Trapdoor Hash Functions and Their Applications

Nico Döttling* Sanjam Garg† Yuval Ishai‡ Giulio Malavolta§ Tamer Mour¶
Rafail Ostrovsky‖

Abstract

We introduce a new primitive, called *trapdoor hash functions* (TDH), which are hash functions $\mathsf{H} : \{0,1\}^n \to \{0,1\}^\lambda$ with additional trapdoor function-like properties. Specifically, given an index $i \in [n]$, TDHs allow for sampling an encoding key ek (that hides i) along with a corresponding trapdoor. Furthermore, given $\mathsf{H}(x)$, a hint value $\mathsf{E}(\mathsf{ek}, x)$, and the trapdoor corresponding to ek, the i^{th} bit of x can be efficiently recovered. In this setting, one of our main questions is: How small can the hint value $\mathsf{E}(\mathsf{ek}, x)$ be? We obtain constructions where the hint is only *one bit* long based on DDH, QR, DCR, or LWE.

This primitive opens a floodgate of applications for low-communication secure computation. We mainly focus on two-message protocols between a receiver and a sender, with private inputs x and y, resp., where the receiver should learn $f(x, y)$. We wish to optimize the *(download) rate* of such protocols, namely the asymptotic ratio between the size of the output and the sender's message. Using TDHs, we obtain:

1. The first protocols for (two-message) *rate-1 string OT* based on DDH, QR, or LWE. This has several useful consequences, such as:
   (a) The first constructions of PIR with communication cost poly-logarithmic in the database size based on DDH or QR. These protocols are in fact rate-1 when considering block PIR.
   (b) The first constructions of a *semi-compact* homomorphic encryption scheme for branching programs, where the encrypted output grows only with the program *length*, based on DDH or QR.
   (c) The first constructions of lossy trapdoor functions with input to output ratio approaching 1 based on DDH, QR or LWE.
   (d) The first *constant-rate* LWE-based construction of a 2-message "statistically sender-private" OT protocol in the plain model.
2. The first *rate-1* protocols (under any assumption) for n parallel OTs and matrix-vector products from DDH, QR or LWE.

We further consider the setting where f evaluates a RAM program y with running time $T \ll |x|$ on x. We obtain the first protocols with communication sublinear in the size of x, namely $T \cdot \sqrt{|x|}$ or $T \cdot \sqrt[3]{|x|}$, based on DDH or, resp., pairings (and correlated-input secure hash functions).

*CISPA Helmholtz Center for Information Security, doettling@cispa.saarland.
†University of California, Berkeley, sanjamg@berkeley.edu.
‡Technion, yuvali@cs.technion.ac.il.
§Carnegie Mellon University, giulio.malavolta@hotmail.it. Part of the work done while at Friedrich-Alexander-Universität Erlangen-Nürnberg.
¶Weizmann Institute of Science, tamer@weizmann.ac.il. Part of the work done while at UCLA and Technion.
‖University of California, Los Angeles, rafail@cs.ucla.edu.

Contents

1 Introduction

Seminal results from the 1980s [Yao86, GMW87] showed that it is possible for a group of mutually distrustful parties to compute a joint function on their private inputs without revealing anything more than the output of the computation. These foundational results were seen as providing the first theoretical proofs of concept. However, significant theoretical and practical advances over the years provide support for the idea that perhaps secure computation can be as practical and ubiquitous as public-key cryptography.

In the quest to make secure computation efficient, realizing *communication efficient* secure computation protocols has emerged as a central theme of research. Moreover, secure computation protocols with large communication cost can often be prohibitive in practice. Consequently, substantial effort has been put towards realizing communication efficient protocols. Nonetheless, our understanding of communication efficient secure computation protocols remains significantly limited. Specifically, known protocols for circuits with communication independent of the circuit size are only known using fully homomorphic encryption (FHE) [Gen09] and can only be based on variants of LWE. In the two-party case, the communication complexity of such protocols is comparable to the length of the *shorter input* plus the length of the output. For simpler functions that can be represented by log-depth circuits or polynomial-size branching programs, similar protocols were recently constructed from other assumptions such as DDH [BGI16] or a circular-secure flavor of DCR [FGJI17]. Here the communication is comparable to the total length of *both inputs* plus the length of the output.

The above state of affairs leaves several types of gaps between secure and insecure communication complexity.[1] First, even when applying the best known FHE schemes, there is a constant-factor gap for functions whose output length is comparable to (or longer than) the length of the shorter input.[2] Second, the communication gap can be even bigger when considering restricted interaction. For instance, when one input is much shorter than the other, FHE cannot be used to get communication-efficient 2-message protocols where the party holding the *long* input sends the first message. Finally, and most importantly for this work, under standard assumptions other than LWE, the gaps between secure and insecure communication are much bigger, especially when considering functions with unbalanced input sizes.

To illustrate the current gaps, consider the fundamental problem of private information retrieval (PIR) [KO97, CGKS95] over m-bit records, where a client wants to privately learn the i^{th} record of a server's database that consists of n records of length m each. Here, a protocol that achieves near-optimal communication from the server to the client (i.e., roughly m bits) is only known under DCR [DJ01, Lip05]. For the case of retrieving m different 1-bit records, the situation is even worse. In the best known protocol, the gap between the server's message length and the output length is a big multiplicative constant [GKL10]. Finally, even for the case $m = 1$, obtaining polylog(n) communication under (subexponential variants of) standard assumptions such as DDH or QR is open.

[1] It seems plausible that these gaps can be closed using indistinguishability obfuscation [GGH+13]. However, the focus of this work is on constructions that can be based on more traditional assumptions.

[2] A simple "hybrid FHE" technique [GGI+15] can generically convert any FHE scheme into one whose encrypted (long) input is roughly as long as the input. However, no such generic technique is known for compressing the encrypted output.

1.1 Our Setting and Questions of Interest

Setting: Two-Message Secure Computation. We consider two-party protocols in which a receiver and a sender have private inputs x and y, respectively. We consider protocols for evaluating a function $f(x, y)$ using only two messages. First, based on its input x, the receiver sends the first message msg_1 to the sender who, based on its input y, responds with the second message msg_2. Finally, the receiver uses its secret state and msg_2 to compute $f(x, y)$. *Sender's privacy* requires that the receiver learns nothing more about y than $f(x, y)$ and the length of y. Similarly, *receiver's privacy* requires that the sender learns nothing about x other than its length. By default, we only consider security against *semi-honest* parties.[3]

Case I: Large Receiver Output. We are primarily interested in the case where the output of f is long, and define the *download rate* of such a 2-message protocol (or *rate* for short) as the asymptotic ratio between $|f(x, y)|$ and $|\mathsf{msg}_2|$. We will also consider the *overall rate*, defined as the asymptotic ratio between the insecure communication complexity of f and that of the protocol. A fundamental functionality in this regime is oblivious transfer (OT). We start with the special case of *string OT*, implemented via a two-message protocol. Recall that in the string OT functionality the inputs of the sender and receiver are two strings $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}^n$ and a bit $i \in \{0,1\}$, respectively. For this functionality, the receiver's output should be s_i. Here the download and overall rate are the asymptotic ratios $\frac{n}{|\mathsf{msg}_2|}$ and $\frac{n}{|\mathsf{msg}_1|+|\mathsf{msg}_2|}$, respectively, when the security parameter λ tends to infinity and n is a sufficiently big polynomial in λ (see Definition 5.2 for a precise formulation). We also consider *batch OT*; this functionality allows n parallel instances of bit-OT (string OT with 1-bit strings).

Even for the special case of OT, the state-of-the-art with optimal overall rate (or optimal download rate) is quite unsatisfactory.[4] Any 2-message string-OT protocol can be compiled into a similar protocol with rate 1/2 using *hybrid encryption* as follows: Given a string-OT protocol for messages of size λ, the sender uses the OT protocol to transmit one out of two symmetric keys to the receiver, and uses these keys with a rate-1 symmetric encryption scheme to encrypt the actual messages. The two ciphertexts are sent along with the OT sender message. The receiver recovers one of the two keys and decrypts the corresponding ciphertext. However, going beyond rate 1/2 seems to hit barriers! Interestingly enough, for information-theoretic OT in the correlated randomness model, rate 1/2 (as e.g. in Beaver's standard reduction [Bea95a]) is optimal [WW10, IKM+13]. In the computational setting, constructions based on additively homomorphic encryption or homomorphic secret sharing hit a similar barrier [Ste98, BGI17b, JVC18a]. Currently, the only construction of OT known to achieve rate better than 1/2 is based on the Damgård-Jurik cryptosystem [DJ01], which relies on the DCR assumption. Even here, optimal rate in only achieved by undesirably letting the size of the group used in the scheme grow with the size of the inputs.[5] Moreover, in the more general case of *batch OT*, rate 1 could not even be achieved based on DCR. This brings us to our first motivating question:

[3] Our protocols can be efficiently extended to provide security against malicious parties (under the same assumptions) using sublinear arguments [NN01]. This increases the number of rounds, but does not affect the asymptotic communication.

[4] The work of Cho et al. [CDG+17] on *laconic OT* gives a batch OT protocol where $|\mathsf{msg}_1|$ is independent of $|x|$. This generalizes to arbitrary functions f; however, even in the simple case of batch OT the download rate is sub-constant. The same applies to the more recent work of Quach et al. [QWW18] on laconic function evaluation.

[5] In this work, we consider this question in the more stringent two-message setting. However, we note that no other protocols with rate $> \frac{1}{2}$ are known even when additional rounds of communication are allowed.

Can we realize OT with rate $> \frac{1}{2}$ from assumptions other than DCR? Can we realize such batch OT from any assumption?

Why care about OT with rate $> \frac{1}{2}$? As mentioned earlier, there is a large body of work on minimizing the communication complexity of secure computation. The special case of OT is not only natural and useful as a standalone application, but it also serves as a stepping stone for other applications. Indeed, high-rate 2-message OT implies: (i) high-rate PIR with polylogarithmic communication complexity in the number of records [KO97, IP07]; (ii) a semi-compact homomorphic encryption scheme that supports evaluation of bounded-length branching programs (in particular, finite automata, decision trees and OBDDs) over encrypted data [IP07], (iii) a high-rate lossy-trapdoor function [PW08], and (iv) statistically sender-private (SSP) two-message OT with constant rate [BGI+17a, BD18]. We will elaborate on these applications below. To sum up, while high-rate OT is a powerful primitive with important consequences, very little is known about how to construct it.

Case II: Large Receiver Input. Up to this point, we were mainly concerned with functions $f(x, y)$ that have a long output, where our goal was to make the communication from the sender to the receiver very close to $|f(x, y)|$. Even multi-round protocols of this type were not known prior to our work. A second setting we consider applies to two-round protocols in the case where $|x| \gg |y|$ and the output is short. In this case, an insecure protocol for f can simply have the sender communicate y to the receiver. Since secure computation with only one message is impossible (except in trivial cases), our goal is to obtain a *two-message* secure protocol with similar efficiency features, namely where the total communication complexity is comparable to $|y|$ rather than $|x|$. As a motivating example, consider the case where the receiver has a large n-bit database x, the sender's input y describes a small RAM machine M whose running time is $T \ll n$, and the receiver's output is $M(x)$. For instance, M can select a single entry $y \in [n]$ of x, outputting $M(x) = x_y$. Note that a natural FHE-based protocol where the receiver sends an encryption of x and receives an encryption of $M(x)$ does not meet our efficiency goal of using less than n bits of communication. On the other hand, allowing for a higher round complexity, our goal can be met using any PIR protocol [NN01].

From here on, we restrict the attention to 2-message protocols with $o(n)$ communication. The recent *laconic function evaluation* primitive [QWW18] provides such a protocol with overall communication of $|y| + \mathsf{poly}(\lambda, T)$, where λ is a security parameter. However, results in this setting are only known under LWE (with subexponential modulus-to-noise ratio). This brings us to our second main question:

Are there 2-message protocols computing $M(x)$ with $o(n)$ bits of communication from any assumptions other than LWE?

1.2 Our Results

In this work, we introduce a new primitive that we call trapdoor hash functions (TDHs).[6] TDHs are hash functions $\mathsf{H} : \{0,1\}^n \to \{0,1\}^\lambda$ with additional trapdoor-function-like properties. Specifically, given an index $i \in [n]$, TDHs allow for sampling an encoding key ek (that hides i) along with a corresponding trapdoor. Furthermore, given $\mathsf{H}(x)$, a hint value $\mathsf{E}(\mathsf{ek}, x)$, and the trapdoor

[6] The notion of trapdoor hash functions is inspired by the closely related notion of hash encryption [DG17, DGHM18, BLSV18, GH18] and somewhere statistically binding hash functions [HW14, KLW15, OPWW15].

corresponding to ek, the i^{th} bit of x can be efficiently recovered. In this setting, one of our main questions is: how small can the hint value $\mathsf{E}(\mathsf{ek}, x)$ be? We define the rate of TDH as the inverse of the size of the hint.

We obtain constructions of rate-1 TDHs from standard assumptions, namely DDH, QR, DCR, and LWE. The surprising twist in these constructions is the close integration of techniques developed in two very recent and seemingly unrelated lines of investigation: (i) A new type of hash function for constructing identity-based encryption by Döttling and Garg [DG17] and its extension to constructions of trapdoor functions by Garg, Gay and Hajiabadi [GH18, GGH18] and (ii) techniques for homomorphic secret sharing by Boyle, Gilboa and Ishai [BGI16].

Main Result: Rate-1 Two-Message String OT. Our TDHs yield the first construction of string OT with rate-1 from the {DDH, QR, LWE} assumption. Additionally, we get a new construction under DCR, for which, unlike the Damgård-Jurik construction [DJ01], the size of the group used is independent of the size of the inputs. We stress that while our constructions use only two messages; previously, even multi-round constructions with rate $> \frac{1}{2}$ were not known under these assumptions.[7] This allows us to obtain the following new results:

1. *Private Information Retrieval*: We obtain the first constructions of private information retrieval (PIR) from {DDH, QR, LWE} with download rate 1 (for retrieving long records). The total communication complexity grows only logarithmically with the number of records.[8] Previously, such PIR protocols were only known under DCR [Lip05]. This also resolves the longstanding open question of building PIR with polylogarithmic communication (for 1-bit records) from {DDH, QR} [KO97]. Such protocols were only known under DCR, LWE, and the Phi-hiding assumptions [CMS99, Lip05, Cha04, OI07]. For example, the best known construction from DDH required $O(2^{\sqrt{\log n}} \cdot \lambda)$ bits of communication, for database size n and security parameter λ [KO97, Ste98].

