 or $\tilde{g}^0$ respectively.

More practically, Groth and Sahai observe that some non-trivial statements can be proven in Zero Knowledge by applying the simulation trapdoor for the Groth-Sahai commitment scheme. This trapdoor allows the simulator to open a random commitment to any g^x or $\tilde{g}^x$ (for known x), and can be applied such that the same commitment is opened *differently* for each equation within the statement. In some cases, we may need to re-write a statement in order to construct a ZK proof. For example, consider a proof of the statement $e(a,d) = e(g,h)$ made on variable a and constants d, g, h. By adding a second variable b and a further equation, we obtain an equivalent statement which can be proven using the following zero knowledge proof:

$$NIZK_{GS}\{(a,b) : e(a,d)e(b,h^{-1}) = 1 \ \wedge \ e(b,g)e(g^{-1},g) = 1\}$$

Note that the equivalence holds by the property that $b = g$ is the only valid solution to the revised equation. However, using the simulation trapdoor we can open the appropriate commitments such that $a = b = g^0$ in the first equation, while in the second equation $b = g$. We will use similar techniques to simulate the Zero-Knowledge proofs in our constructions.

3.2 Additional Tools

Modified CL Signatures. Our constructions use a variant of the Camenisch-Lysanskyaya signature scheme [9], altered to operate on messages in $\mathbb{G}_1$. Whereas CL signatures rely on the interactive LRSW assumption to achieve security against adaptive chosen-message attacks, in the context of our construction we will require only a non-interactive q-Hidden LRSW assumption to achieve a weaker property (unforgeability given a set of signatures on *random* messages).

$\mathsf{CLKeyGen}(\gamma, g, \tilde{g})$**.** On input $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \dots)$ and generators $(g, \tilde{g})$, select $s, t \xleftarrow{\$} \mathbb{Z}_p$ and set $\tilde{S} \leftarrow \tilde{g}^s, \tilde{T} \leftarrow \tilde{g}^t$. Output $vk = (\gamma, g, \tilde{g}, \tilde{S}, \tilde{T})$, and $sk = (vk, s, t)$.

$\mathsf{CLSign}_{sk}(m)$**.** On input a message $m \in \mathbb{G}_1$, select $w \xleftarrow{\$} \mathbb{Z}_p$ and output the signature $\mathsf{sig} = (g^w, m^w, g^{ws}m^{wst}, m^{wt}, \tilde{g}^w) \in \mathbb{G}_1^4 \times \mathbb{G}_2$.

$\mathsf{CLVerify}_{vk}(\mathsf{sig}, m)$**.** On input the value $m \in \mathbb{G}_1$ and $\mathsf{sig} = (a_1, a_2, a_3, a_4, \tilde{a}_5)$, verify that $e(g, \tilde{a}_5) = e(a_1, \tilde{g}) \wedge e(m, \tilde{a}_5) = e(a_2, \tilde{g}) \wedge e(a_2, \tilde{T}) = e(a_4, \tilde{g}) \wedge e(a_3, \tilde{g}) = e(a_1 a_4, \tilde{S})$.

Note that the verification algorithm can be represented as a set of pairing product equations, and thus it is possible to prove knowledge of a pair (m, sig) using the GS proof system. To prove knowledge of m, sig, first select $y \xleftarrow{\$} \mathbb{Z}_p$, compute $\mathsf{sig}' = \langle a_1', a_2', a_3', a_4', \tilde{a}_5' \rangle = \langle a_1^y, a_2^y, a_3^y, a_4^y, \tilde{a}_5^y \rangle$ and release the pair $a_1', \tilde{a}_5'$ along with the following witness indistinguishable proof:

$$\pi = NIWI_{GS}\{(m, a_2', a_3', a_4') :$$
$$e(m, \tilde{a}_5')e(a_2', \tilde{g}^{-1}) = 1 \wedge e(a_2', \tilde{T})e(a_4', \tilde{g}^{-1}) = 1 \wedge e(a_3', \tilde{g})e(a_4'^{-1}, \tilde{S}) = e(a_1', \tilde{S})\}$$

The verifier checks both the proof and the fact that $e(a_1', \tilde{g}) = e(g, \tilde{a}_5')$.

Selective-message Secure Boneh-Boyen Signatures. Our constructions also make use of a weak signature scheme built from the Boneh-Boyen selective-ID IBE scheme [4] (§4).

