Tighter Concrete Security for the Simplest OT

Iftach Haitner* Gil Segev†

Abstract

The Chou-Orlandi batch oblivious transfer (OT) protocol is a particularly attractive OT protocol that bridges the gap between practical efficiency and strong security guarantees and is especially notable due to its simplicity. The security analysis provided by Chou and Orlandi bases the security of their protocol on the hardness of the computational Diffie-Hellman (CDH) problem in prime-order groups. Concretely, in groups in which no better-than-generic algorithms are known for the CDH problem, their security analysis yields that an attacker running in time t and issuing q random-oracle queries breaks the security of their protocol with probability at most $\epsilon \leq q^2 \cdot t/2^{\kappa/2}$, where κ is the bit-length of the group's order. This concrete bound, however, is somewhat insufficient for 256-bit groups (e.g., for $\kappa = 256$, it does not provide any guarantee already for $t = 2^{48}$ and $q = 2^{40}$).

In this work, we establish a tighter concrete security bound for the Chou-Orlandi protocol. First, we introduce the *list square Diffie-Hellman* (ℓ-sqDH) problem and present a *tight* reduction from the security of the protocol to the hardness of solving ℓ-sqDH. That is, we completely shift the task of analyzing the concrete security of the protocol to that of analyzing the concrete hardness of the ℓ-sqDH problem. Second, we reduce the hardness of the ℓ-sqDH problem to that of the decisional Diffie-Hellman (DDH) problem without incurring a multiplicative loss. Our key observation is that although CDH and DDH have the same assumed concrete hardness, relying on the hardness of DDH enables our reduction to efficiently test the correctness of the solutions it produces.

Concretely, in groups in which no better-than-generic algorithms are known for the DDH problem, our analysis yields that an attacker running in time t and issuing $q \leq t$ random-oracle queries breaks the security of the Chou-Orlandi protocol with probability at most $\epsilon \leq t/2^{\kappa/2}$ (i.e., we eliminate the above multiplicative q^2 term). We prove our results within the standard real-vs-ideal framework considering static corruptions by malicious adversaries, and provide a concrete security treatment by accounting for the statistical distance between a real-model execution and an ideal-model execution.

*Stellar Development Foundation and Tel-Aviv University. Email: iftachh@tauex.tau.ac.il. Part of this work was conducted while at Coinbase.

†Hebrew University of Jerusalem and Coinbase. Email: segev@cs.huji.ac.il. Partially supported by the Israel Science Foundation (Grant No. 1336/22) and by the European Union (ERC, FTRC, 101043243). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction

The pursuit of oblivious transfer (OT) protocols [Rab81, EGL85] that provide both practical efficiency and strong security guarantees is a driving force in cryptographic research. Whereas early approaches to constructing OT protocols resulted in somewhat unsatisfying trade-offs between their efficiency and security, over the past three decades the cryptography community has made significant progress towards bridging the gap between practical efficiency and strong security guarantees (see, for example, [BM90, NP01, AIR01, Lin08, PVW08] and the many references therein).

The Chou-Orlandi batch OT protocol. A particularly attractive OT protocol bridging this gap was presented by Chou and Orlandi [CO15a, CO15b], and is especially notable due to its simplicity. Specifically, Chou and Orlandi constructed an exceedingly simple and practical batch OT protocol by cleverly manipulating the Diffie-Hellman key-exchange protocol. Although indications were provided for the protocol's lack of UC security (see [CO15b, Sec. 1.1]), the protocol is UC-secure when augmented with a non-interactive proof of knowledge for each party (as we discuss in Section 1.1, our results in this work provide along the way a formal proof of this standard observation). Specifically, for the sender, it suffices to provide a proof of knowledge for the discrete logarithm of a single group element, while for the receiver, it suffices to provide a proof of knowledge for the openings of Pedersen commitments.

Although the required proofs of knowledge can be derived from practical Σ-protocols which can be made non-interactive using the Fiat-Shamir transform [FS87], augmenting the protocol with these proofs may degrade its practical performance to some extent. However, it does not affect the *simplicity* of the protocol, especially when modeling the proofs of knowledge via ideal functionalities, as elegantly enabled by the modularity of the UC framework [Can01].

The concrete security of the Chou-Orlandi protocol. The security analysis provided by Chou and Orlandi bases the security of their protocol on the hardness of the computational Diffie-Hellman (CDH) problem. Therefore, in prime-order groups in which no better-than-generic algorithms are known for the CDH problem (e.g., in widely used 256-bit elliptic-curve groups such as Secp256k1 and P-256), the assumed concrete hardness of the CDH problem enables to derive a concrete security bound that depends on the running time of the attacker, the number of random-oracle queries issued by the attacker, and the order of the group.

Concretely, the analysis of Chou and Orlandi shows that any attacker A that runs in time t_{A}, issues q_{A} random-oracle queries, and breaks the security of their batch OT protocol with probability ϵ_{A}, can be transformed into an algorithm B that runs in time $t_{\mathsf{B}} \approx t_{\mathsf{A}}$ and solves the square Diffie-Hellman (sqDH) problem with probability $\epsilon_{\mathsf{B}} = \epsilon_{\mathsf{A}}/q_{\mathsf{A}}^2$.[1] The sqDH problem considers the task of receiving two group elements, G and $a \cdot G$, where G is a generator of a cyclic group of a κ-bit prime order q and $a \in \mathbb{Z}_q$ is uniformly sampled, and outputting the group element $a^2 \cdot G$. Bresson, Chevassut, and Checkpoint [BCP04] showed that solving the sqDH problem reduces to solving the CDH problem on two independent instances. Therefore, the algorithm B can be transformed into an algorithm C that runs in time $t_{\mathsf{C}} \approx t_{\mathsf{B}}$ and solves the CDH problem with probability $\epsilon_{\mathsf{C}} = \epsilon_{\mathsf{B}}^2$. Shoup's concrete hardness result for CDH problem [Sho97] provides us with the bound $\epsilon_{\mathsf{C}} \leq t_{\mathsf{C}}^2/2^{\kappa}$, which leads to the bound

$$\epsilon_{\mathsf{A}} \leq \frac{q_{\mathsf{A}}^2 \cdot t_{\mathsf{A}}}{2^{\kappa/2}} \tag{1.1}$$

[1] Chou and Orlandi proved that if an attacker A breaks the security of their protocol with probability ϵ_{A}, then A must have issued two random-oracle queries that can be used to solve the sqDH problem. Thus, guessing the indices of these two queries leads to the q_{A}^2 factor.

on the concrete security of the Chou-Orlandi protocol. For various realistic ranges of the attacker's running time t_{A} and number of random-oracle queries q_{A}, the bound stated in Eq. (1.1) is somewhat insufficient for 256-bit groups (e.g., for $\kappa = 256$, it does not provide any guarantee already for $t_{\mathsf{A}} = 2^{48}$ and $q_{\mathsf{A}} = 2^{40}$).

