Efficient, Adaptively Secure, and Composable Oblivious Transfer with a Single, Global CRS

Seung Geol Choi[*] Jonathan Katz[†] Hoeteck Wee[‡] Hong-Sheng Zhou[§]

February 20, 2013

Abstract

We present a general framework for efficient, universally composable oblivious transfer (OT) protocols in which a *single*, global, common reference string (CRS) can be used for multiple invocations of oblivious transfer by arbitrary pairs of parties. In addition:

- Our framework is round-efficient. E.g., under the DLIN or SXDH assumptions we achieve round-optimal protocols with static security, or 3-round protocols with adaptive security (assuming erasure).
- Our resulting protocols are more efficient than any known previously, and in particular yield protocols for string OT using $O(1)$ exponentiations and communicating $O(1)$ group elements.

Our result improves on that of Peikert et al. (Crypto 2008), which uses a CRS whose length depends on the number of parties in the network and achieves only static security. Compared to Garay et al. (Crypto 2009), we achieve adaptive security with better round complexity and efficiency.

[*] Columbia University. Email: sgchoi@cs.columbia.edu. Portions of this work were done at the University of Maryland. This work was supported in part by the Intelligence Advanced Research Project Activity (IARPA) via DoI/NBC contract Number D11PC20194. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

[†] Dept. of Computer Science, University of Maryland. Email: jkatz@cs.umd.edu. This work is supported in part by DARPA and by NSF awards #0447075, #1111599, and #1223623.

[‡] George Washington University. Email: hoeteck@alum.mit.edu. Supported by NSF CAREER award #1237429.

[§] Dept. of Computer Science, University of Maryland. Email: hszhou@cs.umd.edu. Supported by an NSF CI postdoctoral fellowship.

Contents

1 Introduction

In this work we study the construction of efficient protocols for universally composable (UC) [Can01] oblivious transfer (OT). Our work is motivated by the fact that, although UC *commitments* are complete for UC multiparty computation [CLOS02], the most efficient multiparty computation protocols (e.g., [LP11, LOP11]) rely on universally composable OT as a building block. Relative to UC commitments (see [Lin11, FLM11] and references therein), however, universally composable OT has received less attention.

There is a long series of work on efficient OT protocols in the stand-alone setting (e.g., [NP01, AIR01, HK12, Lin08]). Lindell [Lin09] (also [IPS08, Appendix A]) gave a generic transformation from static security to adaptive security (assuming erasure) that applied in the semi-honest setting and the stand-alone malicious setting, but not in the UC setting.

Constructions of UC oblivious transfer from general assumptions were given in [CLOS02]; these constructions are relatively inefficient. Garay, MacKenzie, and Yang [GMY04] constructed a constant-round protocol for *committed* OT under the DDH and strong RSA assumptions. Their protocol yields *bit* OT rather than *string* OT, so results in protocols for string OT with complexity linear in the length of the sender's inputs. Jarecki and Shmatikov show a four-round protocol for committed string OT under the decisional composite residuosity (DCR) assumption [JS07]. A round-optimal OT protocol appears in [HK07].

The most efficient known protocol for UC oblivious transfer is that of Peikert et al. [PVW08]. Their work, however, has several disadvantages. First, it requires an independent common reference string[1] (CRS) for *every party* in the network or, equivalently, a single CRS of length linear in the number of parties. (Any pair of parties can then run the protocol of Peikert et al. using the CRS of the receiver.) Their protocols also only achieve security against a *static* adversary who decides which parties to corrupt before the protocol begins (and even before the CRS is chosen). They do not handle an *adaptive* adversary who may choose which parties to corrupt during the course of the protocol execution.

Garay et al. [GWZ09] constructed efficient UC oblivious-transfer protocols that address both the above-mentioned drawbacks. In their constructions, the parties run a coin-tossing protocol whose outcome is then used as a common random string for an OT protocol. Since the resulting schemes are not explicitly described in the literature, we do so in Appendix A. This approach is not entirely satisfactory. First, it increases the overall computation, communication, and round complexity; second, it can (in general) only be instantiated with OT protocols that work in the common *random* string model rather than the more general common *reference* string model. Choi et al. [CDMW09b, CDMW09a] showed other approaches for obtaining adaptively secure, constant-round UC oblivious transfer.

1.1 Our Results

Here, we present a new framework for constructing UC oblivious-transfer protocols that require only a *single*, global CRS. We aim for efficient protocols having low round complexity, and incurring only *constant* computation and communication even when the sender's inputs are long strings. We are also interested in achieving *adaptive* security, under the assumption that parties erase portions of their local state that are no longer needed. (Note, however, that the works of [CDMW09b, GWZ09, CDMW09a] do not make this assumption.)

[1] *Some* form of setup is known to be necessary for universally composable OT [CF01, CKL06].

reference	assumption	rounds	communication complexity	CRS size
[PVW08]	DDH	2	6	n
[PVW08]+[GWZ09]+[FLM11]	DLIN	4	78	12
Protocol 1* (Sec. 4.1)	DLIN	2	54	12
[PVW08]+[GWZ09]+[Lin11]	DDH	6	38	7
Protocol 2 (Sec. 5.1, Sec. 6.1)	DDH	4	32	6
[JS07]	DCR	4	$35\ (\mathbb{Z}_{N^2}) + 16\ (\mathbb{Z}_N)$	10
Protocol 2 (Sec. 5.1, Sec. 6.2)	DCR	4	$18\ (\mathbb{Z}_{N^2}) + 7\ (\mathbb{Z}_N)$	12

Protocols with static security.

reference	assumption	rounds	communication complexity	CRS size
[PVW08]+[GWZ09]+[FLM11]	DLIN	4	83	12
Protocol 1* (Sec. 4.1)	DLIN	3	59	12
[PVW08]+[GWZ09]+[Lin11]	DDH	8	51	7
Protocol 2* (Sec. 5.2, Sec. 6.1)	DDH	4	35	6
Protocol 2* (Sec. 5.2, Sec. 6.2)	DCR	4	$21\ (\mathbb{Z}_{N^2}) + 7\ (\mathbb{Z}_N)$	12

Protocols with adaptive security (assuming erasure).

Table 1: Efficient universally composable protocols for string OT. Here, n is the number of parties. Communication complexity and CRS size are measured in terms of group elements, with other values ignored. The numbers for [JS07] include the cost of the pre-processing (off-line) stage.

Our framework is fairly general and can be instantiated from several assumptions. Specifically:

- We obtain efficient, *round-optimal* OT protocols with static security under the decisional linear (DLIN) [BBS04] or symmetric external Diffie-Hellman (SXDH) assumptions [Sco02, BBS04]. These protocols can be modified to achieve adaptive security (assuming erasure) with one additional round and a slight increase in communication and computation.
- We obtain efficient, four-round OT protocols under the decisional Diffie-Hellman (DDH) or DCR [Pai99] assumptions. Our basic constructions achieve static security, and we present variants that are secure against adaptive corruptions (assuming erasure) *without* any additional rounds, but with a slight increase in communication and computation.

We compare our constructions with previous work in Table 1[2]

Overview of our constructions. The starting point of our approach is the Halevi-Kalai construction [HK12] of 2-round OT based on smooth projective hashing. Their construction only achieves indistinguishability-based security (and not even stand-alone simulation-based security) against a malicious receiver. We show how to overcome this with the following modifications:

1. We require the receiver to commit to its input using CCA-secure encryption.

[2] The numbers for the adaptively secure protocol of [PVW08]+[GWZ09]+[Lin11] in Table 1 are based on a preliminary version of [Lin11], and could change once the author publishes the fix to a bug in the protocol.

2. The receiver proves in zero knowledge that it is behaving consistently in the underlying OT protocol (with respect to the input it committed to).

A similar high-level approach was taken in [HK07], but using generic simulation-sound non-interactive zero knowledge [DDO$^+$01]. Here, following recent constructions of efficient UC commitments [Lin11, FLM11], we rely instead on efficient zero-knowledge protocols that admit straight-line simulation in the CRS model. In particular, for our two-round OT protocols we instantiate the underlying zero-knowledge proofs using Groth-Sahai proofs [GS08], as in [FLM11]. For our four-round OT protocols, we rely on Damgård's three-round zero-knowledge proof system [Dam00].

Achieving adaptive security. To achieve adaptive security, we first modify our protocols so the final message is sent over an adaptively secure channel (cf. functionality $\mathcal{F}_{\text{SMT}}$ in [Can01]). The latter can be realized at low cost if erasure is assumed [BH92]. With this modification, security against adaptive corruption of the sender is achieved automatically by simply having the sender erase its local state at appropriate times. In our two-round protocols, security against adaptive corruption of the receiver is similarly achieved. For our 4-round protocols, we use techniques similar to those in [Lin11, FLM11]. Unlike this prior work, however, we do not introduce any additional overhead in communication or round complexity. (We incur a modest increase in computational cost.)

1.2 Organization

We review preliminaries in Section 2. Our general framework and concrete instantiations for two-round OT is described in Sections 3 and 4, and for four-round OT in Sections 5 and 6.

2 Preliminaries

Let λ be the security parameter. For a set S, we let $x \leftarrow S$ denote choosing x uniformly at random from S. For a randomized algorithm A, we let $y \leftarrow A(x)$ denote running $A(x; r)$ with r chosen uniformly at random.

2.1 Ideal Functionalities

The ideal OT functionality $\mathcal{F}_{\mathsf{MOT}}$ described in Figure 1 is a reformulation of the multi-session OT functionality in [Can01].

The CRS functionality in Figure 2 is a reformulation of that in [Can07].

2.2 Smooth Projective Hash Proof Systems

We recall the notion of a hard subset membership problem and smooth projective hashing defined by Cramer and Shoup [CS02], following the notation of [HK12]. A hash family $\mathcal{H}$ consists of the following PPT algorithms:

- The *parameter-generator* $\mathsf{HashPG}(1^\lambda) \rightarrow \textsc{pp}$. We assume that the security parameter λ can be inferred from $\textsc{pp}$. Let $\lambda(\textsc{pp})$ denote the security parameter corresponding to $\textsc{pp}$.

Functionality $\mathcal{F}_{\mathsf{MOT}}$

$\mathcal{F}_{\mathsf{MOT}}$ interacts with parties $P_1, \ldots, P_n$ and an adversary Sim, and proceeds as follows:

Upon receiving an input $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \langle m_0, m_1 \rangle)$ from P_i with $m_0, m_1 \in \{0,1\}^\ell$, record the tuple $\langle \mathsf{ssid}, P_i, P_j, m_0, m_1 \rangle$, and reveal $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ to the adversary. Ignore further $(\textsc{send}, \ldots)$ inputs with the same ssid from P_i.

Upon receiving an input $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, b)$ from P_j with $b \in \{0,1\}$, record the tuple $\langle \mathsf{ssid}, P_i, P_j, b \rangle$, and reveal $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ to the adversary. Ignore further $(\textsc{receive}, \ldots)$ inputs with the same ssid from P_j.

Upon receiving a message $(\textsc{sent}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, P_i)$ from the adversary, ignore the message if $\langle \mathsf{ssid}, P_i, P_j, m_0, m_1 \rangle$ or $\langle \mathsf{ssid}, P_i, P_j, b \rangle$ is not recorded; Otherwise, return $(\textsc{sent}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ to P_i; Ignore further $(\textsc{sent}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, P_i)$ messages from the adversary.

Upon receiving a message $(\textsc{received}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, P_j)$ from the adversary, ignore the message if $\langle \mathsf{ssid}, P_i, P_j, m_0, m_1 \rangle$ or $\langle \mathsf{ssid}, P_i, P_j, b \rangle$ is not recorded; Otherwise, return $(\textsc{received}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, m_b)$ to P_j; Ignore further $(\textsc{received}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, P_j)$ messages from the adversary.

Figure 1: The ideal $\mathcal{F}_{\mathsf{MOT}}$ functionality for muti-session string-OT.

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

Upon receiving an input $(\textsc{crs}, \mathsf{sid})$ from P, $\mathsf{sid} = (\mathcal{P}, sid')$ where $\mathcal{P}$ is a set of identities, and $P \in \mathcal{P}$; else ignore the input. Next if there is no value crs recorded then choose $\mathsf{crs} \leftarrow \mathcal{D}()$ and record it. Send $(\textsc{crs}, \mathsf{sid}, \mathsf{crs}, P)$ to $\mathcal{S}$; when receiving $(\textsc{crs-return}, \mathsf{sid})$ from $\mathcal{S}$, send $(\textsc{crs-return}, \mathsf{sid}, \mathsf{crs})$ to P.

Figure 2: The ideal $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{P},\mathcal{D}}$ functionality for common reference string.

- A pair of disjoint sets Λ_{YES} and Λ_{NO} are associated to $\textsc{pp}$ corresponding to YES and NO instances respectively. There exists a YES *instance-sampler* $\mathsf{SampYes}(\textsc{pp}) \rightarrow (x, w)$ where x is uniformly distributed over Λ_{YES} and w is the corresponding witness. There also exists a NO *instance-sampler* $\mathsf{SampNo}(\textsc{pp}) \rightarrow x'$ where x' is uniformly distributed over Λ_{NO}.

- The *hash-key generator* $\mathsf{HashKG}(\textsc{pp}) \rightarrow (\textsc{hk}, \textsc{pk})$. Here HK is the primary hashing key and PK is a projective key.

- The *primary hash algorithm* $\mathsf{Hash}(\textsc{hk}, x) \rightarrow y$ for all $x \in \Lambda_{\text{YES}} \cup \Lambda_{\text{NO}}$.

- The *secondary (projection) hash algorithm* $\mathsf{pHash}(\textsc{pk}, x, w) \rightarrow y$ for all $(x, w) \leftarrow \mathsf{SampYes}(\textsc{pp})$.

Definition 1 *A family* $\mathcal{H} = (\mathsf{HashPG}, \mathsf{SampYes}, \mathsf{SampNo}, \mathsf{HashKG}, \mathsf{Hash}, \mathsf{pHash})$ *is a* smooth projective hash family *if for every* $\textsc{pp} \in \mathsf{support}(\mathsf{HashPG})$*, the following holds:*

Correctness:

$$\Pr[(\textsc{hk}, \textsc{pk}) \leftarrow \mathsf{HashKG}(\textsc{pp}); (x, w) \leftarrow \mathsf{SampYes}(\textsc{pp}) : \mathsf{pHash}(\textsc{pk}, x, w) = \mathsf{Hash}(\textsc{hk}, x)\ ] = 1.$$

Smoothness: *Let* $(\textsc{hk}, \textsc{pk}) \leftarrow \mathsf{HashKG}(\textsc{pp})$*. For all* $x \in \Lambda_{\text{NO}}$*, the distribution of* $\mathsf{Hash}(\textsc{hk}, x)$ *given* PK *is statistically close to uniform. That is, the statistical difference between the following two distributions is negligible in* $\lambda(\textsc{pp})$*.*

$$\{y \leftarrow \mathsf{Hash}(\textsc{hk}, x) :\ (\textsc{pk}, y, x)\} \overset{s}{\equiv} \{y \leftarrow \Gamma :\ (\textsc{pk}, y, x)\}$$

where Γ *denotes the set of possible hash values with parameter* PP*.*

Definition 2 *A smooth projective hash family* $\mathcal{H} = (\mathsf{HashPG}, \mathsf{SampYes}, \mathsf{SampNo}, \mathsf{HashKG}, \mathsf{Hash}, \mathsf{pHash})$ *is said to have a* hard subset membership *property if the following two ensembles are computationally indistinguishable:*

- $\left\{\textsc{pp} \leftarrow \mathsf{HashPG}(1^\lambda);\ (x, w) \leftarrow \mathsf{SampYes}(\textsc{pp}) :\ (\textsc{pp}, x)\right\}_{\lambda \in \mathbb{N}}$

- $\left\{\textsc{pp} \leftarrow \mathsf{HashPG}(1^\lambda);\ x \leftarrow \mathsf{SampNo}(\textsc{pp}) :\ (\textsc{pp}, x)\right\}_{\lambda \in \mathbb{N}}$.

2.3 Dual-Mode NIZK

Groth introduced composable non-interactive zero-knowledge proofs [Gro06]. In this work, we sometimes call these proofs as *dual-mode NIZK* proofs. In the system, a common reference string crs is generated in one of two modes called *soundness* mode and *zero knowledge* (ZK) mode. It is required that given the crs, no efficient adversary can distinguish between the modes. In addition, when crs is generated in soundness mode, the proof system is statistically sound. On the other hand, when crs is generated in ZK mode, the simulation is perfect. Groth and Sahai [GS08] provide efficient dual mode NIZK proofs for various equations in bilinear groups.

Definition 3 *A* non-interactive proof system *for a language* $L \in \mathcal{NP}$ *consists of three algorithms* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *where* $\mathcal{K}$ *is a CRS generation algorithm,* $\mathcal{P}$ *and* $\mathcal{V}$ *are a prover and a verifier algorithm respectively. The system is required to satisfy the following properties:*

Completeness: *For any λ, any $x \in L$, and any witness w for x, it holds that*

$$\Pr[\mathsf{crs} \leftarrow \mathcal{K}(1^\lambda);\ \pi \leftarrow \mathcal{P}(1^\lambda, \mathsf{crs}, x, w):\ \mathcal{V}(1^\lambda, \mathsf{crs}, x, \pi) = 1] = 1.$$

Adaptive soundness: *For any λ and any adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathsf{crs} \leftarrow \mathcal{K}(1^\lambda);\ (x, \pi) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}):\ \mathcal{V}(1^\lambda, \mathsf{crs}, x, \pi) = 1\ \wedge x \notin L] = \mathsf{negl}(\lambda).$$

Definition 4 *A non-interactive proof system $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ for a language $L \in \mathcal{NP}$ is said to be* dual-mode NIZK *if there is a pair of efficient algorithms $(\mathcal{S}_1, \mathcal{S}_2)$ such that for any $\lambda \in \mathbb{N}$ and for all non-uniform polynomial time adversary $\mathcal{A}$, it holds the following:*

Indistinguishability of modes:

$$\left| \Pr[\mathsf{crs} \leftarrow \mathcal{K}(1^\lambda):\ \mathcal{A}(1^\lambda, \mathsf{crs}) = 1] - \Pr[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\lambda):\ \mathcal{A}(1^\lambda, \mathsf{crs}) = 1] \right| = \mathsf{negl}(\lambda).$$

Perfect simulation in ZK mode: *The following two probabilities are equal.*

- $\Pr[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\lambda);\ (x, w) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \tau);\ \pi \leftarrow \mathcal{P}(1^\lambda, \mathsf{crs}, x, w) : \mathcal{A}(\pi) = 1]$
- $\Pr[(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^\lambda);\ (x, w) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \tau);\ \pi \leftarrow \mathcal{S}_2(\tau, x) : \mathcal{A}(\pi) = 1]$

Here, $\mathcal{A}$ has to generate a valid pair (x, w), i.e., w should be a witness for x.

2.4 Labeled CCA-Secure Encryption

We use the standard notion of chosen-ciphertext security for public-key encryption supporting labels [CS03].