$\mathsf{BBKeyGen}(\gamma, g_1, \tilde{g}_1)$**.** On input $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \ldots)$ and bases $(g_1, \tilde{g}_1)$, select $\alpha, z \xleftarrow{\$} \mathbb{Z}_p$, $g \leftarrow g_1^{1/\alpha}$, $\tilde{g} \leftarrow \tilde{g}_1^{1/\alpha}$, $g_2 \leftarrow g^z$, $\tilde{g}_2 \leftarrow \tilde{g}^z$, $h \xleftarrow{\$} \mathbb{G}_1$. Output $vk = (\gamma, g, \tilde{g}, g_1, g_2, h, \tilde{g}_2)$, and $sk = (vk, g_2^\alpha)$.

$\mathsf{BBSign}_{sk}(m)$**.** On input a message $m \in \mathbb{G}_1$, select $r \xleftarrow{\$} \mathbb{Z}_p$ and output the signature $\mathsf{sig} = ((mh)^r g_2^\alpha, \tilde{g}^r, g^r) \in \mathbb{G}_1^2 \times \mathbb{G}_2$.

$\mathsf{BBVerify}_{vk}(\mathsf{sig}, m)$**.** On input $m \in \mathbb{G}_1$ and $\mathsf{sig} = (s_1, \tilde{s}_2, s_3)$, verify that $e(s_1, \tilde{g}) \;/\; e(mh, \tilde{s}_2) = e(g_1, \tilde{g_2})$ and $e(g, \tilde{s}_2) = e(s_3, \tilde{g})$.

We can prove knowledge of a pair (m, sig) as follows. Select $y \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{sig}' = (s_1', \tilde{s}_2', s_3') = (s_1(mh)^y, \tilde{s}_2\tilde{g}^y, s_3 g^y)$. Output $\tilde{s}_2', s_3'$ and the WI proof:

$$\pi = NIWI_{GS}\{(m, s_1') : e(s_1', \tilde{g})e(m, \tilde{s}_2'^{-1}) = e(h, \tilde{s}_2')e(g_1, \tilde{g_2})\}$$

The verifier checks the proof and the fact that $e(g, \tilde{s}_2') = e(s_3', \tilde{g})$.

Double-Trapdoor BBS Encryption. Our OT constructions employ an encryption scheme with a "double-trapdoor" (so that both the simulator in charge of the crs and the sender in charge of the pk can extract the messages of the ciphertext.) It is crucial that the holder of either secret key can verify the consistency of the ciphertext with respect to the other secret key (*i.e.,* that decryption using the other key would reveal the same plaintext.) We use a variant of Boneh-Boyen-Shacham encryption [5], which has a public consistency check.

Let $\mathsf{BMsetup}(1^\kappa) \to \gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$. Publish global parameters $\gamma, h, \tilde{h}$ such that $e(g, \tilde{h}) = e(\tilde{g}, h)$, and for $i \in [1, 2]$ select $sk_i \leftarrow (x_i, y_i \in_R \mathbb{Z}_p)$ and $pk_i = (u_i, v_i, \tilde{u}_i, \tilde{v}_i) \leftarrow (h^{1/x_i}, h^{1/y_i}, \tilde{h}^{1/x_i}, \tilde{h}^{1/y_i})$. To encrypt a message $m \in \mathbb{G}_1$ under pk_1/pk_2, first select random values $r, s \in \mathbb{Z}_p$ and output the ciphertext $(u_1^r, v_1^s, u_2^r, v_2^s, h^{r+s}m)$. To decrypt a message $(c_1, \ldots, c_5)$ under $sk_1 = (x_1, y_1)$, output $c_5/(c_1^{x_1} \cdot c_2^{y_1})$. To decrypt under $sk_2 = (x_2, y_2)$, output $c_5/(c_3^{x_2} \cdot c_4^{y_2})$. Note that the structure of a ciphertext can be verified using the bilinear map, by checking that $e(c_1, \tilde{u}_2) = e(c_3, \tilde{u}_1) \;\wedge\; e(c_2, \tilde{v}_2) = e(c_4, \tilde{v}_1)$ In the full version [23] we show that scheme above is semantically-secure under the DLIN assumption.

Protocol OTA

OTA is parameterized by the algorithms $(\mathsf{OTGenCRS}, \mathsf{OTInitialize}, \mathsf{OTRequest}, \mathsf{OTRespond}, \mathsf{OTComplete})$.

