Amortizing Rate-1 OT and Applications to PIR and PSI

Melissa Chase[1], Sanjam Garg[2*], Mohammad Hajiabadi[3†], Jialin Li[4], and Peihan Miao[5‡]

[1] Microsoft Research, melissac@microsoft.com

[2] University of California, Berkeley and NTT Research, sanjamg@berkeley.edu

[3] University of Waterloo, mdhajiabadi@uwaterloo.ca

[4] University of California, Berkeley, j.li98@berkeley.edu

[5] University of Illinois Chicago, peihan@uic.edu

Abstract

Recent new constructions of rate-1 OT [Döttling, Garg, Ishai, Malavolta, Mour, and Ostrovsky, CRYPTO 2019] have brought this primitive under the spotlight and the techniques have led to new feasibility results for private-information retrieval, and homomorphic encryption for branching programs. The receiver communication of this construction consists of a quadratic (in the sender's input size) number of group elements for a single instance of rate-1 OT. Recently [Garg, Hajiabadi, Ostrovsky, TCC 2020] improved the receiver communication to a linear number of group elements for a single string-OT. However, most applications of rate-1 OT require executing it multiple times, resulting in large communication costs for the receiver.

In this work, we introduce a new technique for amortizing the cost of multiple rate-1 OTs. Specifically, based on standard pairing assumptions, we obtain a two-message rate-1 OT protocol for which the amortized cost per string-OT is asymptotically reduced to only four group elements. Our results lead to significant communication improvements in PSI and PIR, special cases of SFE for branching programs.

1. *PIR*: We obtain a rate-1 PIR scheme with client communication cost of $O(\lambda \cdot \log N)$ group elements for security parameter λ and database size N. Notably, after a one-time setup (or one PIR instance), any following PIR instance only requires communication cost $O(\log N)$ number of group elements.
2. *PSI with unbalanced inputs*: We apply our techniques to private set intersection with unbalanced set sizes (where the receiver has a smaller set) and achieve receiver communication of $O((m+\lambda)\log N)$ group elements where m, N are the sizes of the receiver and sender sets, respectively. Similarly, after a one-time setup (or one PSI instance), any following PSI instance only requires communication cost $O(m \cdot \log N)$ number of group elements. All previous sublinear-communication non-FHE based PSI protocols for the above unbalanced setting were also based on rate-1 OT, but incurred at least $O(\lambda^2 m \log N)$ group elements.

1 Introduction

Oblivious transfer (OT) [Rab05] is a foundational primitive in cryptography. In this work, we are interested in *two-message* OT protocols between: (i) a receiver with an input bit b who sends the

*Supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation and Visa Inc.

†Supported in part by NSF CNS Award 2055564.

‡Supported in part by NSF CNS Award 2055358 and a 2020 DPI Science Team Seed Grant.

first message otr of the protocol, and (ii) a sender with input two (equal length) strings m_0, m_1 who sends the second message ots. Correctness requires that at the end of execution, the receiver should learn m_b, while security requires that the receiver does not learn m_{1-b} and that the sender does not learn the bit b. Over the years, significant progress has been made in constructing two-message OT protocols, either from general assumptions [EGL82,GMW87], or from specific assumptions but with enhanced security/functionality/efficiency, such as OT based on DDH [NP01,AIR01,PVW08], CDH [DGH+20], factoring related [HK12] and LWE [PVW08].

Rate-1 OT. In this work, we are interested in constructing rate-1 two-message OT protocols. We say that an OT protocol is rate-1 if the ratio $\frac{|m_0|}{|\mathsf{ots}|}$ approaches 1, as $n := |m_0|$ grows. As shown by Ishai and Paskin [IP07], rate-1 OT enables powerful applications such as (i) semi-compact homomorphic encryption for branching programs (where the ciphertext grows only with the depth but not the size of the program) as well as (ii) communication-efficient private-information retrieval (PIR) protocols.

The rate-1 property is crucial in realizing these applications, allowing a sender to compress a large database for a receiver who is interested only in a small portion of it. To give some intuition, suppose we want to use a rate-1 OT to implement a 1-out-of-4 OT for a sender with four elements $\mathbf{m} := (m_{00}, m_{01}, m_{10}, m_{11})$. Thinking about the corresponding binary tree, the receiver on an input $uw \in \{0,1\}^2$ will send two messages otr and otr', the first one for choice bit u and the second one for w. The sender will use otr' once against (m_{00}, m_{01}) and once against (m_{10}, m_{11}) to get two outgoing messages ots_0 and ots_1. The receiver is only interested in ots_u, but the sender does not know which one it is. So, the sender compresses $(\mathsf{ots}_0, \mathsf{ots}_1)$ using otr, allowing the receiver to learn ots_u, and consequently m_{uw}.

The above construction employs a *self-eating* process, where a pair of ots messages is used as the sender input for the next OT, and so on. Employing a low rate 1-out-of-2 OT to build 1-out-of-n OT will blow up the communication, falling short for PIR. To see this, suppose $|\mathsf{ots}| \geq 2|m_0|$, as is the case with most 1-out-of-2 OT protocols. Then, if $n = 2^k$, as the sender packs up the tree from bottom-up, in each OT invocation the size of the resulting ots message (which either packs two previous ots messages, or two leaf messages) doubles, resulting in a final message of size at least $O(2^k f)$, where f is the size of each initial individual message of the sender. While the protocol is a 1-out-of-n OT, it is not a sublinear PIR, because the size of the sender's protocol message is not sublinear in its total input size, nf. Moreover, as we will see later, in some applications involving branching programs, such as Private Set Intersection (PSI) with unbalanced set sizes, the sender will need to pack a tree of depth polynomial in the security parameter (as opposed to logarithmic size as in PIR), so using low rate 1-out-of-2 OT will result in an exponential size blow-up.

Building rate-1 OT. Recent work of Döttling, Garg, Ishai, Malavolta, Mour, and Ostrovsky [DGI+19] provides a framework for constructing rate-1 OT based on a variety of assumptions such as DDH, QR, and LWE. This in turn led to PIR protocols with sender messages of only a logarithmic size dependence on the server size, and, more generally, branching-program protocols with sender messages whose size only grows with the depth of the program. In addition to these applications, the underlying techniques have been used in building collision-intractable hash functions and non-interactive zero-knowledge (NIZK) proofs [BKM20]. This has made the notion of rate-1 OT fundamental both from a theory and applications point of view.

How about the receiver communication? An overlooked aspect of rate-1 OT is the receiver communication cost. This is an important metric because, as stated above, the self eating process involve producing many otr messages (proportional to the depth of the tree/program), and hence sending a fresh otr for each depth results in large first-round messages. Concretely, in the DDH-based rate-1 OT construction of [DGI+19], for a sender with $(m_0 \in \{0,1\}^n, m_1 \in \{0,1\}^n)$, the receiver should send a linear $(O(n))$ number of group elements for each bit of the sender, resulting in overall $O(n^2)$ group elements. This incurs high receiver communication in the respective applications. Addressing this issue, Garg, Hajiabadi and Ostrovsky [GHO20] obtained rate-1 OT for which otr consists of only a linear $O(n)$ number of group elements in total, as opposed to $O(n^2)$.

One limitation of [GHO20] is that it only improves the communication efficiency of the base rate-1 OT, but still requires the receiver to send a fresh otr message for each new OT execution. This constitutes a prohibitive overhead for the receiver in applications in which the depth of the branching program is large, and where the receiver needs to engage with a sender holding a branching program BP on many different inputs $x_1, \ldots, x_n$ (e.g., PSI). Addressing this communication bottleneck is the goal of our paper. We achieve this by introducing and realizing a new primitive that we call receiver-amortized (or amortized, for short) rate-1 OT.

1.1 Our Results

We put forth a cryptographic primitive that we call *amortized* rate-1 OT, and show how to realize it using standard assumptions on bilinear groups. As applications we obtain significant efficiency improvements, shaving a factor of $\mathsf{poly}(\lambda)$ off the receiver communication in various protocols involving secure branching program computation (e.g., unbalanced PSI).

An amortized rate-1 OT breaks up the computation of a receiver into an *offline* and *online* phase. The offline phase is performed by the receiver once and for all, prior to receiving any choice bits. Specifically, we have an algorithm $\mathsf{PreP}(1^\lambda)$, run by the receiver, which outputs a private state str for the receiver, and a reusable parameter prm. Next, we have an algorithm OT_1 run by the receiver on a choice bit b to obtain $\mathsf{otr} \xleftarrow{\$} \mathsf{OT}_1(\mathsf{str}, b)$. A sender with messages $\mathbf{m} := (m_0 \in \{0,1\}^n, m_1 \in \{0,1\}^n)$ runs $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), \mathbf{m})$ to obtain ots. Finally, the receiver can recover m_b by running $\mathsf{OT}_3(\mathsf{str}, \mathsf{ots})$. One notable aspect is that the state str used by OT_1 and OT_3 is the same as the initial state outputted by PreP — the state is not updated as a result of OT_1 executions. This property is in fact exploited in some of our applications, such as PSI cardinality. Also, the message prm is reused across all communications, so the receiver may send it only once. We require the following properties:

1. Sender rate-1 communication: $|\mathsf{ots}| = n + \mathsf{poly}(\lambda)$, where poly is a fixed polynomial (e.g., the size of a group element) independent of how large n is.

2. Receiver non-reusable compactness: $|\mathsf{otr}| = \mathsf{poly}'(\lambda)$, where $\mathsf{poly}'(\lambda)$ is independent of n.

3. Receiver privacy: We require indistinguishability security for the receiver against adaptive adversaries. If $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^\lambda)$, an adaptive adversary who is given prm and who sends many pairs of choice bits in an adaptive fashion cannot determine whether his received otr messages (all made relative to str) were built using the first choice bits or the second choice bits of his submitted pairs. Notice that since otr messages are all produced based on the same private state str, we should give the adversary the ability to submit many pairs.

4. Sender privacy: Standard indistinguishability security against honest receivers.[1]

Assuming an SXDH-hard bilinear map $\mathsf{e}: \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ on prime-order groups, we give a construction of amortized rate-1 OT in which prm consists of $O(n^2)$ group elements in $\mathbb{G}_1$ and otr consists of 4 group elements in $\mathbb{G}_2$. Recall that the SXDH assumption [BGdMM05] (Symmetric External Diffie-Hellman) states that both $\mathbb{G}_1$ and $\mathbb{G}_2$ are DDH hard. Our construction is based on a new re-randomization trick that allows us to obtain a structured matrix, as required for rate-1 OT, from a reusable initial matrix and and a re-randomizing term involving four group elements.

The above reusable parameter prm is still quite large, even though it can be amortized among many OT executions. We show by relying on a stronger assumption on $\mathbb{G}_1$, called $2n$-power-DDH, we can make prm consist only of $O(n)$ group elements in $\mathbb{G}_1$. We achieve this by relying on a sliding window technique, introduced in [GHO20], that implicitly builds a Toeplitz matrix in the exponent using a linear number of group elements. The t-power DDH assumption says the distribution $(g, g^a, \ldots, g^{a^t})$ is pseudorandom.

Efficiency gained. For performing t rate-1 OTs where the size of each message of the sender is n, our receiver communication consists of $O(n^2)$ reusable group elements in $\mathbb{G}_1$ and $4t$ group elements in $\mathbb{G}_2$, relying on SXDH. Assuming power DDH on $\mathbb{G}_1$ the receiver communication becomes $O(n)$ group elements in $\mathbb{G}_1$ and $4t$ group elements in $\mathbb{G}_2$. In comparison, the most receiver compact bilinear SXDH-based rate-1 OT, due to [DGI+19], involves sending $O(tn\sqrt{n})$ both in $\mathbb{G}_1$ and $\mathbb{G}_2$. As we will see in Section 1.2 in many applications of rate-1 OT, we have $t >> \sqrt{n}$, allowing us to cut off large multiplicative polynomial factors from the receiver communication. We compare our receiver communication with prior rate-1 OT protocols in Table 1. We only include receiver communication, since the sender communication in all these protocols is the same (rate-1 for each instance of the OT).

Work	Receiver Reusable Comm	Receiver Non-Reusable Comm	Receiver Total Comm	Assumption
[DGI+19]	N/A	$O(tn^2)$ $\mathbb{G}$	$O(tn^2)$ $\mathbb{G}$	DDH
[DGI+19]	N/A	$O(tn\sqrt{n})$ $\mathbb{G}_1$ + $O(tn\sqrt{n})$ $\mathbb{G}_2$	$O(tn\sqrt{n})$ $\mathbb{G}_1$ + $O(tn\sqrt{n})$ $\mathbb{G}_2$	Bilinear SXDH
[GHO20]	N/A	$O(tn)$ $\mathbb{G}$	$O(tn)$ $\mathbb{G}$	Power-DDH
Ours	$O(n^2)$ $\mathbb{G}_1$	$O(t)$ $\mathbb{G}_2$	$O(n^2)$ $\mathbb{G}_1$ + $O(t)$ $\mathbb{G}_2$	Bilinear SXDH
Ours	$O(n)$ $\mathbb{G}_1$	$O(t)$ $\mathbb{G}_2$	$O(n)$ $\mathbb{G}_1$ + $O(t)$ $\mathbb{G}_2$	Bilinear Power DDH

Table 1: Receiver communication complexity for t executions of a rate-1 OT. Here n denotes the bit size of each message of the sender in the OT executions.

1.2 Applications

Our results allow us to realize SFE for branching programs with significantly lower receiver communication. To illustrate our improvements, we first review the concept of branching programs. A deterministic κ-bit input branching program BP is a directed acyclic graph, where every leaf node has a label 0 or 1 (reject or accept), and every non-leaf node v has a label $\mathsf{lb}(v) \in \{1, \ldots, \kappa\}$. The root node is labeled with 1. Every non-leaf node has two outgoing edges labeled 0 and 1.

[1] For applications involving non-oblivious branching programs we need to strengthen sender privacy, along the lines of [IP07]. For oblivious branching programs, from which all our applications are obtained, the stated requirement suffices.