1.1 Our Contributions

In this work, we establish a tighter concrete security bound for the Chou-Orlandi protocol (when augmented with proofs of knowledge, as noted above). Our tighter analysis consists of the following two steps:

- First, we introduce the *list square Diffie-Hellman* (ℓ-sqDH) problem and present a *tight* reduction from the security of the protocol to the hardness of solving ℓ-sqDH. The ℓ-sqDH problem is a generalization of the sqDH problem, which considers the task of receiving two group elements, G and $a \cdot G$ (as in the sqDH problem), and outputting a list of at most ℓ group elements that must contain the group element $a^2 \cdot G$. The length ℓ of the list in our reduction is exactly the number of random-oracle queries q_{A} issued by the attacker of the protocol, and this already improves upon the reduction of Chou and Orlandi (i.e., even without our second step) by reducing the dependence on the number of such queries from q_{A}^2 to q_{A}. The fact that our reduction to the hardness of the ℓ-sqDH problem is tight enables us to completely shift the task of analyzing the concrete security of the protocol to that of analyzing the concrete hardness of the ℓ-sqDH problem.

- Second, we prove that the hardness of the ℓ-sqDH problem reduces to that of the decisional Diffie-Hellman (DDH) problem without incurring a *multiplicative* loss that depends on ℓ in either the success probability or the running time. By relying on the hardness of the DDH problem instead of the CDH problem, we only incur an *additive* loss of $\ell^2/2^\kappa$ in the success probability. Here, our key observation is that although the CDH and DDH problems have the exact same assumed concrete hardness [Sho97], relying on the DDH problem enables our reduction to *test* various candidates that it produces (using the idea underlying the above-mentioned reduction of Bresson, Chevassut, and Checkpoint [BCP04]) for the group element $a^2 \cdot G$. Equipped with this observation, we show that any algorithm B that runs in time t_{B} and solves the ℓ-sqDH problem with probability ϵ_{B} can be transformed into an algorithm C that runs in time $t_{\mathsf{C}} \approx t_{\mathsf{B}}$ and solves the DDH problem with probability $\epsilon_{\mathsf{C}} \geq \epsilon_{\mathsf{B}}^2 - \ell^2/2^\kappa$.

Putting together our two steps and relying on the assumed concrete hardness of the DDH problem [Sho97], we obtain the following bound on the security of the Chou-Orlandi protocol (which, for simplicity, we state here in an informal manner):

Theorem 1.1 (Informal)**.** *Let* A *be an attacker that runs in time* t_{A}*, issues* q_{A} *random-oracle queries, and breaks the security of the Chou-Orlandi protocol with probability* ϵ_{A} *in a* κ*-bit prime-order group. Then,*

$$\epsilon_{\mathsf{A}} \leq \sqrt{\frac{t_{\mathsf{A}}^2 + q_{\mathsf{A}}^2}{2^\kappa}} \tag{1.2}$$

Note that although we have included the additive $q_{\mathsf{A}}^2/2^\kappa$ term in the bound stated in Eq. (1.2) (to explicitly account for the minor security loss in reducing ℓ-sqDH to DDH), the number q_{A} of random-oracle queries issued by an attacker is clearly upper bounded by the attacker's running time t_{A}. Therefore, up to a multiplicative factor of $\sqrt{2}$, our bound implies the bound $t_{\mathsf{A}}/2^{\kappa/2}$, which should be compared to the bound $q_{\mathsf{A}}^2 \cdot t_{\mathsf{A}}/2^{\kappa/2}$ provided by the analysis of Chou and Orlandi.

We provide our analysis within the standard real-vs-ideal framework considering static corruptions by malicious adversaries. Within this framework, we provide a concrete security treatment by accounting for the statistical distance between a real-model execution with a given adversary and an ideal-model execution with a corresponding simulator (the simulator we construct is black-box and non-rewinding, which implies UC security [KLR10]). More specifically, we bound the statistical distance between an ideal-model execution and an execution in a hybrid model in which the underlying zero-knowledge functionalities are modeled via ideal functionalities. This guarantees that our concrete analysis focuses on the security of the protocol in a modular manner that distinguishes between any security loss that may result from the protocol itself and any such loss that may result from any particular choice for implementing the zero-knowledge functionalities.

1.2 Paper Organization

The remainder of this paper is organized as follows. First, in Section 2, we present some standard notation and basic cryptographic assumptions, as well as briefly provide the required background on the real-vs-ideal framework for two-party computation. In Section 3, we introduce the List Square Diffie-Hellman problem and analyze its concrete hardness. In Section 4, we present the augmented Chou-Orlandi protocol and analyze its concrete security.

2 Preliminaries

For any distribution X, we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for any set $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For any two distributions, X and Y, we denote by $\mathsf{SD}(X, Y)$ their statistical distance. A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for any polynomial $p(\cdot)$ there exists an integer N such that for all $\kappa > N$ it holds that $\nu(\kappa) \leq 1/p(\kappa)$. For any two distribution ensembles, $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$, we let $X \equiv^s Y$ denote the existence of a negligible function ν such that $\mathsf{SD}(X_\kappa, Y_\kappa) \leq \nu(\kappa)$ for all $\kappa \in \mathbb{N}$.

2.1 Groups and Hardness Assumptions

We rely on cryptographic hardness assumptions in prime-order groups, and model such groups as generated via a group-generation algorithm GGen. On input 1^κ, where $\kappa \in \mathbb{N}$ is the security parameter, such a group-generation algorithm outputs the description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime.

Relative to groups $\mathcal{G} = (\mathbb{G}, G, q)$ that are produced by such a group-generation algorithm, our starting point for computational hardness is based on the computational Diffie-Hellman (CDH), decisional Diffie-Hellman (DDH), and square Diffie-Hellman (sqDH) problems, defined as follows:

Definition 2.1. Let GGen be a group-generation algorithm. For $\kappa \in \mathbb{N}$ and algorithm A, let

$$\mathbf{Adv}^{\mathsf{CDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \stackrel{\mathsf{def}}{=} \Pr_{(a,b)\leftarrow\mathbb{Z}_q^2}\left[\mathsf{A}\left(\mathcal{G}, a\cdot G, b\cdot G\right) = ab\cdot G\right]$$

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \stackrel{\mathsf{def}}{=} \left|\Pr_{(a,b)\leftarrow\mathbb{Z}_q^2}\left[\mathsf{A}\left(\mathcal{G}, a\cdot G, b\cdot G, ab\cdot G\right) = 1\right] - \Pr_{(a,b,c)\leftarrow\mathbb{Z}_q^3}\left[\mathsf{A}\left(\mathcal{G}, a\cdot G, b\cdot G, c\cdot G\right) = 1\right]\right|$$

$$\mathbf{Adv}^{\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \stackrel{\mathsf{def}}{=} \Pr_{a\leftarrow\mathbb{Z}_q}\left[\mathsf{A}\left(\mathcal{G}, a\cdot G\right) = a^2\cdot G\right]$$

where the probability is additionally taken over the choice of $\mathcal{G} = (\mathbb{G}, G, q) \leftarrow \mathsf{GGen}(1^\kappa)$ and over A's internal randomness.