2. *Semi-Compact Homomorphic Encryption for Branching Programs:* We obtain the first encryption schemes based on {DDH, QR} that allow evaluating a branching program on an encrypted input, where the encrypted output grows only with the *length* of the branching program and not with its size. Previously, such schemes were only known under {DCR, LWE} [IP07].

3. *Lossy Trapdoor Functions:* We obtain the first construction of lossy trapdoor functions [PW08] with rate 1 from the {DDH, QR, LWE} assumption. Here, rate is defined as the ratio of the input length and output length for the trapdoor function. Very recently, Garg et al. [GGH18] obtained construction from DDH with a small constant rate. However, besides that, no constructions with constant rate were known under these assumptions.

4. *Malicious Statistically Sender Private OT:* We obtain the first LWE-based 2-message OT protocol in the plain model that offers statistical sender privacy against a malicious receiver and has a *constant* rate. This improves over the $1/\log(\lambda)$ rate of a recent LWE-based protocol

[7] Our protocols achieve asymptotic download rate 1, which is clearly optimal. However, the (additive) difference between the sender's message length and the output length grows with the security parameter λ. In the full version we show that that this gap is necessary even in the more liberal setting of secure computation with preprocessing.

[8] More specifically, as our group-based constructions are black-box in the underlying group, we can count the communication complexity in terms of the number of group elements, which in our case is $\log n \cdot \mathsf{poly}(\lambda)$, where n is the size of the database and λ is the security parameter.

of Brakerski and Döttling [BD18]. Similar protocols were previously known under {DDH, DCR} [NP01, AIR01, HK12].

Rate-1 Protocols for Functionalities Generalizing OT. We generalize the techniques for rate-1 OT to yield secure 2-message protocols with *download* rate 1 for other useful functionalities. In these cases, we obtain the first protocols under *any* assumption. We obtain such protocols for the following functionalities.

1. *Batch OT:* Batch OT allows n instances of bit-OT to be performed in parallel. We obtain 2-message batch OT protocol with download rate 1 (but sub-constant upload rate) from QR and LWE. Allowing for inverse polynomial error probability, we obtain a similar protocol under DDH. Protocols with smaller constant download rates (and constant overall rate) were known under a variety of assumptions; see [IKOS08, BGI17b, BMN18] and references therein.

2. *Batch OLE:* An oblivious linear function evaluation (OLE) scheme allows the sender to evaluate an affine function $f(x) = ax + b$ over the receiver's private input x. We obtain the first batch OLE (over either a field of a small characteristic or smooth modulus) with download rate 1 based on QR or LWE. We also get a DDH-based construction if we allow inverse-polynomial error. For the case of fields, smaller constant download rate (and constant overall rate) could be realized under LWE [DHRW16, JVC18a] or code-based assumptions [NP99, IPS09, ADI+17].

3. *Matrix-Vector Products:* We generalize the above to oblivious matrix-vector product evaluation (OMV), where the sender has a matrix M, the receiver holds a vector v, and the output is Mv. A two-message OMV protocol can be thought as a relaxed form of additively homomorphic encryption. Our techniques can be generalized to construct OMVs over $\mathbb{F}_2$ with optimal download rate, based on QR or LWE. We can also generalize the LWE-based construction to fields modulo small primes or smooth integers. Compared to previous LWE-based constructions (e.g., [JVC18a]), we get better (optimal) download rate but worse overall rate.

As mentioned for rate-1 OT, all the aforementioned results were known only under the DCR assumption (and in the case of functionalities generalizing OT, were not known under any standard assumptions), where optimal rate was achieved by letting the size of the group grow with the size of the inputs. Our work improves in this setting. Specifically, assuming only DCR, our work implies all of the above results in groups of size independent of the message length.

As in the context of rate-1 OT, while we consider only two-message protocols, we stress that, prior to our work, none of the above-mentioned results were known even when additional rounds of communication are allowed.

Beyond OT: Two-Message SFE with Sublinear Communication. Armed with our new techniques, we attempt to broaden the class of functionalities for which two-message secure-function evaluation (SFE) can be achieved with sublinear communication. Specifically, we start with the following example setting: Alice would like to share her DNA sequence online so that various medical researchers can use it to provide her with valuable insights about her health. However, Alice wants to keep her "large" genetic information confidential and each researcher wants to keep the specific parts of the genetic code it looks at private. In a bit more detail, Alice wants to publish a hash $h(\mathsf{x})$ of her input x (of length n) online, such that any contractor Bob, with a private machine M with "small running time" (denoted by T) can send Alice a "short" message, enabling her to learn

M^{x}, where M has random access to x. In summary, we are interested in a setting that allows Alice to evaluate Bob's private *small machine* on her private *large input* with sublinear communication.

Positive results for the above setting with sublinear communication are only known from lattice assumptions — namely, using laconic function evaluation [QWW18]. In contrast, for the case of DDH-based constructions, such protocols need communication complexity proportional to n. We note that constructions based on laconic OT [CDG+17] do not keep the locations accessed by M private and thus, do not suffice for this application.[9]

We obtain the first protocol for non-interactive secure computation on large inputs from DDH with communication proportional to $T \cdot \sqrt{n}$, where T is the running time of the machine and n is the size of the database. Furthermore, using pairings (and appropriate correlated-input secure hash functions [IKNP03, GL10, BC10, GOR11, AMN+18]) we obtain a protocol with communication cost proportional to $T \cdot \sqrt[3]{n}$.

Further, in a scenario where Bob's machine M is repeatedly executed over different large inputs (possibly owned by different Alices), we achieve protocols with communication proportional to T, and *independent* of n, per execution, assuming a non-interactive "offline phase" where Bob publishes an "encoding" of M of length proportional to n or $\sqrt{n}$ (from DDH or pairings, resp.), which can be amortized over all executions.

Our results are obtained by constructing a variant of laconic OT [CDG+17], that keeps the locations accessed by M private. We call this primitive *private laconic OT*. The key technical challenge here is to realize this primitive with communication cost sublinear in the size of Alice's large input. By using private laconic OT, rather than laconic OT, in the constructions from in [CDG+17], we obtain SFE for RAM programs with sublinear communication which, as opposed to the protocol from [CDG+17], also hides the access pattern made by the machine to the input database and therefore achieves a full notion of security.

2 Technical Outline

We will now provide an outline of the technical parts of this work.

2.1 Trapdoor Hash Functions

We start by providing an overview of our basic construction for trapdoor hash. Recall that a trapdoor hash scheme (TDH) defines a family of samplable publicly-parameterized hash functions $\mathsf{H}_{\mathsf{hk}} : \{0,1\}^n \to \{0,1\}^\eta$, accompanied with the following three algorithms:

- *Key generation:* given the public hash key, Bob generates a pair of an *encoding key* and a *trapdoor* $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{G}(\mathsf{hk}, i)$, corresponding to a private index $i \in [n]$.
- *Encoding:* using the encoding key ek, Alice, with a private input $\mathsf{x} \in \{0,1\}^n$, can compute a *hint* $\mathsf{e} \leftarrow \mathsf{E}(\mathsf{ek}, \mathsf{x})$, which essentially encodes the bit $\mathsf{x}[i]$.
- *Decoding:* Bob, who has the secret trapdoor td, can now decode any encoding e generated for some input x as above, to recover x_i, given only the hash $\mathsf{H}_{\mathsf{hk}}(\mathsf{x})$. In fact, Bob would be able to generate two encodings $(\mathsf{e}_0, \mathsf{e}_1) \leftarrow \mathsf{D}(\mathsf{td}, \mathsf{h})$, where it is guaranteed that $\mathsf{e} = \mathsf{e}_{\mathsf{x}[i]}$.

[9] For this application, we insist on the two-message setting. Allowing $O(T)$ rounds of interaction, similar protocols can be based on any single-server PIR scheme [NN01].

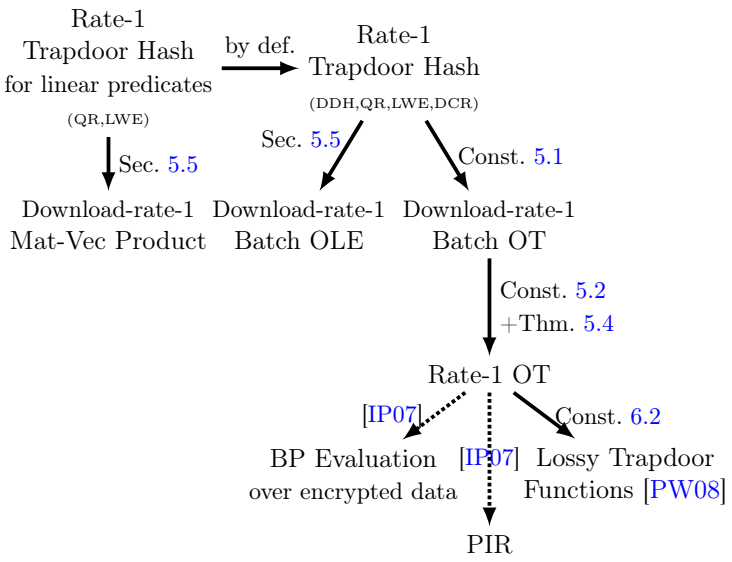

Figure 1: Overview of the results in this work, Part I: optimal-rate protocols for OT-like sender-receiver functionalities and their applications. Dotted lines correspond to corollaries from prior work.

We actually consider a more general notion of TDH where Bob with a private *predicate* $f : \{0,1\}^n \to \{0,1\}$, chosen from a predefined class of predicates $\mathcal{F}$, generates a key ek, using which Alice encodes the bit $f(x)$, and a corresponding trapdoor, using which Bob decodes. Such a scheme is called trapdoor hash for $\mathcal{F}$, and the above special case is referred to as trapdoor hash for *index predicates*.

A formal definition of trapdoor hash can be found in Section 4.1.

For this outline, we focus on TDH for index predicates and think of trapdoor hashing as a protocol where Alice and Bob play the roles of a sender with input x and, respectively, a receiver who wants to learn $\mathsf{x}[i]$. For now, we will mostly focus on receiver privacy (which we refer to as *function privacy* in the general setting), as sender's privacy (generally, *input privacy*) is much easier to achieve. Receiver privacy means that the encoding key ek hides the index i (or, generally, the predicate $f \in \mathcal{F}$). We also require that the hash function will be compressing, specifically that the length of its output is independent of the size of the input. Our main efficiency goal, however, is to construct trapdoor hash where the hint e is as small as possible. The size of e is referred to as the *rate* of the scheme. Our target is to achieve rate-1 trapdoor hash (which is clearly optimal).

We start with our DDH-based construction, which is presented in details in Section 4.2.

The Basic Hash Function. The starting point of this work is the following group-based hash function mapping $\{0,1\}^n$ to a group $\mathbb{G}$

$$\mathsf{H}(\mathbf{A}, \mathsf{x}) = \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}$$

where $\mathsf{x} \in \{0,1\}^n$ is the input and $\mathbf{A} = (g_{j,b})_{j\in[n],b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}^{2\times n}$ is chosen uniformly at random and serves as the hash key hk. By choosing n larger than the representation size of a group element in $\mathbb{G}$,

this function becomes compressing. Collision resistance of this function can be routinely established from the discrete logarithm assumption in $\mathbb{G}$.

This surprisingly powerful function plays a central role in recent constructions of identity based encryption [DG17], trapdoor functions [GH18], deterministic encryption and lossy trapdoor functions [GGH18].

Adding Trapdoors. We show how this function can be made invertible, using techniques of [GGH18]. Clearly, the hash value $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, \mathsf{x})$ is too short to information-theoretically specify x. Thus we will add additional *hints*, which we also call *encodings*, to allow recovery of x. We will first discuss how the receiver can recover a single bit $\mathsf{x}[i]$ of x.

Let $i \in [n]$ be an index of the receiver's choice. The receiver will generate a matrix $\mathbf{B} \in \mathbb{G}^{2\times n}$, that serves as an encoding key ek, such that the following holds for all $\mathsf{x} \in \{0,1\}^n$: If $\mathsf{H}(\mathbf{A}, \mathsf{x}) = \mathsf{h}$, then $\mathsf{H}(\mathbf{B}, \mathsf{x}) = \mathsf{h}^s \cdot g^{\mathsf{x}[i]}$ for some $s \in \mathbb{Z}_p$. We can construct such a matrix $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$ by choosing $s \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random, and setting

$$u_{j,b} = g_{j,b}^s$$

for all $j \neq i$ and

$$u_{i,b} = g_{i,b}^s \cdot g^b. \tag{2.1}$$

Since s is uniform, we immediately get, via the DDH assumption, that all $g_{j,b}^s$ are pseudorandom, and consequently, the matrix $\mathbf{B}$ is pseudorandom as well. Thus, the matrix $\mathbf{B}$ computationally hides the index i.

Given the values $\mathsf{h} = \mathsf{H}(\mathbf{A}, \mathsf{x})$ and the hint $\mathsf{e} = \mathsf{H}(\mathbf{B}, \mathsf{x})$, as well as a trapdoor consisting of s, the receiver can recover x as follows. As by the above property it holds that $\mathsf{e} = \mathsf{h}^s \cdot g^{\mathsf{x}[i]}$, we can recover $\mathsf{x}[i]$ by testing $\mathsf{e} \stackrel{?}{=} \mathsf{h}^s \cdot g^b$ for both $b \in \{0,1\}$ and setting $\mathsf{x}[i] \leftarrow b$ for the b which satisfies this test. While we can construct a trapdoor hash function in this way, its rate will be far from 1: To encode a single bit $\mathsf{x}[i]$ of x, we need to spend one full group element e. Assuming that a group element has size λ, this will give us a construction of rate $1/\lambda$ (see Construction 4.1 for a technical description).

Towards Rate 1. Clearly, sending a group element e to encode a single bit $\mathsf{x}[i]$ is wasteful. However, we make the following observation: The term e can only assume two different values, namely h^s and $\mathsf{h}^s \cdot g$, depending on whether the bit $\mathsf{x}[i]$ is 0 or 1. So what we need is a way for the sender to signal to the receiver that either $\mathsf{e} = \mathsf{h}^s$ or $\mathsf{e} = \mathsf{h}^s \cdot g$, without actually sending e. Yet, since the sender does not know i, he generally does not know whether he is encoding 0 or 1, that is, he does not know whether e is of the form h^s or $\mathsf{h}^s \cdot g$.