An input $x \in \{0,1\}^\kappa$ induces a unique computation path from the root to a leaf node, where the computation from a node v will branch out to one of its two children depending on the value of x_i, where $i = \mathsf{lb}(v)$. We say $\mathsf{BP}(x) = b$ if the underlying computation path ends in a b-labeled leaf node. The size of a branching program is the number of nodes, and the depth, ℓ, is the length of the longest path. A branching program is *oblivious* if $\kappa = \ell$ and if all nodes at level i (where the root is considered level 1) are labeled i.[2]

As an example, consider a client who wants to know whether her input $x \in \{0,1\}^\lambda$ is in the set $\mathsf{D} \subset \{0,1\}^\lambda$ of a server. This reduces to evaluating an oblivious branching program PSI on x where PSI is constructed as follows: for every string $a \in \{\epsilon\} \cup \{0,1\} \cup \cdots \{0,1\}^\lambda$ such that a is a prefix of a string in D, we put a node v_a in the graph. We designate v_ϵ as the root node, and all v_a such that $a \in \mathsf{D}$ as accept leaf nodes. The label of a node v_a for $|a| < \lambda$ is $\mathsf{lb}(v_a) = |a| + 1$. For a node v_a, for $|a| < \lambda$, and for $b \in \{0,1\}$, if a node v_{ab} exists, we put a b-labeled edge from v_a to v_{ab}; otherwise, we create a new reject leaf node and put a b-labeled edge from v_a to this node. The depth of PSI is λ and its size is $O(\lambda|\mathsf{D}|)$.

Now if a client wants to learn the intersection of her set $\mathsf{S} = \{x_1, \ldots, x_m\}$ with D, she needs to learn the values of all $\mathsf{PSI}(x_i)$ for $i \in [m]$, leading to m evaluations of PSI.

Shorter client communication for PSI. Ishai and Paskin [IP07] give a construction of SFE for branching programs from rate-1 OT, where, for an oblivious branching program BP of depth d, the receiver sends d otr messages, each prepared for a sender whose input messages are of size $O(d\lambda)$. Returning to the PSI problem for a client with set $\mathsf{S} = \{x_1, \ldots, x_m\}$ and a server with set D, we need to evaluate the oblivious branching program PSI m times. Recall that the depth of PSI is λ. Hence, setting $t = m\lambda$ and $n = \lambda^2$ in Table 1, our PSI-client communication consists of $O(m\lambda)$ non-reusable group elements in $\mathbb{G}_2$ (in either SXDH or power-DDH cases) and $O(\lambda^4)$ reusable group elements in $\mathbb{G}_1$ (in the case of SXDH), and $O(\lambda^2)$ reusable group elements in $\mathbb{G}_1$ (in the case of bilinear power DDH). In contrast, [DGI+19] results in $O(m\lambda^4)$ group elements in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Thus, we drop a multiplicative factor of m by relying on the same SXDH assumption, and a factor of $m\lambda^2$ by relying on bilinear power DDH. The results of [GHO20] give $O(m\lambda^3)$ group elements for the receiver using (pairing-free) power DDH. This is again significantly larger than what we achieve.

In Section 7.3 we describe some PSI optimization techniques that further reduce the client communication, replacing a multiplicative factor of λ with $\log N$, where $N = |\mathsf{D}|$. These techniques may be of independent interest. We also give more applications, involving PSI/PIR, in Section 7.

SFE for non-oblivious branching programs. Ishai and Paskin [IP07] show how to realize SFE for non-oblivious branching programs (in which at any given level the program might branch over several variables, not known to the receiver) by relying on a stronger sender privacy notion for the underlying rate-1 OT. Informally, the stronger property requires that a sender's response message should hide the previous protocol message of the receiver, even for the receiver herself. In Section 8 we show that simple variants of our amortized rate-1 OT satisfy the stronger sender security requirement, without affecting the efficiency parameters. All our applications are obtained based on oblivious branching programs, however.

[2]The standard definition of oblivious branching programs is more general than what we give here, but we stick to our own definition since it captures our application needs.

We summarize our efficiency parameters for branching programs in Table 2. See Table 3 (Page 24) for a more detailed comparison.

Work	Assumption	Primitive	Recv Reuse Comm	Recv Non-Reuse Comm
Ours	Bilinear SXDH	Oblivious BP	$\lambda(h+\lambda\ell)^2$	$m\lambda\ell$
Ours	Bilinear Power DDH	Oblivious BP	$\lambda(h+\lambda\ell)$	$m\lambda\ell$
[**GHO20**]	Power DDH	Oblivious BP	N/A	$m\lambda\ell(h+\lambda\ell)$
[**DGI$^+$19**]	Bilinear SXDH	Oblivious BP	N/A	$O(m\lambda\ell(h+\lambda\ell)^{3/2})$
[**DGI$^+$19**]	DDH	Oblivious BP	N/A	$O(m\lambda\ell(h+\lambda\ell)^2)$
Ours	Bilinear SXDH	BP	$\lambda(h+\lambda\ell)^2$	$m\kappa\lambda\ell$
Ours	Bilinear Power DDH	BP	$\lambda(h+\lambda\ell)$	$m\kappa\lambda\ell$
[**GHO20**]	Power DDH	BP	N/A	$m\kappa\lambda\ell(h+\lambda\ell)$
[**DGI$^+$19**]	Bilinear SXDH	BP	N/A	$O(m\lambda\ell\kappa(h+\lambda\ell)^{3/2})$
[**DGI$^+$19**]	DDH	BP	N/A	$O(m\lambda\ell\kappa(h+\lambda\ell)^2)$

Table 2: Bit-complexity for receiver communication, omitting $O(\cdot)$ notation. We assume $O(\lambda)$ is the bit size of a group element (in the case of pairings, for both source and target group elements). m denotes the number of branching programs executions. For (oblivious) branching programs (BP), h is the bit size of the output, κ is the bit size of receiver message and ℓ is the depth of the BP program.

1.3 Comparison with Prior Work

The rate-1 OT constructions of [DGI$^+$19] built upon ideas developed in the context of trapdoor functions (TDFs) [GH18, GGH19], identity-based encryption [CDG$^+$17, DG17, BLSV18] and homomorphic secret sharing [BGI16]. The TDF techniques in turn led to notions such as hinting PRGs [KW19], which found extensive applications, e.g., [LQR$^+$19, KMT19, HKW20, GVW20].

OT extension. One might wonder about the difference between amortized rate-1 OT and OT extension [Bea96, IKNP03]. The primary goal of OT extension is to minimize the number of public-key operations: Performing $n := n(\lambda)$ OTs at the cost of doing a fewer, λ, number of OTs and some private key operations. On the other hand, we are concerned with amortizing receiver communication for rate-1 OT; doing t rate-1 OTs, but in a way that the receiver total communication is less than the sum of t individual rate-1 OT executions. OT extension techniques do not provide this feature. Moreover, OT extension techniques destroy the rate-1 property of the sender. For example, Beaver's protocol, which is round preserving, results in sender's OT protocol messages which are larger than $|m_0| + |m_1|$, where (m_0, m_1) is the sender's initial input pair. We leave it as open problem whether one can achieve some form of OT extension and amortized rate-1 OT at the same time.

PSI Private set intersection (PSI) enables two parties, each holding a private set of elements, to compute the intersection of the two sets while revealing nothing. PSI and its variants have found many real-world applications including online advertising [IKN$^+$20], password breach alert [TPY$^+$19, APP, MIC], mobile private contact discovery [KRS$^+$19], privacy-preserving contact tracing [TSS$^+$20, CCF$^+$20]. In the recent years, there has been tremendous progress made towards realizing PSI efficiently in various settings, including Diffie-Hellman-based [HFH99, IKN$^+$20], RSA-based [ADT11],

OT-extension-based [KKRT16,PRTY19,PRTY20,CM20], FHE-based [CLR17], circuit-based [HEK12, PSSZ15,PSWW18,PSTY19], Vector-OLE-based [RS21] approaches.

Most of the existing approaches require the communication complexity to grow with the size of the *larger* set, the only exception being the FHE-based protocol [CLR17] (where communication grows linearly in the receiver set and logistically in the sender set) and RSA-based protocol [ADT11] (where the receiver has the bigger set and the communication grows linearly in the smaller, sender set). We consider the dual setting of [ADT11], meaning that in our case the receiver has the smaller set. In many real-world applications such as password breach alert [TPY+19, APP, MIC] and mobile private contact discovery [KRS+19], we need to perform *unbalanced* PSI between a constrained device (e.g. cellphone) holding a small set and a service provider holding a large set, thus having communication grow the larger set (especially the sender set) is a big concern. Our work presents unbalanced PSI with communication complexity linear in the size of the receiver set and logarithmic in the sender set. Furthermore, our approach is easily adapted to PSI with advanced functionalities such as PSI-Cardinality, PSI-Sum, PSI-Test, etc., which could only be achieved from Diffie-Hellman-based or circuit-based approaches. See Section 7 for more details.

2 Technical Overview

One tool used in our constructions (and in all recent rate-1 OT constructions) is a compressed version of n-bit packed ElGamal encryption. We review these compression features, formalized in [BBD+20], building on [BGI16]. A secret key is an n-bit tuple of exponents $\mathsf{sk} := (\rho_1, \ldots, \rho_n)$ and the public key is $\mathsf{pk} := (g, g^{\rho_1}, \ldots, g^{\rho_n})$. Given $\mathsf{pk} := (g, g_1, \ldots, g_n)$ we can encrypt an n-bit message $\mathsf{Enc}(m_1, \ldots, m_n)$ as $\mathsf{ct} : (g^r, g_1^{r+m_1}, \ldots, g_n^{r+m_n})$. We have two additional algorithms Shrink and $\mathsf{ShrinkDec}$, where $\mathsf{Shrink}(\mathsf{ct})$ shrinks $\mathsf{ct} \in \mathbb{G}^{n+1}$ to obtain $\mathsf{Shrink}(\mathsf{ct}) \to (g', K, b_1, \ldots, b_n) \in \mathbb{G} \times \{0,1\}^{\lambda+n}$. We have shrinking correctness: $\Pr[\mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{ct})) = (m_1, \ldots, m_n)] = 1$.

Approach of [DGI+19]. Let $\mathbb{G}$ be a group of prime order p with a generator g. We let $\vec{\boldsymbol{e}}_i \in \mathbb{G}^{2n}$ denote a vector which has g in its ith position, and the identity element 1 everywhere else.

The receiver on a choice bit b samples $\vec{\mathsf{hk}} \xleftarrow{\$} \mathbb{G}^{2n}$ and for every $i \in [n]$ samples $\rho_i \xleftarrow{\$} \mathbb{Z}_p$ and sets $\vec{\mathsf{ek}}_i := \vec{\mathsf{hk}}^{\rho_i} \cdot \vec{\boldsymbol{e}}_{i+nb}$, where $\vec{\mathsf{hk}}^{\rho_i}$ denotes entry-wise exponentiation, and $(\cdot)$ denotes entry-wise group multiplication. She sends $\mathsf{otr} := (\vec{\mathsf{hk}}, \{\vec{\mathsf{ek}}_i\})$ to the sender.

Let $\vec{\boldsymbol{m}} = (m_0, m_1) \in \{0,1\}^{2n}$ be a vector concatenating the two strings of the sender. Let $g' := \vec{\boldsymbol{m}} \cdot \vec{\mathsf{hk}}$, and for $i \in [n]$ let $g'_i := \vec{\boldsymbol{m}} \cdot \vec{\mathsf{ek}}_i$, where we overload the $(\cdot)$ notation to define $(b_1, \ldots, b_{2n}) \cdot (g_1, \ldots, g_{2n}) = \prod g_i^{b_i}$. Letting $\mathsf{pk} := (g, g^{\rho_1}, \ldots, g^{\rho_n})$, we have $(g', g'_1, \ldots, g'_n) \in \mathsf{Enc}(\mathsf{pk}, m_b)$, where Enc denotes n-bit packed ElGamal. With this in mind, the sender sends $\mathsf{ots} = \mathsf{Shrink}(\mathsf{ct})$ to the receiver, and the receiver, who has $\mathsf{sk} := (\rho_1, \ldots, \rho_n)$ can recover m_b as $\mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ots})$. We have $\mathsf{ots} \in \mathbb{G} \times \{0,1\}^{\lambda+n}$, so the OT is sender rate-1.

In the above, each vector $\vec{\mathsf{ek}}_i$ is a ρ_i exponentiation of $\vec{\mathsf{hk}}$ but with a *bump* on its $(n + ib)$'s location: namely, we multiply its $(n + b)$'s location by g.

Our techniques: SXDH. We now give a new technique based on pairings that allows us to produce many bumpy vectors $\vec{\mathsf{ek}}_i$'s in the target group, using only 4 group elements and a reusable initial parameter in the source groups. The receiver samples $2n$ vectors $\vec{\boldsymbol{r}}_i \xleftarrow{\$} \mathbb{Z}_p^2$, and let $\mathbf{M}$ contain

all these vectors in the exponent in $\mathbb{G}_1$, namely

$$\mathbf{M} := ([\vec{\boldsymbol{r}}_1]_1, \ldots, [\vec{\boldsymbol{r}}_n]_1 \mid [\vec{\boldsymbol{r}}_{n+1}]_1, \ldots, [\vec{\boldsymbol{r}}_{2n}]_1),$$

where $[\vec{\boldsymbol{r}}]_1 := g^{\vec{\boldsymbol{r}}}$. We similarly define $[\vec{\boldsymbol{r}}]_2 := h^{\vec{\boldsymbol{r}}}$ and $[\vec{\boldsymbol{r}}]_T := \mathsf{e}(g,h)^{\vec{\boldsymbol{r}}}$.

Also, let

$$\begin{aligned}
\vec{\boldsymbol{\nu}}_1 &:= ([p_1\vec{\boldsymbol{r}}_1 + \vec{u}]_1, [p_1\vec{\boldsymbol{r}_2}]_1, \cdots, [p_1\vec{\boldsymbol{r}}_n]_1 \mid [p_1\vec{\boldsymbol{r}}_{n+1} + \vec{u}]_1, [p_1\vec{\boldsymbol{r}}_{n+2}]_1, \cdots, [p_1\vec{\boldsymbol{r}}_{2n}]_1) \\
&\vdots \\
\vec{\boldsymbol{\nu}}_n &:= ([p_n\vec{\boldsymbol{r}}_1]_1, [p_n\vec{r_2}]_1, \cdots, [p_n\vec{\boldsymbol{r}}_n + \vec{u}]_1 \mid [p_n\vec{\boldsymbol{r}}_{n+1}]_1, [p_n\vec{\boldsymbol{r}}_{n+2}]_1, \cdots, [p_n\vec{\boldsymbol{r}}_{2n} + \vec{u}]_1),
\end{aligned}$$

The receiver sets $\mathsf{prm} := (\mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n)$ and her private state as $\mathsf{str} := (\vec{\boldsymbol{u}}, p_1, \ldots, p_n)$.