Definition 2.2. For $\mathsf{X} \in \{\mathsf{CDH}, \mathsf{DDH}, \mathsf{sqDH}\}$, we say that the X *problem is hard with respect to* GGen if for any probabilistic polynomial-time non-uniform algorithm A there exists a negligible function ν such that for all $\kappa \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\mathsf{X}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \nu(\kappa) \ .$$

In addition, for any functions $t = t(\kappa)$ and $\epsilon = \epsilon(\kappa)$, we say that the X *problem is* (t, ϵ)*-hard with respect to* GGen if for any non-uniform algorithm A that runs in time t and for all $\kappa \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\mathsf{X}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \epsilon(\kappa) \ .$$

Concrete security baseline: Generic-group CDH and DDH hardness. As our baseline for analyzing the concrete security guarantees of our protocol, we rely on the classic generic-group hardness of the CDH and DDH problems (which is identical to that of the discrete logarithm problem) [Sho97, Mau05]. Specifically, following standard practice in groups where no better-than-generic algorithms are currently known for these problems, we rely on the following assumption with respect to some underlying group-generation algorithm:

Assumption 2.3 (Concrete hardness of CDH and DDH)**.** For each $\mathsf{X} \in \{\mathsf{CDH}, \mathsf{DDH}\}$ and for any $t = t(\kappa)$, the X*-problem is* (t, ϵ)*-hard with respect to an underlying* GGen, where $\epsilon(\kappa) = t^2(\kappa)/2^\kappa$.

Although an analogous assumption can be made for the concrete hardness of the sqDH problem, the sqDH problem is somewhat less studied compared to the extensively explored CDH and DDH problems. To minimize the extent to which its concrete hardness is dependent on idealized models, Bresson, Chevassut, and Checkpoint [BCP04] proved the following lemma, which reduces the task of solving the sqDH problem to that of solving the CDH problem on two independent instances:

Lemma 2.4 ([BCP04])**.** *Let* GGen *be a group-generation algorithm. For any* sqDH*-algorithm* A *that runs in time* $t_\mathsf{A} = t_\mathsf{A}(\kappa)$ *there exists a* CDH*-algorithm* B *that runs in time* $t_\mathsf{B}(\kappa) \leq 2t_\mathsf{A}(\kappa) + 5\tau_{\mathsf{mult}}(\kappa) + 2\tau_{\mathsf{add}}(\kappa) + 3\tau_{\mathsf{inv}}(\kappa)$ *such that*

$$\mathbf{Adv}^{\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \sqrt{\mathbf{Adv}^{\mathsf{CDH}}_{\mathsf{GGen},\mathsf{B}}(\kappa)},$$

for all $\kappa \in \mathbb{N}$*, where* $\tau_{\mathsf{mult}}(\kappa)$*,* $\tau_{\mathsf{add}}(\kappa)$ *and* $\tau_{\mathsf{inv}}(\kappa)$ *denote upper bounds on the running times required for computing the group multiplication, addition and inversion operations, respectively, in groups produced by* $\mathsf{GGen}(1^\kappa)$*.*

2.2 Secure Two-Party Computation

We consider the standard real-vs-ideal framework for static corruptions by malicious adversaries. In what follows, for any function $g(x_1, x_2)$ we denote by $\mathcal{F}_g$ the functionality that computes $g(x_1, x_2)$ when given inputs x_1 and x_2 from P_1 and P_2, respectively. We now formally define executions in the real model, the ideal model, and the hybrid model for a protocol Π and functionality $\mathcal{F}_g$.

Execution in the real model. When executing a two-party protocol Π in the real model, the real-model adversary A first receives the input of the corrupted party and an arbitrary auxiliary input aux. Then, the adversary takes the role of the corrupted party by sending all messages on their behalf using an arbitrary polynomial-time strategy, whereas the honest party follows the instructions of the protocol using their prescribed input. We denote by $\mathrm{REAL}_{\Pi,\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)$ the joint distribution of the random variables corresponding to the output of the adversary A and the output of the honest party in an execution of Π in the real model, where P_1 holds input $(1^\kappa, x_1)$ and P_2 holds input $(1^\kappa, x_2)$.

Execution in the ideal model. When computing a functionality $\mathcal{F}_g$ in the ideal model, the ideal-model adversary Sim first receives the input of the corrupted party and an arbitrary auxiliary input aux. Then, both parties send inputs to a trusted party, where the honest party sends their prescribed input, and the corrupted party controlled by Sim may send any value of their choice. We denote by (x_1, x_2) the prescribed inputs of the two parties, and by (x'_1, x'_2) the inputs sent to the trusted party (we assume that if one of these inputs is invalid then the trusted party substitutes it with some default input).

In turn, the trusted party computes $(y_1, y_2) = g(x'_1, x'_2)$, and sends ideal-model adversary Sim the output y_i of the corrupted party P_i. If the adversary responds with $\mathsf{continue}$, then the trusted party sends y_{3-i} to the honest party P_{3-i} who outputs y_{3-i}. Otherwise, if the adversary responds with abort, then the trusted party sends abort to the honest party P_{3-i} who outputs abort. We denote by $\mathrm{IDEAL}_{\mathcal{F}_g,\mathsf{Sim}(\mathsf{aux})}(x_1, x_2, \kappa)$ the joint distribution of the random variables corresponding to the output of the adversary Sim and the output of the honest party in an ideal-model computation of $\mathcal{F}_g$, where P_1 holds input $(1^\kappa, x_1)$ and P_2 holds input $(1^\kappa, x_2)$.

Execution in the $\mathcal{F}$-hybrid model. The above formulation of executing a protocol Π in the real model extends to that of executing a protocol Π in the $\mathcal{F}$-hybrid model, where both parties may access a trusted party that computes $\mathcal{F}$ with abort as in the ideal model. That is, whenever either one of the parties or both parties send inputs to the trusted party computing $\mathcal{F}$, the trusted party first sends the output of the corrupted party to the adversary. Then, the adversary responds with either $\mathsf{continue}$ or abort, instructing the trusted party to send the honest party either their output or abort, respectively. We denote by $\mathrm{HYBRID}^{\mathcal{F}}_{\Pi,\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)$ the joint distribution of the random variables corresponding to the output of the adversary A and the output of the honest party in an execution of Π in the $\mathcal{F}$-hybrid model, where P_1 holds input $(1^\kappa, x_1)$ and P_2 holds input $(1^\kappa, x_2)$.