Definition 5 *A* labeled public-key encryption scheme *consists of three PPT algorithms* (Gen, Enc, Dec) *such that:*

- *The* key generation algorithm Gen *takes as input a security parameter 1^λ returns a public key pk and a secret key sk.*
- *The* encryption algorithm Enc *takes as input a public key pk, a label L, and a message m. It returns a ciphertext $C \leftarrow \mathsf{Enc}^L_{pk}(m)$.*
- *The* decryption algorithm Dec *takes as input a secret key sk, a label L, and a ciphertext C. It returns a message m or a distinguished symbol $\perp$. We write this as $m = \mathsf{Dec}^L_{sk}(C)$ or $m = \mathsf{Dec}_{sk}(L, C)$.*

We require that for any label $L \in \{0,1\}^*$ and any m in the message space, it should hold that

$$\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda);\ \mathsf{Dec}^L_{sk}(\mathsf{Enc}^L_{pk}(m)) = m] = 1.$$

Our definition of security against chosen-ciphertext attacks is standard except for our inclusion of labels. In the following, we define a left-or-right encryption oracle $\mathbf{LR}_{pk,\beta}(\cdot,\cdot,\cdot)$ with $\beta \in \{0,1\}$ as follows:

$$\mathbf{LR}_{pk,\beta}(L, m_0, m_1) \overset{\text{def}}{:=} \mathsf{Enc}^L_{pk}(m_\beta).$$

Definition 6 *A labeled public-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* secure against adaptive chosen-ciphertext attacks *(CCA-secure) if for any non-uniform PPT algorithms* $\mathcal{A}$*, it holds that* $|2p-1| = \mathsf{negl}(\lambda)$ *where*

$$p = \Pr\left[(pk, sk)\leftarrow\mathsf{Gen}(1^\lambda); \beta\leftarrow\{0,1\} : \mathcal{A}^{\mathbf{LR}_{pk,\beta}(\cdot,\cdot,\cdot),\mathsf{Dec}_{sk}(\cdot,\cdot)}(1^\lambda, pk) = \beta\right].$$

Here, $\mathcal{A}$*'s queries are restricted as follows: if* $\mathcal{A}$ *makes a query* $\mathbf{LR}_{pk,\beta}(L, m_0, m_1)$ *then* $|m_0| = |m_1|$*; furthermore, if* $\mathcal{A}$ *receives ciphertext* C *in response to a query with label* L*, then* $\mathcal{A}$ *cannot later query* $\mathsf{Dec}_{sk}(L, C)$*; however, it is allowed to query* $\mathsf{Dec}_{sk}(L', C)$ *with* $L \neq L'$*.*

2.5 Σ-Protocols

A Σ-protocol is a 3-round honest-verifier zero-knowledge protocol. We denote by (a, e, z) the messages exchanged between the prover $\mathcal{P}_\Sigma$ and the verifier $\mathcal{V}_\Sigma$. Please see Figure 3. We say that a transcript (a, e, z) is an accepting transcript for x if the protocol instructs $\mathcal{V}_\Sigma$ to accept based on the values (x, a, e, z). Formally:

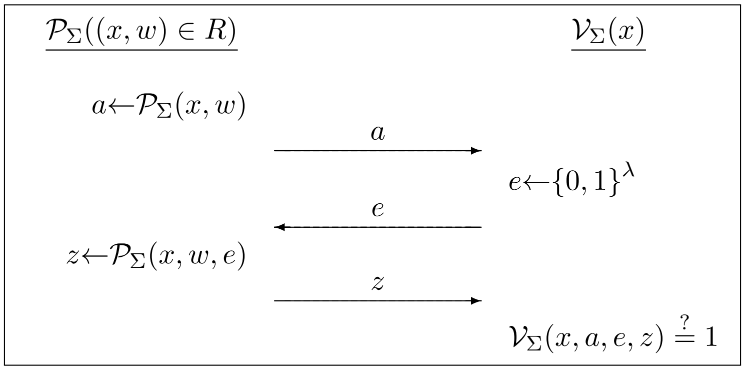

$\mathcal{P}_\Sigma((x,w) \in R)$
$\mathcal{V}_\Sigma(x)$
$a\leftarrow\mathcal{P}_\Sigma(x, w)$
a
$e\leftarrow\{0,1\}^\lambda$
e
$z\leftarrow\mathcal{P}_\Sigma(x, w, e)$
z
$\mathcal{V}_\Sigma(x, a, e, z) \overset{?}{=} 1$

Figure 3: A Σ-protocol $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ for relation R.

Definition 7 *A protocol is a* Σ*-protocol for relation* R *if it is a three-round public-coin protocol and the following requirements hold:*

Completeness: *If* $\mathcal{P}_\Sigma$ *and* $\mathcal{V}_\Sigma$ *follow the protocol on input* x *and private input* w *to* $\mathcal{P}_\Sigma$ *where* $(x, w) \in R$*, then* $\mathcal{V}_\Sigma$ *always accepts.*

Special soundness: *There exists a polynomial-time algorithm* $\mathcal{A}$ *that given any* x *and any pair of accepting transcripts* (a, e, z)*,* (a, e', z') *for* x *where* $e \neq e'$*, outputs* w *s.t.* $(x, w) \in R$*.*

Special honest verifier zero knowledge: *There exists a probabilistic polynomial-time simulator* $\mathcal{S}_\Sigma$*, which on input* x *and* e *outputs a transcript of the form* (a, e, z) *with the same probability distribution as transcripts between the honest* $\mathcal{P}_\Sigma$ *and* $\mathcal{V}_\Sigma$ *on common input* x*. Formally, for every* x *and* w *such that* $(x, w) \in R$ *and every* $e \in \mathsf{support}(\mathcal{V}_\Sigma(x))$ *it holds that*

$$\{\mathcal{S}_\Sigma(x, e)\} = \{\textsc{view}_{\mathcal{V}_\Sigma}\langle\mathcal{P}_\Sigma(x, w), \mathcal{V}_\Sigma(x, e)\rangle\}$$

where $\mathcal{S}_\Sigma(x, e)$ *denotes the output of simulator* $\mathcal{S}_\Sigma$ *upon input* x *and* e*, and* $\textsc{view}_{\mathcal{V}_\Sigma}\langle\mathcal{P}_\Sigma(x, w),$ $\mathcal{V}_\Sigma(x, e)\rangle$ *denotes the view of* $\mathcal{V}_\Sigma$ *of an execution between* $\mathcal{P}_\Sigma$ *and* $\mathcal{V}_\Sigma$*, where* $\mathcal{P}_\Sigma$ *has input* (x, w)*,* $\mathcal{V}_\Sigma$ *has input* x*, and* $\mathcal{V}_\Sigma$*'s random tape (determining its query) equals* e*.*

2.6 Equivocal Commitments

We define equivocal commitment scheme as follows:

Definition 8 *Let* $(\mathcal{K}_{com}, \mathsf{Com})$ *be a non-interactive commitment scheme with CRS where* $\mathcal{K}_{com}$ *is a CRS generation algorithm, and* Com *is a commitment algorithm. The scheme is said to be* equivocal *if there exists a tuple of PPT algorithm* $(\mathcal{S}_{com1}, \mathcal{S}_{com2}, \mathcal{S}_{com3})$ *that satisfies the following properties:*

Computational binding: *Let* $\mathcal{M}$ *and* $\mathcal{R}$ *be the message space and the randomness space implicitly defined by the commitment scheme. For any* $\lambda \in \mathbb{N}$*, and any non-uniform polynomial time adversary* $\mathcal{A}$ *the following probability is negligible in* λ*:*

$$\Pr\left[\mathsf{crs}\leftarrow\mathcal{K}_{com}(1^\lambda);\ (m, m', r, r')\leftarrow\mathcal{A}(\mathsf{crs}):\ m \neq m' \text{ and } \mathsf{Com}_{\mathsf{crs}}(m; r) = \mathsf{Com}_{\mathsf{crs}}(m'; r')\right]$$

Indistinguishability of modes:

$$\left\{\mathsf{crs}\leftarrow\mathcal{K}_{com}(1^\lambda):\ \mathsf{crs}\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{(\mathsf{crs}, t)\leftarrow\mathcal{S}_{com1}(1^\lambda):\ \mathsf{crs}\right\}_{\lambda\in\mathbb{N}}$$

Equivocality: *For any* $\lambda \in \mathbb{N}$*, any* $(\mathsf{crs}, t) \in \mathsf{support}(\mathcal{S}_{com1}(1^\lambda))$*, and any adversary* $\mathcal{A}$*, the following distributions are identical.*

- $\left\{m\leftarrow\mathcal{A}(\mathsf{crs});\ r\leftarrow\mathcal{R};\ c = \mathsf{Com}_{\mathsf{crs}}(m; r):\ (m, r, c)\right\}$
- $\left\{m\leftarrow\mathcal{A}(\mathsf{crs});\ (c, s)\leftarrow\mathcal{S}_{com2}(t);\ r\leftarrow\mathcal{S}_{com3}(s, m):\ (m, r, c)\right\}$

2.7 A Family of Collision-Resistant Hash Functions

$\mathcal{HF} = \left\{h_k : \{0,1\}^* \to \{0,1\}^{\ell(\lambda)}\right\}_{k\in\{0,1\}^\lambda}$ with a polynomial ℓ is a family of collision-resistant hash functions if for any non-uniform PPT algorithm $\mathcal{A}$, it holds that

$$\Pr[k \leftarrow \{0,1\}^\lambda : \mathcal{A}(k) = (x_1, x_2) \text{ s.t. } x_1 \neq x_2 \text{ and } h_k(x_1) = h_k(x_2)] = \mathsf{negl}(\lambda).$$

3 A Generic Framework for Two-Round OT (Protocol 1*)

Let $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ be a dual-mode NIZK proof system, $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure labeled public-key encryption scheme, and $\mathcal{H} = (\mathsf{HashPG}, \mathsf{SampYes}, \mathsf{SampNo}, \mathsf{HashKG}, \mathsf{Hash}, \mathsf{pHash})$ be a smooth hash proof system with a hard subset membership property. We assume for simplicity that $\{0,1\}^\ell$ is the range of the hash functions in $\mathcal{H}$; known constructions can be modified to achieve this property. Based on these components, we construct an OT protocol between a sender P_i and a receiver P_j in the CRS model; refer also to Figure 4.

Common reference string: Compute $\textsc{pp}\leftarrow\mathsf{HashPG}(1^\lambda)$, $(pk, sk)\leftarrow\mathsf{Gen}(1^\lambda)$, and $\mathsf{crs}_{nizk}\leftarrow\mathcal{K}(1^\lambda)$. The common reference string is $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$.

Oblivious transfer: The protocol starts by having the receiver, holding selection bit b, send two instances (x_0, x_1) for the hash proof system $\mathcal{H}$ with x_{1-b} a NO-instance; the receiver sends $\mathsf{Enc}_{pk}(b)$ and a NIZK proof that x_{1-b} is a NO-instance as well. In the second round, for $\sigma \in \{0,1\}$ the sender generates primary and projection hash keys $(\textsc{hk}_\sigma, \textsc{pk}_\sigma)$ and sends $(\textsc{pk}_\sigma, \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma) \oplus m_\sigma)$ to the receiver. The receiver recovers m_b in the standard way. In more detail:

- On input a selection bit b, the receiver P_j proceeds as follows:

  1. Generate YES and NO instances according to the input b, i.e., $(x_b, w) \leftarrow \mathsf{SampYes}(\textsc{pp})$ and $x_{1-b} = \mathsf{SampNo}(\textsc{pp}; \gamma)$ for uniformly random γ. Compute $\Phi = \mathsf{Enc}^L_{pk}(b; \xi)$ with uniformly random ξ, where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. Generate a proof $\pi \leftarrow \mathcal{P}(\mathsf{crs}_{nizk}, (\textsc{pp}, pk, L, x_0, x_1, \Phi), (b, \gamma, \xi))$ for language
  $$\mathcal{L}^* = \big\{(\textsc{pp}, pk, L, x_0, x_1, \Phi) : \exists (b, \gamma, \xi) \text{ s.t. } x_{1-b} = \mathsf{SampNo}(\textsc{pp}; \gamma), \Phi = \mathsf{Enc}^L_{pk}(b; \xi)\big\}.$$
  2. Send $\langle x_0, x_1, \Phi, \pi \rangle$.

- On input $m_0, m_1 \in \{0,1\}^\ell$, and after receiving the first-round message $\langle x_0, x_1, \Phi, \pi \rangle$ from the receiver, the sender P_i proceeds as follows:

  1. If the proof π does not verify, abort.
  2. For $\sigma \in \{0,1\}$ compute $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$ and $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$.
  3. Send $\langle \textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1 \rangle$ to P_j.

- Upon receiving the second-round message $\langle \textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1 \rangle$, the receiver P_j computes the output $m_b = Z_b \oplus \mathsf{pHash}(\textsc{pk}_b, x_b, w)$.

Informally, security against a malicious sender holds because the sender cannot guess the receiver's selection bit from (x_0, x_1) due to the hard subset membership property. On the other hand, a malicious receiver gets no information about m_{1-b} if x_{1-b} is a NO-instance, and this property is enforced by the NIZK proof.

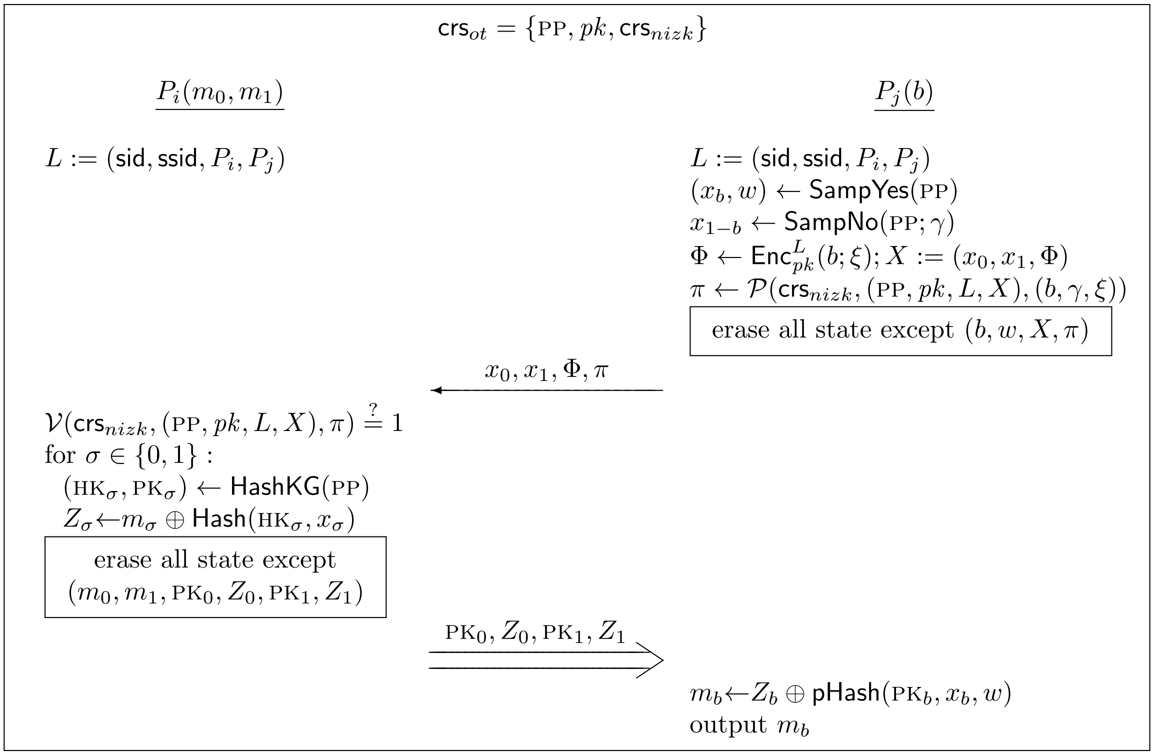
Theorem 1 *Say* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CCA-secure labeled public-key encryption scheme,* $(\mathsf{HashPG}, \mathsf{SampYes}, \mathsf{SampNo}, \mathsf{HashKG}, \mathsf{Hash}, \mathsf{pHash})$ *is a smooth projective hash proof system with a hard subset membership property, and* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *is a dual-mode NIZK proof system. Then the protocol described above securely realizes* $\mathcal{F}_{\mathsf{MOT}}$ *in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model, for static corruptions.*

Here, we give an informal overview of the proof. The simulator works as follows:

- The simulator sets the CRS $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$. It generates PP honestly. For the public key pk, it additionally keeps the corresponding secret key sk. For crs_{nizk}, it runs the CRS generation algorithm in the ZK mode rather than in the soundness mode.

- In simulating an honest sender against a malicious receiver, it uses the secret key sk to decrypt the encryption Φ and extract the receiver's input $\hat{b}$ (and thereby $m_{\hat{b}}$ from the OT functionality). It sets the honest sender's $m_{1-\hat{b}}$ to 0^ℓ.

- In simulating an honest receiver against a malicious sender, it generates (x_0, x_1) where both are YES-instances; additionally, $\Phi = \mathbf{E}(0)$, and π is simulated. Given the sender messages, it extracts the sender input (m_0, m_1) using the witnesses for (x_0, x_1).



$\mathsf{crs}_{ot} = \{\text{PP}, pk, \mathsf{crs}_{nizk}\}$

$\underline{P_i(m_0, m_1)}$

$L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$

$\underline{P_j(b)}$

$L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
$(x_b, w) \leftarrow \mathsf{SampYes}(\text{PP})$
$x_{1-b} \leftarrow \mathsf{SampNo}(\text{PP}; \gamma)$
$\Phi \leftarrow \mathsf{Enc}^L_{pk}(b; \xi); X := (x_0, x_1, \Phi)$
$\pi \leftarrow \mathcal{P}(\mathsf{crs}_{nizk}, (\text{PP}, pk, L, X), (b, \gamma, \xi))$
$\boxed{\text{erase all state except } (b, w, X, \pi)}$

$\xleftarrow{x_0, x_1, \Phi, \pi}$

$\mathcal{V}(\mathsf{crs}_{nizk}, (\text{PP}, pk, L, X), \pi) \overset{?}{=} 1$
for $\sigma \in \{0, 1\}$:
$\quad (\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$
$\quad Z_\sigma \leftarrow m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$
$\boxed{\text{erase all state except } (m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)}$

$\xRightarrow{\text{PK}_0, Z_0, \text{PK}_1, Z_1}$

$m_b \leftarrow Z_b \oplus \mathsf{pHash}(\text{PK}_b, x_b, w)$
output m_b

Figure 4: An adaptively secure OT protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model (**Protocol 1***). The second round message is sent over a secure channel (for static security, the message can be simply sent over an authenticated channel).

Indistinguishability is shown by going over the following hybrids:

- H_0: Real execution.
- H_1: The simulator sets up the CRS in such a way that it knows the decryption key for pk. When the receiver is corrupted, the simulator decrypts Φ to obtain $\hat{b}$ and switches the honest sender's $m_{1-\hat{b}}$ to 0^ℓ. Indistinguishability follows from the soundness of the proof π and smoothness of the hash proof system.
- H_2: The simulator generates crs_{nizk} in the ZK mode.
- H_3: When interacting with a corrupted sender, the simulator generates a simulated proof π. Indistinguishability follows from the zero-knowledge property of the proof system.
- H_4: When interacting with a corrupted sender, the simulator sets Φ to an encryption of 0. Indistinguishability follows from the CCA security of the encryption scheme.
- H_5: When interacting with a corrupted sender, the simulator generates both (x_0, x_1) as YES instances. Indistinguishability follows from the hard subset membership property.
- H_6: When interacting with a corrupted sender, the simulator uses witnesses for (x_0, x_1) to extract (m_0, m_1). H_5 and H_6 are identically distributed.

A full proof of Theorem 1 is given in Section 3.1.

The protocol as described is secure against static corruptions. Interestingly, it achieves adaptive security if the second round message is sent over a secure channel, and parties can erase portions of their local state at the appropriate times. Namely, before sending the message (x_0, x_1, Φ, π), the receiver erases all its internal state except the information needed (i.e., the input b and the witness w for x_b as well as (x_0, x_1, Φ, π)) to later compute the output. Before sending the message $(\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$, the sender erases all its internal state except its input (m_0, m_1) and the message. The details are in Figure 4.

Theorem 2 *Under the same assumptions as Theorem 1, the protocol described in Figure 4, where the second round message is sent over a secure channel, securely realizes $\mathcal{F}_{\mathsf{MOT}}$ in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model, for adaptive corruptions (assuming erasure).*

The proof appears in Section 3.2.

3.1 Proof of Theorem 1

Let Π denote the OT protocol under consideration. To show the security of the protocol, we need to construct a simulator Sim for any non-uniform PPT environment $\mathcal{Z}$ such that $\textsc{exec}^{\mathcal{F}_{\mathsf{CRS}}}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \textsc{ideal}_{\mathcal{F}_{\mathsf{MOT}},\mathsf{Sim},\mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary. We first construct a simulator and then argue the indistinguishability between the two ensembles.

Let $(\mathcal{S}_1, \mathcal{S}_2)$ be the simulator for the dual-mode NIZK proof system $(\mathcal{K}, \mathcal{P}, \mathcal{V})$.

3.1.1 The Simulator

Initialization step: The simulator Sim generates the common reference string as follows:

1. Compute $\textsc{pp} \leftarrow \mathsf{HashPG}(1^\lambda)$;
2. Compute $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$;
3. Compute $(\mathsf{crs}_{nizk}, \tau_{nizk}) \leftarrow \mathcal{S}_1(1^\lambda)$;
4. Set $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$.