Receiver's non-reusable messages. To send a short otr message for a choice bit b, the receiver samples two random vectors $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ s.t. $\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{u}} \rangle = 0$ and $\langle \vec{\boldsymbol{w}}, \vec{\boldsymbol{u}} \rangle = 1$. The receiver sends $\mathsf{otr} := ([\vec{\boldsymbol{f}}]_2, [\vec{\boldsymbol{h}}]_2)$, where $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) = (\vec{\boldsymbol{w}}, \vec{\boldsymbol{v}})$ if $b = 0$, and $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) = (\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ if $b = 1$.

Sender's protocol messages. Given $\mathsf{prm} := (\mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n)$ and $\mathsf{otr} := ([\vec{\boldsymbol{f}}]_2, [\vec{\boldsymbol{h}}]_2)$, the sender uses the pairing to computes the inner product of $\vec{\boldsymbol{f}}$ with all the vectors in the left-hand side of $\mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n$, and the inner product of $\vec{\boldsymbol{h}}$ with all the vectors in the right-hand side of the $\mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n$. That is, using the notation above, letting $\alpha_j := \langle \vec{\boldsymbol{r}}_j, \vec{\boldsymbol{f}} \rangle$ if $j \in [n]$, and $\alpha_j := \langle \vec{\boldsymbol{r}}_j, \vec{\boldsymbol{h}} \rangle$ if $j \in \{n+1, \ldots, 2n\}$ the sender will compute

$$\vec{\mathbf{hk}} := ([\alpha_1]_T \cdots, [\alpha_n]_T \mid [\alpha_{n+1}]_T \cdots, [\alpha_{2n}]_T)$$

$$\mathbf{EK} := \left[\begin{array}{ccc|ccc} [p_1\alpha_1 + 1]_T & \ldots & [p_1\alpha_n]_T & [p_1\alpha_{n+1}]_T & \ldots & [p_1\alpha_{2n}]_T \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ ([p_n\alpha_1]_T & \ldots & [p_n\alpha_n + 1]_T & [p_n\alpha_{n+1}]_T & \ldots & [p_n\alpha_{2n}]_T \end{array}\right] \quad \text{if } b = 0$$

$$\mathbf{EK} := \left[\begin{array}{ccc|ccc} [p_1\alpha_1]_T & \ldots & [p_1\alpha_n]_T & [p_1\alpha_{n+1} + 1]_T & \ldots & [p_1\alpha_{2n}]_T \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ ([p_n\alpha_1]_T & \ldots & [p_n\alpha_n]_T & [p_n\alpha_{n+1}]_T & \ldots & [p_n\alpha_{2n} + 1]_T \end{array}\right] \quad \text{if } b = 1$$

The sender has now built $(\vec{\mathbf{hk}}, \mathbf{IK})$ that satisfies the bump structure explained in the first paragraph. Namely, think of the ith row of $\mathbf{EK}$ as $\vec{\mathbf{ek}}_i$ in that paragraph. Moreover, the receiver knows all the underlying exponent values $\mathsf{sk} := (p_1, \ldots, p_n)$. Now the sender can perform the step explained in the first paragraph to send a rate-1 message ots, and the receiver will be able to use sk to decrypt it to obtain m_b.

Notice that the protocol has rate-1 sender communication, and that otr consist of only 4 group elements in $\mathbb{G}_2$.

To argue about receiver privacy, let us, for simplicity, argue that an adversary $\mathcal{A}$ cannot distinguish between a world in which otr always encrypts the bit 0 from a world in which otr encrypts 1; the proof for the case where the adversary can submits adaptively-chosen pairs of choice bits will be similar. We should show that $\mathcal{A}$ for a random pair $([\vec{\boldsymbol{f}}]_2, [\vec{\boldsymbol{h}}]_2)$ of vectors cannot tell which one

is orthogonal to $\vec{\boldsymbol{u}}$ and which one has inner product one. This should be argued in the presence of prm, known to $\mathcal{A}$. We will first remove the presence of $\vec{\boldsymbol{u}}$ from prm, relying on DDH for $\mathbb{G}_1$. Let prm' be the same as prm but with $\vec{\boldsymbol{u}}$ removed. By DDH, $(\vec{\boldsymbol{u}}, \mathsf{prm}) \stackrel{c}{\equiv} (\vec{\boldsymbol{u}}, \mathsf{prm}')$. If we want to replace prm with prm' for $\mathcal{A}$, we should be able to reply to $\mathcal{A}$'s subsequent OT_1 queries. The reason this can be done is because OT_1 responses are produced based on only $\vec{\boldsymbol{u}}$ and the underlying choice bit, and $\vec{\boldsymbol{u}}$ is included in both distributions. Thus, we can remove $\vec{\boldsymbol{u}}$ from the prm view of $\mathcal{A}$. Once this is done, we will then show that the entire otr view of $\mathcal{A}$ can be simulated by knowing a pair of vectors $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ where $\vec{\boldsymbol{v}}$ is orthogonal to $\vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{w}}$ has inner product one with $\vec{\boldsymbol{u}}$. In particular, to sample from $\mathsf{OT}_1(\mathsf{str}, b)$, we return $(k_1\vec{\boldsymbol{v}} + (1-b)\vec{\boldsymbol{w}}, k_2\vec{\boldsymbol{v}} + b\vec{\boldsymbol{w}})$, where k_1 and k_2 are random exponents. Next we show that the distribution of $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ is identical to uniformly random vectors. This can be argued because information about $\vec{\boldsymbol{u}}$ has been already removed from prm. Finally, we rely on DDH for $\mathbb{G}_2$ to show that by using a random $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ in the above simulation, the entire otr view of $\mathcal{A}$ will be pseudorandom, masking the value of the choice bit b.

Our techniques: Bilinear Power DDH. We sketch how to adapt our cancellation technique to a sliding window setting, developed in [GHO20], to reduce the size of prm into a linear number of group elements. The receiver samples a random exponent a and a vector $\vec{\boldsymbol{r}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$ and sets

$$\mathbf{M} := \left([a\vec{\boldsymbol{r}}]_1, [a^2\vec{\boldsymbol{r}}]_1, \cdots, [a^{2n}\vec{\boldsymbol{r}}]_1\right)$$

$$\begin{aligned}\vec{\boldsymbol{w}} :=& ([ka\vec{\boldsymbol{r}}]_1, \cdots, [ka^{n-1}\vec{\boldsymbol{r}}]_1, [ka^n\vec{\boldsymbol{r}} + \vec{\boldsymbol{u}}]_1, [ka^{n+1}\vec{\boldsymbol{r}}]_1, \cdots, [ka^{2n-1}\vec{\boldsymbol{r}}]_1,\\ & [ka^{2n}\vec{\boldsymbol{r}} + \vec{\boldsymbol{u}}]_1, [ka^{2n+1}\vec{\boldsymbol{r}}]_1, \cdots, [ka^{3n-1}\vec{\boldsymbol{r}}]_1),\end{aligned}$$

where k is a random exponent. The receiver sets $\mathsf{prm} := (\mathbf{M}, \vec{\boldsymbol{w}})$.

The receiver samples a non-reusable message $\mathsf{otr} = ([\vec{\boldsymbol{f}}]_2, [\vec{\boldsymbol{h}}]_2)$ for a choice bit b exactly as in the SXDH case — by sampling it based on $\vec{\boldsymbol{u}}$ and b.

A sender given $(\mathsf{prm}, \mathsf{otr})$ builds n vectors $\vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n$ as follows. For $i \in [n]$ let $\vec{\boldsymbol{\nu}_i} = \vec{\boldsymbol{w}}[n+1-i, 3n-i]$, where $\vec{\boldsymbol{w}}[i,j]$ denotes the elements in positions i all the way up to j. Once the vectors $\vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n$ are formed, the sender will proceed exactly like the SXDH case. Correctness will then follow. The proof of receiver privacy follow similarly to the SXDH case, but we should replace DDH with power DDH in the appropriate places. We omit the details.

3 Preliminaries and Definitions

We use λ for the security parameter. We use $\stackrel{c}{\equiv}$ and $\stackrel{s}{\equiv}$ for computational and statistical indistinguishability, respectively. We let $\equiv$ denote that two distributions are identical. For a distribution $\mathbf{S}$ we use $x \stackrel{\$}{\leftarrow} \mathbf{S}$ to mean x is sampled according to $\mathbf{S}$ and use $y \in \mathbf{S}$ to mean $y \in \sup(\mathbf{S})$, where sup denotes the support of a distribution. For a set S we overload the notation to use $x \stackrel{\$}{\leftarrow} \mathsf{S}$ to indicate that x is chosen uniformly at random from S. If $\mathsf{A}(x_1, \ldots, x_n)$ is a randomized algorithm, then $\mathsf{A}(a_1, \ldots, a_n)$, for deterministic inputs $a_1, \ldots, a_n$, denotes the random variable obtained by sampling random coins r uniformly at random and returning $\mathsf{A}(a_1, \ldots, a_n; r)$. We use $[n] := \{1, \ldots, n\}$ and $[i, i+s] := \{i, i+1, \ldots, i+s\}$. For a vector $\vec{\boldsymbol{v}} = (v_1, \ldots, v_n)$ we define $\vec{\boldsymbol{v}}[i, i+s] := (v_i, v_{i+1}, \ldots, v_{i+s})$.

Definition 3.1 (Pairings and SXDH hardness)**.** *A bilinear map is given by* $(\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h) \xleftarrow{\$} \mathsf{G}(1^\lambda)$*, where* p *is a prime number and is the order of* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_T$*, and* g *and* h *are random generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively. The function* e *is a non-degenerate map, satisfying* $\mathsf{e}(g^a, h^b) = \mathsf{e}(g, h)^{ab}$ *for all exponents* a *and* b*. The Symmetric External Diffie-Hellman (SXDH) assumption [BGdMM05] says* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, sampled as above, are DDH-hard.*

Computing inner product in the exponent. Given $\vec{\boldsymbol{g}} := (g_1, \ldots, g_k) \in \mathbb{G}_1^k$ and $\vec{\boldsymbol{h}} := (h_1, \ldots, h_k) \in \mathbb{G}_2^k$ we define $\mathsf{e}(\vec{\boldsymbol{g}}, \vec{\boldsymbol{h}}) := \prod_{i \in [k]} \mathsf{e}(g_i, h_i)$.

Inner product with integer vectors. Given $\vec{\boldsymbol{b}} := (b_1, \ldots, b_k) \in \mathbb{Z}_p^k$ and $\vec{\boldsymbol{g}} := (g_1, \ldots, g_k) \in \mathbb{G}_1^k$, we define $\vec{\boldsymbol{b}} \cdot \vec{\boldsymbol{g}} := \prod_{i \in [k]} g_i^{b_i}$.

3.1 Amortized Rate-1 OT: Definition

We define our new notion of amortized rate-1 OT, which allows a receiver to reuse part of her protocol message across many independent OT executions. In the definition below, think of n as the maximum size of each input message of a sender. The receiver will generate a reusable parameter prm, based on n, which will allow her later to send a short protocol message otr whenever she wants to perform a new OT. The sender will use $(\mathsf{prm}, \mathsf{otr})$ to complete an OT transfer for any pair of messages $(m_0 \in \{0,1\}^{n_1}, m_1 \in \{0,1\}^{n_1})$, as long as $n_1 \leq n$.

Definition 3.2 (Amortized Rate-1 OT)**.** *Let* $n := n(\lambda)$ *be a polynomial. An amortized rate-1 OT* $\mathsf{OT} := (\mathsf{PreP}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *is defined as follows.*

- $\mathsf{PreP}(1^\lambda, n) \to (\mathsf{str}, \mathsf{prm})$*: Takes as input a security parameter* 1^λ *and* n*, denoting the maximum length of each of the sender's messages, and outputs a private state* str *and a reusable message* prm*.*
- $\mathsf{OT}_1(\mathsf{str}, b) \to \mathsf{otr}$*: Takes as input a security parameter* 1^λ *and a choice bit* $b \in \{0,1\}$*, and outputs a a protocol message* otr*. We refer to* otr *as a fresh receiver's message, to distinguish it from the reusable message* prm*.*
- $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1)) \to \mathsf{ots}$*: Takes as input a reusable message* prm*, a fresh message* otr *and a pair of messages* $(m_0, m_1) \in \{0,1\}^{n_1} \times \{0,1\}^{n_1}$*, for some* $n_1 \leq n$*, and outputs* ots*.*
- $\mathsf{OT}_3(\mathsf{str}, \mathsf{ots}) \to m$*: Takes as input a private state* str *and* ots *and outputs* $m \in \{0,1\}^n$*.*

We require

- ***Correctness****: For any polynomial* $n := n(\lambda)$*,* $b \in \{0,1\}$*,* $n_1 \leq n$ *and* $(m_0, m_1) \in \{0,1\}^{n_1} \times \{0,1\}^{n_1}$*,* $\Pr[\mathsf{OT}_3(\mathsf{str}, \mathsf{ots}) = m_b] = 1$*, where* $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^\lambda, n)$*,* $\mathsf{otr} \xleftarrow{\$} \mathsf{OT}_1(\mathsf{str}, b)$ *and* $\mathsf{ots} \xleftarrow{\$} \mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1))$*.*
- ***Rate-1 sender communication****: There exists a fixed polynomial* poly *such that for all* n *and* $n_1 \leq n$*,* $|\mathsf{ots}| = n_1 + \mathsf{poly}(\lambda)$*, where* ots *is formed as above.*

- ***Receiver amortized compactness:*** *The length of* otr *is independent of* n*. There exists a fixed polynomial* poly' *such that for all polynomials* $n := n(\lambda)$*,* $|\mathsf{otr}| = \mathsf{poly}'(\lambda)$*, where* otr *is formed as above.*

- ***Receiver privacy****: An adaptive sender cannot determine the choice bits of a receiver. Any PPT adversary* $\mathcal{A}$ *has at most* $1/2 + \mathsf{negl}(\lambda)$ *advantage in the following game. The challenger samples* $b \xleftarrow{\$} \{0,1\}$ *and* $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^n, \lambda)$ *and gives* prm *to* $\mathcal{A}$*. Then,* $\mathcal{A}$ *adaptively submits queries* $(s_0, s_1) \in \{0,1\}^2$*, and receives* $\mathsf{OT}_1(\mathsf{str}, s_b)$*.* $\mathcal{A}$ *has to guess the value of* b*.*

Sender privacy? Notice that Definition 3.2 does not impose any sender security requirements. The reason for this is that sender security can be generically realized for rate-1 OT using known techniques [BGI+17], as sketched below. Let poly be the polynomial defined in the rate-1 sender property of Definition 3.2. The new sender on a pair of messages $(m_0, m_1) \in \{0,1\}^n \times \{0,1\}^n$ samples two seeds (r_0, r_1) whose length is sufficiently larger than $\mathsf{poly}(\lambda)$ but independent of n. The sender sends $(\mathsf{ots}'_1, \mathsf{ots}'_2)$ to the receiver, where $\mathsf{ots}'_1 \xleftarrow{\$} \mathsf{OT}_2((r_0, r_1), (\mathsf{prm}, \mathsf{otr}))$ and $\mathsf{ots}'_2 \xleftarrow{\$} \mathsf{OT}_2((\mathsf{ct}_0, \mathsf{ct}_1), (\mathsf{prm}, \mathsf{otr}))$, where $\mathsf{ct}_0 := \mathsf{PRG}(\mathsf{Ext}(r_0)) \oplus m_0$ and $\mathsf{ct}_1 := \mathsf{PRG}(\mathsf{Ext}(r_1)) \oplus m_1$, and Ext is a randomness extractor. The protocol is still sender rate-1. It now provides computational sender privacy against honest receivers: This is because given ots'_1 the value of $\mathsf{Ext}(r_{1-b})$ is statistically close to uniform, where b is the receiver's choice bit.