Definition 2.5. Let Π, $\mathcal{F}$ and g be as above. Then, Π is said to securely compute $\mathcal{F}_g$ with abort if for any $i \in \{1, 2\}$ and for every probabilistic polynomial-time adversary A in the $\mathcal{F}$-hybrid model corrupting the party P_i, there exists a probabilistic polynomial-time adversary Sim in the ideal model corrupting the same party P_i such that for any $x_1, x_2, \mathsf{aux} \in \{0,1\}^*$ it holds that

$$\left\{\mathrm{HYBRID}^{\mathcal{F}}_{\Pi,\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)\right\} \equiv^s \left\{\mathrm{IDEAL}_{\mathcal{F}_g,\mathsf{Sim}(\mathsf{aux})}(x_1, x_2, \kappa)\right\}$$

UC security. The proof security we present in Section 4 provides an ideal-model adversary S that simulates the view of any adversary A in the hybrid model without rewinding it. Therefore, if the ideal functionalities in the hybrid model are instantiated with UC-secure protocols [Can01], then the entire protocol is UC-secure as proven by Kushilevitz, Lindell and Rabin [KLR10].[2]

[2]For computationally-secure protocols, Kushilevitz, Lindell and Rabin [KLR10] require input availability (or start synchronization), which is satisfied in our setting since we consider static corruptions in a two-party setting.

2.3 Ideal Functionalities

We now present the (batch) oblivious transfer functionality that we prove to be realized by the augmented Chou-Orlandi protocol, as well as the ideal functionalities on which the protocol relies within the hybrid model:

- $\mathcal{F}_{\mathsf{OT}}$**.** The OT functionality with batch parameter $n \in \mathbb{N}$ is defined as follows:
  - Upon receiving $(\mathsf{sender}, \{(m_{j,0}, m_{j,1})\}_{j\in[n]})$ from party P_i and $(\mathsf{receiver}, \boldsymbol{c})$ from party P_{3-i}, where $i \in \{1,2\}$ and $\boldsymbol{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$, send $(\mathsf{receive}, \{m_{j,c_j}\}_{j\in[n]})$ to party P_{3-i}.
- $\mathcal{F}_{\mathsf{GGen}}$**.** The group-generation functionality is defined as follows:
  - Upon receiving $(\mathsf{generate}, 1^\kappa)$ from both parties, produce a description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime, and send $(\mathsf{group}, \mathcal{G})$ to both parties.
- $\mathcal{F}_{\mathsf{RO}}$**.** The random-oracle functionality with output-length parameter $n \in \mathbb{N}$ is defined as follows:
  - Upon receiving $(\mathsf{evaluate}, x)$ from party P_i, where $i \in \{1,2\}$, if a pair (x, y) has been previously stored then send (output, x, y) to P_i. If a tuple (x, y) has not been previously stored, uniformly sample $y \leftarrow \{0,1\}^n$, store the pair (x, y), and send (output, x, y) to P_i.
- $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$ **and** $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$**.** The ZK-DL and (batched) ZK-PED functionalities over a group $\mathcal{G} = (\mathbb{G}, G, q)$ are defined as the following standard zero-knowledge functionality for the relations $\mathcal{R}_{\mathsf{DL}} = \{((G, A), a) \in \mathbb{G}^2 \times \mathbb{Z}_q \mid A = a \cdot G\}$ and $\mathcal{R}_{\mathsf{PED}} = \{((G, A, \{C_i\}_{i\in[n]}), \{(c_i, r_i)\}_{i\in[n]}) \in \mathbb{G}^{n+2} \times \mathbb{Z}_q^{2n} \mid C_i = c_i \cdot A + r_i \cdot G \ \forall i \in [n]\}$, respectively:
  - Upon receiving (proof, x, w) from party P_i, where $i \in \{1,2\}$, if $(x, w) \notin \mathcal{R}$ then send (reject, x) to party P_{3-i}, and otherwise send $(\mathsf{verified}, x)$ to party P_{3-i}.

3 The List-sqDH Problem and its Concrete Hardness

The concrete security guarantees we provide in this work rely on the following assumption, which is parameterized by a function $\ell = \ell(\kappa)$ of the security parameter $\kappa \in \mathbb{N}$. In what follows, recall that GGen is a group-generation algorithm that on input 1^κ produces a description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime.

Definition 3.1 (List-square Diffie-Hellman (ℓ-sqDH))**.** For any $\kappa \in \mathbb{N}$, function $\ell = \ell(\kappa)$, and algorithm A, let

$$\mathbf{Adv}^{\ell\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{A}\left(\mathcal{G}, a \cdot G\right) = (G_1, \ldots, G_\ell) \wedge \exists i \in [\ell] \text{ s.t. } G_i = a^2 \cdot G\right] ,$$

where the probability is taken over the choices of $\mathcal{G} = (\mathbb{G}, G, q) \leftarrow \mathsf{GGen}(1^\kappa)$ and $a \leftarrow \mathbb{Z}_q$, and over A's internal randomness.

Definition 3.2. We say that the ℓ-sqDH *problem is hard with respect to* GGen if for any probabilistic polynomial-time non-uniform algorithm A there exists a negligible function ν such that for all $\kappa \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\ell\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \nu(\kappa) .$$

In addition, for any functions $t = t(\kappa)$ and $\epsilon = \epsilon(\kappa)$, we say that the ℓ-sqDH *problem is* (t, ϵ)*-hard with respect to* GGen if for any non-uniform algorithm A that runs in time t and for all $\kappa \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\ell\text{-sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \epsilon(\kappa) \ .$$

Note that, for $\ell = 1$, the 1-sqDH problem is identical to the sqDH problem, and as ℓ increases, then algorithms for solving the ℓ-sqDH problem are allowed to output more potential candidates for $a^2 \cdot G$. Nevertheless, for any $\ell \in \mathbb{N}$, any algorithm A for solving the ℓ-sqDH problem can be transformed into an algorithm B for solving the sqDH problem by choosing uniformly one the ℓ outputs produced by A and outputting it. The running time t_{B} of the algorithm B is essentially identical to the running time t_{A} of the algorithm A, and for any $\kappa \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\ell\text{-sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \ell \cdot \mathbf{Adv}^{\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{B}}(\kappa) \ .$$

For large values of ℓ, such as the number q_{A} of random-oracle queries issued by the algorithm A (as obtained by our security proof in Section 4), this leads to a rather loose concrete security guarantee. Specifically, when relying on the assumed generic hardness of the CDH problem as our baseline (see Assumption 2.3), Lemma 2.4 of Bresson, Chevassut and Pointcheval [BCP04] yields the bound

$$\mathbf{Adv}^{\ell\text{-sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \ell \cdot \sqrt{\frac{(t_{\mathsf{A}})^2}{2^\kappa}} = \frac{\ell \cdot t_{\mathsf{A}}}{2^{\kappa/2}} \ .$$

In the following lemma, we present a more direct approach for establishing concrete security guarantees for the ℓ-sqDH problem by reducing its security directly to that of the DDH problem (instead of reducing its security to the CDH problem via the sqDH problem). This enables us to obtain a concrete security bound that is essentially identical to that of the sqDH problem by relying on the assumed generic hardness of the DDH problem as our baseline. This direct approach provides a much stronger concrete security guarantee for the ℓ-sqDH problem already for 256-bit groups.