However, assume the sender could somehow determine the *distance to a nearby reference point* of e which is insensitive to small perturbations. This would for instance be the case if the group $\mathbb{G}$ had a subgroup $\mathbb{G}'$, such that we can efficiently test membership in $\mathbb{G}'$ and the quotient $\mathbb{G}/\mathbb{G}'$ is of polynomial size. Since $|\mathbb{G}/\mathbb{G}'|$ is only of polynomial size, we can efficiently compute the *distance* to $\mathbb{G}'$ for every $\mathsf{e} \in \mathbb{G}$. That is, the function $\mathsf{Dist}(\mathsf{e})$ which exhaustively searches for the smallest $z \in \mathbb{Z}$ such that $\mathsf{e} \cdot g^z \in \mathbb{G}'$ is efficiently computable. Assuming further for simplicity that $|\mathbb{G}/\mathbb{G}'|$ is even, it holds for every $\mathsf{e} \in \mathbb{G}$ that

$$\mathsf{Dist}(\mathsf{e} \cdot g) \mod 2 = (\mathsf{Dist}(\mathsf{e}) + 1) \mod 2.$$

This means that h^s and $\mathsf{h}^s \cdot g$ never map to the same bit under the function $\mathsf{Dist}(\cdot) \mod 2$. Via this observation, the sender can signal to the receiver whether e is h^s or $\mathsf{h}^s \cdot g$ as follows. Instead of sending e itself to the receiver, he just sends the bit $\hat{\mathsf{e}} = \mathsf{Dist}(\mathsf{e}) \mod 2 \in \{0,1\}$ to the receiver.

Modifying the recovery procedure of above, the receiver can recover $\mathsf{x}[i]$ by testing $\hat{\mathsf{e}} \stackrel{?}{=} \mathsf{Dist}(\mathsf{h}^s g^b) \mod 2$ for $b \in \{0,1\}$ and setting $\mathsf{x}[i] \leftarrow b$ for the b which satisfies this test. This procedure recovers the correct bit $\mathsf{x}[i]$ with $\hat{\mathsf{e}} = \mathsf{Dist}(\mathsf{h}^s \cdot g^{\mathsf{x}[i]})$, as the value e computed by the sender must have been either h^s or $\mathsf{h}^s \cdot g$, and by the above $\mathsf{Dist}(\mathsf{h}^s) \mod 2 \neq \mathsf{Dist}(\mathsf{h}^s \cdot g) \mod 2$.

Achieving Rate 1. Alas, since $\mathbb{G}$ is typically a cyclic group of prime order, it has no non-trivial subgroups. But upon closer inspection, the signalling technique above does not really rely on any additional group structure. All we need is that $\mathsf{Dist}(\mathsf{e} \cdot g) = \mathsf{Dist}(\mathsf{e}) + 1$.

Fortunately, a technique to determine the distance to a reference point was recently proposed by Boyle, Gilboa and Ishai [BGI16]. In a nutshell, instead of computing the distance to a subgroup, we compute the distance to a *moderately dense pseudorandom subset* of $\mathbb{G}$. Such a pseudorandom subset can be succinctly represented via the key of a pseudorandom function by setting S_K to be the set of all points $h \in \mathbb{G}$ for which $\mathsf{PRF}_K(h)$ starts with $k = O(\log(\lambda))$ zeros. By tuning the parameter k appropriately, we can achieve an average separation of the points in S_K by an arbitrary polynomial amount. We can now define $\mathsf{Dist}(\mathsf{e})$ to be the smallest $z \in \mathbb{Z}$ such that $\mathsf{e} \cdot g^z \in S_K$, i.e. $\mathsf{PRF}_K(\mathsf{e} \cdot g^z)$ starts with k zeros. Note that this function can be computed efficiently for the above choice of k.

However, as the vigilant reader might have observed already, when using this distance function, the above signalling procedure does not have perfect correctness anymore. If h^s and $\mathsf{h}^s \cdot g$ decode to different points in S_K, it might be that $\mathsf{Dist}(\mathsf{h}^s) \mod 2 = \mathsf{Dist}(\mathsf{h}^s \cdot g) \mod 2$, in which case the receiver cannot infer whether $\mathsf{x}[i] = 0$ or $\mathsf{x}[i] = 1$ and must declare an erasure.

Fortunately, by choosing k large enough, we can make the probability of such an erasure happening an arbitrarily small polynomial fraction $1/p(\lambda)$, while still ensuring that the decoding procedure runs in polynomial time[10]. As it turns out, in many applications, we can deal with this small erasure probability by resorting to standard coding techniques.

Construction 4.2 contains full technical details of the above.

Sender Privacy. So far we have not addressed issues concerning the privacy of the sender's inputs. However, in our DDH-based construction this is easy to achieve by providing an additional random input to the hash function H. That is, we define H as

$$\mathsf{H}(\mathbf{A}, \mathsf{x}; r) = g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]},$$

for a uniformly random $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. The hash value $\mathsf{h} = \mathsf{H}(\mathbf{A}, \mathsf{x}; r)$ is now uniformly random (over the choice of r) and therefore does not leak information about x. Furthermore, given the trapdoor s and a single bit $\mathsf{x}[j]$ of the input x we can perfectly simulate e by computing $\mathsf{e} \leftarrow \mathsf{h}^s \cdot g^{\mathsf{x}[j]}$. From e we can compute $\hat{\mathsf{e}}$ as before. Thus, the modified construction has perfect sender privacy.

[10] We can ensure that both sender and receiver run in strict polynomial time by introducing a suitable polynomial upper bound for the number of tries in the exhaustive search step of $\mathsf{Dist}(\cdot)$.

2.2 Trapdoor Hash from QR and LWE

We will now briefly discuss instantiations of these techinques based on the Quadratic Residuosity (QR) and Learning With Errors (LWE) assumptions (Constructions 4.3 and 4.4, resp.). As it turns out, in both these cases we will have structures with exact subgroups. However, in both cases there will also be new challenges which will have to be addressed with slightly different ideas.

Construction from QR. We will start with the QR-based construction. Instead of relying on QR directly, we will use the fact that we can use QR to construct a group $\mathbb{G}$ in which the subgroup indistinguishability problem is hard [BG10]. More specifically, the group $\mathbb{G}$ we use has a subgroup $\mathbb{G}'$ such that $|\mathbb{G}/\mathbb{G}'| = 2$. We can represent every $h \in \mathbb{G}$ as $h = (-1)^b \cdot a$, where $b \in \{0,1\}$ and $a \in \mathbb{G}'$. For the hash function H, we can use exactly the same construction as above, that is $\mathsf{H}(\mathsf{hk}, \mathsf{x}) = \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}$. The only difference is that we choose the elements in the key $\mathsf{hk} = \mathbf{A}$ from the subgroup $\mathbb{G}'$ instead of $\mathbb{G}$, that is, $\mathbf{A} = (g_{j,b})_{j\in[n],b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}'^{2\times m}$. Similar as in the DDH-based construction, for an index $i \in [n]$, the matrix $\mathbf{B}$ generated by G now has the form $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$ where

$$u_{j,b} = g_{j,b}^s$$

for all $j \neq i$ and

$$u_{i,b} = g_{i,b}^s \cdot (-1)^b.$$

Here, s is uniformly random in an appropriate domain. The crucial difference is that in $u_{i,b}$, we have replaced the generator g in the DDH-based construction by -1. It follows directly via the subgroup indistinguishabilty assumption that $g_{i,b}^s \cdot (-1)^b$ is indistinguishable from $g_{i,b}^s$. Thus, as before, the index i is hidden.

By a similar analysis as before, it holds that if $\mathsf{h} = \mathsf{H}(\mathbf{A}, \mathsf{x})$, then $\mathsf{e} = \mathsf{H}(\mathbf{B}, \mathsf{x}) = \mathsf{h}^s \cdot (-1)^{\mathsf{x}[i]}$. However, there is a crucial difference now: As $\mathsf{e} = \mathsf{h}^s \cdot (-1)^{\mathsf{x}[i]}$, the sender can also compute $\mathsf{e} \cdot (-1) = \mathsf{h}^s \cdot (-1)^{1-\mathsf{x}[i]}$. That is, one of these two elements is h^s and the other one is $\mathsf{h}^s \cdot (-1)$. Recall that the receiver can also compute these two elements using the hash value h and the trapdoor s. Thus, the only task left for the sender is to signal to the receiver which one of the two elements the element e he got is. This can be easily done by communicating a single bit: The sender compares e and $\mathsf{e} \cdot (-1)$ under some total order $\succ$, say, by representing both elements as bit strings, and computing the lexicographic order. Now, he sends the bit $\hat{\mathsf{e}} = 0$ if $\mathsf{e} \succ \mathsf{e} \cdot (-1)$ and $\hat{\mathsf{e}} = 1$ otherwise. The receiver can recover $\mathsf{x}[i]$ as follows: If $\mathsf{h}^s \succ \mathsf{h}^s \cdot (-1)$ and $\hat{e} = 0$ he sets $\mathsf{x}[i] = 0$, otherwise $\mathsf{x}[i] = 1$.

The main difference of this instantiation compared to our DDH-based construction is that there is no decoding error. We can even leverage this fact to achieve a stronger functionality. So far, we have only discussed how the receiver can recover individual bits $\mathsf{x}[i]$ of the sender's input, namely realize trapdoor hash for *index predicates*. We will now show how this can be upgraded in a way such that the receiver can learn an inner product $\langle \mathsf{y}, \mathsf{x} \rangle \mod 2$, and therefore obtain trapdoor hash for the more general class of *linear predicates*. The vector y is chosen by the receiver and is used to generate the matrix $\mathbf{B}$. Concretely, for a vector $\mathsf{y} \in \{0,1\}^n$ the receiver sets

$$u_{j,b} = g_{j,b}^s \cdot (-1)^{b \cdot \mathsf{y}[j]}$$

for all $j \in [n]$ and $b \in \{0,1\}$. As before, we can use the subgroup indistinguishability assumption to establish that the matrix $\mathbf{B}$ hides the vector y.

A simple calculation shows that $\mathsf{H}(\mathbf{B},\mathsf{x}) = \mathsf{h}^s \cdot (-1)^{\langle \mathsf{y},\mathsf{x}\rangle}$. The encoding and decoding procedures are exactly the same as before, with the difference that now the receiver learns the inner product $\langle \mathsf{y},\mathsf{x}\rangle \mod 2$. While this modification to our construction is nearly straightforward, it has several important applications.

Refer to Section 4.3 for details and analysis.

Construction from LWE. We will finally turn to our construction from LWE (Construction 4.4). On a conceptual level, the construction is very similar to the QR-based construction. We will directly explain the construction for linear predicates, i.e. inner products over $\mathbb{F}_2$. In this instantiation, let $q = 2p$ be an even modulus. The hashing key $\mathsf{hk} = \mathbf{A}$ is a $2 \times n$ matrix of uniformly random column vectors $\mathbf{a}_{j,b} \in \mathbb{Z}_q^k$, that is, each component of this matrix is a vector itself. The hash of an input $\mathsf{x} \in \{0,1\}^n$ is now computed as the sum of the corresponding $\mathbf{a}_{j,b}$, that is

$$\mathsf{H}(\mathbf{A},\mathsf{x}) = \sum_{j=1}^{n} \mathbf{a}_{j,\mathsf{x}[j]}.$$

The encoding key contains a matrix $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$, which consists of elements $u_{j,b} \in \mathbb{Z}_q^k$ which are computed by

$$u_{j,b} = \mathbf{s}^\top \mathbf{a}_{j,b} + e_{j,b} + \mathsf{y}[j] \cdot b \cdot (q/2),$$

where $\mathbf{s}$ is chosen uniformly from $\mathbb{Z}_q^k$ and the $e_{j,b}$ are sampled from a short LWE-error distribution such as a discrete gaussian. By the LWE assumption, we immediately get that the values $\mathbf{s}^\top \mathbf{a}_{j,b} + e_{j,b}$ are pseudorandom, and consequently the matrix $\mathbf{B}$ hides the vector y. Assume further that PRF is a pseudorandom function from $\mathbb{Z}_q^k$ to $\mathbb{Z}_q$. For this instantiation, the receiver will also include a uniformly random PRF-key $K \xleftarrow{\$} \{0,1\}^\lambda$ into the encoding key.

As before, the sender computes $\mathsf{h} = \mathsf{H}(\mathbf{A},\mathsf{x})$ and $\mathsf{e} = \mathsf{H}(\mathbf{B},\mathsf{x})$. Notice that it holds that

$$\mathsf{e} = \sum_{j=1}^{n} u_{j,\mathsf{x}[j]} = \mathbf{s}^\top \sum_{j=1}^{n} \mathbf{a}_{j,\mathsf{x}[j]} + \sum_{j=1}^{n} e_{j,\mathsf{x}[j]} + \langle \mathsf{y},\mathsf{x}\rangle \cdot (q/2) = \mathbf{s}^\top \mathsf{h} + e' + \langle \mathsf{y},\mathsf{x}\rangle \cdot (q/2),$$

where $e' = \sum_{j=1}^{n} e_{j,\mathsf{x}[j]}$ is a small error.

The challenge in this instantiation is that e is noisy, so the comparison-based technique from the QR-based construction will not work here. Nevertheless, a standard tool to robustly deal with this kind of error in the world of LWE is the rounding technique, introduced by Banerjee, Peikert and Rosen [BPR12]. Define the rounding function $\lfloor\cdot\rceil_2$ by $\lfloor z\rceil_2 = \lfloor z \cdot 2/q\rceil \mod 2$. The sender now computes $\hat{\mathsf{e}}$ by

$$\hat{\mathsf{e}} = \lfloor \mathsf{H}(\mathbf{B},\mathsf{x}) + \mathsf{PRF}_K(\mathsf{h})\rceil_2$$

and sends h along with the bit $\hat{\mathsf{e}}$ to the receiver. The receiver now computes and outputs $(\hat{\mathsf{e}} - \lfloor \mathbf{s}^\top \mathsf{h} + \mathsf{PRF}_K(\mathsf{h})\rceil_2) \mod 2$.

To establish correctness, we will use the fact that, for a sufficiently large q, the rounding function is insensitive to small perturbations. That is, for a uniformly random $z \xleftarrow{\$} \mathbb{Z}_q$, and a sufficiently *small* noise e, it holds that $\lfloor z+e\rceil_2 = \lfloor z\rceil_2$, except with small probability over the choice of z. Now, since the term $\mathsf{PRF}_K(\mathsf{h})$ is pseudorandom in $\mathbb{Z}_q$, it holds that

$$\begin{aligned}\hat{\mathsf{e}} = \lfloor \mathsf{H}(\mathbf{B},\mathsf{x}) + \mathsf{PRF}_K(\mathsf{h})\rceil_2 &= \lfloor \mathbf{s}^\top \mathsf{h} + e' + \langle \mathsf{y},\mathsf{x}\rangle \cdot (q/2) + \mathsf{PRF}_K(\mathsf{h})\rceil_2 \\ &= \lfloor \mathsf{s}^\top \mathsf{h} + \mathsf{PRF}_K(\mathsf{h})\rceil_2 + \langle \mathsf{y},\mathsf{x}\rangle,\end{aligned}$$

except with small probability over the choice over K. This is the reason why we include the key K in the receiver's message, that is, to enable the sender to randomize $\mathsf{H}(\mathbf{B}, \mathsf{x})$ without increasing the size of the sender message. Correctness of the scheme follows.