When $\mathbf{S}$ is activated with $(\mathsf{sid}, \mathsf{sender}, \langle M_1, \ldots, M_N \in \{0,1\}^\ell \rangle)$:

1. $\mathbf{S}$ queries $\mathcal{F}_{CRS}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and receives $(\mathsf{sid}, \mathsf{crs})$. $\mathbf{R}$ then queries $\mathcal{F}_{CRS}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and receives $(\mathsf{sid}, \mathsf{crs})$.[a]
2. $\mathbf{S}$ computes $(T, sk) \leftarrow \mathsf{OTInitialize}(\mathsf{crs}, M_1, \ldots, M_N)$, sends (sid, T) to $\mathbf{R}$ and stores (sid, T, sk).

When $\mathbf{R}$ is activated with $(\mathsf{sid}, \mathsf{receiver}, \sigma)$, and $\mathbf{R}$ has previously received (sid, T) and $(\mathsf{sid}, \mathsf{crs})$:

1. $\mathbf{R}$ runs $(Q, Q_{priv}) \leftarrow \mathsf{OTRequest}(\mathsf{crs}, T, \sigma)$, sends (sid, Q) to $\mathbf{S}$ and stores (sid, Q_{priv}).
2. $\mathbf{S}$ gets (sid, Q) from $\mathbf{R}$, runs $R \leftarrow \mathsf{OTRespond}(\mathsf{crs}, T, sk, Q)$, and sends (sid, R) to $\mathbf{R}$.
3. $\mathbf{R}$ receives (sid, R) from $\mathbf{S}$, and outputs $(\mathsf{sid}, \mathsf{OTComplete}(\mathsf{crs}, T, R, Q_{priv}))$.

[a] $\mathcal{F}_{CRS}$ computes computes $\mathsf{crs} \leftarrow \mathsf{OTGenCRS}(1^\kappa)$.

Fig. 4. A high-level outline of the $\mathsf{OT}^N_{k\times 1}$ protocol, with details of each algorithm described in Section 4. We make no explicit mention of the value k, the total transfers permitted by the Sender, because our protocol does not depend on it. The Sender may choose to stop answering the Receiver's queries at any point, in which case $\mathsf{OTRespond}$ outputs "reject" and $\mathsf{OTComplete}$ accepts this as the message $\perp$.

4 A UC-Secure Adaptive OT Construction

Our adaptive oblivious transfer protocol, $\mathsf{OT}^N_{k\times 1}$ follows the framework described in Figure 4. We now describe one instantiation of the algorithms ($\mathsf{OTGenCRS}$, $\mathsf{OTInitialize}$, $\mathsf{OTRequest}$, $\mathsf{OTRespond}$, $\mathsf{OTComplete}$). In the full version [23], we provide a second instantiation, under different assumptions.

$\mathsf{OTGenCRS}(1^\kappa)$. Given security parameter κ, generate parameters for a bilinear mapping $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathsf{BMsetup}(1^\kappa)$. Compute $GS_S \leftarrow \mathsf{GSSetup}(\gamma)$ and $GS_R \leftarrow \mathsf{GSSetup}(\gamma)$. Choose $a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_p$, and set $(g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h}) \leftarrow (g^a, g^b, g^c, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c)$. Output $\mathsf{crs} = (\gamma, GS_S, GS_R, g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h})$. (In the full version [23], we describe how this common reference string can be replaced by a common random string.)

$\mathsf{OTInitialize}(\mathsf{crs}, m_1, \ldots, m_N)$. This algorithm is executed by the Sender. On input a collection of N messages and the crs, it outputs a commitment to the database, T, for publication to the Receiver, as well as a Sender secret key, sk. We treat messages as elements of $\mathbb{G}_1$, since there exist efficient mappings between strings in $\{0,1\}^\ell$ and elements in $\mathbb{G}_1$ (e.g., [1, 6]).

1. Parse crs to obtain $GS_S, g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h}$ and γ.
2. Choose random values $x_1, x_2 \in \mathbb{Z}_p$.