As a technical ingredient, our proof relies on the ability to decide, as fast as possible, whether two sets have a non-empty intersection. Each set consists of ℓ strings of length κ bits (representing group elements), and the running time required for this task affects the tightness of our statement. For stating our lemma without restricting ourselves to a particular implementation, we denote by $t_{\mathsf{SI}}(\ell, \kappa)$ an upper bound on the time required for deciding whether such two sets have a non-empty intersection.

Lemma 3.3. *Let $\ell = \ell(\kappa)$ be a function of the security parameter $\kappa \in \mathbb{N}$. For any ℓ-sqDH algorithm A that runs in time $t_{\mathsf{A}} = t_{\mathsf{A}}(\kappa)$ there exists a* DDH *algorithm* D *that runs in time $t_{\mathsf{D}} \leq 2t_{\mathsf{A}} + t_{\mathsf{SI}}(\ell, \kappa) + (\ell + 4) \cdot \tau_{\mathsf{add}}(\kappa) + \tau_{\mathsf{inv}}(\kappa)$ such that*

$$\mathbf{Adv}^{\ell\text{-sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa) \leq \sqrt{\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GGen},\mathsf{D}}(\kappa) + \frac{\ell^2}{2^\kappa}}$$

for every $\kappa \in \mathbb{N}$, where $\tau_{\mathsf{add}}(\kappa)$ and $\tau_{\mathsf{inv}}(\kappa)$ denote the running times required for computing the group addition and inversion operations, respectively, in groups produced by $\mathsf{GGen}(1^\kappa)$.

We note that it is quite simple to avoid a quadratic overhead of $t_{\mathsf{SI}} = \ell^2$ (which will not yield any concrete security improvement) by sorting the two lists in time $O(\ell \log \ell)$. More subtle approaches may be used to reduce this overhead to $O(\ell)$. For example, given a dictionary data structure that supports (with high probability) a linear time initialization and constant-time lookups in the unit-cost RAM model (see, for example, [Pag00, PR04, ANS10] and the many references therein), one can initialize the dictionary to hold the ℓ strings of the first set, then perform a lookup for

each string of the second set (for simplicity, we ignore the polynomially-small error probability that such a dictionary may introduce). Accounting for group addition at unit cost, we then obtain $t_{\mathsf{D}} \leq 2t_{\mathsf{A}} + O(\ell)$, and therefore the concrete security bound provided by Lemma 3.3 for the ℓ-sqDH problem is equivalent to the bound $t_{\mathsf{A}}/2^{\kappa/2}$ provided by Lemma 2.4 for the sqDH problem.

Proof of Lemma 3.3. Let $\ell = \ell(\kappa)$ be a function of the security parameter $\kappa \in \mathbb{N}$, and let A be an algorithm for the ℓ-sqDH problem. Let D be the following DDH algorithm:

1. On input $(\mathcal{G}, H_a, H_b, W)$, where $\mathcal{G} = (\mathbb{G}, G, q)$ and $H_a, H_b, W \in \mathbb{G}$, compute
$$(G_1, \ldots, G_\ell) \leftarrow \mathsf{A}(\mathcal{G}, H_a + H_b)$$
$$(G'_1, \ldots, G'_\ell) \leftarrow \mathsf{A}(\mathcal{G}, H_a - H_b)$$
2. Let $L = \{G_1, \ldots, G_\ell\}$ and $L' = \{4 \cdot W + G'_1, \ldots, 4 \cdot W + G'_\ell\}$.
3. If $L \cap L' \neq \emptyset$ then output 1, and otherwise output 0.

For analyzing D's success probability, note that if the group element W is uniformly distributed and independent of H_a and H_b, then it is also independent of the inputs, $H_a + H_b$ and $H_a - H_b$, provided to A. Thus, it is also independent of the outputs, $(G_1, \ldots, G_\ell)$ and $(G'_1, \ldots, G'_\ell)$, produced by A. Therefore, when considering the case in which $H_a = a \cdot G$, $H_b = b \cdot G$ and $W = c \cdot G$ for uniformly and independently sampled $a, b, c \leftarrow \mathbb{Z}_q$, it holds that

$$\begin{aligned}
\Pr\left[\mathsf{D}(\mathcal{G}, H_a, H_b, W) = 1\right] &= \Pr\left[\exists i, j \in [\ell] \text{ s.t. } G_i = 4 \cdot W + G'_j\right] \\
&\leq \sum_{i,j \in [\ell]} \Pr\left[4 \cdot W = G_i - G'_j\right] \\
&= \frac{\ell^2}{q} \\
&\leq \frac{\ell^2}{2^\kappa}. \qquad (3.1)
\end{aligned}$$

For considering the case in which $H_a = a \cdot G$, $H_b = b \cdot G$ and $W = ab \cdot G$ for uniformly and independently sampled $a, b \leftarrow \mathbb{Z}_q$, we denote by $\mathsf{Success}$ and $\mathsf{Success}'$ the events in which A is successful in their first and second execution, respectively. Since the transformation $(a, b) \to (a + b, a - b)$ is invertible over $\mathbb{Z}_q$, the inputs $H_a + H_b = (a + b) \cdot G$ and $H_a - H_b = (a - b) \cdot G$ are similarly uniformly and independently distributed, and therefore the events $\mathsf{Success}$ and $\mathsf{Success}'$ are independent. Conditioned on these two events, there exist $i, j \in [\ell]$ such that $G_i = (a + b)^2 \cdot G$ and $G'_j = (a - b)^2 \cdot G$, and therefore

$$G_i - G'_j = 4ab \cdot G = 4 \cdot W.$$

Therefore, conditioned on the events $\mathsf{Success}$ and $\mathsf{Success}'$, the algorithm D always outputs 1, and we have

$$\begin{aligned}
\Pr\left[\mathsf{D}(\mathcal{G}, a \cdot G, b \cdot G, ab \cdot G) = 1\right] &\geq \Pr\left[\mathsf{Success} \wedge \mathsf{Success}'\right] \\
&= \left(\Pr\left[\mathsf{Success}\right]\right)^2 \\
&= \left(\mathbf{Adv}^{\ell\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa)\right)^2. \qquad (3.2)
\end{aligned}$$

Putting together Eq. (3.1) and (3.2) we obtain

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GGen},\mathsf{D}}(\kappa) &= \left|\left[\mathsf{D}(\mathcal{G}, a \cdot G, b \cdot G, ab \cdot G) = 1\right] - \left[D(\mathcal{G}, a \cdot G, b \cdot G, c \cdot G) = 1\right]\right| \\
&\geq \left(\mathbf{Adv}^{\ell\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{A}}(\kappa)\right)^2 - \frac{\ell^2}{2^\kappa},
\end{aligned}$$

where $a, b, c \leftarrow \mathbb{Z}_q$, as required. Finally, in terms of D's running time, note that D invokes A twice, performs $\ell+4$ additions of group elements and a single inversion of a group element, and determines whether $L \cap L' \neq \emptyset$. Thus, $t_{\mathsf{D}} \leq 2t_{\mathsf{A}} + t_{\mathsf{SI}}(\ell, \kappa) + (\ell+4) \cdot \tau_{\mathsf{add}} + \tau_{\mathsf{inv}}$. ■

4 The Augmented Oblivious Transfer Protocol

In this section we present the augmented Chou-Orlandi batch oblivious transfer protocol and analyze its security. As discussed in Section 1.1, we provide our analysis within the standard real-vs-ideal framework considering static corruptions by malicious adversaries. In this framework, we consider a hybrid model in which the underlying zero-knowledge functionalities are modeled via ideal functionalities. This guarantees that our concrete analysis focuses on the security of the protocol in a modular manner that distinguishes between any security loss that may result from the protocol itself and any such loss that may result from any particular choice for implementing the zero-knowledge functionalities. In addition to the zero-knowledge functionalities, the protocol relies on ideal functionalities for modeling a random oracle and a group-generation algorithm (see Section 2.3 for formal descriptions of the ideal functionalities).