The magnitude of the correctness error depends on the modulus-to-noise ratio. If we choose a superpolynomial modulus-to-noise ratio, the correctness error becomes negligible. For a polynomial modulus-to-noise ration the correctness error will be inverse polynomial and we have to compensate with coding techniques.

2.3 Rate-1 Oblivious Transfer and More

Equipped with our newly developed tool, we show how to construct OT with rate 1 given any trapdoor hash with the same rate.

Batch OT. Recall that a trapdoor hash scheme for index predicates allows one to recover the i^{th} bit of a string x given the hash value $\mathsf{H}(\mathsf{hk}, \mathsf{x})$ and a single additional bit e (which we denote $\hat{\mathsf{e}}$ above). With this tool at hand, we can realize the 1-out-of-2 bit OT functionality by letting the receiver specify the hash key hk and the encoding key ek corresponding to the choice bit $i \in \{0, 1\}$. The sender then sets its input $\mathsf{x} := \mathsf{s}_0 \| \mathsf{s}_1$ to be the concatenation of the two secret bits and computes $\mathsf{h} = \mathsf{H}(\mathsf{hk}, \mathsf{x})$ together with the encoding e. Given such an information, the receiver can recover the chosen secret bit by running the decoding algorithm. The obvious shortcoming of this approach is that it is wasteful in terms of download rate, in the sense that the hash of the string must be included to recover a single bit.

The key observation here is that the hash key hk can be reused across several executions. Therefore the size of the hash h can be amortized across multiple independent bit OT protocols. That is, if the bit OTs are executed in a batch, we can boost the download rate of the construction to approach 1: Given n independent instances of bit OT, the receiver samples a hash key hk as before, this time for inputs of length $2n$ rather than 2, and samples a set of encoding keys $(\mathsf{ek}_1, \ldots, \mathsf{ek}_n)$, where the j^{th} key allows the receiver to learn the input bit at position $(2j + i_j)$, where $i_j \in \{0, 1\}$ is the choice bit of the j^{th} OT instance. It is important that all of the encoding keys are generated with respect to the same hk, since it will allow us to re-use the corresponding hash. The sender defines $\mathsf{x} := \mathsf{s}_{1,0} \| \mathsf{s}_{1,1} \| \ldots, \| \mathsf{s}_{n,0} \| \mathsf{s}_{n,1}$, where $\mathsf{s}_{j,0}, \mathsf{s}_{j,1}$ are the secrets for the j^{th} instance, and computes the hash $\mathsf{h} = \mathsf{H}(\mathsf{hk}, \mathsf{x})$ as before, in addition to the additional hints, i.e. TDH encodings, $(\mathsf{e}_1, \ldots, \mathsf{e}_n)$. The recovery procedure is then run in parallel for each bit OT instance. Note that the sender's message consists of a hash (i.e., a single group element) and n bits. That is, the impact of h in the communication vanishes as n grows, and thus, the download rate of the scheme approaches 1.

We elaborate in Section 5.3.

String OT. We showed how to obliterate the impact of the hash value h in the second OT message by executing multiple bit OT instances in a batch. The same can be accomplished for a single OT instance, when executed on sufficiently long secret strings (rather than single bits)[11]. The protocol can be derived generically from the batch OT by adapting the encodings of the inputs: The receiver executes the batch OT protocol of above by replicating the same choice bit i over each of the n instances, whereas the sender parses the two strings $(\mathsf{s}_0, \mathsf{s}_1) \in \{0, 1\}^n$ as n pairs of bits and encodes

[11] In fact, string OT can be thought of as a special case of batch OT, where all the choice bits i_j are equal.

the string x as before. Since the choice bit of the receiver is the same in all positions, the decoding algorithm will recover the string s_i in its entirety.

In the above discussion we omitted a few important aspects of our transformation that need to be addressed in order to obtain a fully-fledged rate-1 OT. More specifically, (i) some instances of trapdoor hash have a correctness error, in the sense that the secret might not be recoverable with a certain probability ϵ. Furthermore, (ii) the upload rate of the construction is inverse polynomial in λ. To resolve the former point we preprocess the sender's inputs with a sufficiently strong error-correcting code. One has to be careful that the encoding function does not affect the download rate of the protocol. Fortunately, our error probability ϵ lies in a regime of parameters that allow us to efficiently instantiate the encoding function. For the latter issue, we show that any string OT with download rate 1 can be generically bootstrapped to a string OT with overall rate 1. Our method is based upon the simple observation that the first message of an OT is always reusable and therefore can be amortized by executing the same OT over blocks of a sufficiently long string.

The same techniques can be generalized to 1-out-of-k OT, for any $k \in \mathbb{N}$. Please refer to Section 5.4 for the details of the above construction.

We now discuss few interesting applications of rate-1 OT.

Private Information Retrieval. Given a 1-out-of-2 string OT with rate 1, a (block) single-server PIR protocol [KO97], with optimal download rate and polylogarithmic overall communication, follows as a simple corollary of the main theorem of Ishai and Paskin [IP07]. We hereby recall the transformation for completeness.

Recall that in (block) PIR, a client queries a server, that holds a database consisting of N blocks, each of length β bits, in order to privately retrieve a block of his choice. Observe that a 1-out-of-2 string OT can be seen as a hash function that compresses the size of its input by a factor of roughly two. The idea is to use such a hash function and let the server compute a Merkle tree over the database $\mathsf{x} \in \{0,1\}^{N \cdot \beta}$. Every node in the tree consists of a block and, for simplicity, we assume that $N = 2^d$ for some $d \in \mathbb{N}$, which is the depth of the tree. Thus, the lowest level in the tree consists of the N database blocks: $\mathsf{x}_0, \ldots, \mathsf{x}_{N-1}$, and every other level $\ell = 1, \ldots, d$ in the tree consists of $N/2^\ell$-many blocks: $\mathsf{h}_{\ell,0}, \ldots, \mathsf{h}_{\ell,N/2^\ell-1}$, that are hashes of the nodes in level ℓ. Notice that every index $i \in \{0, \ldots, N-1\}$ corresponds to a path in the tree, which we denote by $(i_1, \ldots, i_d)$, which represents the path from database block x_i to the root of the tree.

The protocol proceeds as follows: First, the client generates the receiver message $\mathsf{msg}_1^{(\ell)}$ of an OT for strings of appropriate length, for each layer $\ell = 1, \ldots, d$ in the tree, where the choice bit is set to be the index i_ℓ. Then the client sends $(\mathsf{msg}_1^{(1)}, \ldots, \mathsf{msg}_1^{(\ell)})$ to the server, who computes all of the hash values in the Merkle tree, i.e. OT sender messages, and sends the root $\mathsf{msg}_2^{(d)}$ to the client. The client can recover the entry of interest by recursively applying the decoding algorithm of the OT, starting from the top level d.

See Section 6 for details and more applications of rate-1 OT.

Evaluating Branching Programs over Encrypted Data. Another result in the work of Ishai and Paskin [IP07], which can be seen as a generalization of the above, is a compiler that takes any 2-message rate-1 OT[12] into a *semi-compact* homomorphic encryption scheme for branching

[12]In fact they require an OT protocol with a strong notion of sender privacy, which is satisfied by all of our constructions.

programs (a superclass of NC^1), where the size of the evaluated ciphertexts depends only on the length of the branching program but not on its size. This immediately yields a sublinear secure function evaluation protocol where the client's work is independent of the size of the branching program (which is in fact hidden to its eyes).

Lossy Trapdoor Functions. As a yet another application, we show a simple construction of lossy trapdoor functions [PW08, HO12] with optimal rate from any 2-message rate-1 OT, and therefore obtain schemes based on DDH, QR, or LWE. Prior to our work, rate optimal schemes were known to exist only under the DCR assumption.

Rate-Optimal Protocols for Other OT-like Functionalities. It turns out that using trapdoor hash for index predicates, we can already capture a wide variety of predicate classes through a simple transformation. More specifically, if a given predicate class $\mathcal{F}$ is "small", i.e. contains $\mathsf{poly}(n)$ predicates for input size n, then we can obtain TDH for $\mathcal{F}$ on input x by applying TDH for index predicates on input x', where the i^{th} bit in x' is the evaluation of the i^{th} predicate in $\mathcal{F}$ on x.

We use this observation to extend the range of functionalities for which we can construct rate-optimal protocols. For instance, an interesting special case of small predicate classes are functions $f(x) = ax + b$ over $\mathbb{F}_2$, which essentially allow realizing batch oblivious linear function evaluation (OLE) [NP99] by replacing the TDH for index predicates, in the batch OT construction described above, with TDH for such predicates. Further, one can extend the idea to OLE over other constant size rings (e.g. fields $\mathbb{F}_p$ for constant prime p), by evaluating each output bit separately.

An even more general functionality, that allows evaluating matrix-vector products over $\mathbb{F}_2$ (with the vector and matrix respectively being the receiver's and sender's input), can be realized using the same technique by relying on TDH for linear predicates, which can be instantiated, as mentioned earlier, under the LWE and QR assumptions. The LWE-based TDH scheme can be further extended to allow trapdoor-evaluation of linear functions over small fields, thus yielding *oblivious matrix vector multiplication* (OMV) over such fields. It is worth mentioning that OMV can be also seen as a variant of rate-1 additively homomorphic encryption, where inner products (and in particular matrix multiplication) can be evaluated over encrypted vectors.

Lastly, we note that using OLE and OMV schemes over small fields, we can realize similar functionalities over larger algebraic structures through standard algebraic manipulations. More specifically, we can get OLE and OMV over smooth rings, via the Chinese Remainder Theorem, and over extension fields of small characteristic using basic extension field algebra.

We provide more details in Section 5.5.

2.4 Private Laconic Oblivious Transfer

Next, we outline another application of trapdoor hash: *private laconic oblivious transfer*. As discussed in the introduction, private laconic OT has strong applications in secure computation. In particular, following the outline presented in [CDG+17] to utilize laconic OT for non-interactive secure RAM computation with unprotected memory access, we can use private laconic OT to obtain secure RAM computation where the access pattern to the memory is also hidden, and therefore achieve a stronger notion of security.

[13] We also assume correlated-input secure hash over bilinear groups.

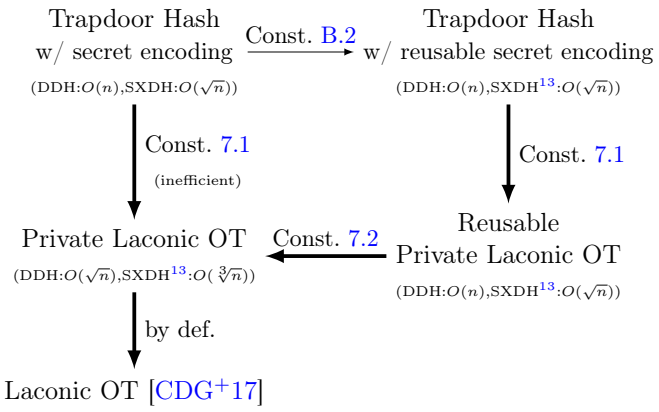

Figure 2: Overview of the results in this work, Part II: safe-function evaluation with sublinear communication. Thin lines correspond to non-generic transformations.

Recall that in laconic OT ($\ell\mathsf{OT}$) [CDG+17], a receiver with an input database $\mathsf{x} \in \{0,1\}^n$ communicates with a sender, with two secrets $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}$ and an index $i \in [n]$ as input, in order to learn $\mathsf{s}_{\mathsf{x}[i]}$,while keeping both x and $\mathsf{s}_{1-\mathsf{x}[i]}$ private. In *private* laconic OT (formally defined in Section 7.1), we also require that the index i remains hidden from the receiver.

Recall that in laconic OT ($\ell\mathsf{OT}$) [CDG+17], a receiver with an input database $\mathsf{x} \in \{0,1\}^n$ communicates with a sender, with two secrets $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}$ and an index $i \in [n]$ as input, in order to learn $\mathsf{s}_{\mathsf{x}[i]}$,while keeping both x and $\mathsf{s}_{1-\mathsf{x}[i]}$ private. In *private* laconic OT ($p\ell\mathsf{OT}$), we also require that the index i remains hidden from the receiver.

Our end goal is to realize the $p\ell\mathsf{OT}$ functionality through a two-message protocol where the overall communication is sublinear in n in order to obtain sublinear SFE protocols (due to [CDG+17]).

As a start, however, we aim for *receiver-compact* $p\ell\mathsf{OT}$ where the upload communication (i.e., the communication from the receiver to the sender) is independent of the receiver's database size n, and set no restrictions on the communication from the sender to the receiver. We then describe such a receiver-compact $p\ell\mathsf{OT}$ construction with linear sender-receiver communication through our DDH-based trapdoor hash, and then show to get sublinear communication (namely $\sqrt{n}$) using pairings.

Lastly, we show that if we are willing to compromise receiver-compactness, then we can balance our protocols using what we call *reusable private laconic OT* and obtain more efficient SFE protocols with sublinear communication under both DDH and pairings.

Basic Construction from Trapdoor Hash. Let us first try to realize the relaxed notion of $\ell\mathsf{OT}$ using trapdoor hash in a straight-forward way. In order to that, the roles of Alice and Bob from above, as a sender and a receiver, must be swapped.

Given a TDH for index predicates, the construction proceeds as follows. The public parameters of the $\ell\mathsf{OT}$ scheme simply consist of a hash key hk of the TDH. The receiver (which is now played by Alice) computes a hash value of his database $\mathsf{h} = \mathsf{H}(\mathsf{hk}, \mathsf{x})$, which he sends to the sender (now Bob). Observe that the size of h is independent of the size of x, and thus satisfying our requirement regarding upload communication.

The sender generates a pair $(\mathsf{ek}, \mathsf{td})$ of an encoding key and a trapdoor corresponding to the hash key hk and his input index i. Using td and h, he computes two symmetric encryption keys

$(\mathsf{e}_0, \mathsf{e}_1) = \mathsf{D}(\mathsf{td}, \mathsf{h})$, using which he encrypts his secret inputs s_0 and s_1, respectively, to obtain two ciphertexts. He now sends the key ek as well as the two ciphertexts to the receiver, who will be able to decrypt one of them by recovering $\mathsf{e}_{\mathsf{x}[i]}$ using the encoding algorithm E of the TDH.