3. Set $(u_1, u_2) \leftarrow (h^{1/x_1}, h^{1/x_2})$, $(\tilde{u}_1, \tilde{u}_2) \leftarrow (\tilde{h}^{1/x_1}, \tilde{h}^{1/x_2})$.
4. Set $(vk_1, sk_1) \leftarrow \mathsf{CLKeyGen}(\gamma, u_1, \tilde{u}_1)$, $(vk_2, sk_2) \leftarrow \mathsf{CLKeyGen}(\gamma, u_2, \tilde{u}_2)$ and $(vk_3, sk_3) \leftarrow \mathsf{BBKeyGen}(\gamma, u_1, \tilde{u}_1)$.
5. Set $pk \leftarrow (u_1, u_2, \tilde{u}_1, \tilde{u}_2, vk_1, vk_2, vk_3)$.
6. For $j = 1, \ldots, N$ encrypt each message m_j as:
   (a) Select random $r, s, t \in \mathbb{Z}_p$.
   (b) Compute $\mathsf{sig}_1 \leftarrow \mathsf{CLSign}_{sk_1}(u_1^r)$, $\mathsf{sig}_2 \leftarrow \mathsf{CLSign}_{sk_2}(u_2^s)$, and $\mathsf{sig}_3 \leftarrow \mathsf{BBSign}_{sk_3}(u_1^r u_2^s)$.
   (c) Set $C_j \leftarrow (u_1^r,\ u_2^s,\ g_1^r,\ g_2^s,\ m_j \cdot h^{r+s},\ \mathsf{sig}_1,\ \mathsf{sig}_2,\ \mathsf{sig}_3)$.
7. Set $T \leftarrow (pk, C_1, \ldots, C_N)$ and $sk \leftarrow (x_1, x_2)$. Output (T, sk).

Each ciphertext C_j above can be thought of as a signcryption where it is the *randomness* for each ciphertext that is signed, rather than the plaintext itself. Each plaintext m_j is encrypted under **S**'s public key u_1, u_2, as well as a "key" g_1, g_2 drawn from crs. This "double-trapdoor" encryption is necessary for the security proof of the OT scheme.

To verify the format of each ciphertext $C_j = (c_1, \ldots, c_5,\ \mathsf{sig}_1,\ \mathsf{sig}_2,\ \mathsf{sig}_3)$ in T, anyone can check that $\mathsf{CLVerify}_{vk_1}(c_1, \mathsf{sig}_1)$, $\mathsf{CLVerify}_{vk_2}(c_2, \mathsf{sig}_2)$, and $\mathsf{BBVerify}_{vk_3}(c_1 c_2, \mathsf{sig}_3)$ each succeed, and that $e(c_1, \tilde{g}_1) = e(c_3, \tilde{u}_1) \wedge e(c_2, \tilde{g}_2) = e(c_4, \tilde{u}_2)$.

$\mathsf{OTRequest}(\mathsf{crs}, T, \sigma)$. This algorithm is executed by a Receiver. On input T generated by the Sender, along with an item index σ, generates a query Q for transmission to the Sender.

1. Parse T as $(pk, C_1, \ldots, C_N)$, and ensure that it is correctly formed (see above). If T is not correctly formed, abort the protocol. (This is only necessary on the first transfer.)
2. Parse crs to obtain $(GS_R, \tilde{h})$, and parse pk as $(u_1, u_2, \tilde{u}_1, \tilde{u}_2, vk_1, vk_2, vk_3)$. Parse the σ^{th} ciphertext C_σ as $(c_1, \ldots, c_5, \mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3)$.
3. Select random $v_1, v_2 \in \mathbb{Z}_p$.
4. Set $d_1 \leftarrow (c_1 \cdot u_1^{v_1}), d_2 \leftarrow (c_2 \cdot u_2^{v_2})$, $t_1 \leftarrow h^{v_1}$, $t_2 \leftarrow h^{v_2}$.
5. Use the Groth-Sahai techniques and reference string GS_R to compute a Witness Indistinguishable proof π that the values d_1, d_2 pertaining to the ciphertext C_σ (which the Receiver wishes to have the Sender help him open) have the correct structure:

$$\begin{aligned}\pi = NIWI_{GS_R}\{&(c_1, c_2, t_1, t_2, \mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3) :\\ &e(c_1, \tilde{h})e(t_1, \tilde{u}_1) = e(d_1, \tilde{h}) \ \wedge\ e(c_2, \tilde{h})e(t_2, \tilde{u}_2) = e(d_2, \tilde{h}) \ \wedge\\ &\mathsf{CLVerify}_{vk_1}(c_1, \mathsf{sig}_1) = 1 \ \wedge\ \mathsf{CLVerify}_{vk_2}(c_2, \mathsf{sig}_2) = 1 \ \wedge\\ &\mathsf{BBVerify}_{vk_3}(c_1 c_2, \mathsf{sig}_3) = 1\}\end{aligned}$$

6. Set request $Q \leftarrow (d_1, d_2, \pi)$, and private state $Q_{priv} \leftarrow (Q, \sigma, v_1, v_2)$. Output (Q, Q_{priv}).

To explain what is happening in the statement of step (5), first observe that the signature proofs of knowledge ensure that the values c_1, c_2 and the product $(c_1 c_2)$ each correspond to a valid signature held by the Receiver. The remaining equations ensure that the values d_1, d_2 correspond to "blinded" versions of the elements c_1, c_2. These checks guarantee that the witness used by the Receiver, and thus the decryption request being made, corresponds to one of the N ciphertexts published by the Sender.