The protocol, described in Figure 1, is parameterized by the security parameter $\kappa \in \mathbb{N}$, and by a "batch parameter" $n = n(\kappa)$ that determines the number of oblivious transfer instances whose executions are batched together. For simplicity, we assume that the sender's input strings are of length κ bits.

Correctness. As shown by Chou and Orlandi [CO15a], for any inputs $\{(m_{i,0}, m_{i,1})\}_{i\in[n]}$ and $\boldsymbol{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$, and for any $i \in [n]$, the correctness of the protocol is established by distinguishing between the cases $c_i = 0$ and $c_i = 1$:

- Case I: $c_i = 0$. In this case, the receiver computes $C_i = r_i \cdot G$ and the sender computes $D_{i,0} = a \cdot C_i = (a \cdot r_i) \cdot G$. Then, the receiver recovers the exact same $D_{i,0}$ by computing $r_i \cdot A = r_i \cdot (a \cdot G)$, and this enables to retrieve $m_{i,0}$.
- Case II: $c_i = 1$. In this case, the receiver computes $C_i = A + r_i \cdot G = (a + r_i) \cdot G$ and the sender computes $D_{i,1} = a \cdot C_i - T = (a \cdot (a + r_i) - a^2) \cdot G = (a \cdot r_i) \cdot G$. Then, the receiver recovers the exact same $D_{i,1}$ by computing $r_i \cdot A = r_i \cdot (a \cdot G)$, and this enables to retrieve $m_{i,1}$.

Security. The following lemma captures the concrete security guarantees of the protocol Π_{OT}:

Lemma 4.1. *Let $\mathcal{F} = (\mathcal{F}_{\mathsf{GGen}}, \mathcal{F}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{ZK\text{-}DL}}, \mathcal{F}_{\mathsf{ZK\text{-}PED}})$, let $t_{\mathsf{A}} = t_{\mathsf{A}}(\kappa)$ and $q_{\mathsf{A}} = q_{\mathsf{A}}(\kappa)$ be functions of the security parameter $\kappa \in \mathbb{N}$, and let A be an adversary in the $\mathcal{F}$-hybrid model that runs in time t_{A} and issues at most q_{A} queries to the random-oracle functionality. Then, there exist a non-rewinding ideal-model adversary Sim that runs in time $t_{\mathsf{Sim}} = O(t_{\mathsf{A}})$ and a non-uniform algorithm B for solving the q_{A}-sqDH problem that runs in time $t_{\mathsf{B}} = O(t_{\mathsf{A}})$ for which*

$$\mathsf{SD}\left(\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa), \mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}(\mathsf{aux})}(x_1, x_2, \kappa)\right) \leq \mathbf{Adv}^{q_{\mathsf{A}}\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{B}}(\kappa)$$

for any security parameter $\kappa \in \mathbb{N}$, input $(x_1, x_2) = \left(\{(m_{i,0}, m_{i,1})\}_{i\in[n]}, \boldsymbol{c}\right)$, and auxiliary information $\mathsf{aux} \in \{0,1\}^*$.

We note that the non-uniformity of the q_{A}-sqDH algorithm B is used for providing it with a triplet (x_1, x_2, aux) that essentially maximizes the statistical distance between the hybrid and ideal

The Batch Oblivious Transfer Protocol Π_{OT}

Parameters: Security parameter $\kappa \in \mathbb{N}$, batch parameter $n \in \mathbb{N}$.

Ideal functionalities: $\mathcal{F}_{\mathsf{GGen}}$, $\mathcal{F}_{\mathsf{RO}} = H : \{0,1\}^* \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$ and $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$.

Parties and inputs:

- Sender S: $\{(m_{i,0}, m_{i,1})\}_{i\in[n]}$, where $m_{i,b} \in \{0,1\}^\kappa$ for every $i \in [n]$ and $b \in \{0,1\}$.
- Receiver R: $\boldsymbol{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$.

Execution:

1. The parties jointly call $\mathcal{F}_{\mathsf{GGen}}$ on input 1^κ, and obtain a description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime.
2. S: Sample $a \leftarrow \mathbb{Z}_q$, send $A = a \cdot G$ to R, and send $(\mathsf{proof}, (G, A), a)$ to $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$.
3. R: Upon receiving A from S and $(\mathsf{verified}, (G, A))$ from $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$, sample $\boldsymbol{r} = (r_1, \ldots, r_n) \leftarrow \mathbb{Z}_q^n$, and for every $i \in [n]$ compute $C_i = c_i \cdot A + r_i \cdot G$. Then, send $(\mathsf{proof}, (G, A, C_1, \ldots, C_n), (\boldsymbol{c}, \boldsymbol{r}))$ to $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$.
4. S: Upon receiving $(\mathsf{verified}, (G, A, C_1, \ldots, C_n))$ from $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$, set $T = a^2 \cdot G$, and for every $i \in [n]$ compute

$$\begin{aligned} D_{i,0} &= a \cdot C_i \\ D_{i,1} &= D_{i,0} - T \\ k_{i,b} &= H(i, D_{i,b}) \text{ for each } b \in \{0,1\} \\ w_{i,b} &= k_{i,b} \oplus m_{i,b} \text{ for each } b \in \{0,1\}. \end{aligned}$$

   Then, send $\{(w_{i,0}, w_{i,1})\}_{i\in[n]}$ to R.
5. R: Upon receiving $\{(w_{i,0}, w_{i,1})\}_{i\in[n]}$ from S, for every $i \in [n]$ compute

$$\begin{aligned} D_{i,c_i} &= r_i \cdot A \\ k_{i,c_i} &= H(i, D_{i,c_i}) \\ m_{i,c_i} &= k_{i,c_i} \oplus w_{i,c_i}, \end{aligned}$$

   and output $(m_{1,c_1}, \ldots, m_{n,c_n})$.

Figure 1: The Batch Oblivious Transfer Protocol Π_{OT}.

executions for our ideal-model adversary Sim for every $\kappa \in \mathbb{N}$ (see the proof below). This form of non-uniformity is not essential, and can be avoided by using the above protocol to implement a random-sender OT, in which the sender randomly samples the input $\{(m_{i,0}, m_{i,1})\}_{i\in[n]}$ instead of being provided with a prescribed input. Then, a full-fledged OT protocol is easily derived from a random-sender one using the standard technique of masking the sender's actual input with the randomly-sampled one (this does not affect the concrete security of the protocol).