Simulating the communication with $\mathcal{Z}$: Upon receiving an input value from $\mathcal{Z}$, the simulator Sim writes it on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$); upon obtaining an output value from $\mathcal{A}$, the simulator Sim writes it on $\mathcal{Z}$'s output tape (as if coming from $\mathcal{A}$).

Case 1: Simulating honest receiver P_j with honest sender P_i:

1. Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator delivers $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ that intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, \pi)$ which is generated as follows:

   Generate two YES instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\textsc{pp})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\textsc{pp})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, and then obtain a simulated proof π by using the simulator $\mathcal{S}_2$ for the NIZK, i.e., compute $\pi \leftarrow \mathcal{S}_2(\tau_{nizk}, (\textsc{pp}, pk, L, x_0, x_1, \Phi))$, where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

2. Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, and the first round message has been delivered from P_j to P_i, the simulator Sim then sets $m_0 = m_1 = 0^\ell$, and computes $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$ for $\sigma \in \{0,1\}$, and generates a second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ that it intends to send from P_i to P_j, where $M_2 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$. The second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ is then delivered to $\mathcal{A}$.

Case 2: Simulating honest receiver P_j with corrupted sender P_i:

1. Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator delivers $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ that intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, \pi)$ which is generated as follows:

   Generate two YES instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\textsc{pp})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\textsc{pp})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, and then obtain a simulated proof π by using the simulator $\mathcal{S}_2$ for the NIZK, i.e., compute $\pi \leftarrow \mathcal{S}_2(\tau_{nizk}, (\textsc{pp}, pk, L, x_0, x_1, \Phi))$, where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

2. Upon receiving a second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ from the adversary $\mathcal{A}$ that intends to send from P_i to P_j, where $M_2 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$, the simulator use (w_0, w_1) to compute $\hat{m}_0 = Z_0 \oplus \mathsf{pHash}(\textsc{pk}_1, x_0, w_0)$ and $\hat{m}_1 = Z_1 \oplus \mathsf{pHash}(\textsc{pk}_1, x_1, w_1)$. The simulator Sim sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \langle \hat{m}_0, \hat{m}_1 \rangle)$ to $\mathcal{F}_{\mathsf{MOT}}$.

Case 3: Simulating corrupted receiver P_j with honest sender P_i:

1. Upon receiving a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ from $\mathcal{A}$ that it intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, \pi)$, the simulator Sim first verifies if π is valid. If the verification fails, then Sim aborts; otherwise, Sim decrypts Φ into $\hat{b}$, i.e., $\hat{b} = \mathsf{Dec}^L_{sk}(\Phi)$ with $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. If $\hat{b} \notin \{0,1\}$, Sim aborts.

2. Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator Sim then sends $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \hat{b})$ to the ideal functionality $\mathcal{F}_{\mathsf{MOT}}$ in the name of dummy receiver and obtains $(\textsc{received}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, m_{\hat{b}})$. Set $m_{1-\hat{b}} = 0^\ell$. Then the simulator computes $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$ for $\sigma \in \{0,1\}$, and sends a second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ to $\mathcal{A}$ that it intends to send from P_i to P_j, where $M_2 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$.

Case 4: Simulating corrupted receiver P_j with corrupted sender P_i: This is the trivial case. Now the simulator Sim just runs $\mathcal{A}$ internally. Note that now $\mathcal{A}$ itself generates the messages between the sender and the receiver.

3.1.2 Indistinguishability

Here we show by hybrid arguments that $\textsc{exec}^{\mathcal{F}_{\mathsf{CRS}}}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \textsc{ideal}_{\mathcal{F}_{\mathsf{MOT}},\mathsf{Sim},\mathcal{Z}}$. Since the case in which both players are corrupted (or honest) is easy (i.e., Case 1 and 4), we here ignore the analysis for those two cases and focus the remaining two difficult cases: honest receiver with corrupted sender, and honest sender with corrupted receiver. Here, We highlight the main features of each hybrid.

- H_0: Real execution.

- H_1: When simulating honest senders, the simulator decrypts Φ from a corrupted receiver to obtain $\hat{b}$ and switches the honest sender's $m_{1-\hat{b}}$ to 0^ℓ.
- H_2: Use a simulated crs_{nizk}.
- H_3: In simulating honest receivers, the simulator generates a simulated proof π.
- H_4: In simulating honest receivers, the simulator sends a dummy encryption $\Phi = \mathsf{Enc}(0)$.
- H_5: In simulating honest receivers, the simulator generates (x_0, x_1) with both YES instances.
- H_6: In simulating honest receivers, the simulator uses witnesses for (x_0, x_1) to extract (m_0, m_1) from a dishonest sender.

Hybrid H_0: This is $\text{EXEC}^{\mathcal{F}_{\mathsf{CRS}}}_{\Pi,\mathcal{A},\mathcal{Z}}$.

Hybrid H_1: In this hybrid, when the environment feeds inputs, *the input of an honest receiver is magically forwarded to the simulator* Sim_1. The simulator Sim_1 generates CRS, and simulates the behavior of all the honest parties. At the end of the protocol execution, the output from Sim_1 (as an honest party) is forwarded to $\mathcal{Z}$.

In simulating the behavior of honest receivers, Sim_1 executes the protocol Π with the adversary $\mathcal{A}$, simply following the protocol specification using the magically given input; however, the input to an honest sender will be suppressed as in the ideal world, and more care is necessary.

Simulation of CRS. As in H_0. However, the simulator Sim_1 stores the secret key *sk* for the CCA-secure cryptosystem in order to use it later.

Simulating honest receiver P_j with corrupted sender P_i. As in H_0.

Simulating honest sender P_i with corrupted receiver P_j. The simulator Sim_1 as P_i just follows the protocol description except the following:

- Upon receiving a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ from $\mathcal{A}$ that it intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, \pi)$, the simulator Sim first verifies if π is valid. If valid, Sim decrypts Φ into $\hat{b}$, i.e., $\hat{b} = \mathsf{Dec}^L_{sk}(\Phi)$ with $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.
- Upon receiving $(\text{SEND}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator Sim then sends $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \hat{b})$ to the ideal functionality $\mathcal{F}_{\mathsf{MOT}}$ in the name of dummy receiver and obtains $(\text{RECEIVED}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, m_{\hat{b}})$. The simulator sets $m_{1-\hat{b}} = 0^\ell$ and computes $(\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$, $Z_\sigma \leftarrow m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$ for $\sigma \in \{0, 1\}$. It sends a second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ to $\mathcal{A}$ that it intends to send from P_i to P_j, where $M_2 = (\text{PK}_0, Z_0, \text{PK}_1, Z_1)$.

We claim that the hybrids H_0 and H_1 are statistically indistinguishable. The proof proceeds via a case analysis:

Case (I). The proof π is accepting, the ciphertext Φ correctly decrypts to some value $\hat{b} \in \{0, 1\}$, and $x_{1-\hat{b}}$ is a YES instance. By adaptive soundness of the proof system $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, this happens with negligible probability (over crs_{nizk} and the coin tosses of $\mathcal{Z}$).

Case (II). The proof π is not accepting. In this case, the sender in both H_0 and H_1 will abort, so H_0 and H_1 are identically distributed.

Case (III). The proof π is accepting and $x_{1-\hat{b}}$ is a NO instance. In this case, the only difference between the distributions H_0 and H_1 is that in H_0, $Z_{1-\hat{b}}$ is honestly generated based on the real plaintext $m_{1-\hat{b}}$, but in H_1 it is based on a dummy plaintext 0^ℓ; from the smoothness property of the hash proof system, the two distributions are statistically indistinguishable.

Combining the three cases, we may deduce that the two hybrids H_0 and H_1 are statistically indistinguishable.

Hybrid H_2: Similar to H_1. The simulator Sim_2 works as follows:

Simulation of CRS. As in H_1 except that now compute $(\mathsf{crs}_{nizk}, \tau_{nizk}) \leftarrow \mathcal{S}_1(1^\lambda)$.

Simulating honest receiver P_j with corrupted sender P_i. As in H_1.

Simulating honest sender P_i with corrupted receiver P_j. As in H_1.

The only difference between the above two hybrids is: in H_1, the crs_{nizk} is in the soundness mode while in H_2 it is in the ZK mode. Since the CRSs in two modes are computationally indistinguishable, we conclude that H_1 and H_2 are computationally indistinguishable.

Hybrid H_3: Similar to H_2. The simulator Sim_3 works as follows:

Simulation of CRS. As in H_2.

Simulating honest receiver P_j with corrupted sender P_i. The simulator Sim_3 is the same as in H_2 except the following: in $M_1 = (x_0, x_1, \Phi, \pi)$, now use the trapdoor τ_{nizk} to simulate the proof $\pi \leftarrow \mathcal{S}_2(\tau_{nizk}, (\textsc{pp}, pk, L, x_0, x_1, \Phi))$.

Simulating honest sender P_i with corrupted receiver P_j. As in H_2.

Now the only difference between the above two hybrids is: in H_2, π is generated honestly, while in H_3, π is simulated using the trapdoor τ_{nizk}. In the ZK mode, the two proofs are statistically close. Therefore, we conclude that H_2 and H_3 are statistically indistinguishable.

Hybrid H_4: Similar to H_3. The simulator Sim_4 works as follows:

Simulation of CRS. As in H_3.

Simulating honest receiver P_j with corrupted sender P_i. The simulator Sim_4 is the same as in H_3 except the following: in $M_1 = (x_0, x_1, \Phi, \pi)$, now compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$. Note that here we still use the trapdoor τ_{nizk} generate a simulated proof π.

Simulating honest sender P_i with corrupted receiver P_j. As in H_3.

Based on the CCA security of the encryption, we claim H_3 and H_4 are computationally indistinguishable. Towards contradiction, assume there exists a distinguisher $\mathcal{Z}$ who can distinguish H_3 from H_4. We next show an adversary $\mathcal{B}$ that breaks the CCA-security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

1. Upon receiving pk, $\mathcal{B}$ runs $\textsc{pp} \leftarrow \mathsf{HashPG}(1^\lambda)$ and $(\mathsf{crs}_{nizk}, \tau_{nizk}) \leftarrow \mathcal{S}_1(1^\lambda)$; then, it sets $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$.

2. $\mathcal{B}$ then internally simulates $\mathcal{Z}$ as follows:

   Simulating honest receiver P_j with corrupted sender P_i. $\mathcal{B}$ works the same as in H_3 except the following: in $M_1 = (x_0, x_1, \Phi, \pi)$, $\mathcal{B}$ queries LR-oracle **LR** with $(L, b, 0)$ where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$; in turn it gets back a ciphertext Φ which is $\mathsf{Enc}^L_{pk}(b)$ or $\mathsf{Enc}^L_{pk}(0)$ from the oracle.

   Simulating honest sender P_i with corrupted receiver P_j. As in H_3 except that whenever $\mathcal{Z}$ instructs $\mathcal{A}$ (in the name of receiver P_j) to send $M_1 = (x_0, x_1, \Phi, \pi)$ to sender P_i, $\mathcal{B}$ queries its decryption oracle to obtain the plaintext of Φ (instead of using the secret key for pk). Note that since the label $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ is unique, $\mathcal{B}$ can always use the decryption oracle for the given Φ. Therefore, both of the decryption behaviors are identical.

3. Finally, $\mathcal{B}$ outputs whatever $\mathcal{Z}$ outputs.

Let β be the hidden random bit embedded in the **LR** oracle. We note that when $\beta = 0$, $\mathcal{B}$ exactly simulates the hybrid H_3 to $\mathcal{Z}$; when $\beta = 1$, $\mathcal{B}$ simulates exactly the hybrid H_4 to $\mathcal{Z}$. Under the assumption that $\mathcal{Z}$ is able to distinguish the two hybrids in non-negligible probability, then the constructed machine $\mathcal{B}$ is successful CCA attacker against $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, which is a contradiction. Therefore, H_3 and H_4 are computationally indistinguishable.

Hybrid H_5: Similar to H_4. The simulator Sim_5 works as follows:

Simulation of CRS. As in H_4.

Simulating honest receiver P_j with corrupted sender P_i. As in H_4 except that now in $M_1 = (x_0, x_1, \Phi, \pi)$, x_0 and x_1 are two YES instance. That is, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\textsc{pp})$, and $(x_1, w_1) \leftarrow \mathsf{SampYes}(\textsc{pp})$.

Simulating honest sender P_i with corrupted receiver P_j. As in H_4.

The only difference is the in H_4, C_{1-b} is a NO instance while in H_5 it is a YES instance. Under the hard subset membership assumption, the two are computationally indistinguishable. Therefore, H_4 and H_5 are computationally indistinguishable.

Hybrid H_6: Similar to H_5 except that in this hybrid, *the input b of each honest receiver is not magically forwarded to the simulator any more.* The simulator Sim_6 works as follows:

Simulation of CRS. As in H_5.

Simulating honest receiver P_j with corrupted sender P_i. The simulator Sim_6 is the same as in H_5 except the following: Upon receiving a second round message $(\mathsf{sid}, \mathsf{ssid}, M_2)$ from the adversary $\mathcal{A}$ that intends to send from P_i to P_j, where $M_2 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$, the simulator use w_σ to compute $\hat{m}_\sigma = Z_\sigma \oplus \mathsf{pHash}(\textsc{pk}_\sigma, x_\sigma, w_\sigma)$ for all $\sigma \in \{0, 1\}$. Sim_6 sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \langle \hat{m}_0, \hat{m}_1 \rangle)$ to $\mathcal{F}_{\mathsf{MOT}}$.

Simulating honest sender P_i with corrupted receiver P_j. As in H_5.

Given the fact that the hash proof system is correct, H_5 and H_6 are identically distributed.

We conclude the proof by observing that $\mathsf{Sim} = \mathsf{Sim}_6$.

3.2 Proof of Theorem 2

The proof is very similar to that for the static case but with careful treatment of corruptions.

3.2.1 The Simulator

Initialization step: Same as the static case simulator in the proof of Theorem 1.

Simulating the communication with $\mathcal{Z}$: Same as the static case simulator in the proof of Theorem 1.

Case 1: Simulating honest receiver P_j with honest sender P_i:

1. Upon receiving $(\text{RECEIVE}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator delivers $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ that intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, \pi)$ which is generated as follows:

   Generate two YES instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\text{PP})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\text{PP})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, and then obtain a simulated proof π by using the simulator $\mathcal{S}_2$ for the NIZK, i.e., compute $\pi \leftarrow \mathcal{S}_2(\tau_{nizk}, (\text{PP}, pk, L, x_0, x_1, \Phi))$, where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.
2. Upon receiving $(\text{SEND}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, inform the adverary $\mathcal{A}$ that a message transfer over the secure channel took place.

Case 2: Simulating honest receiver P_j with corrupted sender P_i: Same as the static case simulator in the proof of Theorem 1.

Case 3: Simulating corrupted receiver P_j with honest sender P_i: Same as the static case simulator in the proof of Theorem 1.

Case 4: Simulating corrupted receiver P_j with corrupted sender P_i: Same as the static case simulator in the proof of Theorem 1.

Dealing with adaptive corruption of sender P_i:

Before the second message. This case is equivalent to static corruption. That is, the simulator receives the original input (m_0, m_1) and returns it to the adversary.

After the second message. The simulator receives the original input (m_0, m_1).

- If the receiver P_i is already corrupted (Case 3), at this moment, the second round message $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$ is fixed by the simulator itself. Now the simulator has to return the internal state of the sender to the adversary, and it returns $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)$.
- Otherwise (Case 1), the simulator has to additionally generate the second round message. The simulator Sim computes $(\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$ for $\sigma \in \{0, 1\}$. Now the simulator has to return the internal state of the sender to the adversary, and it returns $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)$.

Depending on whether P_j is honest, the simulator continues the simulation by following Case 2 (for honest P_j) or Case 4 (for corrupted P_j).

Dealing with adaptive corruption of receiver P_j:

Before the first message. This case is equivalent to static corruption. That is, the simulator receives the original input b and returns it to the adversary.

Between the first message and the second. The simulator receives the original input b. It returns (b, w_b, X, π) to the adversary $\mathcal{A}$ as P_j's internal state.

After the second message The simulator receives the original input b, and it obtains the receiver output m_b.

- If the sender P_i is already corrupted (Case 2), at this moment, the second round message $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$ is fixed either by the adversary $\mathcal{A}$ or by the simulator itself. Now the simulator has to return the internal state of the receiver to the adversary, and it returns (b, w, X, π) and $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$.
- Otherwise (Case 1), the simulator has to additionally generate the second round message. The simulator Sim sets $m_{1-b} = 0^\ell$, and computes $(\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$ for $\sigma \in \{0, 1\}$. Now the simulator has to return the internal state of the receiver to the adversary, and it returns (b, w, X, π) and $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$.

Note that in either case, it holds that $m_b = Z_b \oplus \mathsf{pHash}(\text{PK}_b, x_b, w)$.

Depending on whether P_i is honest, the simulator continues the simulation by following Case 3 (for honest P_i) or Case 4 (for corrupted P_i).

3.2.2 Indistinguishability

The change of simulation (from the static corruption model as to Theorem 1) lies only in the case where both parties are honest. In this case, the simulation is simpler since the protocol now uses a secure channel, and showing static security is easy. Therefore, the simulation shown above achieves static security. With that said, we only need to show that the internal state returned to the adversary upon adaptive corruption is indistinguishable from that in the real-world protocol.

Corrupting the sender. When the simulator returns $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_0)$, the simulation is indistinguishable due to the hard subset membership property for x_{1-b} and the smoothness of the underlying hash proof system.

Corrupting the receiver. When the simulator returns (b, w_b, X, π) to the adversary $\mathcal{A}$ (where $X = (x_0, x_1, \Phi)$), the simulated internal state is indistinguishable due to the hard subset membership property for x_{1-b} and CCA security for Φ and zero-knowledge property for π. Even when the simulator additionally returns $(\text{PK}_0, Z_0, \text{PK}_1, Z_0)$ to the adversary $\mathcal{A}$, the simulation is indistinguishable (as with the sender corruption case).

4 Instantiations of Protocol 1*

4.1 Instantiation from the DLIN Assumption

We show a CCA-secure labeled public-key encryption scheme, a smooth hash proof system, and a dual-mode NIZK proof system under the DLIN assumption. We then obtain a two-round OT protocol by combining these building blocks.

4.1.1 Decision Linear Assumption (DLIN)

Let $\mathcal{G}_{\text{dlin}}$ a randomized algorithm that takes a security parameter λ as input and outputs $\mathsf{desc} = (p, \mathbb{G}, \mathbb{G}_T, \hat{\mathsf{e}}, g)$ such that

1. p is a prime.
2. $\mathbb{G}$ and $\mathbb{G}_T$ are descriptions of groups of order p.
3. $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map, i.e., $\forall u, v \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p : \hat{\mathsf{e}}(u^a, v^b) = \hat{\mathsf{e}}(u, v)^{ab}$.
4. g is a random generator of $\mathbb{G}$ and $\hat{\mathsf{e}}(g, g)$ generates $\mathbb{G}_T$.
5. Deciding group membership, group operations and the bilinear map are all efficiently computable.

The decisional linear assumption was first introduced by Boneh et al. [BBS04]. The decisional linear problem is to distinguish a linear tuple from a random tuple.

Definition 9 (Decisional Linear Assumption) *We say the decisional linear assumption holds for the bilinear group generator $\mathcal{G}_{\text{dlin}}$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ it holds that $|p_1 - p_2| \leq \mathsf{negl}(\lambda)$, where*

$$p_1 = \Pr[(p, \mathbb{G}, \mathbb{G}_T, \hat{\mathsf{e}}, g) \leftarrow \mathcal{G}_{\text{dlin}}(1^\lambda); x, y \leftarrow \mathbb{Z}_p^*; r, s \leftarrow \mathbb{Z}_p : \mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, \hat{\mathsf{e}}, g, g^x, g^y, g^{xr}, g^{ys}, g^{r+s}) = 1]$$
$$p_2 = \Pr[(p, \mathbb{G}, \mathbb{G}_T, \hat{\mathsf{e}}, g) \leftarrow \mathcal{G}_{\text{dlin}}(1^\lambda); x, y \leftarrow \mathbb{Z}_p^*; r, s, d \leftarrow \mathbb{Z}_p : \mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, \hat{\mathsf{e}}, g, g^x, g^y, g^{xr}, g^{ys}, g^d) = 1].$$

4.1.2 CCA-Secure Labeled Public-Key Encryption Scheme

Based on the DLIN assumption, Shacham constructed a CCA-secure encryption called Linear Cramer-Shoup Encryption [Sha07]. As is done in [FLM11], slightly changing the scheme to support labels leads to a CCA-secure labeled public-key encryption scheme: hash functions with stronger security of collision-resistance (rather than UOWHF) are used, and in the encryption algorithm, a given label is also applied to the hash function. Let $\mathsf{desc} \leftarrow \mathcal{G}_{\text{dlin}}(1^\lambda)$.