$\mathsf{OTRespond}(\mathsf{crs}, T, sk, Q)$. This algorithm is executed by the Sender. If the Sender does not wish to answer any more requests for the Receiver, then the Sender outputs the message "reject". Otherwise, the Sender processes the Receiver's request Q as:

1. Parse crs to obtain $(GS_R, \tilde{g}, \tilde{h})$, and parse T as $(pk, C_1, \ldots, C_N)$, and sk as (x_1, x_2).
2. Parse pk (from T) as $(u_1, u_2, \tilde{u}_1, \tilde{u}_2, vk_1, vk_2, vk_3)$.
3. Parse Q as (d_1, d_2, π) and verify proof π using GS_R. Abort if check fails.
4. Set $a_1 \leftarrow d_1^{x_1}$, $a_2 \leftarrow d_2^{x_2}$, and $s \leftarrow a_1 \cdot a_2$.
5. Use the Groth-Sahai techniques and reference string GS_S to formulate a zero-knowledge proof[3] that the decryption value s is properly computed:

$$\delta = NIZK_{GS_S}\{(a_1, a_2) : e(a_1, \tilde{u}_1)e(d_1^{-1}, \tilde{h}) = 1 \\ \wedge\ e(a_2, \tilde{u}_2)e(d_2^{-1}, \tilde{h}) = 1\ \wedge\ e(a_1 a_2, \tilde{h})e(s^{-1}, \tilde{h}) = 1\}$$

   The third equation ensures that $s = a_1 \cdot a_2$, while the first two, since the values $(u_1, d_1, u_2, d_2, \tilde{h})$ are known to both parties, ensure that $a_1 = d_1^{x_1}$ and $a_2 = d_2^{x_2}$.
6. Output $R \leftarrow (s, \delta)$.

$\mathsf{OTComplete}(\mathsf{crs}, T, R, Q_{priv})$. This algorithm is executed by the Receiver. On input R generated by the Sender in response to a request Q, along with state Q_{priv}, outputs a message m or $\perp$. If R is the message "reject", then the Receiver outputs $\perp$. Otherwise, the Receiver does:

1. Parse crs to obtain (GS_S, h). Parse T as $(pk, C_1, \ldots, C_N)$, R as (s, δ), and Q_{priv} as (Q, σ, v_1, v_2).
2. Verify proof δ using GS_S. If verification fails, output $\perp$.
3. Parse C_σ to obtain the first five elements $(c_1, \ldots, c_5)$ and output $m = c_5/(s \cdot h^{-v_1} \cdot h^{-v_2})$. Map this element to a value in $\{0,1\}^\ell$ [1].

4.1 Efficiency Analysis

When the protocol in Figure 4 is implemented using the algorithms described above, we obtain a $(k+1/2)$-round protocol with communications cost $O(N+k)$, where $k \leq N$. More concretely, the crs is comprised of 7 elements in $\mathbb{G}_1$ and 7 elements of $\mathbb{G}_2$, the Sender's public key contains 5 elements in $\mathbb{G}_1$ and 6 elements in $\mathbb{G}_2$. Each of the N ciphertexts in T requires 15 elements in $\mathbb{G}_1$ and 3 elements in $\mathbb{G}_2$. Moreover, each item transfer involves transmission of 68 elements of $\mathbb{G}_1$

[3] We present a simplified version of this proof above. However, to permit simulation, we must add a third variable $\tilde{a}_3 = \tilde{h}$ and re-write the proof as $NIZK_{GS_S}\{(a_1, a_2, \tilde{a}_3) : e(a_1, \tilde{u}_1)e(d_1^{-1}, \tilde{a}_3) = 1\ \wedge\ e(a_2, \tilde{u}_2)e(d_2^{-1}, \tilde{a}_3) = 1\ \wedge\ e(a_1 a_2, \tilde{a}_3)e(s^{-1}, \tilde{a}_3) = 1\ \wedge\ e(u_1, \tilde{a}_3) = e(u_1, \tilde{h})\}$. See the full version for details.

and 38 elements of $\mathbb{G}_2$ from Receiver to Sender, and then 20 elements of $\mathbb{G}_1$ and 18 elements of $\mathbb{G}_2$ from Sender to Receiver. The message space of our OT protocol is elements in $\mathbb{G}_1$, which will be sufficient for transferring a symmetric encryption key to unlock a file of arbitrary size.