Finally, we mention that we may modify our constructions so that they achieve sender privacy for free, without using the above generic randomness extraction method.

4 Amortized Rate-1 OT from SXDH

Our amortized rate-1 OT protocol makes use of a shrinking algorithm, that allows one to shrink ciphertexts of ElGamal encryption, as long as the underlying plaintexts are coming from a small space, say, $\{0,1\}$. An n-bit packed ElGamal encryption has a secret key $\mathsf{sk} := (x_1, \ldots, x_n)$ and a public key $\mathsf{pk} := (g, g^{x_1}, \ldots, g^{x_n})$. Given $\mathsf{pk} := (g, g_1, \ldots, g_n)$ we can encrypt an n-bit message $\mathsf{Enc}(m_1, \ldots, m_n)$ as $\mathsf{ct} : (g^r, g_1^{r+m_1}, \ldots, g_n^{r+m_n})$. We have a shrinking procedure for n-bit ElGamal encryption that will shrink a ciphertext into one group element plus n bits, while allowing for efficient decryption. The procedure below, presented in [BBD+20], enables perfect decryption correctness, improving upon the previous procedures [BGI16, DGI+19] that had a decryption error.

Lemma 4.1 ([BBD+20])**.** *There exists a pair of (expected) PPT algorithms* $(\mathsf{Shrink}, \mathsf{ShrinkDec})$ *such that if* $(\mathsf{pk}, \mathsf{sk})$ *is as above and* $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, \mathbf{m})$ *is a packed ElGamal ciphertext encrypting a message* $\mathbf{m} \in \{0,1\}^n$*,*

(1) $\mathsf{Shrink}(\mathsf{ct}) \to (g', K, b_1, \ldots, b_n) \in \mathbb{G} \times \{0,1\}^{\lambda+n}$*.*

(2) $\Pr[\mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{ct})) = \mathbf{m}] = 1$*.*

Our amortized rate-1 OT makes us of the following procedure $\mathsf{OrthSam}$ that given a vector $\vec{\boldsymbol{u}} \in \mathbb{Z}_p^2$ and a bit $b \in \{0,1\}$, samples two random vectors $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{w}}$ such that $\langle \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \rangle = 0$ and $\langle \vec{\boldsymbol{u}}, \vec{\boldsymbol{w}} \rangle = 1$, and it outputs these two vectors in a shuffled order based on the value of b.

Definition 4.2. *The algorithm* $\mathsf{OrthSam}(\vec{\boldsymbol{u}} \in \mathbb{Z}_p^2, b \in \{0,1\})$ *works as follows. It Samples random vectors* $\vec{\boldsymbol{w}}, \vec{\boldsymbol{v}}$ *such that* $\langle \vec{\boldsymbol{w}}, \vec{\boldsymbol{u}} \rangle = 1$ *and* $\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{u}} \rangle = 0$*, and returns* $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) \in \mathbb{Z}_p^4$*, where*

$$(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) = \begin{cases} (\vec{\boldsymbol{w}}, \vec{\boldsymbol{v}}) & b = 0 \\ (\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) & b = 1 \end{cases}$$

4.1 Our Construction

We now present our construction. For notational clarity, we assume the size of each message of the sender is exactly n, as opposed to an arbitrary value $n_1 \leq n$. Adapting the construction to work with respect to varying lengths for the sender messages will be immediate.

Construction 4.3 (Amortized rate-1 OT: SXDH)**.** *Build* $\mathsf{OT} := (\mathsf{PreP}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *as follows.*

- $\mathsf{PreP}(1^\lambda, n)$*: Sample* $\mathsf{pp} := (\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h) \xleftarrow{\$} \mathsf{G}(1^\lambda)$*. Then*

  1. *For* $i \in [2n]$*, sample* $\vec{\boldsymbol{r}}_i \xleftarrow{\$} \mathbb{Z}_p^2$*. Let*

  $$\mathbf{M} := [g^{\vec{\boldsymbol{r}}_1}, g^{\vec{\boldsymbol{r}}_2}, \cdots, g^{\vec{\boldsymbol{r}}_n} \mid g^{\vec{\boldsymbol{r}}_{n+1}}, g^{\vec{\boldsymbol{r}}_{n+2}}, \cdots, g^{\vec{\boldsymbol{r}}_{2n}}].$$

  2. *Sample* $\vec{\boldsymbol{u}} \xleftarrow{\$} \mathbb{Z}_p^2$ *and for* $i \in [n]$ *sample a random exponent* $p_i \xleftarrow{\$} [p]$*. Let* $\vec{\boldsymbol{D}} := (\vec{\boldsymbol{\nu_1}}, \ldots, \vec{\boldsymbol{\nu_n}})$*, where*

  $$\begin{aligned} \vec{\boldsymbol{\nu}}_1 &:= [g^{p_1 \vec{\boldsymbol{r}}_1 + \vec{\boldsymbol{u}}}, g^{p_1 \vec{\boldsymbol{r}}_2}, \cdots, g^{p_1 \vec{\boldsymbol{r}}_n} \mid g^{p_1 \vec{\boldsymbol{r}}_{n+1} + \vec{\boldsymbol{u}}}, g^{p_1 \vec{\boldsymbol{r}}_{n+2}}, \cdots, g^{p_1 \vec{\boldsymbol{r}}_{2n}}] \\ &\vdots \\ \vec{\boldsymbol{\nu}}_n &:= [g^{p_n \vec{\boldsymbol{r}}_1}, g^{p_n \vec{\boldsymbol{r}}_2}, \cdots, g^{p_n \vec{\boldsymbol{r}}_n + \vec{\boldsymbol{u}}} \mid g^{p_n \vec{\boldsymbol{r}}_{n+1}}, g^{p_n \vec{\boldsymbol{r}}_{n+2}}, \cdots, g^{p_n \vec{\boldsymbol{r}}_{2n} + \vec{\boldsymbol{u}}}], \end{aligned} \tag{1}$$

  3. *Return private state* $\mathsf{str} := (\vec{\boldsymbol{u}}, p_1, \ldots, p_n)$ *and reusable message* $\mathsf{prm} := (\mathsf{pp}, \mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n)$*.*

- $\mathsf{OT}_1(\mathsf{str}, b \in \{0,1\})$*: Parse* str *and all its inside variables as above. Sample* $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) \xleftarrow{\$} \mathsf{OrthSam}(\vec{\boldsymbol{u}}, b)$ *(Definition 4.2). Return return* $\mathsf{otr} := (h^{\vec{\boldsymbol{f}}}, h^{\vec{\boldsymbol{h}}}) \in \mathbb{G}_2^4$*.*

- $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1) \in \{0,1\}^n \times \{0,1\}^n)$*: Parse* $\mathsf{prm} := (\mathsf{pp}, \mathbf{M}, \vec{\boldsymbol{\nu}}_1, \ldots, \vec{\boldsymbol{\nu}}_n)$*,* $\mathsf{otr} := (\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\chi}}_2) \in \mathbb{G}_2^4$*,* $\mathbf{M} := (\vec{\boldsymbol{m}}_1, \ldots, \vec{\boldsymbol{m}}_{2n})$ *and* $\vec{\boldsymbol{\nu}}_i := (\vec{\boldsymbol{\nu}}_{i,1}, \ldots, \vec{\boldsymbol{\nu}}_{i,2n})$ *for* $i \in [n]$*. Let*

  $$\vec{\boldsymbol{hk}} := (\mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{m}}_1), \ldots \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{m}}_n) \mid \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{m}}_{n+1}), \ldots \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{m}}_{2n}))$$

  $$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\nu}}_{1,1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\nu}}_{1,n}) & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{\nu}}_{1,n+1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{\nu}}_{1,2n}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\nu}}_{n,1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\nu}}_{n,n}) & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{\nu}}_{n,n+1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{\nu}}_{n,2n}) \end{array}\right].$$

  Let $\vec{\boldsymbol{m}} := (m_0, m_1) \in \{0,1\}^{2n}$*. Let* $\vec{\boldsymbol{y}}_j \in \mathbb{G}_T^{2n}$ *be the* j*th row of* $\mathbf{IK}$*. The sender then sends*

  $$\mathsf{ots} := \mathsf{Shrink}(\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n) \in \mathbb{G}_T \times \{0,1\}^{n+\lambda}.$$

- $\mathsf{OT}_3(\mathsf{str}, \mathsf{ots})$*: Parse* $\mathsf{str} := (\vec{\boldsymbol{u}}, p_1, \ldots, p_n)$ *and set* $\mathsf{sk} := (p_1, \ldots, p_n)$*. Return* $m' := \mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ots})$*.*

Correctness. We prove $m' = m_b$, where, following the notation of Construction 4.3, m' is the string output by OT_3, and (m_0, m_1) are the input strings to OT_2 and b is the choice bit for OT_1.

Let $\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}$ and $\vec{\boldsymbol{r}}_{\boldsymbol{1}}, \ldots, \vec{\boldsymbol{r}}_{2n}$ be as in Construction 4.3. Let $\alpha_j := \langle \vec{\boldsymbol{r}}_j, \vec{\boldsymbol{f}} \rangle$ if $j \in [n]$, and $\alpha_j := \langle \vec{\boldsymbol{r}}_j, \vec{\boldsymbol{h}} \rangle$ if $j \in \{n+1, \ldots, 2n\}$. Letting $\vec{\boldsymbol{h}}\vec{\boldsymbol{k}}$ and $\mathbf{IK}$ be as in Construction 4.3, we have

$$\vec{\boldsymbol{h}}\vec{\boldsymbol{k}} := [\mathsf{e}(g,h)^{\alpha_1} \cdots, \mathsf{e}(g,h)^{\alpha_n} \mid \mathsf{e}(g,h)^{\alpha_{n+1}} \cdots, \mathsf{e}(g,h)^{\alpha_{2n}}] \in \mathbb{G}_T^{2n}$$

$$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(g,h)^{p_1\alpha_1} \cdot \mathsf{e}(g,h) & \ldots & \mathsf{e}(g,h)^{p_1\alpha_n} & \mathsf{e}(g,h)^{p_1\alpha_{n+1}} & \ldots & \mathsf{e}(g,h)^{p_1\alpha_{2n}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(g,h)^{p_n\alpha_1} & \ldots & \mathsf{e}(g,h)^{p_n\alpha_n} \cdot \mathsf{e}(g,h) & \mathsf{e}(g,h)^{p_n\alpha_{n+1}} & \ldots & \mathsf{e}(g,h)^{p_n\alpha_{2n}} \end{array}\right] \in \mathbb{G}_T^{n\times 2n} \quad \text{if } b = 0$$

$$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(g,h)^{p_1\alpha_1} & \ldots & \mathsf{e}(g,h)^{p_1\alpha_n} & \mathsf{e}(g,h)^{p_1\alpha_{n+1}} \cdot \mathsf{e}(g,h) & \ldots & \mathsf{e}(g,h)^{p_1\alpha_{2n}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(g,h)^{p_n\alpha_1} & \ldots & \mathsf{e}(g,h)^{p_n\alpha_n} & \mathsf{e}(g,h)^{p_n\alpha_{n+1}} & \ldots & \mathsf{e}(g,h)^{p_n\alpha_{2n}} \cdot \mathsf{e}(g,h) \end{array}\right] \in \mathbb{G}_T^{n\times 2n} \quad \text{if } b = 1.$$

Thus, $(\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{h}}\vec{\boldsymbol{k}}, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n) \in \mathsf{Enc}(\mathsf{pk}, (m_b[1], \ldots, m_b[n]))$, where $\vec{\boldsymbol{m}} = (m_0, m_1)$, $\vec{\boldsymbol{y}}_j$ is the jth row of $\mathbf{IK}$, $\mathsf{pk} := (\mathsf{e}(g,h), \mathsf{e}(g,h)^{p_1}, \ldots, \mathsf{e}(g,h)^{p_n})$ and Enc is the packed ElGamal encryption algorithm as in Lemma 4.1. By Lemma 4.1, $m' = m_b$, as desired.

Rate-1 sender communication and receiver amortized compactness. We have $|\mathsf{ots}| = n + \lambda + |g| = n + \mathsf{poly}(\lambda)$ and $|\mathsf{otr}| = 4|h|$.

4.2 Receiver Privacy

In the following we say a vector $\vec{\boldsymbol{f}}$ is non-orthogonal to $\vec{\boldsymbol{u}}$ if $\langle \vec{\boldsymbol{f}}, \vec{\boldsymbol{u}} \rangle = 1$. This is an abuse of terminology (because non-orthogonality refers to any non-zero inner product), but we stick to it below.