Key Generation $(pk, sk) \leftarrow \mathsf{Gen}(\mathsf{desc})$: Choose random generators $g_1, g_2 \leftarrow \mathbb{G}$ and exponents $\beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3, \delta_1, \delta_2, \delta_3 \leftarrow \mathbb{Z}_p$ and compute the following:

$$c_1 = g_1^{\beta_1} g^{\beta_3}, \quad d_1 = g_1^{\gamma_1} g^{\gamma_3}, \quad h_1 = g_1^{\delta_1} g^{\delta_3}, \quad c_2 = g_2^{\beta_2} g^{\beta_3}, \quad d_2 = g_2^{\gamma_2} g^{\gamma_3}, \quad h_2 = g_2^{\delta_2} g^{\delta_3}.$$

Choose a hash function $H \leftarrow \mathcal{HF}$ where $\mathcal{HF}$ is a family of collision-resistant hash functions. Now set $pk = (g_1, g_2, g, c_1, c_2, d_1, d_2, h_1, h_2, H)$ and $sk = (\beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3, \delta_1, \delta_2, \delta_3)$.

Encryption $C \leftarrow \mathsf{Enc}_{pk}^L(m; r)$: Given the message $m \in \mathbb{G}$ under label L, choose $r_1, r_2 \leftarrow \mathbb{Z}_p$ and compute $u_1 = g_1^{r_1}, u_2 = g_2^{r_2}, u_3 = g^{r_1+r_2}, e = m \cdot h_1^{r_1} h_2^{r_2}$. Then compute $\alpha = H(u_1, u_2, u_3, e, L) \in \mathbb{Z}_p$ and $v = (c_1 d_1^\alpha)^{r_1} \cdot (c_2 d_2^\alpha)^{r_2}$. Here $r = (r_1, r_2)$ and $C = (u_1, u_2, u_3, e, v)$.

Decryption $\mathsf{Dec}_{sk}^L(C)$: Parse $C = (u_1, u_2, u_3, e, v)$ and $sk = (\beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3, \delta_1, \delta_2, \delta_3)$; compute $\alpha \leftarrow H(u_1, u_2, u_3, e, L)$ and test if $u_1^{\beta_1+\alpha\gamma_1} \cdot u_2^{\beta_2+\alpha\gamma_2} \cdot u_3^{\beta_3+\alpha\gamma_3} \overset{?}{=} v$. If it does not, output `reject`. Otherwise, output $m = e/(u_1^{\delta_1} u_2^{\delta_2} u_3^{\delta_3})$.

4.1.3 Smooth Projective Hashing

Shacham showed a smooth projective hash proof system based on the DLIN assumption [Sha07].

Parameter generation. Choose $g_1, g_2 \leftarrow \mathbb{G}$. The parameter is $\textsc{pp} = (g_1, g_2, \mathbb{G})$.

Instance sampling. To sample a YES instance, choose $t_1, t_2 \leftarrow \mathbb{Z}_p$ and set $t_3 = t_1 + t_2$, and compute $z_1 = g_1^{t_1}$, $z_2 = g_2^{t_2}$ and $z_3 = g^{t_3}$, and then return $x = (z_1, z_2, z_3)$. To sample a NO instance, choose $t_1, t_2 \leftarrow \mathbb{Z}_p$ and set $t_3 = t_1 + t_2 + 1$, and then $z_1 = g_1^{t_1}$, $z_2 = g_2^{t_2}$ and $z_3 = g^{t_3}$, and then return $x = (z_1, z_2, z_3)$.

Hash key generation. Choose $\theta_1, \theta_2, \theta_3 \leftarrow \mathbb{Z}_p$ and compute $f_1 = g_1^{\theta_1} g^{\theta_3}$, $f_2 = g_2^{\theta_2} g^{\theta_3}$. Return $\textsc{hk} = (\theta_1, \theta_2, \theta_3)$ and $\textsc{pk} = (f_1, f_2)$.

Primary hashing. Given $\textsc{hk} = (\theta_1, \theta_2, \theta_3)$ and $x = (z_1, z_2, z_3)$, return $y = z_1^{\theta_1} z_2^{\theta_2} z_3^{\theta_3}$.

Projective hashing. Given a projective hash key $\textsc{pk} = (f_1, f_2)$, an instance $x = (z_1, z_2, z_3)$, and its witness $w = (t_1, t_2)$ such that $z_1 = g_1^{t_1}$, $z_2 = g_2^{t_2}$ and $z_3 = g^{t_1+t_2}$, return $y = f_1^{t_1} f_2^{t_2}$.

4.1.4 Non-Interactive Dual-Mode Zero-Knowledge Proof System

We simply note that the Groth-Sahai (GS) proof system [GS08] satisfies the required definition. Here, we briefly outline only the part of the system that we need.

CRS. The common reference string consists of three vectors $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \in \mathbb{G}^3$, where $\mathbf{g}_1 = (g_1, 1, g)$, $\mathbf{g}_2 = (1, g_2, g)$ for some $g_1, g_2 \in \mathbb{G}$. Let $\mathbf{h}_3 = \mathbf{g}_3 \cdot (1, 1, g)$. In the soundness mode, $\mathbf{g}_3$ is set as $\mathbf{g}_3 = \mathbf{g}_1^{\zeta_1} \cdot \mathbf{g}_2^{\zeta_2}$ with $\zeta_1, \zeta_2 \leftarrow \mathbb{Z}_p^*$. In the ZK mode, $\mathbf{g}_3$ is set as $\mathbf{g}_3 = \mathbf{g}_1^{\zeta_1} \cdot \mathbf{g}_2^{\zeta_2} \cdot (1, 1, g^{-1})$ with $\zeta_1, \zeta_2 \leftarrow \mathbb{Z}_p^*$. Under the DLIN assumption, the two kinds of CRS are indistinguishable.

Commit to an exponent. Committing to an exponent $x \in \mathbb{Z}_p$ needs three group elements. In particular, the commitment is computed as $\mathbf{C} = \mathbf{g}_1^r \cdot \mathbf{g}_2^s \cdot \mathbf{h}_3^x$ with $r, s \leftarrow \mathbb{Z}_p^*$. In the soundness mode, the commitment becomes $(g_1^{r+\zeta_1 x}, g_2^{s+\zeta_2 x}, g^{r+s+x(\zeta_1+\zeta_2)} \cdot g^x)$, and it is perfectly binding, since it is a linear encryption of g^x, which can be decrypted using $a_1 = \log_g g_1, a_2 = \log_g g_2$ [BBS04]. In the ZK mode, it becomes $\mathbf{C} = (g_1^{r+\zeta_1 x}, g_2^{s+\zeta_2 x}, g^{r+s+x(\zeta_1+\zeta_2)})$, and it is a perfectly hiding, since it is a random linear tuple for any $x \in \mathbb{Z}_p$.

Proving equations. Having the variables $\{x_i\}_{i=1}^n$ with $x_i \in \mathbb{Z}_p$ committed using the above commitment scheme, various equations can be proved. In order to prove a linear equation such as

$$\prod_{i=1}^{n} a_i^{x_i} = b,$$

with known constants $a_i, b \in \mathbb{G}$, one needs two group elements in $\mathbb{G}$. On the other hand, six group elements are needed in order to prove a quadratic equation in $\mathbb{Z}_p$ that looks as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} b_i x_i = c,$$

with known constants $a_{ij}, b_i, c \in \mathbb{Z}_p$.

4.1.5 Oblivious Transfer

By plugging in these components into the generic framework for two-round OT, we obtain an OT protocol based on the DLIN assumption. Thus, it is only left to show how to concretely generate the proof π in the receiver side message given crs_{ot}.

Protocol detail. Ignoring the description desc for the bilinear group, the common reference string is $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$, where $\textsc{pp} = (g_1, g_2, g)$, $pk = (g_1, g_2, g, c_1, c_2, d_1, d_2, h_1, h_2, H)$, $\mathsf{crs}_{nizk} = (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$. Here, $\mathbf{g}_1 = (g_1, 1, g)$, $\mathbf{g}_2 = (1, g_2, g)$ and $\mathbf{g}_3 = (g_{31}, g_{32}, g_{33})$ for some $g_{31}, g_{32}, g_{33} \in \mathbb{G}$. Therefore, the CRS can be represented with 12 group elements of $\mathbb{G}$ and one hash function index, along with the description of the bilinear group.

Let $x_0 = (z_{01}, z_{02}, z_{03})$, $x_1 = (z_{11}, z_{12}, z_{13})$, and $\Phi = (u_1, u_2, u_3, e, v)$ with $\alpha = H(u_1, u_3, u_3, e, L)$, with $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. The proof π is a Groth-Sahai proof for the following variables and equations.

Variables: r_1, r_2, b, t_1, t_2

$$u_1 = g_1^{r_1}; \quad u_2 = g_2^{r_2}; \quad u_3 = g^{r_1+r_2}; \quad e = g^b h_1^{r_1} h_2^{r_2}; \quad v = (c_1 d_1^{\alpha})^{r_1} \cdot (c_2 d_2^{\alpha})^{r_2};$$
$$b(b-1) = 0; \quad z_{01}^{b} \cdot z_{11}^{1-b} = g_1^{t_1}; \quad z_{02}^{b} \cdot z_{12}^{1-b} = g_2^{t_2}; \quad z_{03}^{b} \cdot z_{13}^{1-b} = g^{t_1+t_2+1}$$

The first five equations requires that Φ is a valid encryption of g^b. From the sixth equation, it should hold that $b \in \{0, 1\}$. When $b = 0$, the last three equations require that the $x_1 = (z_{11}, z_{12}, z_{13})$ is a non-linear tuple (i.e., $\textsc{no}$-instance) with $z_{11} = g^{t_1}, z_{12} = g^{t_2}, z_{13} = g^{t_1+t_2+1}$. On the other hand, when $b = 1$, x_0 should be a $\textsc{no}$-instance.

Communication complexity. The receiver message (x_0, x_1, Φ, π) needs $3 + 3 + 5 + 37 = 48$ group elements. (In particular, for the proof π five variables need to be committed (i.e., $15 = 5 \cdot 3$), and there are eight linear equations (i.e., $16 = 8 \cdot 2$) and one quadratic equation (i.e, $6 = 1 \cdot 6$).) The sender message $(\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$ needs $2 + 1 + 2 + 1 = 6$ group elements. Therefore, the total communication complexity amounts to 54 group elements.

Adaptive security: realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. The first two rounds can be overlapped with the first round of the OT protocol, and thus the final OT protocol runs in three rounds. We can use linear encryption [BBS04], and the communication overhead amounts to 5 group elements (the public key consists of two elements excluding the generator in the CRS, and the ciphertext consists of three elements).

4.2 Instantiation from the SXDH Assumption

We show a CCA-secure labeled public-key encryption scheme, a smooth hash proof system, and a dual-mode NIZK proof system under the SXDH assumption. We then obtain a two-round OT protocol by combining these building blocks.

4.2.1 Symmetric External Diffie-Hellman Assumption (SXDH)

Let $\mathcal{G}_{\rm sxdh}$ be a randomized algorithm that takes a security parameter λ as input and outputs $\mathsf{desc} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{\mathsf{e}}, g, g')$ such that

1. p is a prime.
2. $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are descriptions of groups of order p.
3. $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, i.e., $\forall u, v \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p : \hat{\mathsf{e}}(u^a, v^b) = \hat{\mathsf{e}}(u, v)^{ab}$.
4. g and g' are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $\hat{\mathsf{e}}(g, g')$ generates $\mathbb{G}_T$.
5. Deciding group membership, group operations and the bilinear map are all efficiently computable.

Under the SXDH assumption [BBS04], the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 10 (Symmetric External Diffie-Hellman Assumption) *We say the symmetric external Diffie-Hellman assumption holds for the bilinear group generator* $\mathcal{G}_{\rm sxdh}$ *if for all non-uniform polynomial time adversaries* $\mathcal{A}$ *it holds that* $|p_{10} - p_{11}| = \mathsf{negl}(\lambda)$ *and* $|p_{20} - p_{21}| = \mathsf{negl}(\lambda)$*, where*

$$\begin{aligned}
p_{10} &= \Pr[\mathsf{desc} \leftarrow \mathcal{G}_{\rm sxdh}(1^\lambda); a, b \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{desc}, g^a, g^b, g^{ab}) = 1] \\
p_{11} &= \Pr[\mathsf{desc} \leftarrow \mathcal{G}_{\rm sxdh}(1^\lambda); a, b, c \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{desc}, g^a, g^b, g^c) = 1] \\
p_{20} &= \Pr[\mathsf{desc} \leftarrow \mathcal{G}_{\rm sxdh}(1^\lambda); a, b \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{desc}, (g')^a, (g')^b, (g')^{ab}) = 1] \\
p_{21} &= \Pr[\mathsf{desc} \leftarrow \mathcal{G}_{\rm sxdh}(1^\lambda); a, b, c \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{desc}, (g')^a, (g')^b, (g')^c) = 1].
\end{aligned}$$

Here, $\mathsf{desc} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{\mathsf{e}}, g, g')$.

4.2.2 CCA-Secure Labeled Public-Key Encryption Scheme

Since the DDH assumption holds in $\mathbb{G}_1$, we can use Cramer-Shoup encryption scheme [CS98]. As in the case for the DLIN assumption, we slightly change the scheme to support labels, that is, we use collision resistant hash functions instead of UOWHF and apply labels to hash functions when performing encryptions and decryptions. Let $\mathsf{desc} \leftarrow \mathcal{G}_{\rm sxdh}(1^\lambda)$.

Key Generation $(pk, sk) \leftarrow \mathsf{Gen}(\mathsf{desc})$: Choose random generators $g_1 \leftarrow \mathbb{G}_1$ and exponents $\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 \leftarrow \mathbb{Z}_p$ and compute the following:

$$c = g_1^{\beta_1} g^{\beta_2} \qquad d = g_1^{\gamma_1} g^{\gamma_2} \qquad h = g_1^{\delta_1} g^{\delta_2}$$

Choose a hash function $H \leftarrow \mathcal{HF}$ where $\mathcal{HF}$ is a family of collision-resistant hash functions. Now set $pk = (g_1, g, c, d, h, H)$ and $sk = (\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2)$.

Encryption $C \leftarrow \mathsf{Enc}_{pk}^L(m; r)$: Given the message $m \in \mathbb{G}$ under label L, choose $r \leftarrow \mathbb{Z}_p$ and compute the following:

$$u_1 = g_1^r, \quad u_2 = g^r, \quad e = m \cdot h^r$$

Then compute $\alpha = H(u_1, u_2, e, L) \in \mathbb{Z}_p$ and $v = (cd^\alpha)^r$. The ciphertext is $C = (u_1, u_2, e, v)$.

Decryption $\mathsf{Dec}^L_{sk}(C)$: Parse $C = (u_1, u_2, e, v)$ and $sk = (\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2)$; compute $\alpha \leftarrow H(u_1, u_2, e, L)$ and test if the following holds

$$u_1^{\beta_1+\alpha\gamma_1} \cdot u_2^{\beta_2+\alpha\gamma_2} \overset{?}{=} v$$

If it does not, output `reject`. Otherwise, output

$$m = \frac{e}{u_1^{\delta_1} u_2^{\delta_2}}$$

4.2.3 Smooth Projective Hashing

We recall the construction of smooth projective hashing based on the DDH assumption [CS98, CS02].

Parameter Generation. Let g_1 be randomly chosen elements in $\mathbb{G}_1$. Then the parameter is $\textsc{pp} = (g_1, g, \mathbb{G}_1)$.

Instance Sampling. To sample a YES instance, randomly choose $t \leftarrow \mathbb{Z}_p$, and compute $z_1 = g_1^t$, $z_2 = g^t$, and then return $x = (z_1, z_2)$.

To sample a NO instance, randomly choose $t \leftarrow \mathbb{Z}_p$, and then $z_1 = g_1^t$, $z_2 = g^{t+1}$, and then return $x = (z_1, z_2)$.

Hash Key Generation. Randomly choose $\theta_1, \theta_2 \leftarrow \mathbb{Z}_p$ and compute $f = g_1^{\theta_1} g^{\theta_2}$. Return $\textsc{hk} = (\theta_1, \theta_2)$, and $\textsc{pk} = f$.

Primary Hashing Given a primary hash key $\textsc{hk} = (\theta_1, \theta_2)$ and an instance $x = (z_1, z_2)$, return $y = z_1^{\theta_1} z_2^{\theta_2}$.

Projective Hashing Given a projective hash key $\textsc{pk} = f$, an instance $x = (z_1, z_2)$, and its witness $w = t$ such that $z_1 = g_1^t$, $z_2 = g^t$, return $y = f^t$.

4.2.4 Dual-Mode Zero-Knowledge Proof System

We simply note that the Groth-Sahai (GS) proof system [GS08] is indeed the case. Here, we briefly outline only the part of the system that we need.

CRS. The common reference string consists of two vectors $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{G}_1^2$, where $\mathbf{g}_1 = (g, g^\alpha)$. Let $\mathbf{h}_2 = \mathbf{g}_2 \cdot (1, g)$. In the soundness mode, $\mathbf{g}_2$ is set as $\mathbf{g}_2 = \mathbf{g}_1^\zeta$ with $\zeta \leftarrow \mathbb{Z}_p^*$. In the ZK mode, $\mathbf{g}_2$ is set as $\mathbf{g}_2 = \mathbf{g}_1^\zeta \cdot (1, g^{-1})$ with $\zeta \leftarrow \mathbb{Z}_p^*$. Under the SXDH assumption, the two kinds of CRS are indistinguishable.

For group $\mathbb{G}_2$, we can define $\mathbf{g}'_1$ and $\mathbf{g}'_2$ similarly.

Commit to an exponent. Committing to an exponent $x \in \mathbb{Z}_p$ needs two group elements. In particular, the commitment is computed as $\mathbf{C} = \mathbf{g}_1^r \cdot \mathbf{h}_2^x$ with $r \leftarrow \mathbb{Z}_p^*$. In the soundness mode, the commitment becomes $\mathbf{C} = (g^{r+\zeta x}, g^{\alpha(r+\zeta x)} \cdot g^x)$, and it is perfectly binding, since it is an ElGamal encryption of g^x, which can be decrypted using α. In the ZK mode, it becomes $\mathbf{C} = (g^{r+\zeta x}, g^{\alpha(r+\zeta x)})$, and it is a perfectly hiding, since it is a random Diffie-Hellman tuple for any $x \in \mathbb{Z}_p$.

We can also commit to an exponent using $\mathbf{g}'_1$ and $\mathbf{g}'_2$ in $\mathbb{G}_2$.

Proving equations. Having the variables $\{x_i\}_{i=1}^{n}$ with $x_i \in \mathbb{Z}_p$ committed using $\mathbb{G}_1$ and the variables $\{y_j\}_{j=1}^{n'}$ with $y_j \in \mathbb{Z}_p$ committed using $\mathbb{G}_2$ using the above commitment scheme, various equations can be proved. In order to prove a linear equation that looks as follows

$$\prod_{i=1}^{n}(a'_i)^{x_i} = b',$$

with known constants $a'_i, b' \in \mathbb{G}_2$, one needs two group elements in $\mathbb{G}_1$. Similarly, for the linear equation

$$\prod_{j=1}^{n'} a_j^{y_i} = b,$$

with known constants $a_j, b \in \mathbb{G}_1$, one needs two group elements in $\mathbb{G}_2$. On the other hand, two group elements in $\mathbb{G}_1$ and two group elements in $\mathbb{G}_2$ are needed in order to prove a quadratic equation in $\mathbb{Z}_p$ that looks as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{n'} a_{ij}x_iy_j + \sum_{i=1}^{n} b_ix_i + \sum_{j=1}^{n'} c_jy_j = d,$$

with known constants $a_{ij}, b_i, c_j, d \in \mathbb{Z}_p$.

4.2.5 Oblivious Transfer

Now we are ready to describe the protocol based on the SXDH assumption.