To establish security of this $\ell\mathsf{OT}$ construction, we need the following to ensure that (i) the hash h hides x (*receiver privacy*), and (ii) the encoding key ek hides $\mathsf{e}_{1-\mathsf{x}[i]}$ (*sender privacy*). While receiver privacy is implied directly from the input privacy of the underlying TDH, sender privacy does not generically follow. Thus, we need to augment our definition of TDH with the requirement that, for every i, the value $\mathsf{e}_{1-\mathsf{x}[i]}$ is uniformly random given hk, ek and x, where $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, i)$ and $(\mathsf{e}_0, \mathsf{e}_1) = \mathsf{D}(\mathsf{td}, \mathsf{H}(\mathsf{hk}, \mathsf{x}))$. A TDH that satisfies this requirement is said to have *secret encoding*.

Notice that the secret encoding property is in conflict with achieving high rate in a TDH. In particular, in any rate-1 TDH, correctness requires that $\mathsf{e}_{1-\mathsf{x}[i]} = 1 - \mathsf{e}_{\mathsf{x}[i]}$ with a high probability.

Fortunately, the basic DDH-based TDH construction without the rate optimization, which is sketched above, fulfills the secret encoding property under DDH. This is not surprising, as so far, this construction is very similar to the original $\ell\mathsf{OT}$ construction of [CDG+17].

In fact, the above outlined protocol, which is presented in details in Construction 7.1, realizes the stronger notion of $p\ell\mathsf{OT}$. By relying on the function privacy of the underlying TDH, we immediately get that the sender's input index i is kept hidden from the receiver, and hence, we get the p in $p\ell\mathsf{OT}$.

As hinted earlier, the above construction suffers from an undesired property: download communication is linear in the size of the receiver's database. We propose two solutions. The first relies on the SXDH assumption over bilinear groups and uses pairings in order to reduce the communication to $O(\sqrt{n})$. In the second, we introduce a *resuablitiy* notion of $p\ell\mathsf{OT}$, that can be realized under both DDH and SXDH with similar communication. We then show how to transform any reusable $p\ell\mathsf{OT}$ into a (non-reusable) $p\ell\mathsf{OT}$ scheme while reducing the overall communication complexity, to obtain efficient $p\ell\mathsf{OT}$ protocols under DDH, resp. SXDH, with download communication proportional to $\sqrt{n}$ and $\sqrt[3]{n}$, respectively.

Shrinking the Keys Using Pairings. The bottleneck in the efficiency of the DDH-based $p\ell\mathsf{OT}$ scheme from above lies in the size of the public parameters and sender's message, namely the keys hk and ek of the trapdoor hash, which both grow linearly in n.

Towards achieving sublinear communication, we start with the following observation. The high entropy of the public parameters, i.e. matrix $\mathbf{A}$ in hk, is not essential for security in the DDH-based TDH scheme. Thus, if we could produce such a matrix $\mathbf{A} = (g_{j,b})_{j\in[n],b\in\{0,1\}}$ using a shorter "seed", and then let Alice compute a short "seed" that expands to a matrix $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$ which can be used as an encoding key ek, then we are able to reduce the size of hk and ek and, therefore, the communication of the resulted $p\ell\mathsf{OT}$.

Roughly speaking, we choose the seed for $\mathbf{A}$ to be two $2 \times \sqrt{n}$ matrices, $\mathbf{A_1} \in \mathbb{G}^{2\times\sqrt{n}}$ and $\mathbf{A_2} \in \mathbb{H}^{2\times\sqrt{n}}$, for two different groups $\mathbb{G}$ and $\mathbb{H}$. We then use a bilinear map $e : \mathbb{G} \times \mathbb{H} \to \hat{\mathbb{G}}$ to pair elements in $\mathbf{A_1}$ with elements in $\mathbf{A_2}$ and get $2 \times n$ elements in $\hat{\mathbb{G}}$, which we use as the hash key $\mathsf{hk} = \mathbf{A} \in \hat{\mathbb{G}}^{2\times n}$. To generate a seed to the corresponding encoding key $\mathbf{B}$, we begin by defining $\mathbf{B_1} = \mathbf{A_1}^{s_1}$ and $\mathbf{B_2} = \mathbf{A_2}^{s_2}$, which would expand to $\mathbf{B} = \mathbf{A}^{s_1+s_2}$ using the pairing. To achieve functionality, we would want to "puncture" the $(i, 1)^{th}$ entry in $\mathbf{B}$ and multiplying it by a random group element (see Equation 2.1). For this task, we use a bilinear pairing with a special property, that allows us to multiply every element in $\mathbf{B_1}$ and $\mathbf{B_2}$ by carefully sampled random elements from $\mathbb{G}$ and $\mathbb{H}$ (resp.). We do this in a way that when pairing elements from the two matrices to generate $\mathbf{B}$, these random factors cancel each other out, except at the $(i, 1)^{th}$ element, which will be randomly

distributed.

Appendix C contains full details of the parings-based TDH.

Having shown how to obtain receiver-compact private laconic OT from DDH and SXDH, we next describe in a high level how to transform such "unbalanced" schemes to $p\ell\mathsf{OT}$ schemes which, despite being non-receiver-compact, have lower overall communication, and in particular, give us sublinear non-interactive secure computation protocols also from DDH.

How to Reuse the Sender's Message. Let us reexamine the $p\ell\mathsf{OT}$ scheme from TDH. The sender's message consists of an encoding key ek and encryptions of the two sender's secrets, each under a corresponding TDH encoding. We observe that the encoding key ek, which is actually the larger part of the sender's message, is actually independent of the hash value $\mathsf{h} = \mathsf{H}(\mathsf{hk}, \mathsf{x})$. It can therefore be *reused* for different $p\ell\mathsf{OT}$ invocations corresponding to different values of the receiver's database x and the sender's secrets $\mathsf{s}_0, \mathsf{s}_1$ (but that share the same index i).

This brings us to define a notion of *reusable* $p\ell\mathsf{OT}$, where we distinguish between two parts of the sender's message: (i) a *reusable* part, of size sublinear in n, that depends only on i and, therefore, can be reused for different inputs $\mathsf{x}, \mathsf{s}_0, \mathsf{s}_1$, and (ii) a *compact* part, of size independent in n, that is generated w.r.t. a specific receiver's database x and sender's secrets s_0 and s_1.

As mentioned above, the $p\ell\mathsf{OT}$ construction from TDH already gives reusability, with ek being reusable. However, a subtle issue concerning the sender privacy has to be resolved. Take for instance the $p\ell\mathsf{OT}$ construction from the DDH-based TDH. Above, we argued that the encoding $\mathsf{e}_{1-\mathsf{x}[i]}$ is uniformly distributed given hk and ek, in what we called the *secret encoding property*. Notice, however, that this is not sufficient for reusable $p\ell\mathsf{OT}$, where many such encodings $\mathsf{e}_{1-\mathsf{x}[i]}$, namely symmetric encryption keys, are generated w.r.t. different values of x. Although each of these encryption keys is individually uniform, they are highly correlated. Encryption under correlated keys is clearly insecure. Thus, we do not get sender privacy when ek is reused.

We handle this issue by defining a related *reusable secret encoding* property for TDH. In Appendix B we show that both the DDH-based and pairings-based TDH schemes can be extended to have reusable secret encoding using suitable *correlated-input secure hash* [AMN+18], which can be fortunately realized under DDH and, resp., appropriate hardness assumptions over bilinear groups.

Reusable $p\ell\mathsf{OT}$ can be useful by itself for applications in secure computation, in particular when we allow to amortize the communication cost over many computations of the same functionality on different inputs. Further, as mentioned earlier, reusable $p\ell\mathsf{OT}$ turns out to be useful to achieve $p\ell\mathsf{OT}$ schemes which, although non-reusable, have smaller download communication. We elaborate below.

Exploiting Reusability for More Efficient Schemes. Lastly, we show how to use reusable $p\ell\mathsf{OT}$ to achieve more efficient $p\ell\mathsf{OT}$ schemes. Our final results are a DDH-based $p\ell\mathsf{OT}$ with communication proportional to $\sqrt{n}$, and a pairing-based $p\ell\mathsf{OT}$ with communication proportional to $\sqrt[3]{n}$. Although the construction is generic, it is parameterized differently according to the underlying reusable $p\ell\mathsf{OT}$. For presentation, we take the DDH-based reusable $p\ell\mathsf{OT}$, where the public parameters and sender's message grow linearly in n as a special case.

The idea is as follows. We divide the receiver's database x to $\sqrt{n}$ smaller databases, $\mathsf{x}_1, \ldots, \mathsf{x}_{\sqrt{n}}$, each of size $\sqrt{n}$. Consequently, every index $j \in [n]$ is interpreted as $(j_1, j_2) \in [\sqrt{n}]^2$ (particularly, $i = (i_1, i_2)$) where $\mathsf{x}_j := \mathsf{x}_{j_1}[j_2]$. On the sender's side, each of the secrets $\mathsf{s}_0, \mathsf{s}_1$ is additively shared

to $\mathsf{s}_{0,1},\ldots,\mathsf{s}_{0,\sqrt{n}} \in \{0,1\}$ and, respectively, $\mathsf{s}_{1,1},\ldots,\mathsf{s}_{1,\sqrt{n}} \in \{0,1\}$ s.t. $\sum_j \mathsf{s}_{j,b} = \mathsf{s}_b$ for $\mathsf{b} \in \{0,1\}$. In fact, the sender generates the shares such that $\mathsf{s}_{j,0} = \mathsf{s}_{j,1}$ for any $j \neq i_1$.

The idea is to use the underlying reusable $p\ell\mathsf{OT}$ to send to the receiver, for every $j \in [\sqrt{n}]$, either $\mathsf{s}_{j,0}$ or $\mathsf{s}_{j,1}$, conditioned on $\mathsf{x}_j[i_2]$. For any $j \neq i_1$, both bits are equal, and therefore, the receiver obtains s_j, regardless of the value of $\mathsf{x}_j[i_2]$. The only database bit that matters is $\mathsf{x}[i] := \mathsf{x}_{i_1}[i_2]$, which determines whether the receiver receives $\mathsf{s}_j := \mathsf{s}_{j,0}$, and therefore can compute $\sum_j \mathsf{s}_j = \mathsf{s}_0$, or $\mathsf{s}_j := \mathsf{s}_{j,1}$, which would allow him to compute $\sum_j \mathsf{s}_j = \mathsf{s}_1$.

The construction is described in detail in Section 7.3.

2.5 Concurrent Work

In a concurrent work, Gentry and Halevi [GH19] constructed an efficient rate-1 FHE schemes from LWE, which in particular also yield rate-1 OT constructions. When instantiated from LWE with polynomial modulus-to-noise ratio, their construction achieves rate $1-\epsilon$ for any constant ϵ. In comparison, our OT constructions achieve rate $1-1/\lambda$ in this regime and can also be based on DDH or QR.

3 Preliminaries

Notation. For an integer $n \in \mathbb{N}$, $[n]$ denotes the set $\{1,\ldots,n\}$. We use λ for the security parameter and $\mathsf{negl}(\lambda)$ for a negligible function in λ. We say that a distribution ensemble $\{x_\lambda\}_{\lambda\in\mathbb{N}}$ is *polynomial-length* if there exists a polynomial $\ell(\lambda)$ such that $|x_\lambda| \leq \ell(\lambda)$ for any $\lambda \in \mathbb{N}$. We use $\overset{c}{\equiv}$ and $\overset{s}{\equiv}$ to denote computational and, resp., statistical indistinguishability between two distribution ensembles, and we use $\equiv$ when the two ensembles are identical. For a distribution D we use $x \overset{\$}{\leftarrow} D$ to say that x is sampled according to D and use $x \in D$ to say that x is in the support of D. For a set S we overload the notation to use $x \overset{\$}{\leftarrow} \mathsf{S}$ to indicate that x is chosen uniformly at random from S. We conventionally denote vectors as $\mathbf{v}$ and matrices as $\mathbf{M}$. For a group $\mathbb{G}$ of order p, an element $g \in \mathbb{G}$ and a vector $\mathbf{v} = (v_1,\ldots,v_n) \in \mathbb{Z}_p^n$, we denote by $g^{\mathbf{v}}$ the vector $(g^{v_1},\ldots,g^{v_n}) \in \mathbb{G}^n$. Given a vector $\mathbf{g} = (g_1,\ldots,g_n)$ and a scalar c, we denote by $(\mathbf{g})^c$ the vector $(g_1^c,\ldots,g_n^c) \in \mathbb{G}^n$.

3.1 Number Theoretical Assumptions

The Decisional Diffie-Hellman Assumption. In the following we state the decisional version of the Diffie-Hellman (DDH) assumption [DH76].

Definition 3.1 (Decisional Diffie-Hellman (DDH) assumption)**.** *A (prime-order)* group generator *is an algorithm $\mathcal{G}$ that takes as an input a security parameter 1^λ and outputs $(\mathbb{G},p,g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group which is always a prime number, and g is a generator of the group. We say that $\mathcal{G}$* satisfies the DDH assumption *(or is* DDH-hard*) if for any PPT adversary, $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}((\mathbb{G},p,g),(g^{a_1},g^{a_2},g^{a_1a_2}))=1] - \Pr[\mathcal{A}((\mathbb{G},p,g),(g^{a_1},g^{a_2},g^{a_3}))=1]| = \mathsf{negl}(\lambda)$$

where $(\mathbb{G},p,g) \overset{\$}{\leftarrow} \mathcal{G}(1^\lambda)$ and $a_1,a_2,a_3 \overset{\$}{\leftarrow} \mathbb{Z}_p$.

The Quadratic Residuosity Assumption. We say that N is a Blum integer if $N = p \cdot q$ for some primes p and q such that $p \pmod 4 = q \pmod 4 = 3$. We denote by $\mathbb{J}_N$ the multiplicative group of the elements in $\mathbb{Z}_N^*$ with Jacobi symbol $+1$ and by $\mathbb{QR}_N$ the multiplicative group of quadratic residues modulo N with generator g. Note that $\mathbb{QR}_N$ is a subgroups of $\mathbb{J}_N$ and they have order $\frac{\varphi(N)}{4}$ and $\frac{\varphi(N)}{2}$, respectively, where $\varphi(\cdot)$ is Euler's totient function. It is useful to write $\mathbb{J}_N : \mathbb{H} \times \mathbb{QR}_N$, where $\mathbb{H}$ is the multiplicative group $(\pm 1, \cdot)$ of order 2. Note that if N is a Blum integer then $\gcd(2, \frac{\varphi(N)}{4}) = 1$ and $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$. We recall the quadratic residuosity (QR) assumption [GM82].