4.2 Security Analysis

Theorem 1. *Instantiated with the above algorithms,* OTA *securely realizes the functionality* $\mathcal{F}_{OT}^{N\times 1}$ *in the* $\mathcal{F}_{CRS}$-hybrid *model under the SXDH, DLIN, and q-Hidden LRSW assumptions.*

Due to space considerations, we provide only a sketch of Theorem 1 below (the complete proof can be found in the full version of this work [23]). When either the Sender or the Receiver is corrupted, we wish to describe a simulator $\mathcal{S}$ such that it can interact with the ideal functionality $\mathcal{F}_{OT}^{N\times 1}$ (which we'll denote simply as $\mathcal{F}$) and the environment $\mathcal{Z}$ appropriately; i.e., $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA},\mathcal{A},\mathcal{Z}}$.

Simulating the case where only S is corrupted. We first consider the case where the real-world adversary $\mathcal{A}$ corrupts the Sender, and thus $\mathcal{S}$ must interact with $\mathcal{F}$ as the ideal Sender and with (an internal copy of) $\mathcal{A}$ as a real-world Receiver. Here $\mathcal{S}$ does the following:

1. Ask $\mathcal{A}$ to begin an OT protocol, and set the crs *for these two parties* by running $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $GS_S \leftarrow \mathsf{GSSetup}(\gamma)$, $GS_R \leftarrow \mathsf{GSSetup}(\gamma)$, selecting random elements $a_1, a_2 \in \mathbb{Z}_p$, and setting $g_1^{a_1} = g_2^{a_2} = h$ (and a corresponding relationship for $\tilde{g}_1, \tilde{g}_2, \tilde{h}$). Set $\mathsf{crs} = (\gamma, GS_S, GS_R, g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h})$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.
2. Obtain the database commitment T from $\mathcal{A}$. Verify that T is well-formed, abort if not. Otherwise, $\forall i \in [1, N]$ use a_1, a_2 to decrypt each ciphertext $C_i = (c_1, \ldots, c_5, \ldots)$ as $m_i = c_5/(c_3^{a_1} c_4^{a_2})$. Map each element $m_i \in \mathbb{G}_1$ to a string in $\{0,1\}^\ell$ [1]. Send $(\mathsf{sid}, \mathbf{S}, m_1, \ldots, m_N)$ to $\mathcal{F}$.
3. Upon receiving $(\mathsf{sid}, \mathsf{request})$ from $\mathcal{F}$, return $\mathsf{OTRequest}(\mathsf{crs}, T, 1)$ to $\mathcal{A}$. This response includes two random values d_1, d_2 and a non-interactive witness indistinguishable proof π with respect to $GS_R \in \mathsf{crs}$ that d_1, d_2 are "blinded" values corresponding to ciphertext C_1. This proof can be performed honestly and without rewinding.
4. If $\mathcal{A}$ issues a "reject" message or responds with anything other than a value in $\mathbb{G}_1$ and a valid NIZK proof, then $\mathcal{S}$ tells $\mathcal{F}$ to fail the request by sending message $(\mathsf{sid}, 0)$. Otherwise, $\mathcal{S}$ sends the message $(\mathsf{sid}, 1)$ to $\mathcal{F}$.

The indistinguishability argument here follows from the indistinguishability of the crs (which is identically distributed to a real crs), the perfect extraction of the messages in step (2),[4] and the Witness Indistinguishability of the

[4] Note that a ciphertext that passes the validity check can be represented as $C = (u_1^r, u_2^s, g_1^r, g_2^s, h^{r+s}m, \ldots)$ for some $r, s \in \mathbb{Z}_p$, and when (g_1, g_2, h) have the relationship described above, decryption using a_1, a_2 always produces m.

GS proof π issued during each request phase, which guarantees that $\mathcal{A}$ (the corrupt Sender) cannot distinguish a request to decrypt C_1 from a request to decrypt any other valid ciphertext. Thus, $\mathcal{S}$ can adequately mimic its response pattern.