CRS. Ignoring the description desc for the bilinear group, the common reference string $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$ is as follows:

$$\begin{aligned}\textsc{pp} &= (g_1, g)\\ pk &= (g_1, g, c, d, h, H)\\ \mathsf{crs}_{nizk} &= (\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}'_1, \mathbf{g}'_2)\end{aligned}$$

Here, $\mathbf{g}_1 = (g, f_1)$, $\mathbf{g}_2 = (f_2, f_3)$ for some $f_1, f_2, f_3 \in \mathbb{G}_1$, and $\mathbf{g}'_1 = (g', f'_1)$, $\mathbf{g}'_2 = (f'_2, f'_3)$ for some $g', f'_1, f'_2, f'_3 \in \mathbb{G}_2$. Therefore, the CRS can be represented with 8 group elements of $\mathbb{G}_1$, 4 group elements of $\mathbb{G}_2$ and one hash function index, along with the description of the bilinear group.

Protocol detail. It suffices to describe how to generate the proof for the receiver's message M_1 given $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{nizk})$. Let $x_0 = (z_{01}, z_{02})$, $x_1 = (z_{11}, z_{12})$, and $\Phi = (u_1, u_2, e, v)$ with $\alpha = H(u_1, u_2, e, (\mathsf{sid}, \mathsf{ssid}, P_i, P_j))$. The proof π is a Groth-Sahai proof for the following variables and equations.

Variables committed using $\mathbb{G}_2$: r, b, t

Variables committed using $\mathbb{G}_1$: b'

$$\begin{array}{llll} u_1 = g_1^r; & u_2 = g^r; & e = g^bh^r; & v = (cd^\alpha)^r;\\ b = b'; & b(b'-1) = 0; & z_{01}^b \cdot z_{11}^{1-b} = g_1^t; & z_{02}^b \cdot z_{12}^{1-b} = g_2^{t+1}; \end{array}$$

The first four equations requires that Φ is a valid encryption of g^b. From the fifth equation and the sixth, it should hold that $b \in \{0,1\}$. When $b = 0$, the last two equations require that the $x_1 = (z_{11}, z_{12})$ is a non-Diffie-Hellman tuple (i.e., NO-instance) with $z_{11} = g_1^t, z_{12} = g_2^{t+1}$. On the other hand, when $b = 1$, x_0 should be a NO-instance.

Communication complexity. The receiver message (x_0, x_1, Φ) needs $2 + 2 + 4 = 8$ group elements in $\mathbb{G}_1$. The proof π takes 6 elements in $\mathbb{G}_1$ and 22 elements in $\mathbb{G}_2$. In particular, three variables need to be committed using $\mathbb{G}_2$ (i.e., $6 = 3 \cdot 2$ in $\mathbb{G}_2$), and one variable needs to be committed using $\mathbb{G}_1$ (i.e., $2 = 1 \cdot 2$ in $\mathbb{G}_1$). There are six linear equations (i.e., $12 = 6 \cdot 2$ in $\mathbb{G}_2$) and two quadratic equations (i.e, $4 = 2 \cdot 2$ in $\mathbb{G}_1$ and $4 = 2 \cdot 2$ in $\mathbb{G}_2$). The sender message $(\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$ needs $(1,1,1,1) = 4$ group elements in $\mathbb{G}_1$. Therefore, the total communication complexity amounts to 18 group elements in $\mathbb{G}_1$ and 22 group elements in $\mathbb{G}_2$.

Adaptive security: realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. The first two rounds can be overlapped with the first round of the OT protocol, and thus the final OT protocol runs in three rounds. We can use ElGamal encryption, and the communication overhead amounts to 3 group elements (the public key consists of one element excluding the generator in the CRS, and the ciphertext consists of two elements).

5 A Generic Framework for Four-Round OT

In this section, we describe a generic framework for constructing four-round OT protocols. We begin by looking at the case of static security, and then show how the ideas can be extended to achieve security against adaptive adversaries.

5.1 Static Security (Protocol 2)

The main idea is to adapt our previous two-round framework by replacing the dual-mode NIZK proof with an interactive equivalent. In particular, the general structure of the protocol is as follows: the protocol starts by having the receiver send two instances (x_0, x_1) for hash proof system where x_{1-b} being a NO-instance; also, in protection against a malicious behavior, $\mathsf{Enc}_{pk}(b)$ and a Sigma protocol (augmented with an equivocal commitment) are attached. Then, the sender generates primary and projective hash keys $(\textsc{hk}_\sigma, \textsc{pk}_\sigma)$ for each instance x_σ and sends $(\textsc{pk}_\sigma, \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma) \oplus m_\sigma)$ to the receiver. The security can be shown similarly to the two-round OT case.

Here, instead of replicating all the details, we only describe how to combine a Sigma protocol with an equivocal commitment scheme in order to replace the NIZK part. The idea is having the prover commit to the first round message of the Sigma protocol, and reveal it in the third round. Refer to Figure 5 for the overall pictorial description of the protocol.

CRS. Compute $\textsc{pp} \leftarrow \mathsf{HashPG}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and $\mathsf{crs}_{com} \leftarrow \mathcal{K}_{com}(1^\lambda)$. The common reference string is $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$.

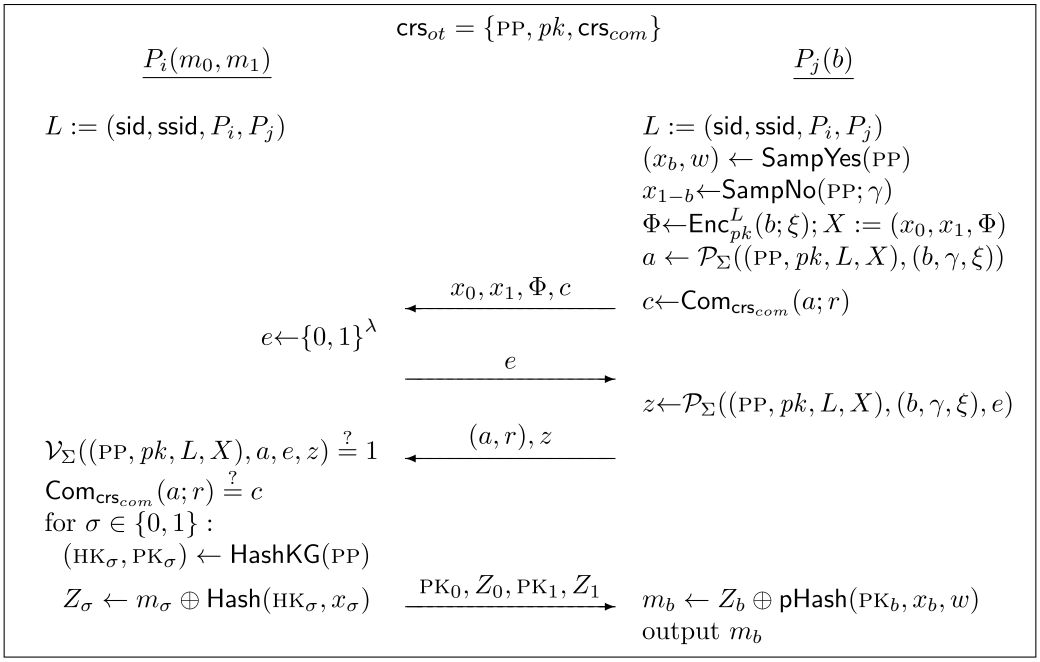

$\mathsf{crs}_{ot} = \{\textsc{pp}, pk, \mathsf{crs}_{com}\}$
$\underline{P_i(m_0, m_1)}$
$\underline{P_j(b)}$
$L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
$L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
$(x_b, w) \leftarrow \mathsf{SampYes}(\textsc{pp})$
$x_{1-b} \leftarrow \mathsf{SampNo}(\textsc{pp}; \gamma)$
$\Phi \leftarrow \mathsf{Enc}^L_{pk}(b; \xi); X := (x_0, x_1, \Phi)$
$a \leftarrow \mathcal{P}_\Sigma((\textsc{pp}, pk, L, X), (b, \gamma, \xi))$
x_0, x_1, Φ, c
$c \leftarrow \mathsf{Com}_{\mathsf{crs}_{com}}(a; r)$
$e \leftarrow \{0, 1\}^\lambda$
e
$z \leftarrow \mathcal{P}_\Sigma((\textsc{pp}, pk, L, X), (b, \gamma, \xi), e)$
$\mathcal{V}_\Sigma((\textsc{pp}, pk, L, X), a, e, z) \overset{?}{=} 1$
$(a, r), z$
$\mathsf{Com}_{\mathsf{crs}_{com}}(a; r) \overset{?}{=} c$
for $\sigma \in \{0, 1\}$:
$(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$
$Z_\sigma \leftarrow m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$
$\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1$
$m_b \leftarrow Z_b \oplus \mathsf{pHash}(\textsc{pk}_b, x_b, w)$
output m_b

Figure 5: A statically secure OT protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model (**Protocol 2**).

Replacing NIZK. Recall in the two-round OT case, the receiver generates a NIZK π to prove that (x_0, x_1, Φ) is valid message, i.e., Φ is an encryption of $b \in \{0, 1\}$ for some b and x_{1-b} is NO-instance. In this protocol, the receiver proves it by running a Sigma protocol $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$, along with an equivocal commitment scheme $(\mathcal{K}_{com}, \mathsf{Com})$, with respect to the following language:

$$\mathcal{L}^* = \{(\textsc{pp}, pk, L, x_0, x_1, \Phi) : \exists (b, \gamma, \xi) \text{ s.t. } x_{1-b} = \mathsf{SampNo}(\textsc{pp}; \gamma), \Phi = \mathsf{Enc}^L_{pk}(b; \xi)\},$$

where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

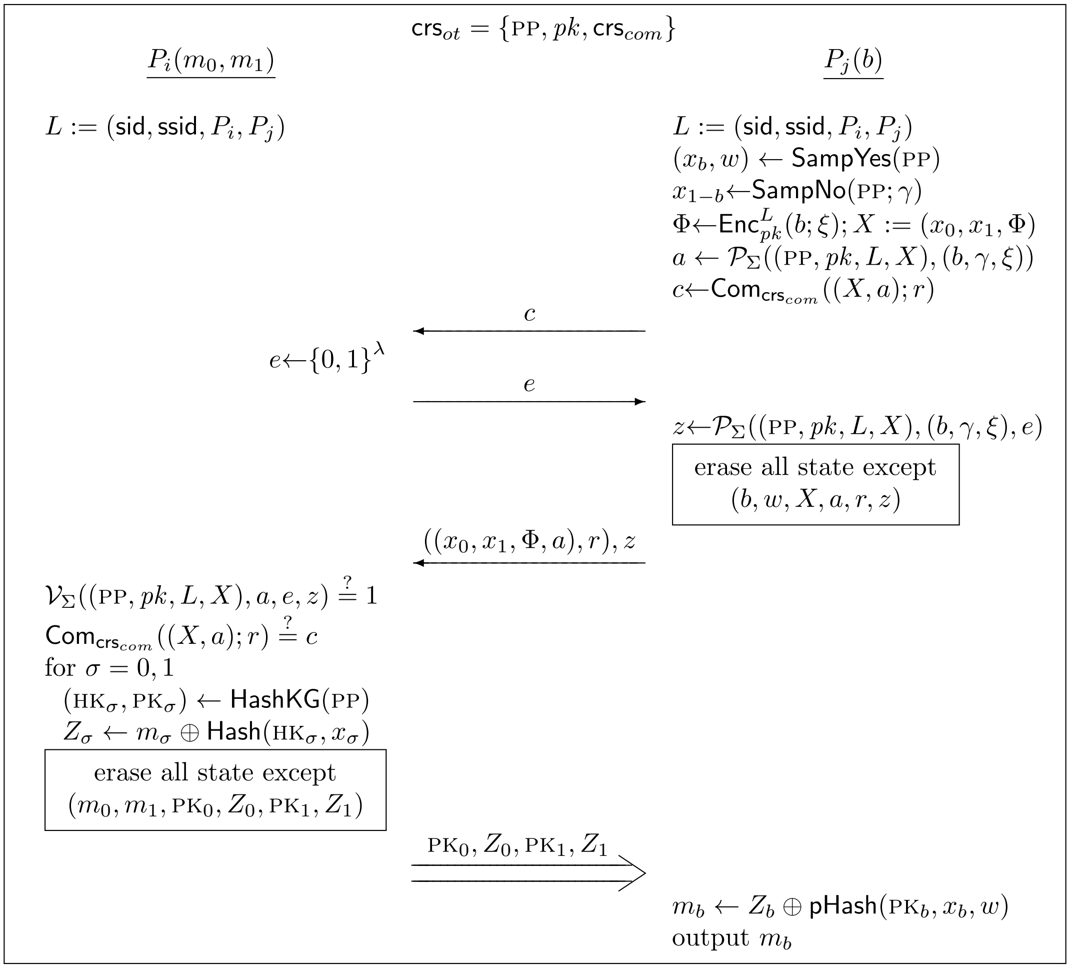
1. The receiver runs $a \leftarrow \mathcal{P}_\Sigma((\textsc{pp}, pk, L, x_0, x_1, \Phi), (b, \gamma, \xi))$, and $c = \mathsf{Com}_{\mathsf{crs}_{com}}(a; r)$ with r chosen uniformly at random. It sends (x_0, x_1, Φ, c).
2. The sender sends the challenge message $e \leftarrow \{0, 1\}^\lambda$ of the Sigma protocol.
3. Upon receiving the challenge e, the receiver generates an answer by running

$$z = \mathcal{P}_\Sigma((\textsc{pp}, pk, L, x_0, x_1, \Phi), (b, \gamma, \xi), e).$$

   It sends the sender the answer z along with the opening of the commitment, i.e., $((a, r), z)$.
4. The sender verifies (a, e, z) is an accepting transcript and (a, r) is a valid opening of c:

$$\mathcal{V}_\Sigma((\textsc{pp}, pk, L, x_0, x_1, \Phi), a, e, z) \overset{?}{=} 1, \quad \mathsf{Com}_{\mathsf{crs}_{com}}(a; r) \overset{?}{=} c.$$

The security of the protocol can be proved similarly to the two-round case.



$\mathsf{crs}_{ot} = \{\textsc{pp}, pk, \mathsf{crs}_{com}\}$

$\underline{P_i(m_0, m_1)}$

$\underline{P_j(b)}$

P_i: $L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$

P_j:
$L := (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
$(x_b, w) \leftarrow \mathsf{SampYes}(\textsc{pp})$
$x_{1-b} \leftarrow \mathsf{SampNo}(\textsc{pp}; \gamma)$
$\Phi \leftarrow \mathsf{Enc}^L_{pk}(b; \xi); X := (x_0, x_1, \Phi)$
$a \leftarrow \mathcal{P}_\Sigma((\textsc{pp}, pk, L, X), (b, \gamma, \xi))$
$c \leftarrow \mathsf{Com}_{\mathsf{crs}_{com}}((X, a); r)$

$\xleftarrow{\quad c \quad}$

P_i: $e \leftarrow \{0,1\}^\lambda$

$\xrightarrow{\quad e \quad}$

P_j: $z \leftarrow \mathcal{P}_\Sigma((\textsc{pp}, pk, L, X), (b, \gamma, \xi), e)$

$\boxed{\text{erase all state except } (b, w, X, a, r, z)}$

$\xleftarrow{\quad ((x_0, x_1, \Phi, a), r), z \quad}$

P_i:
$\mathcal{V}_\Sigma((\textsc{pp}, pk, L, X), a, e, z) \overset{?}{=} 1$
$\mathsf{Com}_{\mathsf{crs}_{com}}((X, a); r) \overset{?}{=} c$
for $\sigma = 0, 1$
  $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$
  $Z_\sigma \leftarrow m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$

$\boxed{\text{erase all state except } (m_0, m_1, \textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)}$

$\xRightarrow{\quad \textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1 \quad}$

P_j:
$m_b \leftarrow Z_b \oplus \mathsf{pHash}(\textsc{pk}_b, x_b, w)$
output m_b

Figure 6: An adaptively secure OT protocol in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model (**Protocol 2***). The last round message is sent over a secure channel.

Theorem 3 *Say* (Gen, Enc, Dec) *is a CCA-secure labeled public-key encryption scheme,* (HashPG, SampYes, SampNo, HashKG, Hash, pHash) *is a smooth projective hash proof system with hard subset membership property,* $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ *is a* Σ*-protocol, and* $(\mathcal{K}_{com}, \mathsf{Com})$ *is an equivocal commitment scheme. Then the protocol of Figure 5 securely realizes* $\mathcal{F}_{\mathsf{MOT}}$ *in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model, for static corruptions.*

The proof appears in Section 5.3.

5.2 Adaptive Security (Protocol 2*)

As with the 2-round framework, the protocol first needs to be changed so that the last round message is sent over a secure channel. This modification (along with erasing the state appropriately), however, is not sufficient to deal with adaptive corruption in the four-round case. For the NIZK, the receiver can generate π and then erase the unnecessary internal state before sending out (x_0, x_1, Φ, π). However, if the statement is composed with the interactive Sigma protocol, some of the internal state cannot be erased until the last move. For example, in the Sigma protocol, the receiver cannot erase the randomness used for generating the NO-instance x_{1-b} until it receives the challenge e, since he has to use the randomness as part of the witness in order to finish the proof. However, recall that both x_0 and x_1 are YES instances in simulation; when the adversary corrupts the receiver right before sending e, the simulator cannot return a valid randomness for x_{1-b}, and so the simulation breaks down.

Changing the order of messages. As in the commitment scheme [Lin11], we resolve this issue by switching the order of messages. That is, the message to be committed to is not only the first message a of the Sigma protocol but also the statement itself (i.e., (x_0, x_1, Φ)), and they are revealed at the last move of the Sigma protocol. Now, thanks to the equivocality of the commitment scheme, the protocol can achieve adaptive security. Refer to Figure 6 for the overall pictorial description. Here, we only describe the aforementioned modification in more detail. Recall in the statically secure protocol described in Section 5.1, the receiver sends (x_0, x_1, Φ) and the commitment c to the first message a of the Sigma protocol $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ for the language

$$\mathcal{L}^* = \{(\text{PP}, pk, L, x_0, x_1, \Phi) : \exists (b, \gamma, \xi) \text{ s.t. } x_{1-b} = \mathsf{SampNo}(\text{PP}; \gamma), \Phi = \mathsf{Enc}^L_{pk}(b; \xi)\},$$

where $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In this protocol, we change the order of messages as follows:

1. The receiver runs $a \leftarrow \mathcal{P}_\Sigma((\text{PP}, pk, L, x_0, x_1, \Phi), (b, \gamma, \xi))$, and $c \leftarrow \mathsf{Com}_{\mathsf{crs}_{com}}((x_0, x_1, \Phi, a); r)$ with r chosen uniformly at random. It sends c.

2. The sender sends the challenge message $e \leftarrow \{0,1\}^\lambda$ of the Sigma protocol.

3. Upon receiving the challenge e, the receiver generates an answer by running

$$z = \mathcal{P}_\Sigma((\text{PP}, pk, L, x_0, x_1, \Phi), (b, \gamma, \xi), e).$$

   It sends the sender the answer z along with the opening of the commitment, i.e., $((x_0, x_1, \Phi, a), r, z)$.

4. The sender verifies (a, e, z) is an accepting transcript and $((x_0, x_1, \Phi, a), r)$ is a valid opening of c:

$$\mathcal{V}_\Sigma((\text{PP}, pk, L, x_0, x_1, \Phi), a, e, z) \overset{?}{=} 1, \quad \mathsf{Com}_{\mathsf{crs}_{com}}((x_0, x_1, \Phi, a); r) \overset{?}{=} c.$$

Theorem 4 *Under the same assumptions as Theorem 3, the protocol in Figure 6 securely realizes $\mathcal{F}_{\mathsf{MOT}}$ in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model, for adaptive corruptions (assuming erasure).*

The proof appears in Section 5.4.

5.3 Proof of Theorem 3

5.3.1 The Simulator

The simulator is very similar to that in the two round case. Let Π denote the OT protocol under consideration. Here we construct a simulator Sim for any non-uniform PPT environment $\mathcal{Z}$ such that $\text{EXEC}^{\mathcal{F}_{\mathsf{CRS}}}_{\Pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\mathsf{MOT}}, \mathsf{Sim}, \mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary. Then we argue the indistinguishability between the two ensembles. Let $(\mathcal{S}_{com1}, \mathcal{S}_{com2}, \mathcal{S}_{com3})$ be the simulator for the dual-mode equivocal commitment $(\mathcal{K}_{com}, \mathsf{Com})$, and $\mathcal{S}_\Sigma$ is the simulator for the Sigma protocol.