Definition 3.2 (Quadratic Residuosity Assumption)**.** *Let N be a uniformly sampled Blum integer and let $\mathbb{QR}_N$ be the multiplicative group of quadratic residues modulo N with generator g. We say the Quadratic Residuosity assumption holds with respect to $\mathbb{QR}_N$ if for any PPT adversary $\mathcal{A}$ it holds that*

$$| \Pr[\mathcal{A}(N, g, a) = 1] - \Pr[\mathcal{A}((N, g, (-1) \cdot a)) = 1] | = \mathsf{negl}(\lambda)$$

where $a \xleftarrow{\$} \mathbb{QR}_N$.

In the following we recall a useful proposition from [BG10].

Proposition 3.1 ([BG10])**.** *Let $\mathbb{QR}_N$ be the multiplicative group of quadratic residues modulo N with generator g, where N is a uniformly sampled Blum integer, and let $\ell = \ell(\lambda)$ be a polynomial. If the quadratic residuosity assumption holds then for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ it holds that*

$$\left| \begin{array}{l} \Pr[\mathcal{A}_1(\tau, (-1)^{\mathbf{a}} \cdot (\mathbf{g})^r) = 1 \mid \mathcal{A}_0(N, g, \mathbf{g}) = (\tau, \mathbf{a})] \\ - \Pr[\mathcal{A}_1(\tau, (\mathbf{g})^r) = 1 \mid \mathcal{A}_0(N, g, \mathbf{g}) = (\tau, \mathbf{a})] \end{array} \right| = \mathsf{negl}(\lambda)$$

where $a \in \{0,1\}^\ell$, $\mathbf{g} \xleftarrow{\$} \mathbb{QR}_N^\ell$, and $r \xleftarrow{\$} \left[\frac{N-1}{2}\right]$.

3.2 The Learning with Errors Assumption

The learning with errors (LWE) problem was defined by Regev [Reg05]. In this work we exclusively use the decisional version.

Definition 3.3 (Learning with Errors (LWE) assumption)**.** *The $\mathsf{LWE}_{m,\tilde{m},q,\chi}$ problem, for $(m, \tilde{m}, q) \in \mathbb{N}$ and for a distribution χ supported over $\mathbb{Z}$, is two distinguish between the distributions $(\mathbf{A}, \mathbf{sA}+\mathbf{e} \mod q)$ and $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times \tilde{m}}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^m$, and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{\tilde{m}}$. The $\mathsf{LWE}_{m,\tilde{m},q,\chi}$ assumption is that the two distributions are computationally indistinguishable.*

The assumption is consider to hold if for *any* $\tilde{m} = \mathsf{poly}(m \log q)$ and we denote this problem by $\mathsf{LWE}_{m,q,\chi}$. As shown in [Reg05, PRSD17], the $\mathsf{LWE}_{m,q,\chi}$ problem with χ being the discrete Gaussian distribution with parameter $\sigma = \alpha q \geq 2\sqrt{m}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of x is proportional to $e^{-\pi(|x|/\sigma)^2}$), is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in *worst case* dimension n lattices. This is proven using a quantum reduction. Classical reductions (to a slightly different problem) exist as well [Pei09, BLP+13] but with somewhat worse parameters. The best known (classical or quantum) algorithms for these problems run in time $2^{\tilde{O}(m/\log \gamma)}$, and in particular they are conjectured to be intractable for $\gamma = \mathsf{poly}(m)$.

A discrete gaussian with parameter αq is $B = \alpha q$ bounded, except with negligible probability. For parameter α the worst-to-average case reduction of [Reg05] gives a worst-case approximation factor of $\tilde{O}(m/\alpha)$ for SIVP. Consequently, in terms of the bound B and the modulus q we get a worst-case approximation factor of $\tilde{O}(mq/B)$ for SIVP.

3.3 Statistics and Information Theory

We recall the notion of statistical distance and some results from information theory.

Definition 3.4 (Statistical Distance)**.** *Let X and Y be two random variables over a finite set $\mathcal{U}$. The statistical distance between X and Y is defined as*

$$\mathbb{SD}\left[X, Y\right] = \sum_{u \in \mathcal{U}} \left|\Pr[X = u] - \Pr[Y = u]\right|$$

Definition 3.5 (Average Conditional Min-Entropy)**.** *Let X be a random-variable supported on a finite set $\mathcal{X}$ and let Z be a (possibly correlated) random variable supported on a finite set $\mathcal{Z}$. The average-conditional min-entropy $\tilde{H}_\infty(X|Z)$ of X given Z is defined as*

$$\tilde{H}_\infty(X|Z) = -\log\left(\mathsf{E}_z\left[\max_{x \in \mathcal{X}} \Pr X = x|Z = z\right]\right).$$

Definition 3.6 (Extractor)**.** *A function $\mathsf{Ext} : \{0,1\}^d \times \mathcal{X} \to \{0,1\}^\ell$ is called a* seeded strong average-case (k, ϵ)-extractor*, if it holds for all random variables X with support $\mathcal{X}$ and Z defined on some finite support that if $\tilde{H}_\infty(X|Z) \geq k$, then it holds that*

$$\mathbb{SD}[(\mathsf{seed}, \mathsf{Ext}(\mathsf{seed}, X), Z), (\mathsf{seed}, U, Z)] = \epsilon,$$

where $\mathsf{seed} \xleftarrow{\$} \{0,1\}^d$ and $U \xleftarrow{\$} \{0,1\}^\ell$

Such extractors can be constructed from universal hash functions [DRS04, DORS08]. In fact, any extractor is an average-case extractor for slightly worse parameters by the averaging principle.

Definition 3.7 (Universal Hash Functions)**.** *An ensemble of functions $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is called universal, if it holds for all $x \neq x' \in \mathcal{X}$ that*

$$\Pr_H[\mathsf{H}(x) = \mathsf{H}(x')] \leq 1/|\mathcal{Y}|,$$

where $H \xleftarrow{\$} \mathcal{H}$ is chosen uniformly random.

Lemma 3.1 (Leftover Hash Lemma [DRS04])**.** *Let X be a random-variable supported on a finite set $\mathcal{X}$ and let Z be a (possibly correlated) random variable supported on a finite set $\mathcal{Z}$ such that $\tilde{H}_\infty(X|Z) \geq k$. Let $\mathcal{H} : \mathcal{X} \to \{0,1\}^\ell$, where $\ell \leq k - 2\log\left(\frac{1}{\epsilon}\right)$, be a family of universal functions. Then $\mathcal{H}$ is a strong average-case (k, ϵ)-extractor.*

Chernoff Bound. We recall the following useful bound for binomial distributions.

Definition 3.8 (Chernoff Bound)**.** *Let X be binomially distributed with parameters $n \in \mathbb{N}$ and $p \in [0,1]$. Let $p' > p$. Then it holds that*

$$\Pr[X > 2p'n] < e^{-p'n/3}.$$

Erasure- and Error-Correcting Codes. We recall the definition of erasure- and error-correcting codes.

Definition 3.9 (Erasure- and Error-Correcting Codes)**.** *A* (binary) (N, n) code *consists of a pair of efficiently computable functions* $(\mathsf{Encode}, \mathsf{Decode})$*, where* $\mathsf{Encode} : \{0,1\}^n \to \{0,1\}^N$ *and* $\mathsf{Decode} : \{0,1,\bot\}^N \to \{0,1\}^n$*. The rate* r *of such a code is defined as* $r := n/N$*.*

- *We say that a code* $(\mathsf{Encode}, \mathsf{Decode})$ *efficiently corrects from* t erasures *if the following holds. Let* $\mathsf{x} \in \{0,1\}^n$ *and* $y' \in \{0,1\}^N$ *be such that* y' *is* $\bot$ *in at most* t *positions but otherwise identical to* $y = \mathsf{Encode}(x)$*. Then it holds that* $\mathsf{Decode}(y') = x$*.*
- *We say that a code* $(\mathsf{Encode}, \mathsf{Decode})$ *efficiently corrects from* t errors *if the following holds. Let* $\mathsf{x} \in \{0,1\}^n$ *and* $y' \in \{0,1\}^N$ *be such that* y' *differs from* $y = \mathsf{Encode}(x)$ *in at most* t *positions. Then it holds that* $\mathsf{Decode}(y') = x$*.*

We will now briefly discuss suitable instantiations of error and erasure correcting codes for our constructions. Our main concern is finding an explicit family with rate approaching 1 which can still be efficiently decoded.

While concatenated codes with an outer Reed Solomon code seem like a natural choice, their rate is insufficient for our purposes. The reason is that since the inner code has length at most $O(log(\lambda))$, their asymptotic rate can be at most $1 - 1/O(log(\lambda))$, which implies a rate of $1 - 1/O(log(\lambda))$ for the concatenated code. But this is below our target of $1 - 1/\lambda$. For erasure correcting codes, we could choose a a random linear code of appropriate parameters and rely on the Gilber-Varshamov bound to get good distance, but this would result in a probabilistic construction.

Instead, our approach will be to instantiate Sipser-Spielman expander codes [SS94, CRVW02] with a suitable expander that allows us to achieve rate $1 - 1/\lambda$. The Guruswami-Umans-Vadhan (GUV) construction [GUV07] provides such an expander.

Definition 3.10 (Expander Graphs)**.** *A bipartite graph* H *with* N *left nodes* L *and* M *right nodes* R *is called* (N, M, D, K, γ) *expander, if the degree of every left node is* D *and it holds for every set* $X \subseteq L$ *of size at most* K *that* $|N(X)| \geq \gamma|X|$*, where* $N(X) \subseteq R$ *is the neighborhood of* X *in* R*.*

Theorem 3.1 (Guruswami-Umans-Vadhan [GUV07])**.** *For all constants* $\alpha, \epsilon > 0$*, for every* N *and* $K \leq N$ *there exists an explicit* $(N, M, D, K, (1-\epsilon)D)$ *expander, where* $D \leq O(\log(N)\log(K)/\epsilon)^{1+1/\alpha}$ *and* $M \leq D^2 \cdot K^{1+\alpha}$*.*

Theorem 3.2 (Sipser-Spielman [SS94, CRVW02])**.** *Let* H *be an* $(N, M, D, K, (1-\epsilon)D)$ *expander for* $\epsilon < 1/12$*. Then there exists a binary error correcting code* $(\mathsf{Encode}, \mathsf{Decode})$ *with rate* $1 - M/N$ *which can efficiently decode* $(1 - 3\epsilon)K$ *errors in time* $O(D \cdot N)$*.*

We will now instantiate the codes of Theorem 3.2 with the expanders from Theorem 3.1. Fix any $\epsilon < 1/12$ and set $\alpha = 1$. Setting $N = \Theta((\log \lambda)^8 \cdot \lambda^3)$ and $K = \Theta(\lambda)$, we get $D = O((\log \lambda)^4)$ and $M = O((\log \lambda)^8 \cdot \lambda^2)$. This gives leads to a rate $1 - O(1/\lambda)$.

Theorem 3.3. *There exists an efficient family of binary error correcting codes* $(\mathsf{Encode}, \mathsf{Decode})$ *of length* $N = \tilde{O}(\lambda^3)$ *and rate* $1 - O(1/\lambda)$ *which can efficiently decode from* $\Omega(\lambda)$ *errors or erasures.*

Notice that since the GUV expander does not have constant degree, the resulting decoding algorithm is not linear time, but still polynomial time (in fact quasi-linear).

We remark, however, that the above requirements are asymptotic in nature. In practice, one would use either a concrete LDPC code [Gal63, Mac02] or a concatenated code with an inner Reed Solomon code, as in practically speaking a rate of $1-1/\log(\lambda)$ is sufficiently close to 1 for reasonable parameters of λ.

4 Trapdoor Hash Functions

In this section, we provide a formalization of our main primitive: a trapdoor hash scheme TDH for a class of predicates $\mathcal{F}$, then show how to realize it for the class of index predicates, and more generally, for linear predicates, under three different cryptographic assumptions: DDH, QR, and LWE.

4.1 Model and Formal Definition

Defining Trapdoor Hash. A *trapdoor hash scheme* (TDH for short), in essence, defines a publicly parameterized hash function $h_{\mathsf{hk}} : \{0,1\}^n \to \{0,1\}^\eta$, through its *sampling algorithm*, that allows two parties, Alice and Bob, to perform the following functionality:

- *Generating a key and trapdoor for encoding:* Alice, who has a private predicate $f \in \mathcal{F}$ over $\{0,1\}^n$ (for a pre-defined class of predicates $\mathcal{F}$), generates a pair of an encoding key ek and a trapdoor td. Alice can publish ek on her website, while keeping f private (hence, *function privacy*).
- *Hashing:* using the public hash key hk, Bob, who has a private input $\mathsf{x} \in \{0,1\}^n$, can compute a hash $h_{\mathsf{hk}}(\mathsf{x})$ that does not reveal x (hence, *input privacy*), and send it to Alice.
- *Encoding:* using the encoding key ek, anyone, including Bob, can compute an encoding $\mathsf{e} := \mathsf{E}(\mathsf{ek}, \mathsf{x})$ for his input $\mathsf{x} \in \{0,1\}^n$.
- *Decoding:* Alice, who has the secret trapdoor td, can decode the encoding e to recover $f(\mathsf{x})$, given only the hash $h_{\mathsf{hk}}(\mathsf{x})$. In fact, Alice would be able to generate two encodings: a 0-encoding e_0 and a 1-encoding e_1, where it is guaranteed that $\mathsf{e} = \mathsf{e}_{f(\mathsf{x})}$ (this is *correctness*). Notice that since the encodings are computable by Alice using the hash, they too do not reveal any information about x.

Definition 4.1 (Trapdoor Hash Scheme (TDH))**.** *Let $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be a class of predicates, where each $\mathcal{F}_n$ is a set of predicates defined over over $\{0,1\}^n$, and let $\omega := \omega(\lambda) \in \mathbb{N}$ for any $\lambda \in \mathbb{N}$. A* rate-$\frac{1}{\omega}$ TDH scheme for $\mathcal{F}$ *is a tuple of five PPT algorithms $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ with the following properties.*

- ***Syntax:***
  - $\mathsf{hk} \leftarrow \mathsf{S}(1^\lambda, 1^n)$. *The sampling algorithm takes as input a security parameter λ and an input length n, and outputs a hash key hk.*
  - $(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{G}(\mathsf{hk}, f)$. *The generating algorithm takes as input a hash key hk and a predicate $f \in \mathcal{F}_n$, and outputs a pair of an encoding key ek and a trapdoor td.*
  - $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$. *The hashing algorithm takes as input a hash key hk, a string $\mathsf{x} \in \{0,1\}^n$ and randomness $\rho \in \{0,1\}^*$, and deterministically outputs a hash value $\mathsf{h} \in \{0,1\}^\eta$.*

– $\mathsf{e} \leftarrow \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$. *The encoding algorithm takes as input an encoding key* ek*, string* $\mathsf{x} \in \{0,1\}^n$ *and randomness* $\rho \in \{0,1\}^*$*, and deterministically outputs an encoding* $\mathsf{e} \in \{0,1\}^\omega$.