To prove receiver OT security, we should argue that a fresh receiver protocol message otr does not reveal the receiver's underlying choice bit. The main difficulty is that all otr values depend on the vector $\vec{\boldsymbol{u}}$.

The core of our argument is in showing that the vector $\vec{\boldsymbol{u}}$ remains hidden in the following sense. Given a sequence of $(g^{\vec{\boldsymbol{f}}_i}, g^{\vec{\boldsymbol{h}}_i})$, an adversary cannot determine the order of orthogonality/non-orthogonality in any given pair, with respect to $g^{\vec{\boldsymbol{u}}}$. To this end, we will first remove $\mathbf{u}$ from all vectors $\vec{\boldsymbol{D}} := (\vec{\boldsymbol{\nu}}_{\boldsymbol{1}}, \ldots, \vec{\boldsymbol{\nu}}_{\boldsymbol{n}})$, given in Equation 1. Once $\vec{\boldsymbol{u}}$ is removed from the reusable message prm, we will then show any receiver's future fresh message otr may be simulated by the underlying choice bit b and a pair of vectors $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ which are orthogonal/non-orthogonal to $\vec{\boldsymbol{u}}$, in a way that if the joint distribution of $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ is pseudorandom, then the entire simulated view will be pseudorandom as well, masking the choice bits. We will then show that the distribution of a random $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ subject to them being orthogonal/non-orthogonal to a random $\vec{\boldsymbol{u}}$ is uniformly random. Taken all together, receiver security will follow.

Definition 4.4 (Distribution **Dual**). *For $\vec{\boldsymbol{u}} \in \mathbb{Z}_p^2$ the distribution* $\mathbf{Dual}(\vec{\boldsymbol{u}})$ *returns $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$, where $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{w}}$ are sampled uniformly subject to $\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{u}} \rangle = 0$ and $\langle \vec{\boldsymbol{w}}, \vec{\boldsymbol{u}} \rangle = 1$.*

We now describe a way of simulating messages otr, for a given choice bit b, without knowing $\vec{\boldsymbol{u}}$, but by knowing a pair $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ sampled according to $\mathbf{Dual}(\vec{\boldsymbol{u}})$.

Definition 4.5 (Simulator Sim)**.** *The algorithm* $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, b)$ *samples* $k, k' \xleftarrow{\$} \mathbb{Z}_p$*, and returns* $(h^{k\vec{\boldsymbol{v}}+(1-b)\vec{\boldsymbol{w}}}, h^{k'\vec{\boldsymbol{v}}+b\vec{\boldsymbol{w}}})$.

Hybrid Hyb_1**:** Real game. Sample $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^\lambda, 1^n)$, a challenge bit $b \xleftarrow{\$} \{0,1\}$, and give prm to the adversary. Parse $\mathsf{str} := (\vec{\boldsymbol{u}}, p_1, \ldots, p_n)$. Reply to an adversary's query $(s_0, s_1) \in \{0,1\}^2$ with $\mathsf{OT}_1(\mathsf{str}, s_b)$. The view of the adversary for otr messages can be produced just by knowing $\vec{\boldsymbol{u}}$, as opposed to $\mathsf{str} := (\vec{\boldsymbol{u}}, p_1, \ldots, p_n)$. In particular, the values $p_1, \ldots, p_n$ do not participate in producing the output of $\mathsf{OT}_1(\mathsf{str}, s_b)$, and are only used in OT_3, which is immaterial to the adversary's view.

Hybrid Hyb_2**:** Replace $\vec{\boldsymbol{D}} = (\vec{\boldsymbol{\nu_1}}, \ldots, \vec{\boldsymbol{\nu_n}})$, Equation 1, with uniformly random vectors of group elements. Thus, information about $\vec{\boldsymbol{u}}$ will be removed from $\vec{\boldsymbol{D}}$, and hence from prm.

Hybrid Hyb_3**:** Same as Hybrid Hyb_2, except we sample $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$, and reply to any adversary's query (s_0, s_1) as $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, s_b)$. The whole view is produced by knowing only $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$.

Hybrid Hyb_4**:** Same as Hyb_3, except we sample $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbb{Z}_p^4$.

Hybrid Hyb_5**:** Same as Hyb_4, except we reply to any adversary's query (s_0, s_1) with a uniformly random vector sampled from $\mathbb{G}_2^4$. This hybrid perfectly hides the value of the challenge bit b.

We will now show that any two adjacent hybrids produce computationally indistinguishable views.

Lemma 4.6 ($\mathsf{Hyb}_1 \stackrel{c}{\equiv} \mathsf{Hyb}_2$)**.** *Assuming DDH hardness of* $\mathbb{G}_1$*,* $(\vec{\boldsymbol{u}}, \mathbf{D}) \stackrel{c}{\equiv} (\vec{\boldsymbol{u}}, \vec{\boldsymbol{D'}})$*, where* $\vec{\boldsymbol{u}}$ *and* $\vec{\boldsymbol{D}}$ *are as in Equation 1, and* $\vec{\boldsymbol{D'}}$ *is a uniformly random matrix of group elements.*

Proof. By DDH hardness of $\mathbb{G}_1$, $(\vec{\boldsymbol{u}}, \vec{\boldsymbol{D}}) \stackrel{c}{\equiv} (\vec{\boldsymbol{u}}, \vec{\boldsymbol{D'}})$. The views in Hybrid Hyb_1 and Hyb_2 can be produced just by knowing $(\vec{\boldsymbol{u}}, \vec{\boldsymbol{D}})$ and $(\vec{\boldsymbol{u}}, \vec{\boldsymbol{D'}})$, respectively. (See the explanation given in Hybrid Hyb_1 on why the view can be simulated just by knowing $(\vec{\boldsymbol{u}}, \vec{\boldsymbol{D}})$.) Thus, $\mathsf{Hyb}_1 \stackrel{c}{\equiv} \mathsf{Hyb}_2$. □

Lemma 4.7. $\mathsf{Hyb}_2 \stackrel{s}{\equiv} \mathsf{Hyb}_3$.

Proof. Let $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{w}}$ be as in Hyb_3, namely $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$. We show that for any choice bit $z \in \{0,1\}$, the output of $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, z)$ is statistically close to $(h^{\vec{\boldsymbol{f}}}, h^{\vec{\boldsymbol{h}}})$, where $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) \xleftarrow{\$} \mathsf{OrthSam}(\vec{\boldsymbol{u}}, z)$.

Let S_0 be the set of all vectors whose inner product with $\vec{\boldsymbol{u}}$ is one, and S_1 be the set of all vectors orthogonal to $\vec{\boldsymbol{u}}$. The vectors $\vec{\boldsymbol{f}}$ and $\vec{\boldsymbol{h}}$ are uniformly distributed over S_z and S_{1-z}, respectively. Also, recall that the output of $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, z)$ is as $(h^{k\vec{\boldsymbol{v}}+(1-z)\vec{\boldsymbol{w}}}, h^{k'\vec{\boldsymbol{v}}+z\vec{\boldsymbol{w}}})$, where $k, k' \xleftarrow{\$} \mathbb{Z}_p$. In what follows, we show $(r\vec{\boldsymbol{v}}, r'\vec{\boldsymbol{v}} + \vec{\boldsymbol{w}})$ for random r and r' is statistically close the uniform distribution over $(\mathsf{S}_0, \mathsf{S}_1)$, and this will complete the proof.

Notice that S_1 is a subspace and has dimension one; i.e., any basis of it has only one vector. Letting $\vec{\boldsymbol{v}} := (v_1, v_2)$ assume $v_1 \neq 0$ and $v_2 \neq 0$. (The probability that either is zero is negligible, so we may ignore it.) Since $v_1 \neq 0$ and $v_2 \neq 0$, if $r \xleftarrow{\$} \mathbb{Z}_p$, then $r\vec{\boldsymbol{v}}$ is uniformly random in S_1.

Next, note that $\mathsf{S}_0 = \vec{\boldsymbol{w}} + \mathsf{S}_1$; i.e., for any $\vec{\boldsymbol{m}}' \in \mathsf{S}_0$, there exists $\vec{\boldsymbol{m}} \in \mathsf{S}_1$ s.t. $\vec{\boldsymbol{m}}' = \vec{\boldsymbol{w}} + \vec{\boldsymbol{m}}$. Since $\vec{\boldsymbol{v}}$ spans S_1, the vector $r'\vec{\boldsymbol{v}} + \vec{\boldsymbol{w}}$ for a random $r' \xleftarrow{\$} \mathbb{Z}_p$ is uniformly distributed over S_0. The above was conditioned on $v_1 \neq 0$ and $v_2 \neq 0$, which is true with all but negligible probability. Thus, we have statistical indistinguishability. □

Lemma 4.8 ($\mathsf{Hyb}_3 \overset{s}{\equiv} \mathsf{Hyb}_4$)**.** *Assuming $\vec{\boldsymbol{u}} \xleftarrow{\$} \mathbb{Z}_p^2$, the output of* $\mathbf{Dual}(\vec{\boldsymbol{u}})$ *is statistically close to the uniform distribution over $\mathbb{Z}_p^4$. Thus, the two hybrids are statistically indistinguishable.*

Proof. The only difference between these two hybrids lies in $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$, sampled as $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$ in Hyb_3 and as completely random in Hyb_4. We show that the marginal distribution of $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ sampled as $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$ is statistically close to the uniform distribution over $\mathbb{Z}_p^4$, assuming $\vec{\boldsymbol{u}}$ is uniformly random. This will complete the proof, because the view in either hybrid can be sampled by knowing $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$, and by knowing prm, from which we have already removed $\vec{\boldsymbol{u}}$, so prm is information-theoretically independent of $\vec{\boldsymbol{u}}$.

Parse $\vec{\boldsymbol{u}} := (a, b) \in \mathbb{Z}_p^2$. First, we know that $\vec{\boldsymbol{u}} = \vec{\boldsymbol{0}}$ with negligible probability. In case $\vec{\boldsymbol{u}} \neq \vec{\boldsymbol{0}}$, without loss of generality assume $b \neq 0$, then we have $\vec{\boldsymbol{v}} = (x, -\frac{a}{b}x)$ and $\vec{\boldsymbol{w}} = (z, -\frac{a}{b}z + \frac{1}{b})$, where $x, z \xleftarrow{\$} \mathbb{Z}_p$. Let $(t, t') := (-\frac{a}{b}, \frac{1}{b})$, and note that (t, t') is uniform over $\mathbb{Z}_p^2$ with $t' \neq 0$, since $\vec{\boldsymbol{u}}$ is uniformly random except that $b \neq 0$. We may then rewrite $\vec{\boldsymbol{v}} = (x, tx)$ and $\vec{\boldsymbol{w}} = (z, tz + t')$, where x, z, t, t' are all independent and uniformly random over $\mathbb{Z}_p$ with the constraint that $t' \neq 0$. Thus, $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ is statistically close to the uniform distribution over $\mathbb{Z}_p^4$. □

Lemma 4.9 ($\mathsf{Hyb}_4 \overset{c}{\equiv} \mathsf{Hyb}_5$)**.** *Assuming DDH hardness of $\mathbb{G}_2$, $\mathsf{Hyb}_4 \overset{c}{\equiv} \mathsf{Hyb}_5$.*

Proof. In Hyb_4 the receiver forms an otr message for an adversary's query $(s_0, s_1) \in \{0,1\}^2$ as $(h^{k\vec{\boldsymbol{v}} + (1-s_b)\vec{\boldsymbol{w}}}, h^{k'\vec{\boldsymbol{v}} + s_b\vec{\boldsymbol{w}}})$. Since $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{w}}$ are independent and uniformly random, and since $k, k' \xleftarrow{\$} \mathbb{Z}_p$, by DDH $(g^{k\vec{\boldsymbol{v}}}, g^{k'\vec{\boldsymbol{v}}})$ is pseudorandom, and hence $(h^{k\vec{\boldsymbol{v}} + (1-b_i)\vec{\boldsymbol{w}}}, h^{k'\vec{\boldsymbol{v}} + b_i\vec{\boldsymbol{w}}})$ is pseudorandom. The proof is now complete. □

Thus, we have the following theorem.

Theorem 4.10. *Assuming DDH hardness for $\mathbb{G}_1$ and $\mathbb{G}_2$, the amortized rate-1 OT protocol of Construction 4.3 provides receiver privacy.*

5 Amortized Rate-1 OT from Bilinear Power DDH

We show how to shorten the reusable parameter using the circulant structure imposed by power-DDH assumptions, following ideas from [GHO20]. We assume $\mathbb{G}_2$ is DDH-hard, and $\mathbb{G}_1$ is m-power-DDH hard, meaning that $(g, g^{\alpha}, g^{\alpha^2}, ..., g^{\alpha^m})$ is pseudorandom. We will need to set $m = O(n)$, where n is the bit length of each of the sender's messages. Concretely, $m = 3n - 1$ suffices.

Construction 5.1 (Amortized rate-1 OT: Bilinear Power DDH)**.** *Build* $\mathsf{OT} := (\mathsf{PreP}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *as follows.*

- $\mathsf{PreP}(1^\lambda, n)$*: Sample* $\mathsf{pp} := (\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h) \xleftarrow{\$} \mathsf{G}(1^\lambda)$*. Then*
  1. *Sample* $\mathbf{M} := [g^{a\vec{\boldsymbol{r}}}, g^{a^2\vec{\boldsymbol{r}}}, \cdots, g^{a^{2n}\vec{\boldsymbol{r}}}]$*, where* $a \xleftarrow{\$} \mathbb{Z}_p$ *and* $\vec{\boldsymbol{r}} \xleftarrow{\$} \mathbb{Z}_p^2$*.*

2. *Sample* $k \xleftarrow{\$} \mathbb{Z}_p$ *and let*

$$\begin{aligned}\vec{\boldsymbol{w}} :=& [g^{ka\vec{\boldsymbol{r}}}, g^{ka^2\vec{\boldsymbol{r}}}, \cdots, g^{ka^{n-1}\vec{\boldsymbol{r}}}, g^{ka^n\vec{\boldsymbol{r}}+\vec{\boldsymbol{u}}}, g^{ka^{n+1}\vec{\boldsymbol{r}}}, \cdots, g^{ka^{2n-1}\vec{\boldsymbol{r}}},\\ & g^{ka^{2n}\vec{\boldsymbol{r}}+\vec{\boldsymbol{u}}}, g^{ka^{2n+1}\vec{\boldsymbol{r}}}, \cdots, g^{ka^{3n-1}\vec{\boldsymbol{r}}}]\end{aligned} \tag{2}$$

3. *Return private state* $\mathsf{str} := (\vec{\boldsymbol{u}}, k, a)$ *and reusable message* $\mathsf{prm} := (\mathsf{pp}, \mathbf{M}, \vec{\boldsymbol{w}})$.