Initialization step: The simulator Sim generates the common reference string as follows:

1. Compute $\textsc{pp}{\leftarrow}\mathsf{HashPG}(1^\lambda)$;
2. Compute $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$;
3. Compute $(\mathsf{crs}_{com}, \tau) \leftarrow \mathcal{S}_{com1}(1^\lambda)$;
4. Set $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$.

Simulating the communication with $\mathcal{Z}$: Same as the simulator in the proof of Theorem 1.

Case 1: Simulating honest receiver P_j with honest sender P_i:

1. Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j\rangle)$ from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator delivers $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ that intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, c)$ which is generated as follows:

   Generate two $\textsc{yes}$ instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\textsc{pp})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\textsc{pp})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, and then run $(c, \xi){\leftarrow}\mathcal{S}_{com2}(\tau)$.

2. Upon receiving a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ from $\mathcal{A}$ that it intends to send from P_j to P_i, the simulator randomly chooses e, and sends $\mathcal{A}$ the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$.

3. Upon receiving the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$ from $\mathcal{A}$ that intends to send from P_i to P_j, the simulator sends $\mathcal{A}$ a third round message $(\mathsf{sid}, \mathsf{ssid}, M_3)$ that intends to send from P_j to P_i, where $X = (x_0, x_1, \Phi)$ and $M_3 = (a, r, z)$ which are generated as follows:

   Run $\mathcal{S}_\Sigma$ for the Sigma protocol, i.e., compute $(a, z){\leftarrow}\mathcal{S}_\Sigma((\textsc{pp}, pk, L, X), e)$. After that the simulator runs $\mathcal{S}_{com3}$ to obtain r, i.e., compute $r{\leftarrow}\mathcal{S}_{com3}(\xi, a)$.

4. Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j\rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, and the third round message has been delivered from P_j to P_i, the simulator Sim then sets $m_0 = m_1 = 0^\ell$, and computes $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$, $Z_\sigma = m_\sigma \leftarrow \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$ for $\sigma \in \{0, 1\}$, and generates a fourth round message $(\mathsf{sid}, \mathsf{ssid}, M_4)$ that it intends to send from P_i to P_j, where $M_4 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$. The fourth round message $(\mathsf{sid}, \mathsf{ssid}, M_4)$ is then delivered to $\mathcal{A}$.

Case 2: Simulating honest receiver P_j with corrupted sender P_i:

1. Upon receiving $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j\rangle)$ from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator delivers $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ that intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, c)$ which is generated as follows:

   Generate two $\textsc{yes}$ instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\textsc{pp})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\textsc{pp})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, and then run $(c, \xi){\leftarrow}\mathcal{S}_{com2}(\tau)$.

2. Upon receiving the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$ from $\mathcal{A}$ that intends to send from P_i to P_j, the simulator sends $\mathcal{A}$ a third round message $(\mathsf{sid}, \mathsf{ssid}, M_3)$ that intends to send from P_j to P_i, where $M_3 = (a, r, z)$ is generated as follows:

Run $\mathcal{S}_\Sigma$ for the Sigma protocol, i.e., compute $(a, z) \leftarrow \mathcal{S}_\Sigma((\textsc{pp}, pk, L, X), e)$. After that the simulator runs $\mathcal{S}_{com3}$ to obtain r, i.e., compute $r \leftarrow \mathcal{S}_{com3}(\xi, a)$.

3. Upon receiving a fourth round message $(\mathsf{sid}, \mathsf{ssid}, M_4)$ from the adversary $\mathcal{A}$ that intends to send from P_i to P_j, where $M_4 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$, the simulator use (w_0, w_1) to compute $\hat{m}_0 = Z_0 \oplus \mathsf{pHash}(\textsc{pk}_1, x_0, w_0)$ and $\hat{m}_1 = Z_1 \oplus \mathsf{pHash}(\textsc{pk}_1, x_1, w_1)$. The simulator Sim sends $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \langle \hat{m}_0, \hat{m}_1 \rangle)$ to $\mathcal{F}_{\mathsf{MOT}}$.

Case 3: Simulating corrupted receiver P_j with honest sender P_i:

1. Upon receiving a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1)$ from $\mathcal{A}$ that it intends to send from P_j to P_i, where $M_1 = (x_0, x_1, \Phi, c)$, the simulator randomly chooses e, and sends $\mathcal{A}$ the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$.
2. Upon receiving a third round message $(\mathsf{sid}, \mathsf{ssid}, M_3)$ with $M_3 = (a, r, z)$ from $\mathcal{A}$ that intends to send from P_j to P_i the simulator Sim verifies (a, e, z) is an accepting transcript and (a, r) is a valid opening of c:

$$\mathcal{V}_\Sigma((\textsc{pp}, pk, L, x_0, x_1, \Phi), a, e, z) \overset{?}{=} 1, \quad \mathsf{Com}_{\mathsf{crs}_{com}}(a; r) \overset{?}{=} c.$$

   If the verification fails, then Sim aborts; otherwise, Sim decrypts Φ into $\hat{b}$, i.e., $\hat{b} = \mathsf{Dec}^L_{sk}(\Phi)$ with $L = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. If $\hat{b} \notin \{0, 1\}$, Sim aborts.
3. Upon receiving $(\textsc{send}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle)$ from functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator Sim then sends $(\textsc{receive}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \hat{b})$ to the ideal functionality $\mathcal{F}_{\mathsf{MOT}}$ in the name of dummy receiver and obtains $(\textsc{received}, \langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, m_{\hat{b}})$. Set $m_{1-\hat{b}} = 0^\ell$. Then the simulator computes $(\textsc{hk}_\sigma, \textsc{pk}_\sigma) \leftarrow \mathsf{HashKG}(\textsc{pp})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\textsc{hk}_\sigma, x_\sigma)$ for $\sigma \in \{0, 1\}$, and sends a fourth round message $(\mathsf{sid}, \mathsf{ssid}, M_4)$ to $\mathcal{A}$ that it intends to send from P_i to P_j, where $M_4 = (\textsc{pk}_0, Z_0, \textsc{pk}_1, Z_1)$.

Case 4: Simulating corrupted receiver P_j with corrupted sender P_i: This is the trivial case. Now the simulator Sim just runs $\mathcal{A}$ internally. Note that now $\mathcal{A}$ itself generates the messages between the sender and the receiver.

5.3.2 Indistinguishability

The proof is quite similar to the two-round case, and thus we highlight only the differences. As before, we show indistinguishability by following the same series of the hybrids as the two-round OT case.

- H_0: Real execution.
- H_1: When simulating honest senders, the simulator decrypts Φ from a corrupted receiver to obtain $\hat{b}$ and switches the honest sender's $m_{1-\hat{b}}$ to 0^ℓ.
- H_2: Use a simulated crs_{com} (note that, in the two-round OT case, it was using a simulated crs_{nizk}).
- H_3: In simulating honest receivers, the simulator generates a fake commitment c as well as fake Sigma protocol transcript (a, r, z) (note that in the two-round OT case, the simulator generates a simulated NIZK proof).

- H_4: In simulating honest receivers, the simulator sends a dummy encryption $\Phi = \mathsf{Enc}(0)$.
- H_5: In simulating honest receivers, the simulator generates (x_0, x_1) with both YES instances.
- H_6: In simulating honest receivers, the simulator uses witnesses for (x_0, x_1) to extract (m_0, m_1) from a dishonest sender.

The simulator is very similar to that in the two-round OT case except in simulating the proof. There, the simulator generates a fake NIZK proof, while here the simulator needs to generate a fake interactive proof (Sigma protocol along with a commitment). Since we only changed the proof-simulation part, it suffices to show the indistinguishability from H_0 to H_3. The rest is the same as the two-round OT.

$\mathbf{H_0 \approx H_1}$. As in the two-round OT, we consider the following cases:

Case (I). The proof is accepting, i.e., $\mathcal{V}_\Sigma((\text{PP}, pk, L, x_0, x_1, \Phi), a, e, z) = 1$ and $\mathsf{Com}_{\mathsf{crs}_{com}}(a; r) = c$, the ciphertext Φ correctly decrypts to some value $\hat{b} \in \{0, 1\}$, and $x_{1-\hat{b}}$ is a YES instance.

Case (II). The proof is not accepting. In this case, the sender in both H_0 and H_1 will abort, so H_0 and H_1 are identically distributed.

Case (III). The proof is accepting and $x_{1-\hat{b}}$ is a NO instance. In this case, the only difference between the distributions H_0 and H_1 is that in H_0, $Z_{1-\hat{b}}$ is honestly generated based on the real plaintext $m_{1-\hat{b}}$, but in H_1 it is based on a dummy plaintext 0^ℓ; from the smoothness property of the hash proof system, the two distributions are statistically indistinguishable.

Having these three cases in mind, next in order to show the two hybrids H_0 and H_1 are indistinguishable, we just need to argue that Case (I) occurs with negligible probability.

Suppose there exists an environment $\mathcal{Z}$ that distinguishes H_0 and H_1 with non-negligible probability. This means that with $\mathcal{Z}$, the probability that Case (I) occurs is non-negligible. Without loss of generality, we assume $\mathcal{Z}$ is deterministic. Then, we can construct an adversary $\mathcal{B}$ that breaks the binding property of the equivocal commitment scheme $(\mathcal{K}_{com}, \mathsf{Com})$. In particular, the adversary $\mathcal{B}$ works as follows:

> $\mathcal{B}$ is given crs_{com}, and it tries to come up with a commitment that opens to two different messages as follows.
>
> 1. $\mathcal{B}$ runs $\text{PP} \leftarrow \mathsf{HashPG}(1^\lambda)$ and $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and sets $\mathsf{crs}_{ot} = (\text{PP}, pk, \mathsf{crs}_{com})$.
> 2. $\mathcal{B}$ exactly follows the specification H_0 while interacting with $\mathcal{Z}$.
> 3. Once $\mathcal{Z}$ finishes execution, $\mathcal{B}$ randomly chooses a sub-session among the sub-sessions with an accepting transcript that $\mathcal{Z}$ invoked during the execution.
> 4. Let $(c, (a, r), e, z)$ be the accepting transcript of the chosen sub-session ssid. Now, $\mathcal{B}$ rewinds $\mathcal{Z}$ to the point right before sending e, and it resumes the protocol execution by sending $e' \leftarrow \{0, 1\}^\lambda$ instead of e in sub-session ssid. If $\mathcal{Z}$ generates another accepting transcript $(c, (a', r'), e', z')$ with $a \neq a'$, $\mathcal{B}$ outputs $(c, (a, r), (a', r'))$; otherwise $\mathcal{B}$ outputs $\perp$.

Let δ be the probability that Case (I) occurs. Then, the probability that Case (I) occurs in the chosen session ssid is at least δ/T, where T is the maximum number of sub-sessions that $\mathcal{Z}$ initiates. Now, conditioned that the chosen session contains a false statement, the probability that $\mathcal{Z}$ outputs another accepting transcript $(c, (a', r'), e', z')$ is δ, since e' is also chosen uniformly at random. Now consider the special soundness of the Sigma protocol: the statement is false, so there is only one accepting challenge (i.e., e) for a. Therefore, $e \neq e'$ implies $a \neq a'$. Considering the event $e \neq e'$ happens with probability $1 - 2^{\lambda}$, we conclude that $\mathcal{B}$ outputs a commitment along with two different openings with non-negligible probability of at least $(1 - 2^{-\lambda})\delta^2/T$.

$\mathbf{H_1} \approx \mathbf{H_2}$**.** Let $(\mathcal{S}_{com1}, \mathcal{S}_{com2}, \mathcal{S}_{com3})$ be the simulator for the equivocal commitment scheme $(\mathcal{K}_{com}, \mathsf{Com})$. In H_2, the CRS for the commitment scheme is generated as follows:

$$(\mathsf{crs}_{com}, \tau) \leftarrow \mathcal{S}_{com1}(1^{\lambda}).$$

The indistinguishability simply holds from the security definition of the equivocal commitment scheme.

$\mathbf{H_2} \approx \mathbf{H_3}$**.** In the interactive setting, the simulation for the zero-knowledge proof goes as follows:

1. Let $\mathcal{S}_{\Sigma}$ be the simulator for the Sigma protocol $(\mathcal{P}_{\Sigma}, \mathcal{V}_{\Sigma})$. Let $X = (x_0, x_1, \Phi)$ be the statement to prove.

2. Run $(c, \xi) \leftarrow \mathcal{S}_{com2}(\tau)$, and send c (along with (x_0, x_1, Φ)) to the adversary.

3. Upon receiving the challenger e, run $(a, z) \leftarrow \mathcal{S}_{\Sigma}((\textsc{pp}, pk, L, X), e)$ and $r \leftarrow \mathcal{S}_{com3}(\xi, a)$; send $((a, r), z)$ to the adversary.

From the special honest zero-knowledge property of the Sigma protocol, (a, e, z) are statistically indistinguishable to the original transcript, and from the equivocality property of the commitment scheme (c, a, r) is identically distributed to the original transcript. Therefore, H_2 and H_3 are statistically indistinguishable.

5.4 Proof of Theorem 4

The proof is very similar to that of Theorem 3 but with careful treatment of corruptions. Here we construct a simulator Sim for any non-uniform PPT environment $\mathcal{Z}$ such that $\textsc{exec}^{\mathcal{F}_{\mathsf{CRS}}}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \textsc{ideal}_{\mathcal{F}_{\mathsf{MOT}},\mathsf{Sim},\mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary. Then we argue the indistinguishability between the two ensembles. Let $(\mathcal{S}_{com1}, \mathcal{S}_{com2}, \mathcal{S}_{com3})$ be the simulator for the equivocal commitment $(\mathcal{K}_{com}, \mathsf{Com})$, and $\mathcal{S}_{\Sigma}$ is the simulator for the Sigma protocol.

5.4.1 The Simulator

Initialization step: Same as the simulator in the proof of Theorem 3.

Simulating the communication with $\mathcal{Z}$: Same as the simulator in the proof of Theorem 3.

Case 1: Simulating honest receiver P_j with honest sender P_i:

1. Upon receiving (RECEIVE, $\langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle$) from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator sends $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1 = c)$ that intends to send from P_j to P_i, where c is obtained by computing $(c, \xi) \leftarrow \mathcal{S}_{com2}(\tau)$.
2. The simulator randomly chooses e, and sends $\mathcal{A}$ the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$.
3. The simulator sends $\mathcal{A}$ a third round message $(\mathsf{sid}, \mathsf{ssid}, M_3)$ that intends to send from P_j to P_i, where $X = (x_0, x_1, \Phi)$ and $M_3 = (X, a, r, z)$ which are generated as follows:

   > Generate two YES instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\text{PP})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\text{PP})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, define $X = (x_0, x_1, \Phi)$, and then run $\mathcal{S}_\Sigma$ for the Sigma protocol, i.e., compute $(a, z) \leftarrow \mathcal{S}_\Sigma((\text{PP}, pk, L, X), e)$. After that the simulator runs $\mathcal{S}_{com3}$ to obtain r, i.e., compute $r \leftarrow \mathcal{S}_{com3}(\xi, (X, a))$.

4. The simulator informs the adverary $\mathcal{A}$ that the fourth round message has been transferred over the secure channel.

Case 2: Simulating honest receiver P_j with corrupted sender P_i:

1. Upon receiving (RECEIVE, $\langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle$) from the functionality $\mathcal{F}_{\mathsf{MOT}}$, the simulator sends $\mathcal{A}$ a first round message $(\mathsf{sid}, \mathsf{ssid}, M_1 = c)$ that intends to send from P_j to P_i, where c is obtained by computing $(c, \xi) \leftarrow \mathcal{S}_{com2}(\tau)$.
2. Upon receiving the second round message $(\mathsf{sid}, \mathsf{ssid}, M_2 = e)$ from $\mathcal{A}$ that intends to send from P_i to P_j, the simulator sends $\mathcal{A}$ a third round message $(\mathsf{sid}, \mathsf{ssid}, M_3)$ that intends to send from P_j to P_i, where $X = (x_0, x_1, \Phi)$ and $M_3 = (X, a, r, z)$ which are generated as follows:

   > Generate two YES instances, i.e, $(x_0, w_0) \leftarrow \mathsf{SampYes}(\text{PP})$, $(x_1, w_1) \leftarrow \mathsf{SampYes}(\text{PP})$, and record (w_0, w_1). Compute a dummy ciphertext $\Phi \leftarrow \mathsf{Enc}_{pk}(0)$, define $X = (x_0, x_1, \Phi)$, and then run $\mathcal{S}_\Sigma$ for the Sigma protocol, i.e., compute $(a, z) \leftarrow \mathcal{S}_\Sigma((\text{PP}, pk, L, X), e)$. After that the simulator runs $\mathcal{S}_{com3}$ to obtain r, i.e., compute $r \leftarrow \mathcal{S}_{com3}(\xi, (X, a))$.

3. Upon receiving a fourth round message $(\mathsf{sid}, \mathsf{ssid}, M_4)$ from the adversary $\mathcal{A}$ that intends to send from P_i to P_j, where $M_4 = (\text{PK}_0, Z_0, \text{PK}_1, Z_1)$, the simulator use (w_0, w_1) to compute $\hat{m}_0 = Z_0 \oplus \mathsf{pHash}(\text{PK}_1, x_0, w_0)$ and $\hat{m}_1 = Z_1 \oplus \mathsf{pHash}(\text{PK}_1, x_1, w_1)$. The simulator Sim sends (SEND, $\langle \mathsf{sid}, \mathsf{ssid}, P_i, P_j \rangle, \langle \hat{m}_0, \hat{m}_1 \rangle$) to $\mathcal{F}_{\mathsf{MOT}}$.

Case 3: Simulating corrupted receiver P_j with honest sender P_i: Similar to the simulator in the proof of Theorem 3. The only difference is in the proof verification according to the change of the protocol specification.

Case 4: Simulating corrupted receiver P_j with corrupted sender P_i: This is the trivial case. Now the simulator Sim just runs $\mathcal{A}$ internally. Note that now $\mathcal{A}$ itself generates the messages between the sender and the receiver.

Dealing with adaptive corruption of sender P_i:

Before the second message. This case is equivalent to static corruption. That is, the simulator receives the original input (m_0, m_1) and returns it to the adversary.

Between the second message and the fourth. Likewise, the simulator receives the original input (m_0, m_1), and return (m_0, m_1, e) to the adversary $\mathcal{A}$ as P_i's internal state.

After the fourth message. The simulator receives the original input (m_0, m_1).

- If the receiver P_i is already corrupted (Case 3), at this moment, the last round message $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$ is fixed by the simulator itself. Now the simulator has to return the internal state of the sender to the adversary, and it returns $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)$.
- Otherwise (Case 1), the simulator has to additionally generate the last round message. The simulator Sim computes $(\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$ for $\sigma \in \{0,1\}$. Now the simulator has to return the internal state of the sender to the adversary, and it returns $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)$.

Depending on whether P_j is honest, the simulator continues the simulation by following Case 2 (for honest P_j) or Case 4 (for corrupted P_j).

Dealing with adaptive corruption of receiver P_j:

Before the first message. This case is equivalent to static corruption. That is, the simulator receives the original input b and returns it to the adversary.

Between the first message and the second. The simulator receives the original input b. The simulator then computes $X = (x_0, x_1, \Phi)$ honestly and record witness (b, γ, ξ), and then generates a honestly i.e. $a \leftarrow \mathcal{P}_\Sigma((\text{PP}, pk, L, X), (b, \gamma, \xi))$. At this point, the simulator runs $r \leftarrow \mathcal{S}_{com3}(c, (X, a))$, and return all internal state, i.e., $(b, X, \gamma, \xi, a, r)$ to the adversary $\mathcal{A}$.

Between the second message and the third. The simulator receives the original input b. The simulator then computes $X = (x_0, x_1, \Phi)$ honestly and record witness W, and then generates a, z honestly i.e. $a \leftarrow \mathcal{P}_\Sigma((\text{PP}, pk, L, X), (b, \gamma, \xi))$, $z \leftarrow \mathcal{P}_\Sigma((\text{PP}, pk, L, X), (b, \gamma, \xi), e)$. At this point, the simulator runs $r \leftarrow \mathcal{S}_{com3}(c, (X, a))$, and returns (b, w, X, a, r, z) to the adversary $\mathcal{A}$ as P_j's internal state.