– $(\mathsf{e}_0, \mathsf{e}_1) \leftarrow \mathsf{D}(\mathsf{td}, \mathsf{h})$. *The decoding algorithm takes as input a trapdoor* td*, a hash value* $\mathsf{h} \in \{0,1\}^\eta$*, and outputs a pair of a 0-encoding and a 1-encoding* $(\mathsf{e}_0, \mathsf{e}_1) \in \{0,1\}^\omega \times \{0,1\}^\omega$.

- ***Correctness:*** *$\mathcal{H}$ is* $(1-\epsilon)$-correct *(or has ϵ error probability), for $\epsilon := \epsilon(\lambda) < 1$, if the following holds for any $\lambda, n \in \mathbb{N}$, any $\mathsf{x} \in \{0,1\}^n$ and any predicate $f \in \mathcal{F}_n$.*

$$\Pr[\mathsf{e} = \mathsf{e}_{f(x)}] \geq 1 - \mathsf{negl}(\lambda) \qquad \Pr[\mathsf{e} \neq \mathsf{e}_{1-f(x)}] \geq 1 - \epsilon - \mathsf{negl}(\lambda)$$

where $\mathsf{hk} := \mathsf{S}(1^\lambda, 1^n)$*,* $(\mathsf{ek}, \mathsf{td}) := \mathsf{G}(\mathsf{hk}, f)$*,* $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$ *and* $\mathsf{e} := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$ *for* $\rho \xleftarrow{\$} \{0,1\}^*$*, and* $(\mathsf{e}_0, \mathsf{e}_1) := \mathsf{D}(\mathsf{td}, \mathsf{h})$*. When $\epsilon = 0$ we say that the scheme is* fully correct*.*

- ***Function Privacy:*** *$\mathcal{H}$ is* function-private *if for any polynomial-length $\{1^{n_\lambda}\}_{\lambda\in\mathbb{N}}$ and any $\{f_n\}_{n\in\mathbb{N}}$ and $\{f'_n\}_{n\in\mathbb{N}}$ such that $f_n, f'_n \in \mathcal{F}_n$ for all $n \in \mathbb{N}$, it holds that*

$$\{(\mathsf{hk}_\lambda, \mathsf{ek}_\lambda)\}_{\lambda\in\mathbb{N}} \stackrel{c}{\equiv} \{(\mathsf{hk}_\lambda, \mathsf{ek}'_\lambda)\}_{\lambda\in\mathbb{N}}$$

where $\mathsf{hk}_\lambda \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^{n_\lambda})$*,* $(\mathsf{ek}_\lambda, \mathsf{td}_\lambda) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}_\lambda, f_{n_\lambda})$ *and* $(\mathsf{ek}'_\lambda, \mathsf{td}'_\lambda) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}_\lambda, f'_{n_\lambda})$*.*

- ***Input Privacy:*** *$\mathcal{H}$ is* input-private *if for any polynomial-length $\{\mathsf{x}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathsf{x}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that $n_\lambda := |\mathsf{x}_\lambda| = |\mathsf{x}'_\lambda|$, it holds that*

$$\{(\mathsf{hk}_\lambda, \mathsf{h}_\lambda)\}_{\lambda\in\mathbb{N}} \stackrel{c}{\equiv} \{(\mathsf{hk}_\lambda, \mathsf{h}'_\lambda)\}_{\lambda\in\mathbb{N}}$$

where $\mathsf{hk}_\lambda \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^{n_\lambda})$*,* $\mathsf{h}_\lambda = \mathsf{H}(\mathsf{hk}_\lambda, \mathsf{x}_\lambda; \rho)$ *and* $\mathsf{h}' = \mathsf{H}(\mathsf{hk}_\lambda, \mathsf{x}'_\lambda; \rho')$ *for* $\rho, \rho' \xleftarrow{\$} \{0,1\}^*$*. When the indistinguishability above is statistical, we say we have statistical input privacy.*

- ***Compactness:*** *we require that the image length of the hash function, η, is independent of n, and is bounded by some polynomial in the security parameter λ.*

We will also define a weaker version of correctness, as one of our LWE-based construction does not achieve the slightly stronger version above.

Definition 4.2 (Weakly Correct TDH)**.** *A* TDH *scheme $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ for the class of predicates $\mathcal{F}$ is* $(1-\epsilon)$-weakly correct*, for $\epsilon := \epsilon(\lambda) < 1$, if the following holds for any $\lambda, n \in \mathbb{N}$, any $\mathsf{x} \in \{0,1\}^n$, and any predicate $f \in \mathcal{F}$.*

$$\Pr[\mathsf{e} = \mathsf{e}_{f(x)}] \geq 1 - \epsilon - \mathsf{negl}(\lambda) \qquad \Pr[\mathsf{e} \neq \mathsf{e}_{1-f(x)}] \geq 1 - \epsilon - \mathsf{negl}(\lambda)$$

where $\mathsf{hk} := \mathsf{S}(1^\lambda, 1^n)$*,* $(\mathsf{ek}, \mathsf{td}) := \mathsf{G}(\mathsf{hk}, f)$*,* $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$ *and* $\mathsf{e} := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$ *for* $\rho \xleftarrow{\$} \{0,1\}^*$*, and* $(\mathsf{e}_0, \mathsf{e}_1) := \mathsf{D}(\mathsf{td}, \mathsf{h})$*.*

Useful Classes of Predicates. In this work, we consider TDH schemes for two classes of predicates over $\{0,1\}^n$. The first is the class of *index predicates*, that output the i^{th} bit of an input x for some index $i \in [n]$, and is formally defined as $\mathcal{I} = \{\mathcal{I}_n\}_{n\in\mathbb{N}}$, where

$$\mathcal{I}_n = \{f_{[i]}(x) = x[i] \mid i \in [n]\} \tag{4.1}$$

The second is a generalization that consist of all *linear predicates* over $\mathbb{F}_2$, $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$, where

$$\mathcal{L}_n = \{f_\alpha(x) = \sum_{i=1}^{n} \alpha_j x_i \mid \alpha \in \{0,1\}^n\} \tag{4.2}$$

where addition and multiplication are defined in $\mathbb{F}_2$.

Remark 4.1 (On the generality of index predicates)**.** *We note that one can obtain a* TDH *scheme for any class of predicates $\mathcal{F}$ using a* TDH *scheme for the class of index predicates $\mathcal{I}$. Given an input $\mathsf{x} \in \{0,1\}^n$, we simply compute the hashing algorithm on the string $\mathsf{x}' := (f_1(\mathsf{x}), \ldots, f_m(\mathsf{x}))$, where $f_1, \ldots, f_m$ are all the predicates in $\mathcal{F}_n$. To encode inputs w.r.t. predicate f_i, we generate an encoding key (and a corresponding trapdoor) for the index predicate $f_{[i]}$, and accordingly, the encoding is computed over x'. The obtained scheme would have a rate and hash length equal to those of the underlying scheme. Observe, however, that the length of the hashing and encoding keys become proportional to m, rather than the input length n. The correctness and security properties are clearly preserved.*

Remark 4.2 (Trapdoor hash for general functions)**.** *Although we define trapdoor hash only w.r.t. predicates, the same functionality can be realized for general classes of functions, with outputs longer than a single bit, through multiple executions. More specifically, every function $f : \{0,1\}^n \to \{0,1\}^m$ can be looked at as a collection of m predicates, each defined for one bit of the output string. If we are able to construct* TDH *scheme for each of these predicates, then we obtain a* TDH *for the function f. The rate in such a general setting is the ratio between m and the length of the* TDH *encoding.*

4.2 Trapdoor Hash for Index Predicates from DDH

In this section, we present our first realization of TDH. We begin with a "basic" construction of a rate-$\frac{1}{\lambda}$ TDH scheme for index predicates. The scheme provides full correctness, statistical input privacy, and function privacy based on the DDH assumption. We then show how to use further techniques to achieve a rate-1 TDH in the expense of a $\frac{1}{\lambda}$ error probability.

We note that the basic construction is not merely an intermediary step, as it provides a property that is crucial for some of our applications (in particular, the *secret encoding* property for private laconic OT - see Section 7), and that is inherently unachievable by a rate-1 TDH.

4.2.1 Basic Construction

The basic scheme is inspired by a construction given in [CDG+17, DG17] in the contexts of laconic OT and identity-based encryption (resp.). Our scheme, however, is a variant closer to the construction from [GH18], which is also based on [CDG+17, DG17], and used to construct a KEM for one-way function with encryption (OWFE). In contrast to prior works we only achieve security under DDH while these previous works based security under CDH. This need of stronger assumption arrise because our constructions need to satisfy a more stringent notion of security.

Overview. Similar to the aforementioned works, we define the public hash key to consist of a $2 \times n$ matrix of random group elements $\mathbf{A} := \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix}$ and a generator g, and define the hash value of an input x to be $\mathsf{h} := H_{g,\mathbf{A}}(\mathsf{x}; r) := g^r \prod_j g_{j,\mathsf{x}[j]}$. Notice that when r is uniform, so is the hash of an input x. Thus, our scheme provides statistical input privacy.

Now, given an index $i \in [n]$, Alice computes a corresponding encoding key as $u := g^s$ and $\mathbf{B} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{n,0} \\ u_{1,1}, u_{2,1}, \ldots, u_{n,1} \end{pmatrix}$, where $s \in \mathbb{Z}_p$ is random and, for every $j \in [n], b \in \{0,1\}$, $u_{j,b} := g_{j,b}^s$. The only exception is $u_{i,1}$ which is set as $g_{i,1}^s g^t$ for another random exponent $t \in \mathbb{Z}_p$. Alice's secret trapdoor is set to be s and t. We rely on the DDH assumption to show that the elements in $\mathbf{B}$ are all indistinguishable from random, and therefore, i remains private given the encoding key.

Given the encoding key $(u, \mathbf{B})$, Bob computes an encoding $\mathsf{e} := H_{u,\mathbf{B}}(\mathsf{x}; r) := u^r \prod_j u_{j,\mathsf{x}[j]}$. Observe that when $\mathsf{x}_i = 0$, then this value is equal to h^s, and that otherwise, it is equal to $\mathsf{h}^s g^t$. Both of these values are computable by Alice given her trapdoor, and that is indeed how we define the output of the decoding algorithm: e_0 and respectively e_1.

Construction 4.1 (Rate-$\frac{1}{\lambda}$ TDH for $\mathcal{I}$ from DDH)**.** *Our basic DDH-based* TDH *scheme consists of the following algorithms.*

- $\mathsf{S}(1^\lambda, 1^n)$:
  1. *Sample* $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}$
  2. *Sample a matrix*
$$\mathbf{A} := \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix} \xleftarrow{\$} \mathbb{G}^{2 \times n}$$
  3. *Output*
$$\mathsf{hk} := ((\mathbb{G}, p, g), \mathbf{A}) \tag{4.3}$$
- $\mathsf{G}(\mathsf{hk}, f_i)$: *parse* hk *as in Equation 4.3 and proceed as follows.*
  1. *Sample* $s, t \xleftarrow{\$} \mathbb{Z}_p$.
  2. *Set*
$$u := g^s$$
  and
$$\mathbf{B} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{n,0} \\ u_{1,1}, u_{2,1}, \ldots, u_{n,1} \end{pmatrix} \qquad \textit{where} \qquad u_{j,b} := \begin{cases} g_{j,b}^s \cdot g^t & \textit{if } (j,b) = (i,1) \\ g_{j,b}^s & \textit{otherwise} \end{cases}$$
  3. *Output*
$$\mathsf{ek} := (u, \mathbf{B}) \qquad\qquad \mathsf{td} := (s, t) \tag{4.4}$$
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *parse* hk *as in Equation 4.3,* $\mathbf{A} = (g_{j,b})_{j \in [n], b \in \{0,1\}}$ *and* ρ *as* $r \in \mathbb{Z}_p$*, and output*
$$\mathsf{h} := g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]} \tag{4.5}$$

- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: parse* ek *as* $(u, \mathbf{B})$, $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$ *and* ρ *as* $r \in \mathbb{Z}_p$ *and output*

$$\mathsf{e} := u^r \cdot \prod_{j=1}^{n} u_{j,\mathsf{x}[j]} \tag{4.6}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$: *parse* $\mathsf{h} \in \mathbb{G}$ *and* td *as in Equation 4.4 and output*

$$\mathsf{e}_0 := \mathsf{h}^s \qquad\qquad \mathsf{e}_1 := \mathsf{h}^s g^t \tag{4.7}$$

Analysis. Full correctness of the above construction is immediate. We now proceed to show that the scheme provides statistical input privacy and (computational) function-privacy under the DDH assumption.

Theorem 4.1 (Input privacy of basic DDH-based construction)**.** *The* TDH *scheme from Construction 4.1 provides statistical input security.*

Proof. It is not hard to see that since the hash value is multiplied by g^r for a uniform $r \in \mathbb{Z}_p$ (see equation 4.5), then it also distributes uniformly given hk. Thus, for any $\mathsf{x} \in \{0,1\}^n$, it holds that $(\mathsf{hk}, \mathsf{h}) \equiv (\mathsf{hk}, \mathsf{h})$ for a uniform $h \in \mathbb{G}$, when $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$ and $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$ where $\rho \xleftarrow{\$} \{0,1\}^*$. □

Theorem 4.2 (Function privacy of basic DDH-based construction)**.** *The* TDH *scheme from Construction 4.1 provides function privacy under the DDH assumption.*

Proof. Fix some $\lambda, n \in \mathbb{N}$. We show that, under the DDH assumption, for any $i \in [n]$, it holds that

$$(\mathsf{hk}, \mathsf{ek}) \overset{c}{\equiv} (\mathsf{hk}, (u', \mathbf{B}')) \tag{4.8}$$

where $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \cdot) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_i)$ and $u' \xleftarrow{\$} \mathbb{G}$ and $\mathbf{B}' \xleftarrow{\$} \mathbb{G}^{2\times n}$. This clearly suffices.