- $\mathsf{OT}_1(\mathsf{str}, b \in \{0,1\})$*: Parse* str *as above. Sample* $(\vec{\boldsymbol{f}}, \vec{\boldsymbol{h}}) \xleftarrow{\$} \mathsf{OrthSam}(\vec{\boldsymbol{u}}, b)$ *(Definition 4.2). Return return* $\mathsf{otr} := (h^{\vec{\boldsymbol{f}}}, h^{\vec{\boldsymbol{h}}}) \in \mathbb{G}_2^4$.
- $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1) \in \{0,1\}^n \times \{0,1\}^n)$*: Parse* $\mathsf{otr} := (\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{\chi}}_2) \in \mathbb{G}_2^4$, $\mathsf{prm} := (\mathsf{pp}, \mathbf{M}, \vec{\boldsymbol{w}})$, $\mathbf{M} := (\vec{\boldsymbol{m}}_1, \ldots, \vec{\boldsymbol{m}}_{2n})$ *and* $\vec{\boldsymbol{w}} := (\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_{3n-1})$*. For* $j \in [n]$ *let* $\vec{\boldsymbol{w}}_j := \vec{\boldsymbol{w}}[j, j+2n-1]$*; namely, the elements of* $\vec{\boldsymbol{w}}$ *in the range* $[j, j+2n-1]$*. Parse* $\vec{\boldsymbol{w}}_j := (\vec{\boldsymbol{w}}_{j,1}, \ldots, \vec{\boldsymbol{w}}_{j,2n})$*. Let*

$$\vec{\boldsymbol{hk}} := (\mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{m}}_1), \ldots \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{m}}_n) \mid \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{m}}_{n+1}), \ldots \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{m}}_{2n}))$$

$$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{w}}_{n,1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{w}}_{n,n}) & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{w}}_{n,n+1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{w}}_{n,2n}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{w}}_{1,1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_1, \vec{\boldsymbol{w}}_{1,n}) & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{w}}_{1,n+1}) & \ldots & \mathsf{e}(\vec{\boldsymbol{\chi}}_2, \vec{\boldsymbol{w}}_{1,2n}) \end{array}\right].$$

Let $\vec{\boldsymbol{m}} := (m_0, m_1) \in \{0,1\}^{2n}$*. Let* $\vec{\boldsymbol{y}}_j \in \mathbb{G}_T^{2n}$ *be the* j*th row of* $\mathbf{IK}$*. The sender then sends*

$$\mathsf{ots} := \mathsf{Shrink}(\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n) \in \mathbb{G}_T \times \{0,1\}^{n+\lambda}.$$

- $\mathsf{OT}_3(\mathsf{str}, \mathsf{ots})$*: Parse* $\mathsf{str} := (\vec{\boldsymbol{u}}, k, a)$ *and set* $\mathsf{sk} := (ka^{n-1}, \ldots, ka, k)$*. Return* $m' := \mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ots})$.

Correctness. We prove $m' = m_b$, where, following the notation of Construction 5.1, m' is the string output by OT_3, and (m_0, m_1) are the input strings to OT_2 and b is the choice bit for OT_1.

Let $\vec{\boldsymbol{f}}$, $\vec{\boldsymbol{h}}$, $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{w}}_j$ for $j \in [n]$ be as in Construction 5.1. Let $\beta = \langle \vec{\boldsymbol{r}}, \vec{\boldsymbol{f}} \rangle$ and $\mu = \langle \vec{\boldsymbol{r}}, \vec{\boldsymbol{h}} \rangle$. Letting $\vec{\boldsymbol{hk}}$ and $\mathbf{IK}$ be as in Construction 5.1, we have

$$\vec{\boldsymbol{hk}} := [\mathsf{e}(g,h)^{\beta a} \cdots, \mathsf{e}(g,h)^{\beta a^n} \mid \mathsf{e}(g,h)^{\mu a^{n+1}} \cdots, \mathsf{e}(g,h)^{\mu a^{2n}}]$$

$$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(g,h)^{k\beta a^n} \cdot \mathsf{e}(g,h) & \ldots & \mathsf{e}(g,h)^{k\beta a^{2n-1}} & \mathsf{e}(g,h)^{k\mu a^{2n}} & \ldots & \mathsf{e}(g,h)^{k\mu a^{3n-1}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(g,h)^{k\beta a} & \ldots & \mathsf{e}(g,h)^{k\beta a^n} \cdot \mathsf{e}(g,h) & \mathsf{e}(g,h)^{k\mu a^{n+1}} & \ldots & \mathsf{e}(g,h)^{k\mu a^{2n}} \end{array}\right] \quad \text{if } b_i = 0$$

$$\mathbf{IK} := \left[\begin{array}{ccc|ccc} \mathsf{e}(g,h)^{k\beta a^n} & \ldots & \mathsf{e}(g,h)^{k\beta a^{2n-1}} & \mathsf{e}(g,h)^{k\mu a^{2n}} \cdot \mathsf{e}(g,h) & \ldots & \mathsf{e}(g,h)^{k\mu a^{3n-1}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathsf{e}(g,h)^{k\beta a} & \ldots & \mathsf{e}(g,h)^{k\beta a^n} & \mathsf{e}(g,h)^{k\mu a^{n+1}} & \ldots & \mathsf{e}(g,h)^{k\mu a^{2n}} \cdot \mathsf{e}(g,h) \end{array}\right] \quad \text{if } b_i = 1$$

Thus, $(\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n) \in \mathsf{Enc}(\mathsf{pk}, (m_b[1], \ldots, m_b[n]))$, where $\vec{\boldsymbol{m}} = (m_0, m_1)$, $\vec{\boldsymbol{y}}_j$ is the jth row of $\mathbf{IK}$, $\mathsf{pk} := (\mathsf{e}(g,h), \mathsf{e}(g,h)^{ka^{n-1}}, \ldots, \mathsf{e}(g,h)^k)$ and Enc is the packed ElGamal encryption algorithm as in Lemma 4.1. By Lemma 4.1, $m' = m_b$, as desired.

5.1 Receiver Privacy

The proof of security follows the same sequence of hybrids as in Section 4.2, so we only sketch the hybrids and the proofs.

Hybrid Hyb_1: Real game. Sample $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^\lambda, 1^n)$, a challenge bit $b \xleftarrow{\$} \{0,1\}$, and give prm to the adversary. Parse $\mathsf{str} := (\vec{\boldsymbol{u}}, *)$. Reply to an adversary's query $(s_0, s_1) \in \{0,1\}^2$ with $\mathsf{OT}_1(\mathsf{str}, s_b)$. The view of the adversary for OT_1 outputs can be produced just by knowing $\vec{\boldsymbol{u}}$.

Hybrid Hyb_2: Replace $\vec{\boldsymbol{w}}$, Equation 2, with uniformly random vectors of group elements. Thus, information about $\vec{\boldsymbol{u}}$ will be removed from $\vec{\boldsymbol{D}}$, and hence from prm.

Hybrid Hyb_3: Same as Hybrid Hyb_2, except we sample $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$ (Definition 4.4), and reply to any adversary's query $(s_0, s_1) \in \{0,1\}^2$ as $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, s_b)$ (Definition 4.5). The whole view is produced by knowing only $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$.

Hybrid Hyb_4: Same as Hyb_3, except we sample $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbb{Z}_p^4$.

Hybrid Hyb_5: Same as Hyb_4, except we reply to any adversary's query (s_0, s_1) with a uniformly random vector sampled from $\mathbb{G}_2^4$. This hybrid perfectly hides the value of the challenge bit b.

$\mathsf{Hyb}_1 \overset{c}{\equiv} \mathsf{Hyb}_2$ is established exactly like Lemma 4.6, except we use the power-DDH assumption instead of DDH. We have $\mathsf{Hyb}_2 \overset{s}{\equiv} \mathsf{Hyb}_3$, $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$ and $\mathsf{Hyb}_4 \equiv \mathsf{Hyb}_5$, and the proofs are exactly the same as those of Lemma 4.7, Lemma 4.8 and Lemma 4.9, respectively. Thus, we have the following theorem.

Theorem 5.2. *Assuming $(3n-1)$-power DDH hardness for $\mathbb{G}_1$, and DDH hardness for $\mathbb{G}_2$, the amortized rate-1 OT protocol of Construction 5.1 provides receiver privacy.*

6 Optimization

In this section, we discuss some techniques to improve the concrete computational efficiency and lower the communication cost in amortized rate-1 OT. These optimizations work for both the basic amortized rate-1 OT from bilinear SXDH and the sliding-window construction from bilinear power DDH. In Section 7 when we describe the applications of amortized rate-1 OT, we will discuss further optimizations specific to these applications.

6.1 Delayed Pairing

Recall that when the sender computes her response message, she needs to compute the hash-key vector $\vec{\boldsymbol{hk}}$, which requires $4n$ pairing operations. In addition, she needs to compute the matrix $\mathbf{IK}$, which requires $4n^2$ pairing operations in the basic construction and $6n$ pairing operations in the sliding-window construction. Since paring operations are orders of magnitude more expensive than the other group operations, we introduce a technique to minimize it.

On Basic Construction. The high-level idea is that we can leverage the bilinear property to delay the pairing operations. Instead of first performing the pairing operations and then computing inner products in the target group, we can first compute the inner products in $\mathbb{G}_1$ and then perform the pairings.

In more detail, in the basic construction, let

$$\mathbf{M}_0 := [g^{\vec{r}_1}, g^{\vec{r}_2}, \cdots, g^{\vec{r}_n}],$$
$$\mathbf{M}_1 := [g^{\vec{r}_{n+1}}, g^{\vec{r}_{n+2}}, \cdots, g^{\vec{r}_{2n}}].$$

Let $\vec{\boldsymbol{m}} = (\vec{\boldsymbol{m}}_0, \vec{\boldsymbol{m}}_1) \in \{0,1\}^{2n}$ be the sender messages. With receiver message $\mathsf{otr} = (\vec{\boldsymbol{\chi}_1}, \vec{\boldsymbol{\chi}}_2) \in \mathbb{G}_2^4$, the inner product of $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}$ can be computed as

$$\mathsf{e}(\vec{\boldsymbol{m}}_0 \cdot \mathbf{M}_0, \vec{\boldsymbol{\chi}_1}) \cdot \mathsf{e}(\vec{\boldsymbol{m}}_1 \cdot \mathbf{M}_1, \vec{\boldsymbol{\chi}_2}).$$

Here $\vec{\boldsymbol{m}}_0 \cdot \mathbf{M}_0$ computes inner products for each vector component of $\mathbf{M}_0$ and results in a vector of two group elements in $\mathbb{G}_1$, and $\mathsf{e}(\mathbf{M}_0 \cdot \vec{\boldsymbol{m}}_0, \vec{\boldsymbol{\chi}_1})$ takes the inner product on the exponent of the two vectors. $\mathsf{e}(\vec{\boldsymbol{m}}_1 \cdot \mathbf{M}_1, \vec{\boldsymbol{\chi}_2})$ is computed in the same way. The same approach can be applied to compute $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n$.

The computational cost of $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}$ in the basic construction includes $4n$ pairing operations and $4n$ multiplications in $\mathbb{G}_T$. By using the above technique, this cost can be reduced to 4 pairing operations, $4n$ multiplications in $\mathbb{G}_1$, and 3 multiplications in $\mathbb{G}_T$. The same improvement applies to each inner product $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n$. Therefore, the total computational cost of the sender is reduced to $4n$ pairing operations, $4n^2$ multiplications in $\mathbb{G}_1$, and $3n$ multiplications in $\mathbb{G}_T$.

On Sliding-Window Construction. The same technique can be applied on the sliding-window construction and the improvements on $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}$ is the same as above. The total cost of computing $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_1, \ldots, \vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{y}}_n$ in the sliding-window construction includes $6n$ pairing operations and $(2n^2 + 3n)$ multiplications in $\mathbb{G}_T$. This can be improved to $4n$ pairing operations, $4n^2$ multiplications in $\mathbb{G}_1$, and $3n$ multiplications in $\mathbb{G}_T$.

6.2 Increasing Vector Dimension

Reducing Hash Value Size. The hash value $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}$ currently contains a single group element in $\mathbb{G}_T$. Since the bit representation of group elements in $\mathbb{G}_T$ is much longer than group elements in $\mathbb{G}_1$, we can reduce that by sending 4 group elements in $\mathbb{G}_1$, namely $\vec{\boldsymbol{m}}_0 \cdot \mathbf{M}_0$ and $\vec{\boldsymbol{m}}_1 \cdot \mathbf{M}_1$, and then let the receiver perform the remaining pairing operations. In applications such as PIR and PSI, the sender message grows with the tree depth and this saving in communication gets accumulated throughout all the levels of the tree. Another benefit of this optimization is that it pushes the pairing operations in computing hashes to the receiver side, which significantly reduces the computational cost in computing hashes because the sender had to compute hashes in every node of the tree while the receiver only needs to compute hashes along a single path of the tree.

Next we discuss another technique to further reduce the cost to 3 group elements in $\mathbb{G}_1$.