Between the third message and the fourth. The simulator receives the original input b, and returns (b, w_b, X, a, r, z) to the adversary $\mathcal{A}$ as P_j's internal state.

After the fourth message. The simulator receives the original input b and the output m_b.

- If the sender P_i is already corrupted (Case 2), at this moment, the last round message $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$ is fixed either by the adversary $\mathcal{A}$ or by the simulator itself. Now the simulator has to return the internal state of the receiver to the adversary, and it returns (b, w_b, X, a, r, z) and $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$.
- Otherwise (Case 1), the simulator has to additionally generate the last round message. The simulator Sim sets $m_{1-b} = 0^\ell$, and computes $(\text{HK}_\sigma, \text{PK}_\sigma) \leftarrow \mathsf{HashKG}(\text{PP})$, $Z_\sigma = m_\sigma \oplus \mathsf{Hash}(\text{HK}_\sigma, x_\sigma)$ for $\sigma \in \{0,1\}$. Now the simulator has to return the internal state of the receiver to the adversary, and it returns (b, w_b, X, a, r, z) and $(\text{PK}_0, Z_0, \text{PK}_1, Z_1)$.

Note that in either case, it holds that $m_b = Z_b \oplus \mathsf{pHash}(\text{PK}_b, x_b, w)$.

Depending on whether P_i is honest, the simulator continues the simulation by following Case 3 (for honest P_i) or Case 4 (for corrupted P_i).

5.4.2 Indistinguishability

The proof is quite similar to that of Theorem 3. We follow the same series of the hybrids as before.

- H_0: Real execution.
- H_1: When simulating honest senders, the simulator decrypts Φ from a corrupted receiver to obtain $\hat{b}$ and switches the honest sender's $m_{1-\hat{b}}$ to 0^ℓ.
- H_2: Use a simulated crs_{com}.
- H_3: In simulating honest receivers, the simulator generates a simulated Sigma protocol transcript and a fake commitment c.
- H_4: In simulating honest receivers, the simulator sends a dummy encryption $\Phi = \mathsf{Enc}(0)$.
- H_5: In simulating honest receivers, the simulator generates (x_0, x_1) with both YES instances.
- H_6: In simulating honest receivers, the simulator uses witnesses for (x_0, x_1) to extract (m_0, m_1) from a dishonest sender.

Since we only changed the proof part, it suffices to show the indistinguishability from H_0 to H_3. The rest are the same as the two-round OT.

$\mathbf{H_0 \approx H_1}$. As in the two-round OT, we consider the following cases:

Case (I). The proof is accepting, i.e., $\mathcal{V}_\Sigma((\textsc{pp}, pk, L, x_0, x_1, \Phi), a, e, z) = 1$ and $\mathsf{Com}_{\mathsf{crs}_{com}}(a; r) = c$, and the ciphertext Φ correctly decrypts to some value $\hat{b} \in \{0, 1\}$, and $x_{1-\hat{b}}$ is a YES instance.

Case (II). The proof is not accepting. In this case, the sender in both H_0 and H_1 will abort, so H_0 and H_1 are identically distributed.

Case (III). The proof is accepting and $x_{1-\hat{b}}$ is a NO instance. In this case, the only difference between the distributions H_0 and H_1 is that in H_0, $Z_{1-\hat{b}}$ is honestly generated based on the real plaintext $m_{1-\hat{b}}$, but in H_1 it is based on a dummy plaintext 0^ℓ; from the smoothness property of the hash proof system, the two distributions are statistically indistinguishable.

Having these three cases in mind, suppose there exists an environment $\mathcal{Z}$ that distinguishes H_0 and H_1 with non-negligible probability. This means that with $\mathcal{Z}$, the probability that Case (I) occurs is non-negligible. Without loss of generality, we assume $\mathcal{Z}$ is deterministic. Then, we can construct an adversary $\mathcal{B}$ that breaks the binding property of the equivocal commitment scheme $(\mathcal{K}_{com}, \mathsf{Com})$. In particular, the adversary $\mathcal{B}$ works as follows:

> $\mathcal{B}$ is given crs_{com}, and it tries to come up with a commitment that opens to two different messages as follows.
>
> 1. $\mathcal{B}$ runs $\textsc{pp}{\leftarrow}\mathsf{HashPG}(1^\lambda)$ and $(pk, sk){\leftarrow}\mathsf{Gen}(1^\lambda)$ and sets $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$.
> 2. $\mathcal{B}$ exactly follows the specification H_0 while interacting with $\mathcal{Z}$.
> 3. Once $\mathcal{Z}$ finishes execution, $\mathcal{B}$ randomly chooses a sub-session among the sub-sessions with an accepting transcript that $\mathcal{Z}$ invoked during the execution.

4. Let $(c, (X, a, r), e, z)$ be the accepting transcript of the chosen sub-session ssid^*. Now, $\mathcal{B}$ rewinds $\mathcal{Z}$ to the point right before sending e, and it resumes the protocol execution by sending $e' \leftarrow \{0,1\}^\lambda$ instead of e in sub-session ssid. If $\mathcal{Z}$ generates another accepting transcript $(c, (X', a', r'), e', z')$ with $(X, a) \neq (X', a')$, $\mathcal{B}$ outputs $(c, ((X, a), r), ((X', a'), r'))$; otherwise $\mathcal{B}$ outputs $\perp$.

Let $\mathsf{Diff}(\mathsf{ssid})$ be a predicate for a given session ssid that is defined to be true if the sub-session ssid satisfies the following conditions:

- The sender is honest.
- The receiver is corrupted.
- Case (I) occurs.

Let $\Pr[\exists \mathsf{ssid}\ s.t.\ \mathsf{Diff}(\mathsf{ssid}) = 1] = \delta$. By assumption, the probability δ is non-negligible. Then, the probability that $\mathsf{Diff}(\mathsf{ssid}^*)$ is true for the chosen session ssid^* is at least δ/T, where T is the maximum number of sub-sessions that $\mathcal{Z}$ initiates. Now, conditioned that Diff is true for the chosen session ssid^*, the probability that $\mathcal{Z}$ outputs another accepting transcript $(c, (X', a', r'), e', z')$, with $\mathsf{Diff}(\mathsf{ssid}^*) = 1$, is δ/T; the challenge e' is also chosen uniformly at random. Now consider the special soundness of the Sigma protocol: the statement is false, so there is only one accepting challenge (i.e., e) for a. Therefore, $e \neq e'$ implies $(X, a) \neq (X', a')$. Considering the event $e \neq e'$ happens with probability $1 - 2^\lambda$, we conclude that $\mathcal{B}$ outputs a commitment along with two different openings with non-negligible probability of at least $(1 - 2^{-\lambda})(\delta/T)^2$.

$\mathbf{H_1} \approx \mathbf{H_2}$. Let $(\mathcal{S}_{com1}, \mathcal{S}_{com2}, \mathcal{S}_{com3})$ be the simulator for the equivocal commitment scheme $(\mathcal{K}_{com}, \mathsf{Com})$. In H_2, the CRS for the commitment scheme is generated as follows:

$$(\mathsf{crs}_{com}, \tau) \leftarrow \mathcal{S}_{com1}(1^\lambda).$$

The indistinguishability simply holds from the security definition of the equivocal commitment scheme.

$\mathbf{H_2} \approx \mathbf{H_3}$. In the interactive setting, the simulation for the zero-knowledge proof goes as follows:

1. Let $\mathcal{S}_\Sigma$ be the simulator for the Sigma protocol $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$. The statement to be proven is $(\textsc{pp}, pk, L, X)$ where $X = (x_0, x_1, \Phi)$.

2. Run $(c, \xi) \leftarrow \mathcal{S}_{com2}(\tau)$, and send c to the adversary.

3. Upon receiving the challenger e, run $(a, z) \leftarrow \mathcal{S}_\Sigma((\textsc{pp}, pk, L, X), e)$ and $r \leftarrow \mathcal{S}_{com}(\xi, (X, a))$; send $((X, a), r, z)$ to the adversary.

From the special honest zero-knowledge property of the Sigma protocol, (a, e, z) are statistically indistinguishable to the original transcript, and from the equivocality property of the commitment scheme (c, a, r) is identically distributed to the original transcript. Therefore, H_2 and H_3 are statistically indistinguishable.

5.4.3 Dealing with Corruptions

We are left to show that the internal state returned to the adversary upon adaptive corruption is indistinguishable from that in the real-world protocol.

Dealing with adaptive corruption of sender P_i.

Before the second message. The simulation is simply perfect, since there has been no simulated protocol transcript so far in the given sub-session.

Between the second message and the fourth. The simulated value e is uniform; so the simulation is perfect.

After the fourth message. Since the adversary is given only the erased state $(m_0, m_1, \text{PK}_0, Z_0, \text{PK}_1, Z_1)$, the simulation is indististinguisahble to the real-world execution due to the hard-subset membership property and the smoothness of the underlying hash proof system.

Dealing with adaptive corruption of receiver P_j.

Before the first message. The simulation is simply perfect, since there has been no simulated protocol transcript so far in the given sub-session.

Between the first message and the second. From the perfect hiding property of the commitment scheme, the simulation is perfect.

Between the second message and the third. From the perfect hiding property of the commitment scheme, the simulation is perfect.

Between the third message and the fourth. Note that (x_0, x_1, Φ) in the simulated transcript is such that x_{1-b} is a yes instance, and $\Phi = \mathsf{Enc}(0)$. However, since the adversary is given only the erased state (b, w_b, X, a, r, z), the simulation is indistinguishable from the real-world execution due to the hard subset membership property for x_{1-b} and CCA security for Φ and zero-knowledge property for Sigma protocol.

After the fourth message. Indistinguishability for simulating (b, w_b, X, a, r, z) holds as in the above case. Even when the simulator additionally returns $(\text{PK}_0, Z_0, \text{PK}_1, Z_0)$ to the adversary $\mathcal{A}$, the simulation is indistinguishable (as with the sender corruption case).

6 Instantiations of Protocol 2 and Protocol 2*

6.1 Instantiation from the DDH Assumption

We show a CCA-secure labeled public-key encryption scheme, a smooth hash proof system, and an equivocal commitment scheme under the DDH assumption. We then obtain a four-round OT protocol by combining these building blocks.

6.1.1 Decisional Diffie-Hellman Assumption

Let $\mathcal{G}_{\text{ddh}}$ be a randomized algorithm that takes a security parameter λ and outputs $\mathsf{desc} = (p, \mathbb{G}, g)$ such that $\mathbb{G}$ is the description of group of prime order p, and g is a generator of $\mathbb{G}$.

Definition 11 (Decisional Diffie-Hellman Assumption) *The* DDH problem is hard relative to $\mathbb{G}$ *if for all* PPT *algorithms $\mathcal{A}$ there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left|\Pr[\mathcal{A}(\mathbb{G}, p, g, g^a, g^b, g^c) = 1] - \Pr[\mathcal{A}(\mathbb{G}, p, g, g^a, g^b, g^{ab}) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where in each case the probabilities are taken over the experiment in which the group-generating algorithm outputs $(\mathbb{G}, p, g)$ and random $a, b, c \in \mathbb{Z}_p$ are chosen.

6.1.2 CCA-Secure Labeled Public-Key Encryption Scheme

Since the DDH assumption holds in $\mathbb{G}_1$, we can use Cramer-Shoup encryption scheme [CS98]. As in the case for the DLIN assumption, we slightly change the scheme to support labels, that is, we use collision resistant hash functions instead of UOWHF and apply labels to hash functions when performing encryptions and decryptions.

Key generation $(pk, sk) \leftarrow \mathsf{Gen}(\mathsf{desc})$: Choose random generators $g_1 \leftarrow \mathbb{G}$ and exponents $\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2 \leftarrow \mathbb{Z}_p$ and compute $c = g_1^{\beta_1} g^{\beta_2}, d = g_1^{\gamma_1} g^{\gamma_2}, h = g_1^{\delta_1} g^{\delta_2}$. Choose a hash function $H \leftarrow \mathcal{HF}$ where $\mathcal{HF}$ is a family of collision-resistant hash functions. Now set $pk = (g_1, g, c, d, h, H)$ and $sk = (\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2)$.

Encryption $C \leftarrow \mathsf{Enc}_{pk}^{L}(m; r)$: Given the message $m \in \mathbb{G}$ under label L, choose $r \leftarrow \mathbb{Z}_p$ and compute $u_1 = g_1^r, u_2 = g^r, e = m \cdot h^r$. Then compute $\alpha = H(u_1, u_2, e, L) \in \mathbb{Z}_p$ and $v = (cd^{\alpha})^r$. The ciphertext is $C = (u_1, u_2, e, v)$.

Decryption $\mathsf{Dec}_{sk}^{L}(C)$: Parse $C = (u_1, u_2, e, v)$ and $sk = (\beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2)$; compute $\alpha \leftarrow H(u_1, u_2, e, L)$ and test if $u_1^{\beta_1 + \alpha\gamma_1} \cdot u_2^{\beta_2 + \alpha\gamma_2} \overset{?}{=} v$. If it does not, output `reject`. Otherwise, output $m = e/(u_1^{\delta_1} u_2^{\delta_2})$.

6.1.3 Smooth Projective Hashing

We recall the smooth projective hashing based on the DDH assumption [CS98, CS02].

Parameter generation. Choose $g_1, g \leftarrow \mathbb{G}$. Then $\textsc{pp} = (g_1, g, \mathbb{G})$.

Instance sampling. To sample a YES instance, choose $t \leftarrow \mathbb{Z}_p$, and compute $z_1 = g_1^t$, $z_2 = g^t$, and then return $x = (z_1, z_2)$. To sample a NO instance, choose $t \leftarrow \mathbb{Z}_p$, and then $z_1 = g_1^t$, $z_2 = g^{t+1}$, and then return $x = (z_1, z_2)$.

Hash key generation. Choose $\theta_1, \theta_2 \leftarrow \mathbb{Z}_p$ and compute $f = g_1^{\theta_1} g^{\theta_2}$. Return $\textsc{hk} = (\theta_1, \theta_2)$, and $\textsc{pk} = f$.

Primary hashing. Given $\textsc{hk} = (\theta_1, \theta_2)$ and $x = (z_1, z_2)$, return $y = z_1^{\theta_1} z_2^{\theta_2}$.

Projective hashing. Given a projective hash key $\textsc{pk} = f$, an instance $x = (z_1, z_2)$, and its witness $w = t$ such that $z_1 = g_1^t$, $z_2 = g^t$, return $y = f^t$.

6.1.4 Equivocal Commitment

We use a variant of the famous equivocal commitment by Pedersen [Ped92]. The main difference from the original Pedersen commitment is that collision resilient hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ is used to commit to arbitrary long message very efficiently. In particular, given the CRS $(g, h_1) \in \mathbb{G}^2$, the commitment to a message m is $g^r h_1^{H(m)}$. We note that the binding property is under the DLOG assumption and the collision resilient property of the hash function. When a trapdoor ζ with $h_1 = g^{\zeta}$ is known, it easy to equivocate a commitment $c = g^s$ into any m by outputting $r = s - \zeta \cdot H(m)$.

6.1.5 Oblivious Transfer

By plugging in these components into the generic framework for four-round OT, we obtain an OT protocol based on the DDH assumption. Thus, it is only left to show the concrete Sigma protocol that is used in the receiver side message.

Protocol detail. Ignoring the description desc of the group $\mathbb{G}$, the CRS is $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$ where $\textsc{pp} = (g_1, g) \quad pk = (g_1, g, c, d, h, H) \quad \mathsf{crs}_{com} = (h_1, g)$. Therefore, the CRS can be represented with 6 group elements of $\mathbb{G}$ and one hash function index, along with the description of the group $\mathbb{G}$.

Let $x_0 = (z_{01}, z_{02})$, $x_1 = (z_{11}, z_{12})$, and $\Phi = (u_1, u_2, e, v)$ with $\alpha = H(u_1, u_2, e, (\mathsf{sid}, \mathsf{ssid}, P_i, P_j))$. Then, we use a standard Sigma protocol for the following language:

$$\mathcal{L}^* = \left\{ \begin{array}{l} (\mathsf{crs}_{ot}, pk, x_0, x_1, \Phi, \alpha) : \\ \exists (r,t) \text{ s.t. } u_1 = g_1^r, u_2 = g^r, e = h^r, v = (cd^\alpha)^r, z_{11} = g_1^t, z_{12} = g^{t+1} \\ \qquad\qquad \text{or } u_1 = g_1^r, u_2 = g^r, e = gh^r, v = (cd^\alpha)^r, z_{01} = g_1^t, z_{02} = g^{t+1} \end{array} \right\}.$$

1. Suppose that $\Phi = \mathsf{Enc}(g^b)$. Let $\bar{b} = 1 - b$. The prover chooses $R, T \leftarrow \mathbb{Z}_p$, $\eta \leftarrow [0, 2^\lambda)$, and $\rho, \tau \leftarrow \mathbb{Z}_p$. Then, it computes and sends the verifier the following:

$$\begin{array}{lll} U_{1b} = g_1^R, & U_{2b} = g^R, & E_b = h^R, \\ V_b = (cd^\alpha)^R, & Z_{1b} = g_1^T, & Z_{2b} = g^T \\ U_{1\bar{b}} = g_1^\rho / u_1^\eta, & U_{2\bar{b}} = g_1^\rho / u_2^\eta, & E_{\bar{b}} = h^\rho / (e/g^{\bar{b}})^\eta, \\ V_{\bar{b}} = (cd^\alpha)^\rho / v^\eta, & Z_{1\bar{b}} = g_1^\tau / z_{b1}^\eta, & Z_{2\bar{b}} = g^\tau / (z_{b2}/g)^\eta. \end{array}$$

2. The verifier chooses $\epsilon \leftarrow [0, 2^\lambda)$ and sends it to the prover.

3. The prover computes the following:

$$\begin{array}{ll} \epsilon_b = \epsilon - \eta \bmod 2^\lambda & \epsilon_{\bar{b}} = \eta \\ \rho_b = R + r\epsilon_b & \rho_{\bar{b}} = \rho \\ \tau_b = T + t\epsilon_b & \tau_{\bar{b}} = \tau. \end{array}$$

   Then, it sends $(\epsilon_0, \rho_0, \tau_0, \rho_1, \tau_1)$ to the verifier.

4. The verifier computes $\epsilon_1 = \epsilon - \epsilon_0 \bmod 2^\lambda$. It also checks if the following holds for $i \in \{0, 1\}$.

$$\begin{array}{lll} g_1^{\rho_i} = U_{1i} \cdot u_1^{\epsilon_i}, & g^{\rho_i} = U_{2i} \cdot u_2^{\epsilon_i}, & h^{\rho_i} = E_i \cdot (e/g^i)^{\epsilon_i}, \\ (cd^\alpha)^{\rho_i} = V_i \cdot v^{\epsilon_i}, & g_1^{\tau_i} = Z_{1i} \cdot z_{\bar{i}1}^{\epsilon_i}, & g^{\tau_i} = Z_{2i} \cdot (z_{\bar{i}2}/g)^{\epsilon_i}. \end{array}$$

Communication complexity. The receiver message (x_0, x_1, Φ) needs $2 + 2 + 4 = 8$ group elements. The proof takes 13 elements in $\mathbb{G}$ and 7 elements in $\mathbb{Z}_p$. In particular, the first message has one commitment (i.e., 1 element in $\mathbb{G}$). The second message has 1 element in ${\mathbb{Z}_p}^3$, and the third messages has 5 elements in $\mathbb{Z}_p$ along with the decommitment (i.e., 12 elements in $\mathbb{G}$ and 1 element in $\mathbb{Z}_p$). The sender message (pk_0, Z_0, pk_1, Z_1) needs $(1, 1, 1, 1) = 4$ group elements in $\mathbb{G}$. Therefore, the total communication complexity amounts to 25 elements in $\mathbb{G}$ and 7 elements in $\mathbb{Z}_p$.

[3] In fact, the second message is in $\{0, 1\}^\lambda$ but we count it as an element of $\mathbb{Z}_p$ for simplicity.