Equipped with Lemma 4.1, the following corollary now follows directly by combining it with Lemma 3.3:

Corollary 4.2. *Based on Assumption 2.3, for any adversary* A *in the* $\mathcal{F}$*-hybrid model that runs in time* t_{A} *and issues at most* q_{A} *queries to the random-oracle functionality there exists a non-rewinding ideal-model adversary* Sim *that runs in time* $t_{\mathsf{Sim}} = O(t_{\mathsf{A}})$ *for which*

$$\mathsf{SD}\left(\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1,x_2,\kappa), \mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}(\mathsf{aux})}(x_1,x_2,\kappa)\right) \leq \sqrt{\frac{t^2_{\mathsf{A}}(\kappa)+q^2_{\mathsf{A}}(\kappa)}{2^\kappa}}$$

for any security parameter $\kappa \in \mathbb{N}$*, input* $(x_1,x_2) = \left(\{(m_{i,0},m_{i,0})\}_{i\in[n]}, \boldsymbol{c}\right)$*, and auxiliary information* $\mathsf{aux} \in \{0,1\}^*$*.*

In the remainder of this section, we provide the proof of Lemma 4.1.

Proof of Lemma 4.1. We separately consider the case where the sender S is corrupted and the case where the receiver R is corrupted.

Case I: S is corrupted. Let A be a probabilistic polynomial-time adversary corrupting S in the $\mathcal{F}$-hybrid model, where $\mathcal{F} = (\mathcal{F}_{\mathsf{GGen}}, \mathcal{F}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{ZK\text{-}DL}}, \mathcal{F}_{\mathsf{ZK\text{-}PED}})$, and denote the inputs held by the parties in the ideal model by $(x_1,x_2) = \left(\{(m_{i,0},m_{i,1})\}_{i\in[n]}, \boldsymbol{c}\right)$. We construct a probabilistic polynomial-time ideal-model adversary Sim as follows:

1. On input $(\{(m_{i,0},m_{i,0})\}_{i\in[n]}, \mathsf{aux})$, perfectly simulate the group-generation functionality $\mathcal{F}_{\mathsf{GGen}}$ by producing a description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime.

2. Invoke A on the input $(\{(m_{i,0},m_{i,0})\}_{i\in[n]}, \mathsf{aux})$ and the description $\mathcal{G}$ of the cyclic group. Throughout the execution of A, whenever A issues a random-oracle query x, perfectly simulate the random-oracle functionality $\mathcal{F}_{\mathsf{RO}}$: If the query x has been previously made then retrieve the stored value $H(x)$ and respond with $(\mathsf{output}, x, H(x))$. Otherwise, uniformly sample $H(x) \leftarrow \{0,1\}^\kappa$, store the pair $(x, H(x))$, and respond with $(\mathsf{output}, x, H(x))$.

3. Obtain the messages A and $(\mathsf{proof}, (G,A), a)$ sent from A to R and $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$, respectively. If $A \neq a \cdot G$, then submit some fixed input on behalf of S to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$, respond to $\mathcal{F}_{\mathsf{OT}}$ with abort, and output A's output. Otherwise (i.e., $A = a \cdot G$), continue to the next step.

4. Sample $\boldsymbol{r} = (r_1,\ldots,r_n) \leftarrow \mathbb{Z}_q^n$, and for every $i \in [n]$ compute $C_i = r_i \cdot G$. Then, send $(\mathsf{verified}, (G, A, C_1, \ldots, C_n))$ to A on behalf of $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$, and obtain the messages $\{(w_{i,0}, w_{i,1})\}_{i\in[n]}$ sent from A to R.

5. Set $T = a^2 \cdot G$, and for every $i \in [n]$ compute

$$\begin{aligned} D_{i,0} &= a \cdot C_i \\ D_{i,1} &= D_{i,0} - T \\ k_{i,b} &= H\left(i, D_{i,b}\right) \text{ for each } b \in \{0,1\} \\ m'_{i,b} &= k_{i,b} \oplus w_{i,b} \text{ for each } b \in \{0,1\}. \end{aligned}$$

6. Submit the input $\left\{\left(m'_{i,0}, m'_{i,1}\right)\right\}_{i\in[n]}$ on behalf of S to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$, respond to $\mathcal{F}_{\mathsf{OT}}$ with $\mathsf{continue}$, and output A's output.

Provided with the above description of Sim, the perfect hiding property of the Pedersen commitments $C_1, \ldots, C_n$ (together with our above analysis of the correctness of the protocol), guarantee that the two distributions $\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{S}(\mathsf{aux})}(x_1, x_2, \kappa)$ are *identical*. Specifically, the only difference is that Sim computes $C_i = r_i \cdot G$ instead of $C_i = c_i \cdot A + r_i \cdot G$ for every $i \in [n]$. However, the resulting distributions are identical since each $r_i \in \mathbb{Z}_q$ is sampled uniformly.

Case II: R is corrupted. Let A be a probabilistic polynomial-time adversary corrupting R in the $\mathcal{F}$-hybrid model, where $\mathcal{F} = (\mathcal{F}_{\mathsf{GGen}}, \mathcal{F}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{ZK\text{-}DL}}, \mathcal{F}_{\mathsf{ZK\text{-}PED}})$, and denote the inputs held by the parties in the ideal model by $(x_1, x_2) = \left(\{(m_{i,0}, m_{i,0})\}_{i\in[n]}, \boldsymbol{c}\right)$. We construct a probabilistic polynomial-time ideal-model adversary Sim as follows:

1. On input $(\boldsymbol{c}, \mathsf{aux})$, perfectly simulate the group-generation functionality $\mathcal{F}_{\mathsf{GGen}}$ by producing a description $\mathcal{G} = (\mathbb{G}, G, q)$ of a cyclic group $\mathbb{G}$ of order q that is generated by $G \in \mathbb{G}$, where q is a κ-bit prime.

2. Invoke A on the input $(\boldsymbol{c}, \mathsf{aux})$ and the description $\mathcal{G}$ of the cyclic group. Throughout the execution of A, whenever A issues a random-oracle query x, perfectly simulate the random-oracle functionality $\mathcal{F}_{\mathsf{RO}}$: If the query x has been previously made then retrieve the stored value $H(x)$ and respond with $(\mathsf{output}, x, H(x))$. Otherwise, uniformly sample $H(x) \leftarrow \{0,1\}^\kappa$, store the pair $(x, H(x))$, and respond with $(\mathsf{output}, x, H(x))$.

3. Sample $A \leftarrow \mathbb{G}$ uniformly, and send A and $(\mathsf{verified}, (G, A))$ to A on behalf of S and $\mathcal{F}_{\mathsf{ZK\text{-}DL}}$, respectively.