On Basic Construction. At a high-level, we will unify $\vec{\boldsymbol{f}}$ and $\vec{\boldsymbol{h}}$ to a single vector by increasing the vector dimension from 2 to 3. In more detail, the base hash key $\mathbf{M}$ is the same as before except that each $\vec{\boldsymbol{r}}_i \xleftarrow{\$} \mathbb{Z}_p^3$ is of dimension 3. The receiver's reusable message is redefined by

$$\vec{\boldsymbol{\nu}}_1 := [g^{p_1 \vec{\boldsymbol{r}_1} + \vec{\boldsymbol{u}}}, g^{p_1 \vec{\boldsymbol{r}_2}}, \cdots, g^{p_1 \vec{\boldsymbol{r}}_n} \mid g^{p_1 \vec{\boldsymbol{r}}_{n+1} + \vec{\boldsymbol{v}}}, g^{p_1 \vec{\boldsymbol{r}}_{n+2}}, \cdots, g^{p_1 \vec{\boldsymbol{r}}_{2n}}]$$

$$\vdots$$

$$\vec{\boldsymbol{\nu}}_n := [g^{p_n \vec{\boldsymbol{r}}_1}, g^{p_n \vec{\boldsymbol{r}}_2}, \cdots, g^{p_n \vec{\boldsymbol{r}}_n + \vec{\boldsymbol{u}}} \mid g^{p_n \vec{\boldsymbol{r}}_{n+1}}, g^{p_n \vec{\boldsymbol{r}}_{n+2}}, \cdots, g^{p_n \vec{\boldsymbol{r}}_{2n} + \vec{\boldsymbol{v}}}],$$

where all p_i's are random exponents and $\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \xleftarrow{\$} \mathbb{Z}_p^3$. For a choice bit b, the receiver samples a single random vector $\vec{\boldsymbol{f}}$ s.t. $\langle \vec{\boldsymbol{u}}, \vec{\boldsymbol{f}} \rangle = 1 - b$ and $\langle \vec{\boldsymbol{v}}, \vec{\boldsymbol{f}} \rangle = b$, and sends a single vector $\vec{\boldsymbol{\chi}} = h^{\vec{\boldsymbol{f}}} \in \mathbb{G}_2^3$.

Next the sender computes $\vec{\boldsymbol{hk}}$ by taking the inner product in the exponent of $\mathbf{M}$ and $\vec{\boldsymbol{\chi}}$. The matrix $\mathbf{IK}$ can be computed by taking the inner product in the exponent of $\vec{\boldsymbol{\nu}}_j$'s and $\vec{\boldsymbol{\chi}}$. We can use delayed pairing to compute $\vec{\boldsymbol{m}} \cdot \vec{\boldsymbol{hk}}$ by

$$\mathsf{e}(\vec{\boldsymbol{m}} \cdot \mathbf{M}, \vec{\boldsymbol{\chi}}).$$

Again, we can reduce the hash value size by sending 3 group elements in the vector $\vec{\boldsymbol{m}} \cdot \mathbf{M}$ and postpone the pairing operations to the receiver side. It also reduces the receiver's non-reusable message from 4 group elements in $\mathbb{G}_2$ to 3.

To summarize, the receiver's reusable message is increased from $(4n^2 + 4n)$ to $(6n^2 + 6n)$ group elements in $\mathbb{G}_1$, but the non-reusable message is reduced from 4 to 3 group elements in $\mathbb{G}_2$. The hash value in the sender's message is reduced from 1 group element in $\mathbb{G}_T$ to 3 group elements in $\mathbb{G}_1$.

On Sliding-Window Construction. The same technique can be applied on the sliding-window construction and the improvements on the communication is the same as above. In particular, the receiver's reusable message is increased from $10n$ to $15n$ group elements in $\mathbb{G}_1$, but the non-reusable message is reduced from 4 to 3 group elements in $\mathbb{G}_2$. The hash value in the sender's message is reduced from 1 group element in $\mathbb{G}_T$ to 3 group elements in $\mathbb{G}_1$.

7 Applications

In this section, we discuss several applications of our amortized rate-1 OT and focus on the communication improvements over prior work. For certain applications, we will discuss optimizations that further improve the communication and/or computational complexity. The communication improvements are summarized in Table 3 at the end of the section.

7.1 Secure Function Evaluation on Branching Programs

The work of Ishai and Paskin [IP07] presents an approach to two-round secure function evaluation (SFE) on (oblivious) branching program (BP) from rate-1 OT where the communication complexity only grows with the depth of the branching program instead of its size. In particular, consider a sender holding a private branching program P and a receiver holding a private input x. They can jointly compute $P(x)$ in two rounds of communication, that is, the receiver first sends an encryption c of the input x to the sender, and the the sender can compute a succinct ciphertext c' which allows the receiver to decrypt $P(x)$ without revealing any further information about P except its depth. The size of c' depends polynomially on the size of x and the depth of P, but does not further depend on the size of P.

In terms of concrete communication complexity, let ℓ be the depth of the oblivious BP and h be the bit length of the output. The recent work of Garg et al. [GHO20] achieves receiver's

communication complexity of $O(\ell \cdot (h + \lambda \cdot \ell))$ group elements and sender's communication complexity of $O(h + \lambda \cdot \ell)$ bits, where the group elements are from a pairing-free group where the power DDH assumption holds. This improves upon prior work of Döttling et al. [DGI+19] based on DDH with receiver's communication complexity of $O(\ell \cdot (h + \lambda \cdot \ell)^2)$ group elements and sender's communication complexity of $O(h + \lambda \cdot \ell)$ bits.

In this work, we consider the problem in the *reusable* setting where the receiver first sends a one-time reusable message to the sender consisting of $O(h + \lambda \cdot \ell)$ group elements in $\mathbb{G}_1$. Afterwards, for any oblivious BP with depth ℓ and output length h and any input x, the receiver's communication complexity is $O(\ell)$ group elements in $\mathbb{G}_2$ and the sender's communication complexity is $O(h + \lambda \cdot \ell)$ bits. Note that the one-time messages can be reused for arbitrary polynomially many times.

Example: Secure Inference of Decision Trees. As an example, we consider a server holding a machine learning model of a decision tree, which takes as input a data point with multiple features. Starting from the root, each node of the tree is a function on some feature (e.g. testing if $x < 10$, $t = \texttt{true}$) that determines whether to go to the left or right child. The client has a single data point and would like to perform a secure inference with the server on the decision tree. The decision tree can be formalized as a branching program and two-round secure inference can be achieved by two-round SFE described above, where the communication only grows with the depth of the tree.

7.2 PSI and PIR

In this section, we illustrate several useful applications that can be viewed as special cases of SFE on oblivious BP, hence they achieve the same improvements over prior work.

Unbalanced Private Set Intersection (PSI) Consider the PSI problem between a server holding a private set $X = \{x_1, \ldots, x_N\}$ and a client holding a private set $Y = \{y_1, \ldots, y_m\}$. They want to jointly compute the set intersection $X \cap Y$ without revealing any other information. Without loss of generality we assume all the set elements $x_i, y_j \in \{0,1\}^\lambda$.[3] We focus on the case with *unbalanced* set sizes, namely $N \gg m$, and present a solution for two-round PSI.

To learn the intersection $X \cap \{y\}$ for any $y \in Y$, we can construct an oblivious BP with depth λ and size $\lambda \cdot N$. To construct the oblivious BP, we can first think of it as a full binary tree of depth λ where each leaf node indicates whether the root-to-leaf path is an element in X. However, this branching program has exponential size. We can prune the full binary tree by replacing each subtree consisting of only 0's with a "dummy node" of the same depth. A dummy node of depth d is connected to two dummy nodes with depth $d - 1$.

Following this approach, the client only needs to performs m instances of SFE on the oblivious BP to learn the intersection $X \cap \{y\}$ for every $y \in Y$. The oblivious BP has depth $\ell = \lambda$, size $\lambda \cdot N$, and single-bit output.

Private Set Intersection (PIR) Consider a server (sender) holding a large database $D \in \{0,1\}^N$ and a client (receiver) who wants to retrieve $D[i]$ for $i \in [N]$ without revealing i to the server. As pointed out in [IP07], single-server two-round PIR can be viewed as two-round SFE on an oblivious BP with depth $\ell = \log N$ and single-bit output.

[3] The set elements can be of arbitrary length, but the parties can first apply a collision-resistant hash function on the elements to make them all have length λ.

PIR-with-Default Consider a PIR variant where the server holds N binary strings $s_1, \ldots, s_N \in \{0,1\}^t$ along with N values $v_1, \ldots, v_N \in \{0,1\}^k$. The server additionally holds a default value $v_{\mathsf{dflt}} \in \{0,1\}^k$. The client holds a binary string $w \in \{0,1\}^t$ and wants to learn a value v such that if $w = s_j$ for some $j \in [N]$, then $v = v_j$; otherwise $v = v_{\mathsf{dflt}}$, without revealing any information about w to the server. This problem is formalized by Lepoint et al. [LPR+20]. Two-round PIR-with-Default can be viewed as two-round SFE on a k-bit output oblivious BP with depth t and polynomial size. Hence the receiver and sender communication follow generically from oblivious BP with many-bit outputs. We mention this PIR variant because it will be used to construct PSI-Cardinality.

PSI-Cardinality Consider a PSI variant where a server holding a private set $X = \{x_1, \ldots, x_N\}$ and a client holding a private set $Y = \{y_1, \ldots, y_m\}$ want to learn the cardinality of the intersection $|X \cap Y|$ instead of the intersection itself.

We can achieve PSI-Cardinality by the client querying PIR-with-Default on every element in the her set, where in each PIR-with-Default instance, the default value v^i_{dflt} is sampled at random such that all the default values sum up to 0, namely $\sum_{i=1}^m v^i_{\mathsf{dflt}} = 0$. All the non-default values in a single instance are set to $v^i_{\mathsf{dflt}} + 1$. At the end, the client sums up all the values retrieved from the PIR-with-Default instances. Similar to PSI, we should prune the full binary tree to obtain an oblivious BP with depth λ and polynomial size.

7.3 Optimization for PSI and PSI-Cardinality

We design optimizations for unbalanced PSI and PSI-Cardinality so as to achieve better communication than the above generic approaches.

Optimized PSI Note that the aforementioned oblivious BP for PSI has depth $\ell = \lambda$. To further improve the communication complexity, we replace small subtrees by small instances of two-round PSI (e.g. DDH-based PSI [HFH99]), which we denote by Π_{PSI}.

In particular, to compute $X \cap \{y\}$, the server first hashes his N elements into N random bins. We know that each bin has at most $O(\log N)$ elements. The client computes the same hash on y to identify the bin b that could possibly contain an element y. Now the client queries the server with PIR-with-Default on a string b. The client additionally sends the round-1 message of the two-round PSI protocol Π_{PSI} on a single element y. The server then computes a round-2 message of Π_{PSI} for each bin with elements in that bin. The server views his database for PIR-with-Default as all the N indices of the bins along with the associated values being the round-2 messages of Π_{PSI}, and generates the response for PIR-with-Default. Finally, the client first recovers the round-2 message of Π_{PSI} from PIR-with-Default, and then recovers the output of Π_{PSI}, namely $X \cap \{y\}$.

The receiver's reusable communication is reduced from $O(\lambda^2)$ to $O(\lambda \cdot \log N)$ group elements in $\mathbb{G}_1$. Then for each $X \cap \{y\}$ query, her online communication is reduced from $O(\lambda)$ to $O(\log N)$ group elements in $\mathbb{G}_2$. The sender's communication is reduced from $O(\lambda^2)$ to $O(\lambda \cdot \log N)$.

PSI-Cardinality We can optimize the PSI-Cardinality protocol by replacing small subtrees by small instances of two-round PSI-Cardinality (e.g. DDH-based PSI-Cardinality [IKN+20]), similarly as in the above PSI protocol. However, this would reveal which elements are in the intersection and which are not.

- Server has a set X of size N, client has a single element y. All the elements are λ-bit strings.
- Let $h : \{0,1\}^\lambda \to [N]$ be a hash function.
- Let Π_{PSI} and $\Pi_{\mathsf{PIR\text{-}Default}}$ be two-round PSI and PIR-with-Default protocols, respectively.

Round 1: Client does the following:

1. Compute $b := H(y)$.
2. Compute round-1 message of Π_{PSI} with a single element y, and round-1 message of $\Pi_{\mathsf{PIR\text{-}Default}}$ with query b, and send them to the server.

Round 2: Server does the following:

1. Let $B[i] := \emptyset$ for each bin $i \in [N]$.
2. For each $j \in [N]$, compute $b_j := H(x_j)$ and let $B[b_j] := B[b_j] \cup \{x_j\}$.
3. For each bin $i \in [N]$:
   (a) Pad $B[i]$ with dummy elements to be a total of $\log N$ elements.
   (b) Based on the round-1 message of Π_{PSI}, compute round-2 message of Π_{PSI} with set $B[i]$. Let the round-2 message be M_i.
4. Based on the round-1 message of $\Pi_{\mathsf{PIR\text{-}Default}}$, compute round-2 message of $\Pi_{\mathsf{PIR\text{-}Default}}$ with N values $M_1, \ldots, M_N$, and send it to the client.

Output: Client does the following:

1. Compute the output of the $\Pi_{\mathsf{PIR\text{-}Default}}$, which gives a round-2 message of Π_{PSI}, namely M_b.
2. Use the round-2 message M_b of Π_{PSI} to compute the PSI output.

Figure 1: Optimized two-round PSI protocol with a single element on the client side.

Nonetheless, we notice that in our reusable rate-1 OT protocol, any OT response from the sender can be decrypted by the receiver using the same secret state str, and the receiver cannot distinguish between different responses. Therefore, the server can randomly shuffle the responses for all the PIR-with-Default instances so that the client can only learn the cardinality of the intersection. This achieves the same improvement as in the above PSI protocol.

7.4 Other Variants of PSI and PIR

In this section, we discuss a few more useful variants of PSI and PIR problems.

PIR-by-Keywords Consider a PIR variant where the server holds N binary strings $s_1, \ldots, s_N \in \{0,1\}^t$. The client holds a binary string $w \in \{0,1\}^t$, who wants to learn whether $w = s_j$ for some $j \in [N]$ without revealing any information about w to the server. This problem was introduced by Chor et al. [CGN98]. As pointed out in [IP07], two-round PIR-by-Keywords can be viewed as two-round SFE on a branching program with depth $\ell = t$ and single-bit-output.

PSI-Sum Consider a server holding a set with weights $(X, W) = \{(x_1, w_1), \ldots, (x_N, w_N)\}$ and a client holding a set $Y = \{y_1, \ldots, y_m\}$. They want to jointly compute the PSI-Cardinality along

with the sum of the weights associated with the elements in the intersection, namely $\sum_{i:x_i \in Y} w_i$. This functionality, introduced by Ion et al. [IKN+20], is a generalization of PSI-Cardinality.

We can achieve PSI-Sum from PIR-with-Default similarly as in the PSI-Cardinality protocol except that all the non-default values v_j in a single instance are set to $v^i_{\mathsf{dflt}} + w_j$ where w_j is the corresponding weight. Note that this approach additionally hides the PSI-Cardinality and only reveals the PSI-Sum.