Adaptive security: realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. The NCE protocol messages can be overlapped with the OT protocol messages, and thus the final OT protocol runs in four rounds. We can use ElGamal encryption, and the communication overhead amounts to 3 group elements (the public key consists of one element excluding the generator in the CRS, and the ciphertext consists of two elements).

6.2 Instantiation from the DCR Assumption

We show a CCA-secure labeled public-key encryption scheme, a smooth hash proof system, and an equivocal commitment under the DCR assumption. We then obtain a four-round OT protocol by combining these building blocks.

6.2.1 Decisional Composite Residuosity Assumption (DCR)

Let N be a Blum integer, i.e., $N = PQ$ for safe primes $P, Q \equiv 3 \bmod 4$ so that $P = 2p + 1$ and $Q = 2q+1$ for primes p, q. The decision composite residuosity (DCR) assumption [Pai99] states that given N, it is hard to distinguish random elements from $\mathbb{Z}^*_{N^2}$ from random N-th power elements in $\mathbb{Z}^*_{N^2}$. Let $\mathsf{Primes}(\lambda)$ denote the set of prime numbers between 2^λ and $2^{\lambda+1}$.

Definition 12 (Decisional Composite Residuosity Assumption) *We say the Decisional Composite Residuosity assumption holds if for any $\lambda \in \mathbb{N}$ and any non-uniform polynomial time adversary $\mathcal{A}$, it holds that $|p_1 - p_2| = \mathsf{negl}(\lambda)$, where*

$$p_1 = \Pr[p, q \leftarrow \mathsf{Primes}(\lambda); N = pq; x \leftarrow \mathbb{Z}^*_{N^2} : \mathcal{A}(N, x) = 1]$$
$$p_2 = \Pr[p, q \leftarrow \mathsf{Primes}(\lambda); N = pq; x \leftarrow \mathbb{Z}^*_{N^2} : \mathcal{A}(N, x^N) = 1].$$

6.2.2 CCA-Secure Labeled Public-Key Encryption Scheme

We use the variant of Camenisch-Shoup encryption scheme [CS03] described in [HK12].

Key Generation Choose $p, q \leftarrow \mathsf{Primes}(\lambda)$ and compute $N = PQ$ where $P = 2p+1$ and $Q = 2q+1$. Choose $g' \leftarrow \mathbb{Z}^*_{N^2}$ and $\beta, \gamma, \delta \leftarrow [N^2/4]$, and compute the following:

$$g = (g')^{2N} \qquad c = g^\beta \qquad d = g^\gamma \qquad h = g^\delta.$$

Choose a hash function $H \leftarrow \mathcal{HF}$ where $\mathcal{HF}$ is a family of collision-resistant hash functions. Now set $pk = (N, g, c, d, h, H)$ and $sk = (\beta, \gamma, \delta)$.

Encryption $C \leftarrow \mathsf{Enc}^L_{pk}(m)$: Given the message $m \in [M]$ under label L, choose $r \leftarrow [N/4]$ and compute the following:

$$u_1 = g^r, \quad e = (1 + N)^m \cdot h^r$$

Then compute $\alpha = H(u_1, e, L)$ and $v = |(cd^\alpha)^r|$, where $|x| \overset{\text{def}}{:=} \min\{x, N^2 - x\}$. The ciphertext is $C == (u, e, v)$.

Decryption $\mathsf{Dec}^L_{sk}(C)$: Parse $C = (u, e, v)$ and $sk = (\beta, \gamma, \delta)$; compute $\alpha = H(u, e, L)$ and $z = (e/u^\delta)^{N+1} \bmod N^2$. Check if the following conditions hold:

$$v \stackrel{?}{=} |v|, \quad u^{2(\beta+\alpha\gamma)} \stackrel{?}{=} v^2, \quad z - 1 \text{ is divisible by } N.$$

If so, output $(z - 1)/N$; otherwise output `reject`.

6.2.3 Smooth Projective Hashing

We use the smooth projective hashing scheme described in [HK12].

Parameter Generation. Choose two random prime numbers $p, q \leftarrow \mathsf{Primes}(\lambda)$ and compute $N = PQ$, where $P = 2p+1$, $Q = 2q+1$. Choose $g' \leftarrow \mathbb{Z}^*_{N^2}$ and compute $g_1 = (g')^N \bmod N^2$. Then the parameter is $\textsc{pp} = (N, g_1)$.

Instance Sampling. Randomly choose $w \leftarrow \mathbb{Z}^*_N$. For a YES instance, set $x = g_1^w \bmod N^2$ and return (x, w); for a NO instance, return $x = g_1^w \cdot (1 + N) \bmod N^2$.

Hash Key Generation. Randomly choose $\theta \leftarrow \mathbb{Z}_N$ and compute $f = g_1^\theta \bmod N^2$. Return $\textsc{hk} = \theta$, and $\textsc{pk} = f$.

Primary Hashing Given a primary hash key $\textsc{hk} = \theta$ and an instance x, return $y = x^\theta \bmod N^2$.

Projective Hashing Given a projective hash key $\textsc{pk} = f$, an instance x, and its witness w such that $x = g_1^w$, return $y = f^w \bmod N^2$.

6.2.4 Equivocal Commitment

CRS generation: The common reference string consists of $(N, \mathbf{g})$. Here, $N = PQ$ and $P = 2p+1$, $Q = 2q + 1$ where $p, q \leftarrow \mathsf{Primes}(\lambda)$. $\mathbf{g} = (g')^N \bmod N^2$, where $g' \leftarrow \mathbb{Z}^*_N$ is the equivocation trapdoor.

Committing: To commit a value m, randomly choose $r \in \mathbb{Z}^*_N$. The commitment value is $c = \mathbf{g}^{H(m)} r^N \bmod N^2$, and the decommitment value is (m, r).

Equivocation: In the committing stage, the simulation algorithm $\mathcal{S}_{com2}$ chooses $s \leftarrow \mathbb{Z}^*_N$ and outputs $s^N \bmod N^2$. Given a message m, the equivocation algorithm $\mathcal{S}_{com3}$ computes $r = (g')^{-H(m)} s \bmod N$. Observe that $\mathbf{g}^{H(m)} r^N = ((g')^N)^{H(m)}((g')^{-H(m)} s)^N = s^N \bmod N^2$.

6.2.5 Oblivious Transfer

Now we are ready to describe the protocol based on the DCR assumption.

CRS. The reference string $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$ is as follows:

$$\begin{aligned} \textsc{pp} &= (N, g_1) \\ pk &= (N, g, c, d, h, H) \\ \mathsf{crs}_{com} &= (N, \mathbf{g}) \end{aligned}$$

Protocol detail. It suffices to describe how to generate the proof in the receiver side message M_1 given $\mathsf{crs}_{ot} = (\textsc{pp}, pk, \mathsf{crs}_{com})$. Let $\Phi = (u, e, v)$ with $\alpha = H(u, e, (\mathsf{sid}, \mathsf{ssid}, P_i, P_j))$. We show a Sigma protocol for the following language:

$$\mathcal{L} = \left\{ \begin{array}{l} (\mathsf{crs}_{ot}, x_0, x_1, \Phi, \alpha) : \exists (r, t) \text{ s.t.} \\ \quad (u = g^r, e = h^r, v^2 = (cd^\alpha)^{2r}, x_1 = g_1^t \cdot (1+N)) \\ \quad \text{or} \\ \quad (u = g^r, e = (1+N)h^r, v^2 = (cd^\alpha)^{2r}, x_0 = g_1^t \cdot (1+N)) \end{array} \right\}.$$

The concrete Sigma protocol proceeds as follows:

1. The prover chooses $R, T \leftarrow \mathbb{Z}_N$, $\eta \leftarrow [1, 2^\lambda)$, and $\rho, \tau, \leftarrow \mathbb{Z}_N$. Let $\bar{b} = 1 - b$. Then, it computes and send the verifier the following:

$$\begin{array}{llll} U_b = g^R, & E_b = h^R, & V_b = (cd^\alpha)^{2R}, & X_b = g_1^T, \\ U_{\bar{b}} = g^\rho / u^\eta, & E_{\bar{b}} = h^\rho / (e/(1+N)^{\bar{b}})^\eta, & V_{\bar{b}} = (cd^\alpha)^{2\rho} / v^{2\eta}, & X_{\bar{b}} = g_1^\tau / (x_b/(1+N))^\eta, \end{array}$$

2. The verifier chooses $\epsilon \leftarrow [1, 2^\lambda)$ and sends it to the prover.

3. The prover computes the following:

$$\begin{array}{ll} \epsilon_b = \epsilon - \eta \bmod 2^\lambda & \epsilon_{\bar{b}} = \eta \\ \rho_b = R + \epsilon_b \cdot r \bmod N & \rho_{\bar{b}} = \rho \\ \tau_b = T + \epsilon_b \cdot t \bmod N & \tau_{\bar{b}} = \tau \end{array}$$

   Then, it sends ϵ_0, and $\{\rho_i, \tau_i\}_{i \in \{0,1\}}$ to the verifier.

4. The verifier computes $\epsilon_1 = \epsilon - \epsilon_0 \bmod 2^\lambda$. It also checks if the following holds for $i \in \{0, 1\}$.

$$\begin{array}{l} g_1^{\rho_i} = U_i \cdot u^{\epsilon_i}, \\ h^{\rho_i} = E_i \cdot (e/(1+N)^i)^{\epsilon_i}, \\ (cd^\alpha)^{2\rho_i} = V_i \cdot v^{2\epsilon_i} \\ g_1^{\tau_i} = X_i \cdot (x_{1-i}/(1+N))^{\epsilon_i} \end{array}$$

Communication complexity. We will count a value in $[0, 2^\lambda)$ as one element in $\mathbb{Z}_N$. Consider the receiver message (x_0, x_1, Φ, π). Firstly, the part (x_0, x_1, Φ) amounts to $5 = 1 + 1 + 3$ elements in $\mathbb{Z}_{N^2}$. For the Sigma protocol for π, the first message has one elements in $\mathbb{Z}_{N^2}$, the second message has one element in $\{0,1\}^\lambda$, and the third message has one element in $\{0,1\}^\lambda$ and four elements in $\mathbb{Z}_N$, along with the decommitment with eight elements in $\mathbb{Z}_{N^2}$ and one element in $\mathbb{Z}_N$. Thus, communication complexity for the proof π amounts to $9 = 1 + 8$ elements in $\mathbb{Z}_{N^2}$ and $7 = 1 + (1 + 4 + 1))$ elements in $\mathbb{Z}_N$. The sender message (pk_0, Z_0, pk_1, Z_1) needs $(1, 1, 1, 1) = 4$ group elements in $\mathbb{Z}_{N^2}$. Therefore, the total communication complexity amounts to 18 elements in $\mathbb{Z}_{N^2}$ and 7 elements in $\mathbb{Z}_N$.

Adaptive security: realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. The NCE protocol messages can be overlapped with the OT protocol messages, and thus the final OT protocol runs in four rounds. We can use the variant of Camenisch-Shoup encryption scheme [CS03] described in [HK12] (we don't need the last checking element for semantic security) and the communication overhead amounts to 3 group elements in $\mathbb{Z}_{N^2}$ (the public key consists of one element excluding the generator in the CRS, and the ciphertext consists of two elements).

Acknowledgments.

We would like to thank the anonymous reviewers for pointing out the need to transmit the sender's messages over an adaptively secure channel, and for additional helpful feedback.

References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — Eurocrypt 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, 2001.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology — Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BH92] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology — Eurocrypt '92*, volume 658 of *LNCS*, pages 307–323. Springer, 1992.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001. Full version available as Cryptology ePrint Archive, Report 2000/067.

[Can07] Ran Canetti. Obtaining universally compoable security: Towards the bare bones of trust (invited talk). In Kaoru Kurosawa, editor, *Advances in Cryptology — Asiacrypt 2007*, volume 4833 of *LNCS*, pages 88–112. Springer, December 2007.

[CDMW09a] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 287–302. Springer, December 2009.

[CDMW09b] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, 2009.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, 2001.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648. ACM Press, May 1996.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *Journal of Cryptology*, 19(2):135–167, April 2006.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, May 2002.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, 2003.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology — Crypto '98*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, 2000.

[DDO$^+$01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 426–437. ACM Press, June 2003.

[DKMQ11] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *8th Theory of Cryptography Conference — TCC 2011*, volume 6597 of *LNCS*, pages 164–181. Springer, 2011.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, 2000.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology — Crypto 2002*, volume 2442 of *LNCS*, pages 581–596. Springer, 2002.

[DNO08] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. Essentially optimal universally composable oblivious transfer. In *11th Intl. Conf. on Information Security and Cryptology (ICISC)*, volume 5461 of *LNCS*, pages 318–335. Springer, 2008.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer, 2011.

[GIS$^+$10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *7th Theory of Cryptography Conference — TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, 2010.

[GMY04] Juan A. Garay, Philip MacKenzie, and Ke Yang. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *1st Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 297–316. Springer, February 2004.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology — Asiacrypt 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, December 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology — Crypto 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, 2009.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *Advances in Cryptology — Crypto 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, 2007.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, 2012.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer — efficiently. In David Wagner, editor, *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, 2008. Full version: `http://www.cs.uiuc.edu/∼mmp/pub/mpc-ot.pdfShare`.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *Advances in Cryptology — Eurocrypt 2007*, volume 4515 of *LNCS*, pages 97–114. Springer, 2007.

[Lin08] Andrew Y. Lindell. Efficient fully-simulatable oblivious transfer. In Tal Malkin, editor, *Cryptographers' Track — RSA 2008*, volume 4964 of *LNCS*, pages 52–70. Springer, April 2008.

[Lin09] Andrew Y. Lindell. Adaptively secure two-party computation with erasures. In *Cryptographers' Track — RSA 2009*, LNCS, pages 117–132. Springer, 2009.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In *Advances in Cryptology — Eurocrypt 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, 2011.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *8th Theory of Cryptography Conference — TCC 2011*, volume 6597 of *LNCS*, pages 329–346. Springer, 2011.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. ACM-SIAM, January 2001.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — Eurocrypt '99*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — Crypto '91*, volume 576 of *LNCS*, pages 129–140. Springer, 1992.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology — Crypto 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, 2008.

[Sco02] Mike Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.

[Sha07] Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

A Constructions Following the Approach of Garay et al.

Garay et al. [GWZ09] presented an elegant solution that addresses the drawbacks of [PVW08]. In the protocol, both parties execute a coin-tossing protocol (which could be realized using UC commitment), whose outcome is used as a CRS for the OT protocol of Peikert et al. [PVW08], which hereafter we denote by PVW.

A.1 DDH-Based Construction

We first consider the instantiation under the DDH assumption. The CRS in the PVW protocol is (g, h, c, d). We can put (g, h) in the CRS, and use coin flipping only to determine (c, d). This requires UC commitments to two group elements. We use Lindell's commitment scheme [Lin11].

Static case. We start with the protocol with static security.

CRS size is seven. The commitment scheme needs seven group elements for the CRS. We can reuse them for the CRS of the PVW protocol.

Round complexity is six. The commitment scheme needs one round for committing and four rounds for opening. However, the second message of the coin-tossing protocol can be overlapped with one of the four rounds in the opening. This implies that the coin-tossing stage needs five rounds. The PVW protocol needs two rounds, but the last round of the coin-tossing protocol and the first round of PVW protocol can also be overlapped. Overall, the round complexity is six.

Communication complexity is 38. The commitment and the proof part of the opening requires 14 group elements [FLM11, Table 1]. Recall that we need two random group elements from the coin-tossing protocol. Therefore, considering the committed messages themselves and the second move of the coin-tossing protocol, communication complexity of the coin tossing protocol is $32 = 2(14 + 1 + 1)$. The PVW protocol has communication complexity of 6. Overall, the communication complexity is 38.

Adaptive case. The CRS size remains the same as the static case. We consider the other measures.

Round complexity is eight. In the adaptive case, we don't know whether the protocol remains secure once we overlap the messages in the same round. Therefore, the round complexity is eight.

Communication complexity is 48. For the adaptive case, the commitment and the proof part of the opening requires 19 group elements [FLM11, Table 1]. This implies that the communication complexity increases by 10 elements.

Realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. The NCE protocol messages can be overlapped with the OT protocol messages. We can use ElGamal encryption, and the communication overhead amounts to 3 group elements (the public key consists of one element excluding the generator in the CRS, and the ciphertext consists of two elements).

A.2 DLIN-Based Construction

Now, we consider the instantiation under the DLIN assumption. The commitment of the Groth-Sahai proof can be adapted to the PVW protocol. We first recall the Groth-Sahai commitment.

CRS. The common reference string consists of three vectors $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3 \in \mathbb{G}^3$, where $\mathbf{g}_1 = (g_1, 1, g)$, $\mathbf{g}_2 = (1, g_2, g)$ for some $g_1, g_2 \in \mathbb{G}$. In the soundness mode, $\mathbf{g}_3$ is set as $\mathbf{g}_3 = \mathbf{g}_1^{\zeta_1} \cdot \mathbf{g}_2^{\zeta_2}$ with $\zeta_1, \zeta_2 \leftarrow \mathbb{Z}_p^*$. In the ZK mode, $\mathbf{g}_3$ is set as $\mathbf{g}_3 = \mathbf{g}_1^{\zeta_1} \cdot \mathbf{g}_2^{\zeta_2} \cdot (1, 1, g^u)$ with $\zeta_1, \zeta_2, u \leftarrow \mathbb{Z}_p^*$. Under the DLIN assumption, the two kinds of CRS are indistinguishable.

Commit to a message m. Committing to an exponent $m \in \mathbb{G}$ needs three group elements. In particular, the commitment is computed as $\mathbf{C} = (1, 1, m) \cdot \mathbf{g}_1^r \cdot \mathbf{g}_2^s \cdot \mathbf{g}_3^t$ with $r, s, t \leftarrow \mathbb{Z}_p^*$.

In the soundness mode, the commitment becomes $(g_1^{r+\zeta_1 t}, g_2^{s+\zeta_2 t}, g^{r+s+t(\zeta_1+\zeta_2)} \cdot m)$, and it is perfectly binding, and it can be decrypted using $a_1 = \log_g g_1, a_2 = \log_g g_2$ [BBS04]. In the ZK mode, it becomes $\mathbf{C} = (g_1^{r+\zeta_1 t}, g_2^{s+\zeta_2 t}, g^{r+s+t(\zeta_1+\zeta_2)} \cdot mg^{tu})$, and it is a perfectly hiding, since it is a random tuple for any $m \in \mathbb{G}$.

The PVW protocol. The PVW protocol can be instantiated as follows:

1. The receiver, with input choice bit b, computes $\mathbf{g}_3' = \mathbf{g}_1^{\xi_1} \cdot \mathbf{g}_2^{\xi_2}$ and sends $\mathbf{g} = \mathbf{g}_3' \cdot \mathbf{g}_3^b$.
2. The sender encrypts the message m_σ by using the Groth-Sahai commitment under CRS $(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}/\mathbf{g}_3^\sigma)$ for $\sigma = 0, 1$. This ensures that one commitment is perfectly binding and the other perfectly hiding.

The CRS in the PVW protocol is $(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$. We can put $(\mathbf{g}_1, \mathbf{g}_2)$ in the CRS, and use coin flipping only to determine $\mathbf{g}_3$. This requires UC commitments to three group elements. We use the commitment scheme of Fischlin et al.[FLM11] (hereafter FLM).

CRS size is 12. The FLM commitment scheme needs 12 group elements for the CRS. We can reuse them for the CRS of the PVW protocol.

Round complexity is four. The FLM scheme is non-interactive. Therefore, we need three rounds for the coin-tossing protocol. The PVW protocol needs two rounds, but the last round of the coin-tossing protocol and the first round of PVW protocol can be overlapped. Overall, the round complexity is four.

Communication complexity is 78. In FLM, the commitment and the proof part of the opening requires 21 group elements [FLM11, Table 1]. Recall that we need three random group elements from the coin-tossing protocol. Therefore, considering the committed messages themselves and the second move of the coin-tossing protocol, communication complexity of the coin tossing protocol is $69 = 3(21 + 1 + 1)$. The PVW protocol has communication complexity of 9. Overall, the communication complexity is 78.

Realizing an adaptively secure channel. Note that the non-committing encryption given in [BH92] runs in three rounds and needs one public key and one ciphertext of a semantically secure public key encryption scheme. We can use linear encryption [BBS04], and the communication overhead amounts to 5 group elements (the public key consists of two elements excluding the generator in the CRS, and the ciphertext consists of three elements).