4. Obtain the message $(\mathsf{proof}, (G, A, C_1, \ldots, C_n), (\boldsymbol{c'}, \boldsymbol{r'}))$ sent by A to $\mathcal{F}_{\mathsf{ZK\text{-}PED}}$, and let $\boldsymbol{c'} = (c'_1, \ldots, c'_n) \in \{0,1\}^n$ and $\boldsymbol{r'} = (r'_1, \ldots, r'_n) \in \mathbb{Z}_q^n$. If for some $i \in [n]$ it holds that $C_i \neq c'_i \cdot A + r'_i \cdot G$, then submit some fixed input on behalf of R to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$, respond to $\mathcal{F}_{\mathsf{OT}}$ with abort, and output A's output. Otherwise (i.e., $C_i = c'_i \cdot A + r'_i \cdot G$ for every $i \in [n]$), continue to the next step.

5. Submit the input $\boldsymbol{c'}$ on behalf of R to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$, and obtain an output $\{m_i\}_{i\in[n]}$ from $\mathcal{F}_{\mathsf{OT}}$.

6. For every $i \in [n]$ compute $D_{i,c'_i} = r'_i \cdot A$ and set $k_{i,c'_i} = H(i, D_{i,c'_i})$. If $H(i, D_{i,c'_i})$ has not yet been defined via a random oracle query previously issued by A, then uniformly sample $H(i, D_{i,c'_i}) \leftarrow \{0,1\}^\kappa$ and store the pair $((i, D_{i,c'_i}), H(i, D_{i,c'_i}))$ for consistently responding to A's following random oracle queries.

7. For every $i \in [n]$ set $w_{i,c'_i} = k_{i,c'_i} \oplus m_i$, and uniformly sample $w_{i,1-c'_i} \leftarrow \{0,1\}^\kappa$.

8. Send $\{(w_{i,0}, w_{i,1})\}_{i\in[n]}$ to A on behalf of S, and output A's output.

Provided with the above description of Sim, we now upper bound the statistical distance between the two distributions $\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{S}(\mathsf{aux})}(x_1, x_2, \kappa)$. First, note that the values $D_{i,c'_i} = r'_i \cdot A$ are identically distributed in both cases. Therefore the only difference is that Sim uniformly sample $w_{i,1-c'_i} \leftarrow \{0,1\}^\kappa$, instead of defining $D_{i,1-c'_i} = a^2 \cdot G - r'_i \cdot A$ and then setting $w_{i,1-c'_i} = m_{i,1-c'_i} \oplus H(i, D_{i,1-c'_i})$. Therefore, as long as for every $i \in [n]$ the adversary A does not query the random-oracle functionality on the input $(i, a^2 \cdot G - r'_i \cdot A)$, then the two distributions $\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa)$ and $\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{S}(\mathsf{aux})}(x_1, x_2, \kappa)$ are *identical*. Thus, letting $\mathcal{E}(x_1, x_2, \mathsf{aux}, \kappa)$ denote the event in which during the simulation by Sim the adversary A queries the random-oracle functionality on the input $(i, a^2 \cdot G - r'_i \cdot A)$ for some $i \in [n]$, it holds that

$$\mathsf{SD}\left(\mathrm{HYBRID}^{\mathcal{F}}_{\Pi_{\mathsf{OT}},\mathsf{A}(\mathsf{aux})}(x_1, x_2, \kappa), \mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}(\mathsf{aux})}(x_1, x_2, \kappa)\right) \leq \Pr\left[\mathcal{E}(x_1, x_2, \mathsf{aux}, \kappa)\right].$$

In the remainder of the analysis of this case, we present an algorithm B for solving the q_{A}-sqDH problem for which

$$\Pr\left[\mathcal{E}(x_1, x_2, \mathsf{aux}, \kappa)\right] \leq \mathbf{Adv}^{q_{\mathsf{A}}\text{-}\mathsf{sqDH}}_{\mathsf{GGen},\mathsf{B}}(\kappa) \tag{4.1}$$

for every $\kappa \in \mathbb{N}$ and for every (x_1, x_2, aux). On input $(1^\kappa, \mathcal{G}, A)$, algorithm B first internally emulates the above-described ideal-model execution $\mathrm{IDEAL}_{\mathcal{F}_{\mathsf{OT}},\mathsf{S}(\mathsf{aux})}(x_1, x_2, \kappa)$ for the triplet (x_1, x_2, aux) that maximizes the probability of the event $\mathcal{E}(x_1, x_2, \mathsf{aux}, \kappa)$ (note that, for every $k \in \mathbb{N}$, we use (x_1, x_2, aux) as a non-uniform advice for the algorithm B).

Let $\boldsymbol{r'} = (r'_1, \ldots, r'_n) \in \mathbb{Z}_q^n$ denote the values extracted by Sim in Step 4 of Sim's description, and let $\{(i_j, H_j)\}_{j\in[q_{\mathsf{A}}]}$ denote the random oracle queries issued by A. For every $j \in [q_{\mathsf{A}}]$, algorithm B computes $G_j = r'_{i_j} \cdot A + H_j$ and outputs $(G_1, \ldots, G_{q_{\mathsf{A}}})$. Clearly, if A queries the random-oracle functionality on the input $(i, a^2 \cdot G - r'_i \cdot A)$ for some $i \in [n]$, then for some $j \in [q_{\mathsf{A}}]$ it holds that $(i_j, H_j) = (i_j, a^2 \cdot G - r'_{i_j} \cdot A)$, and therefore $G_j = a^2 \cdot G$. That is, in this case, the algorithm B solves the given q_{A}-sqDH instance, and therefore Eq. 4.1 holds as required. ∎

References

[AIR01] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, May 2001.

[ANS10] Y. Arbitman, M. Naor, and G. Segev. Backyard Cuckoo hashing: Constant worst-case operations with a succinct representation. In *51st Annual Symposium on Foundations of Computer Science*, pages 787–796. IEEE Computer Society Press, October 2010.

[BCP04] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 145–158. Springer, Heidelberg, March 2004.

[BM90] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 547–557, August 1990.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

[CO15a] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. In K. E. Lauter and F. Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Lecture Notes in Computer Science*, pages 40–58. Springer, Heidelberg, August 2015.

[CO15b] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. Cryptology ePrint Archive, Report 2015/267, 2015. https://eprint.iacr.org/2015/267.

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

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, August 1987.

[KLR10] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39(5):2090–2112, 2010.

[Lin08] A. Y. Lindell. Efficient fully-simulatable oblivious transfer. In T. Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 52–70, April 2008.

[Mau05] U. M. Maurer. Abstract models of computation in cryptography (invited paper). In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, December 2005.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In S. R. Kosaraju, editor, *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. ACM-SIAM, January 2001.

[Pag00] R. Pagh. Faster deterministic dictionaries. In D. B. Shmoys, editor, *11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 487–493. ACM-SIAM, January 2000.

[PR04] R. Pagh and F. F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, August 2008.

[Rab81] M. O. Rabin. How to exchange secret by oblivious transfer. Technical Report TR-81, Harvard University, 1981.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Heidelberg, May 1997.