PSI-Test Consider a PSI variant where a server holding a private set $X = \{x_1, \ldots, x_N\}$ and a client holding a private set $Y = \{y_1, \ldots, y_m\}$ want to learn whether the two sets intersect or not, namely whether $|X \cap Y| = \emptyset$.

We can achieve this from PIR-with-Default similarly as in PSI-Cardinality but all the non-default values in a single instance are all set to $v^i_{\mathsf{dflt}} + r^i$ for some random r^i. At the end, the client checks if all the values obtained from the PIR-with-Default instances sum up to 0. The sum equals 0 if and only if $|X \cap Y| = \emptyset$ except with negligible probability.

Extended-PIR-with-Default An extension to PIR-with-Default, also formalized in [LPR+20], enables two parties to learn random shares of the PIR-with-Default answer multiplied with a weight w supplied from the client. By using the techniques from [LPR+20], we can achieve the same complexity as PIR-with-Default with additively homomorphic encryption. In particular, we make the following changes to the PIR-with-Default protocol. The client additionally sends $\mathsf{Enc}(w)$ to the server (in the online phase) where Enc is an additively homomorphic encryption scheme. The server picks a random value α as his output of Extended-PIR-with-Default and replaces each value v in a leaf node of the PIR-with-Default tree by $\mathsf{Enc}(v \cdot w - \alpha)$. Finally the client needs to decrypt her output from PIR-with-Default to recover her output for Extended-PIR-with-Default. We mention this PIR variant because it will be useful in the following application.

Private Join and Compute (PJC) for Inner Product Consider a server holding a set with weights $(X, W) = \{(x_1, w_1), \ldots, (x_N, w_N)\}$ and a client also holding a set with weights $Y = \{(y_1, v_1), \ldots, (y_m, v_m)\}$. They want to jointly compute the $\sum_{i,j:x_i=y_j} w_i \cdot v_j$. This functionality, introduced by Lepoint et al. [LPR+20], is a generalization of PSI-Sum.

We can achieve this by the client querying Extended-PIR-with-Default on every element in her set, where in each Extended-PIR-with-Default instance, the default values are set to 0 and the two parties learn a secret share of $w_i \cdot v_j$ if $X \cap \{y_j\} \neq \emptyset$. From this the two parties can sum up their own shares to obtain a secret sharing of the the inner product result. The server only needs to additionally send the sum of his shares to the client, from which the client can recover the output. Note that this approach additionally hides the PSI-Cardinality and only reveals the result of the inner product.

8 Amortized Rate-1 OT with Strong Sender Privacy

We will now show that variants of our amortized rate-1 OT constructions satisfy a stronger sender privacy requirement, essential for secure computation on non-oblivious branching programs, as required in [IP07].

Application	Receiver Comm [GHO20]	Receiver Comm Ours (reusable)	Receiver Comm Ours (online)	Sender Comm (same)
SFE on oblivious BP	$O(\ell \cdot (h + \lambda \cdot \ell))$ $\mathbb{G}$	$O(h + \lambda \cdot \ell)$ $\mathbb{G}_1$	$O(\ell)$ $\mathbb{G}_2$	$O(h + \lambda \cdot \ell)$
PSI/PSI-Cardinality/ PSI-Sum/PJC/PSI-Test	$O(\lambda^3 \cdot m)$ $\mathbb{G}$	$O(\lambda^2)$ $\mathbb{G}_1$	$O(\lambda \cdot m)$ $\mathbb{G}_2$	$O(\lambda^2 \cdot m)$
Optimized PSI/ Optimized PSI-Cardinality	$O(\lambda^2 \cdot \log N \cdot m)$ $\mathbb{G}$	$O(\lambda \cdot \log N)$ $\mathbb{G}_1$	$O(\log N \cdot m)$ $\mathbb{G}_2$	$O(\lambda \cdot \log N \cdot m)$
PIR	$O(\lambda \cdot \log^2 N)$ $\mathbb{G}$	$O(\lambda \cdot \log N)$ $\mathbb{G}_1$	$O(\log N)$ $\mathbb{G}_2$	$O(\lambda \cdot \log N)$
PIR-by-Keywords	$O(\lambda \cdot t^2)$ $\mathbb{G}$	$O(\lambda \cdot t)$ $\mathbb{G}_1$	$O(t)$ $\mathbb{G}_2$	$O(\lambda \cdot t)$
(Extended-)PIR-with-Default	$O(t \cdot (k + \lambda \cdot t))$ $\mathbb{G}$	$O(k + \lambda \cdot t)$ $\mathbb{G}_1$	$O(t)$ $\mathbb{G}_2$	$O(k + \lambda \cdot t)$

Table 3: Summary of communication complexity in various applications of rate-1 OT. We compare our work based on bilinear power DDH with the state-of-the-art rate-1 OT based on power DDH [GHO20], and show improvements in terms of the receiver's communication while the sender's communication remain the same. Recall that ℓ is the depth of the oblivious BP and h is the output length in bits, m is the client's set size in PSI, N is the server's set size in PSI and the size of database in PIR, t is the length of the keywords in PIR-by-Keywords, k is the output length in PIR-with-Default. In all the applications, the one-time reusable message sent by the receiver can be reused for arbitrary polynomially many times.

Definition 8.1 (Strong sender privacy [IP07])**.** *Let* $\mathsf{OT} := (\mathsf{PreP}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *be as in Definition 3.2. We say* OT *provides strong sender privacy if there exists a PPT algorithm* OTSim *such that for any bit* b *and any pair of messages* (m_0, m_1)*, sampling* $(\mathsf{str}, \mathsf{prm}) \xleftarrow{\$} \mathsf{PreP}(1^\lambda)$ *and* $\mathsf{otr} \xleftarrow{\$} \mathsf{OT}_1(\mathsf{str}, b)$*, the two distributions* $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1))$ *and* $\mathsf{OTSim}(\mathsf{prm}, m_b)$ *are statistically close.*

Our amortized rate-1 OT constructions, as presented in Sections 4,5, do not provide strong sender privacy, because OT_2 is deterministic. Thus, we will consider a randomized OT_2 version of these constructions, obtained by using random extractors and PRGs, as explained in Section 3.1. Under these new OT_2 algorithms of our constructions, the following holds: for any choice b and any two pairs (m_0, m_1) and (m'_0, m'_1) such that $m_b = m'_b$, any $\mathsf{otr} \in \mathsf{OT}_1(\mathsf{str}, b)$, $\mathsf{otr}' \in \mathsf{OT}_1(\mathsf{str}, b')$, the two distributions $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1))$ and $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}'), (m'_0, m'_1))$ are statistically close. The simulation algorithm OTSim, which is only given m_b, should somehow sample from $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_0, m_1))$. By what just mentioned, OTSim may, instead, sample from $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_b, m_b))$. The main challenge in doing so is that OTSim is only given (prm, m_b), and not otr, which in turn is sampled based on str, not known to OTSim. Luckily, in our proofs we showed an oblivious way of sampling from OT_1 without knowing $\mathsf{str} := (\vec{\boldsymbol{u}}, \dots)$. In particular, assuming OTSim is given $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ sampled as $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) \xleftarrow{\$} \mathbf{Dual}(\vec{\boldsymbol{u}})$ (Definition 4.4), then $\mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, b)$ (Definition 4.5) samples an output statistically close to the output of $\mathsf{OT}_1(\mathsf{str}, b)$ (Lemma 4.7). We may include $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ in prm without harming security, as argued in the security of the constructions.

Once $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}})$ is included as part of prm, the output of $\mathsf{OTSim}(\mathsf{prm}, m_b)$ is formed as follows: sample $\mathsf{otr} \xleftarrow{\$} \mathsf{Sim}(\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, 0)$ and return $\mathsf{OT}_2((\mathsf{prm}, \mathsf{otr}), (m_b, m_b))$. In terms of efficiency, the size of otr remains the same, and the size of prm is increased by four group elements in $\mathbb{G}_1$.

References

[ADT11] G. Ateniese, E. De Cristofaro, and G. Tsudik. (If) size matters: Size-hiding private set intersection. In *PKC 2011*, *LNCS* 6571, pages 156–173, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany. 6, 7

[AIR01] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, *LNCS* 2045, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. 2

[APP] Password Monitoring – Apple Platform Security. `https://support.apple.com/en-al/guide/security/sec78e79fc3b/web`. 6, 7

[BBD+20] Z. Brakerski, P. Branco, N. Döttling, S. Garg, and G. Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In *TCC 2020, Part I*, *LNCS* 12550, pages 58–87, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. 7, 11

[Bea96] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 6

[BGdMM05] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `https://eprint.iacr.org/2005/417`. 4, 10

[BGI16] E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, *LNCS* 9814, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 6, 7, 11

[BGI+17] S. Badrinarayanan, S. Garg, Y. Ishai, A. Sahai, and A. Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT 2017, Part III*, *LNCS* 10626, pages 275–303, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 11

[BKM20] Z. Brakerski, V. Koppula, and T. Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO 2020, Part III*, *LNCS* 12172, pages 738–767, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 2

[BLSV18] Z. Brakerski, A. Lombardi, G. Segev, and V. Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In *EUROCRYPT 2018, Part I*, *LNCS* 10820, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 6

[CCF+20] J. Chan, L. P. Cox, D. P. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. M. Kakade, T. Kohno, J. Langford, J. Larson, P. Sharma, S. Singanamalla, J. E. Sunshine, and S. Tessaro. PACT: privacy-sensitive protocols and mechanisms for mobile contact tracing. *IEEE Data Eng. Bull.*, 2020. 6

[CDG+17] C. Cho, N. Döttling, S. Garg, D. Gupta, P. Miao, and A. Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO 2017, Part II*, *LNCS* 10402, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 6

[CGN98] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003, 1998. https://eprint.iacr.org/1998/003. 22

[CLR17] H. Chen, K. Laine, and P. Rindal. Fast private set intersection from homomorphic encryption. In *ACM CCS 2017*, pages 1243–1255, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 7

[CM20] M. Chase and P. Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In *CRYPTO 2020, Part III*, *LNCS* 12172, pages 34–63, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 7

[DG17] N. Döttling and S. Garg. Identity-based encryption from the Diffie-Hellman assumption. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 6

[DGH+20] N. Döttling, S. Garg, M. Hajiabadi, D. Masny, and D. Wichs. Two-round oblivious transfer from CDH or LPN. In *EUROCRYPT 2020, Part II*, *LNCS* 12106, pages 768–797, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 2

[DGI+19] N. Döttling, S. Garg, Y. Ishai, G. Malavolta, T. Mour, and R. Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 2, 3, 4, 5, 6, 7, 11, 20

[EGL82] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA. 2

[GGH19] S. Garg, R. Gay, and M. Hajiabadi. New techniques for efficient trapdoor functions and applications. In *EUROCRYPT 2019, Part III*, *LNCS* 11478, pages 33–63, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 6

[GH18] S. Garg and M. Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 362–391, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 6

[GHO20] S. Garg, M. Hajiabadi, and R. Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In *TCC 2020, Part I*, *LNCS* 12550, pages 88–116, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. 3, 4, 5, 6, 9, 15, 19, 24

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 2

[GVW20] R. Goyal, S. Vusirikala, and B. Waters. New constructions of hinting PRGs, OWFs with encryption, and more. In *CRYPTO 2020, Part I*, *LNCS* 12170, pages 527–558, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 6

[HEK12] Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*, San Diego, CA, USA, February 5–8, 2012. The Internet Society. 7

[HFH99] B. A. Huberman, M. K. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 78–86. ACM, 1999. 6, 21

[HK12] S. Halevi and Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. 2

[HKW20] S. Hohenberger, V. Koppula, and B. Waters. Chosen ciphertext security from injective trapdoor functions. In *CRYPTO 2020, Part I*, *LNCS* 12170, pages 836–866, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 6

[IKN+20] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 370–389. IEEE, 2020. 6, 21, 23

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, *LNCS* 2729, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 6

[IP07] Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *TCC 2007*, *LNCS* 4392, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 2, 4, 5, 19, 20, 22, 23, 24

[KKRT16] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *ACM CCS 2016*, pages 818–829, Vienna, Austria, October 24–28, 2016. ACM Press. 7

[KMT19] F. Kitagawa, T. Matsuda, and K. Tanaka. CCA security and trapdoor functions via key-dependent-message security. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 33–64, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 6

[KRS+19] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert. Mobile private contact discovery at scale. In *USENIX Security*, 2019. 6, 7

[KW19] V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *CRYPTO 2019, Part II*, *LNCS* 11693, pages 671–700, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 6

[LPR+20] T. Lepoint, S. Patel, M. Raykova, K. Seth, and N. Trieu. Private join and compute from PIR with default. Cryptology ePrint Archive, Report 2020/1011, 2020. https://eprint.iacr.org/2020/1011. 21, 23

[LQR+19] A. Lombardi, W. Quach, R. D. Rothblum, D. Wichs, and D. J. Wu. New constructions of reusable designated-verifier NIZKs. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 670–700, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 6

[MIC] Password Monitor: Safeguarding passwords in Microsoft Edge. https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/. 6, 7

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457, Washington, DC, USA, January 7–9, 2001. ACM-SIAM. 2

[PRTY19] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 401–431, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 7

[PRTY20] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. PSI from PaXoS: Fast, malicious private set intersection. In *EUROCRYPT 2020, Part II*, *LNCS* 12106, pages 739–767, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 7

[PSSZ15] B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *USENIX Security 2015*, pages 515–530, Washington, DC, USA, August 12–14, 2015. USENIX Association. 7

[PSTY19] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai. Efficient circuit-based PSI with linear communication. In *EUROCRYPT 2019, Part III*, *LNCS* 11478, pages 122–153, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 7

[PSWW18] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder. Efficient circuit-based PSI via cuckoo hashing. In *EUROCRYPT 2018, Part III*, *LNCS* 10822, pages 125–157, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 7

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, *LNCS* 5157, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 2

[Rab05] M. O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. https://eprint.iacr.org/2005/187. 1

[RS21] P. Rindal and P. Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In *Advances in Cryptology - EUROCRYPT 2021, International Conference on the Theory and Applications of Cryptographic Techniques*, 2021. 7

[TPY+19] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh, and E. Bursztein. Protecting accounts from credential stuffing with password breach alerting. In *USENIX Security*, 2019. 6, 7

[TSS+20] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song. Epione: Lightweight contact tracing with strong privacy. *IEEE Data Eng. Bull.*, 2020. 6