Simulating the case where only R is corrupted. Next, we consider the case where the real world adversary $\mathcal{A}$ corrupts the Receiver, and thus $\mathcal{S}$ must interact with $\mathcal{F}$ as the ideal Receiver and with (and internal copy of) $\mathcal{A}$ as real-world Receiver. This case requires that the $q = N$ for the *q-Hidden LRSW* assumption. Here $\mathcal{S}$ does the following:

1. Ask $\mathcal{A}$ to begin an OT protocol, and set the crs *for these two parties* by running $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $(GS_S, td_{sim}) \leftarrow \mathsf{GSSimulateSetup}(\gamma)$ and $(GS_R, td_{ext}) \leftarrow \mathsf{GSExtractSetup}(\gamma)$. Select random elements for $g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h}$. Set $\mathsf{crs} \leftarrow (\gamma, GS_S, GS_R, g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h})$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.
2. $\mathcal{S}$ must commit to a database of messages for $\mathcal{A}$ *without* knowing the messages $m_1, \ldots, m_N$. Thus, $\mathcal{S}$ simply commits to random junk messages, and sends the corresponding T to $\mathcal{A}$.
3. When $\mathcal{A}$ makes a transfer request, $\mathcal{S}$ uses td_{ext} to extract the witness W corresponding to $\mathcal{A}$'s decryption request from the NIWI proof. (This extraction is done via opening perfectly-binding commitments which are included in the WI proof and does not require any rewinding.) This witness includes the first two elements (c_1, c_2) of the ciphertext that $\mathcal{A}$ is requesting to decrypt, and from these it is possible to determine the index σ' of the ciphertext that $\mathcal{A}$ has requested to open.
4. $\mathcal{S}$ now sends $(\mathsf{sid}, \mathbf{R}, \sigma')$ to $\mathcal{F}$ to obtain the real $m_{\sigma'}$ message.
5. Finally, $\mathcal{S}$ returns a response to $\mathcal{A}$ which opens $C_{\sigma'}$ to $m_{\sigma'}$ and then uses td_{sim} to simulate an NIZK proof that this opening is correct. The NIZK proof here is designed in such a way that simulation is always possible and no rewinding is necessary.

The indistinguishability argument here follows from the indistinguishability of the crs (from a real crs), the indistinguishability of the "fake" database T, the ability to extract witnesses from the NIWI proofs, and the zero-knowledge property of "fake" NIZK proofs. In particular, note that the *N-Hidden LRSW* assumption ensures that any decryption request made by the receiver corresponds to a valid ciphertext from the database T (if $\mathcal{A}$ produces a proof π embedding invalid ciphertext values, we can use $\mathcal{A}$ to solve *N-Hidden LRSW* or the co-CDH problem [7], which is implied by *N-Hidden LRSW*).[5] Unlike the protocol of [10]

[5] Note that we are using both an existentially unforgeable signature scheme, as well as a selective-ID IBE scheme that has been "retasked" as signature scheme. The latter leads to a signature that is only secure for a polynomial-sized, fixed message space. In the full version, we show that this limitation is acceptable given that we are signing the product of other messages which have been signed using the stronger signature scheme. Since there are at most a polynomial number of such products, the construction is secure.

we are able to base the semantic security of the ciphertexts on a standard decisional assumption (the Decision Linear assumption). This is possible because the full ciphertext can be constructed using only the DLIN input (see the note on Ciphertext security below). Notice that $\mathcal{S}$ is never *both* simulating and extracting via the same (subsection of the) common reference string; indeed, we do not require that the proofs be simulation-sound.

Simulating the remaining cases. When both the Receiver and Sender are corrupted, $\mathcal{S}$ knows the inputs to **S** and **R** and can simulate a protocol execution by generating the real messages exchanged between the two parties. In the case where neither party is corrupted, then: when $\mathcal{S}$ receives messages of the form (sid, b_i) indicating that transfers have occurred, $\mathcal{S}$ generates a simulated transcript between the honest **S** and **R**. In this case, $\mathcal{S}$ runs the protocol as specified, using as **S**'s input a random database $(\hat{m}_1, \ldots, \hat{m}_N)$, and (for each transfer), **R**'s input $\sigma' = 1$. If in the i^{th} transfer $b_i = 0$ then **S**'s responds with an invalid R (the empty string). Else, **S** returns a valid response as in the protocol.

Ciphertext security. We briefly elaborate on the security of the ciphertexts in our scheme. To prove security when Receiver is corrupted, we must show that a ciphertext vector encrypting random messages is indistinguishable from a vector encrypting the real message database. We argue that this is the case under the Decision Linear assumption. Let $D = (g, \tilde{g}, f, \tilde{f}, h, \tilde{h}, g^a, f^b, z_d)$ be a candidate Decision Linear tuple. We consider a simulation that behaves as follows:

1. Set $u_1 = g, u_2 = f, \tilde{u}_1 = \tilde{g}, \tilde{u}_2 = \tilde{f}$. Select random $y_1, y_2 \in \mathbb{Z}_p$, and set $g_1 = u_1^{y_1}, g_2 = u_2^{y_2}$ (and similarly for $\tilde{g}_1, \tilde{g}_2$). Fix $\mathsf{crs} \leftarrow (\gamma, GS'_S, GS'_R, g_1, g_2, h, \tilde{g}_1, \tilde{g}_2, \tilde{h})$.
2. Generate $(vk_1, sk_1), (vk_2, sk_2), (vk_3, sk_3)$ as in normal operation. Set $pk = (u_1, u_2, \tilde{u}_1, \tilde{u}_2, vk_1, vk_2, vk_3)$.
3. For $i = 1$ to N, choose fresh random $s, t_1, t_2 \in \mathbb{Z}_p$ and set $c_1 = g^{as} g^{st_1}, c_2 = f^{bs} f^{st_2}$. Set C_i:
$$C_i = (c_1, c_2, c_1^{y_1}, c_2^{y_2}, z_d^s h^{s(t_1+t_2)} m_j, \mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3)$$
where $\mathsf{sig}_1, \mathsf{sig}_2, \mathsf{sig}_3$ are generated normally using the proper secret keys.
4. Set $T \leftarrow (pk, C_1, \ldots, C_N)$.
5. The simulation answers requests from the malicious Receiver by extracting from its proof and simulating correct responses (as described above.)

Note that in the above, if $z_d = h^{a+b}$, then the above simulation perfectly encrypts $(m_1, \ldots, m_N)$. However, when z_d is a random element of $\mathbb{G}_1$, then the ciphertexts correspond to encryptions of random elements in $\mathbb{G}_1$. Now, suppose for the sake of contradiction, that there exists an environment $\mathcal{Z}$ who can distinguish case one from case two with non-negligible probability ϵ. Then, it is easy to see that we can use $\mathcal{Z}$ to decide Decision Linear.

5 On Multiple Receivers

OT is traditionally described as a two-party protocol between a Sender and Receiver. We presented our main construction in this setting. However, since we are motivated by the application of OT to database systems, we would also like to support applications where multiple users share a single database. Naively this can be accomplished by requiring the database to run separate OT protocol instances with each user. However, this approach can be quite inefficient, and moreover does not ensure *consistency* in the database viewed by individual Receivers. Consider a strengthening of the security definition of $\mathcal{F}_{OT}^{N\times 1}$ (in Figure 3) to include the additional requirement that all Receivers "view" the same database, i.e., the database owner cannot selectively alter the messages in the database when interacting with different receivers – on query σ from *any* receiver, he must return a value in $\{m_\sigma, \bot\}$. In the full version of this work [23] we discuss extensions to our protocol designed to achieve this property.

References

1. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable RFID tags via insubvertible encryption. In: CCS 2005, pp. 92–101. ACM Press, New York (2005)
2. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage from keyword searchable encryption. Cryptology ePrint Archive, Report 2005/417 (2005)
3. Belenkiy, M., Chase, M., Kolweiss, M., Lysyanskaya, A.: Non-interactive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure Identity-Based Encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 45–55. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
8. Brassard, G., Crépeau, C., Robert, J.-M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
9. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
10. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
11. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Sommer, G., Daniilidis, K., Pauli, J. (eds.) CAIP 1997. LNCS, vol. 1296, pp. 410–424. Springer, Heidelberg (1997)

12. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with pre-existing setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
13. Canetti, R.: Universally Composable Security: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145. IEEE Computer Society, Los Alamitos (2001), `http://eprint.iacr.org/2000/067`
14. Canetti, R.: Universally composable security: Towards the bare bones of trust. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 88–112. Springer, Heidelberg (2007)
15. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
16. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503. ACM Press, New York (2002)
17. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
18. Chu, C.-K., Tzeng, W.-G.: Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 172–183. Springer, Heidelberg (2005)
19. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: CRYPTO 1982, pp. 205–210 (1982)
20. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 205–210. Springer, Heidelberg (2005)
21. 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 (1987)
22. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
23. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. Cryptology ePrint Archive, Report 2008/163 (2008), `http://eprint.iacr.org/2008/163`
24. 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)
25. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC 1988, pp. 20–31 (1988)
26. Lindell, Y.: Efficient fully-simulatable oblivious transfer. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
27. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
28. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
29. Ogata, W., Kurosawa, K.: Oblivious keyword search. Special issue on coding and cryptography Special issue on coding and cryptography Journal of Complexity 20(2-3), 356–371 (2004)
30. 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)

31. Rabin, M.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University (1981)
32. Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and pin number (2002), `http://eprint.iacr.org/2002/164`
33. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
34. Yao, A.: How to generate and exchange secrets. In: FOCS, pp. 162–167 (1986)