We prove the indistinguishability using a hybrids argument. More specifically, we define $2n+2$ hybrid distributions: $\mathsf{Hybrid}_0, \mathsf{Hybrid}_{1,0}, \mathsf{Hybrid}_{1,1}, \ldots, \mathsf{Hybrid}_{n,0}, \mathsf{Hybrid}_{n,1}$, where

$$\mathsf{Hybrid}_0 := (\mathsf{hk}, \mathsf{ek}) \qquad\qquad \mathsf{Hybrid}_{n,1} := (\mathsf{hk}, (u', \mathbf{B}'))$$

and show that every two adjacent hybrids in the sequence are computationally indistinguishable using reduction to DDH. This would complete the proof.

For every $k \in [n]$ and $b \in \{0,1\}$, define $\mathsf{Hybrid}_{k,b}$ by taking its preceding hybrid (which is either $\mathsf{Hybrid}_{k,b-1}$ or $\mathsf{Hybrid}_{k-1,b}$) and replacing $u_{k,b}$ in $\mathbf{B}$ with a uniform group element $u'_{k,b}$. More formally,

$$\mathsf{Hybrid}_{k,0} := \left(\mathsf{hk}, \left(u, \begin{pmatrix} u'_{1,0}, \ldots, u'_{k-1,0}, \boldsymbol{u'_{k,0}}, u_{k+1,0}, \ldots, u_{n,0} \\ u'_{1,1}, \ldots, u'_{k-1,1}, u_{k,1}, u_{k+1,1}, \ldots, u_{n,1} \end{pmatrix}\right)\right)$$

$$\mathsf{Hybrid}_{k,1} := \left(\mathsf{hk}, \left(u, \begin{pmatrix} u'_{1,0}, \ldots, u'_{k-1,0}, u'_{k,0}, u_{k+1,0}, \ldots, u_{n,0} \\ u'_{1,1}, \ldots, u'_{k-1,1}, \boldsymbol{u'_{k,1}}, u_{k+1,1}, \ldots, u_{n,1} \end{pmatrix}\right)\right)$$

where $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(u, (u_{j,b})_{j\in[n],b\in\{0,1\}}) := \mathsf{G}(\mathsf{hk}, f_i)$, and $(u'_{j,b})_{j\in[n],b\in\{0,1\}} \xleftarrow{\$} \mathbb{G}^{2\times n}$.

It is easy to see that $\mathsf{Hybrid}_{i-1,1} \equiv \mathsf{Hybrid}_{i,0} \equiv \mathsf{Hybrid}_{i,1}$, and therefore, we focus on the case where $k \neq i$. Actually, w.l.o.g. we show that $\mathsf{Hybrid}_{k-1,1} \stackrel{c}{\equiv} \mathsf{Hybrid}_{k,0}$ for every $k > i$. Proving indistinguishability for $k < i$, or for hybrids $\mathsf{Hybrid}_{k,0}$ and $\mathsf{Hybrid}_{k,1}$ is identical up to technicalities.

Let $\mathcal{D}$ be a PPT distinguisher for which

$$|\Pr[\mathcal{D}(\mathsf{Hybrid}_{k-1,1}) = 1] - \Pr[\mathcal{D}(\mathsf{Hybrid}_{k,0}) = 1]| > \mathsf{negl}(\lambda)$$

for any negligible function $\mathsf{negl}(\cdot)$. We construct an adversary $\mathcal{A}$ that uses $\mathcal{D}$ to break the DDH assumption. $\mathcal{A}$ takes as input a tuple $((\mathbb{G}, p, g), (R, S, T))$, and proceeds as follows.

1. For every $j \in [n] \backslash \{k\}$ and $b \in \{0, 1\}$, sample $r_{j,b} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$.
2. For every $j \in [k-1]$ and $b \in \{0, 1\}$, sample $t_{j,b} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$.
3. For every $j \in [k-1]$, set

$$\begin{aligned} \tilde{g}_{j,0} &:= g^{r_{j,0}} & \tilde{g}_{j,1} &:= g^{r_{j,1}} \\ \tilde{u}_{j,0} &:= g^{t_{j,0}} & \tilde{u}_{j,1} &:= g^{t_{j,1}} \end{aligned}$$

4. Set

$$\begin{aligned} \tilde{g}_{k,0} &:= R & \tilde{g}_{k,1} &:= g^{r_{k,1}} \\ \tilde{u}_{k,0} &:= T & \tilde{u}_{k,1} &:= S^{t_{k,1}} \end{aligned}$$

5. For every $k < j \leq n$, set

$$\begin{aligned} \tilde{g}_{j,0} &:= g^{r_{j,0}} & \tilde{g}_{j,1} &:= g^{r_{j,1}} \\ \tilde{u}_{j,0} &:= S^{r_{j,0}} & \tilde{u}_{j,1} &:= S^{r_{j,1}} \end{aligned}$$

6. Set

$$\widetilde{\mathsf{hk}} := \left((\mathbb{G}, p, g), \begin{pmatrix} \tilde{g}_{1,0}, \tilde{g}_{2,0}, \ldots, \tilde{g}_{n,0} \\ \tilde{g}_{1,1}, \tilde{g}_{2,1}, \ldots, \tilde{g}_{n,1} \end{pmatrix}\right) \qquad \widetilde{\mathsf{ek}} := \left(S, \begin{pmatrix} \tilde{u}_{1,0}, \tilde{u}_{2,0}, \ldots, \tilde{u}_{n,0} \\ \tilde{u}_{1,1}, \tilde{u}_{2,1}, \ldots, \tilde{u}_{n,1} \end{pmatrix}\right)$$

and output $\mathcal{D}\left(\widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}\right)$.

From the definitions of the hybrids, one can see that if $(R, S, T) \equiv (g^r, g^s, g^{rs})$ for $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, then

$$(\widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}) \equiv \mathsf{Hybrid}_{k-1,1}$$

and that otherwise, if $(R, S, T) \equiv (g^r, g^s, g^r)$ for $r, s, t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, then

$$(\widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}) \equiv \mathsf{Hybrid}_{k,0}$$

This completes the proof of the lemma. □

4.2.2 From Rate-$1/\lambda$ to Rate-1

We next show how to use additional techniques to achieve an optimal rate TDH.

Overview. Our rate-1 construction builds over the basic construction from Section 4.2.1 above. Recall that in the basic construction, the encodings e_0 and e_1 are essentially group elements with $\mathsf{e}_1 = \mathsf{e}_0 g^t$ for t that is independent of the input. To achieve rate-1, we "compress" each of the encodings to a single bit using a *distance function*, a variant of which was first used by Boyle et al. [BGI16] in a different context. Roughly speaking, the distance function is defined over a group $\mathbb{G}$, and computes, with high probability, the distance of a group element $h \in \mathbb{G}$ from a global random epoch $h_0 \in \mathbb{G}$, in terms of multiplications by a public generator $g \in \mathbb{G}$. The idea, then, is to compute the distance of both encodings w.r.t. multiplications by g^t. Since e_0 and e_1 are one step far of each other, their distances from the epoch have different parities, and therefore, it is sufficient to encode using the least significant bit of the distance. This idea of compression to one is inspired by Garg et al. [GGH18], although we use very different techniques to achieve the compression.

Below, we formally define the distance function, and state its useful property. We then proceed to a detailed description of the rate-1 construction and its analysis.

The Distance Function. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p, and let $g \in \mathbb{G}$ be a generator. We recall the definition of the subroutine $\mathsf{Dist}_{\mathbb{G},g}$ from [BGI16], which takes as input an element $h \in \mathbb{G}$, bounds on the failure probability $\delta > 0$ and on the input range $M \in \mathbb{N}$, and a pseudo-random function [GGM84] $\mathsf{PRF}_K : \mathbb{G} \to \{0,1\}^{\lceil \log(2M/\delta) \rceil}$. Roughly speaking, the function outputs the parity of the "distance" between h and the next zero of PRF_K (as opposed to the original definition where the output is the actual distance).

$\mathsf{Dist}_{\mathbb{G},g}(h, \delta, M, K)$:

1. Define $T := [2M \log_e(2/\delta)]/\delta$, and set $i := 0$.
2. While $i \leq T$:
   2.1. If $\mathsf{PRF}_K(h \cdot g^i) = 0^{\lceil \log(2M/\delta) \rceil}$ then output $\mathsf{LSB}(i)$, else set $i := i + 1$.
3. Output $\mathsf{LSB}(i)$.

where LSB returns the least significant bit of a certain integer.

We hereby state a property of the distance function.

Proposition 4.1 (Proposition 3.2 in [BGI16])**.** *Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p, and let $g \in \mathbb{G}$, $\delta > 0$, $M \in \mathbb{N}$ with $[2M \log_e(2/\delta)]/\delta < p$. Let $\mathsf{PRF}_K : \mathbb{G} \to \{0,1\}^{\lceil \log(2M/\delta) \rceil}$ be a pseudo-random function with a uniformly random key $K \xleftarrow{\$} \{0,1\}^\lambda$. Then, for any $h \in \mathbb{G}$ and $x \leq M$, it holds that*

$$\Pr[\mathsf{Dist}_{\mathbb{G},g}(h, \delta, M, K) \oplus \mathsf{Dist}_{\mathbb{G},g}(hg^x, \delta, M, K) = \mathsf{LSB}(x)] \geq 1 - \delta$$

where the probability is taken over the choice of K.

We stress that our construction is completely parameteric in the implementation of the distance function and therefore other variants [DKK18] can also be used.

Construction 4.2 (Rate-1 TDH for $\mathcal{I}$ from DDH)**.** *Let $\mathcal{H}' = (\mathsf{S}', \mathsf{G}', \mathsf{H}', \mathsf{E}', \mathsf{D}')$ be the basic DDH-based* TDH *from construction 4.1. Our rate-1 DDH-based* TDH *scheme for index predicates consists of the following algorithms.*

- $\mathsf{S}(1^\lambda, 1^n)$: *output* $\mathsf{hk} \xleftarrow{\$} \mathsf{S}'(1^\lambda, 1^n)$.
- $\mathsf{G}(\mathsf{hk}, f_i)$: *sample* $((u, \mathbf{B}), (s, t)) \xleftarrow{\$} \mathsf{G}'(\mathsf{hk}, f_i)$ *and* $K \xleftarrow{\$} \{0,1\}^\lambda$*, and output*

$$\mathsf{ek} := (u, \mathbf{B}, t, K) \qquad\qquad \mathsf{td} := (s, t, K) \tag{4.9}$$

- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *output* $\mathsf{h} := \mathsf{H}'(\mathsf{hk}, \mathsf{x}; \rho)$.
- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: parse* ek *as in Equation 4.9, set* $e := \mathsf{E}'(\mathsf{ek}, \mathsf{x}; \rho)$*, and output*

$$\mathsf{e} := \mathsf{Dist}_{g^t}(e, 1/\lambda, 1, K) \tag{4.10}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$: *parse* $\mathsf{h} \in \mathbb{G}$ *and* td *as in Equation 4.9, set* $(e_0, e_1) := \mathsf{D}'(s, \mathsf{h})$ *and output*

$$\mathsf{e}_0 := \mathsf{Dist}_{g^t}(e_0, 1/\lambda, 1, K) \qquad\qquad \mathsf{e}_1 := \mathsf{Dist}_{g^t}(e_1, 1/\lambda, 1, K) \tag{4.11}$$

Analysis. In the following we show that the scheme has $\frac{1}{\lambda}$ error probability. The statistical input privacy and function privacy of the scheme are implied (almost) immediately from Theorems 4.1 and 4.2 (resp.).

Theorem 4.3 (Correctness of rate-1 DDH-based construction)**.** *Let* PRF *be a psuedorandom function, then the* TDH *scheme from Construction 4.2* $\left(1 - \frac{1}{\lambda}\right)$*-correct* TDH.

Proof. Fix $\lambda, n \in \mathbb{N}$, $\mathsf{x} \in \{0,1\}^n$ and $i \in [n]$ and consider the execution of $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$ and $\mathsf{D}(\mathsf{td}, \mathsf{h})$ for $\rho \xleftarrow{\$} \{0,1\}^*$, $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_i)$ and $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$ where $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$. If $\mathsf{x}_i = 0$ then, from the full correctness of the basic construction, it holds w.p. at least $1 - \mathsf{negl}(\lambda)$, that $e_0 = e$ and therefore $\mathsf{e} = \mathsf{e}_0$ (see Equations 4.10 and 4.11). We also know that $e \neq e_1$, and specifically, $e = e_1 \cdot g^t$ (see Equation 4.7). Using Proposition 4.1, with $\delta = 1/\lambda$ and $x = M = 1$, we imply that $\mathsf{e} \neq \mathsf{e}_1$ w.p. at least $1 - 1/\lambda$. The case where $\mathsf{x}_i = 1$ is symmetric. This concludes our proof. □

4.3 Trapdoor Hash for Linear Predicates from QR

In the following we introduce our construction based on the QR assumption over composite order groups. Unless differently specified, all of the following arithmetic operation are done modulo N.

Overview. The structure of our QR-based scheme is very similar to the DDH-based construction with a few critical difference. Foremost, the scheme naturally extends the class of functions supported by the TDH from index predicates to linear predicates (computed over $\mathbb{F}_2$). Secondly, the scheme exploits the structure of the group $\mathbb{J}_N$ to achieve full correctness.

Recall that $\mathbb{J}_N$ is the composition of two subgroups $\mathbb{H} \times \mathbb{QR}_N$, where $\mathbb{H}$ is the multiplicative group $(\pm 1, \cdot)$ generated by -1. The matrix $\mathbf{A} = (g_{j,b})_{j \in [n], b \in \{0,1\}}$ is a collection of random elements of $\mathbb{QR}_N$ and the function $f_\alpha = (\alpha_1, \ldots, \alpha_n)$ is encoded in $\mathbf{B} = (u_{j,b})_{j \in [n], b \in \{0,1\}}$ as follows: Each component $g_{j,b}$ of $\mathbf{A}$ is raised to some exponent s and then -1 is multiplied to all components

with index i of the second row of $\mathbf{A}$ where $\alpha_j = 1$. When computing the encoding of some string x, then $\prod_{j=1}^{n} u_{j,\mathsf{x}[j]} = \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}^{s} \cdot (-1)^{f_\alpha(\mathsf{x})}$, which means that the element is either a quadratic residue or not depending on the value of $f_\alpha(\mathsf{x})$. At this point it would be sufficient for the encoding algorithm to output a single bit encoding this information, which however is hard compute since the factorization of N is not given.

Fortunately we can circumvent this issue by exploiting the fact that if an element e belongs to $\mathbb{QR}_N$, then $-e$ does not, and vice versa. The single-bit encoding is computed by simply comparing e and $-e$ according to any ordering, e.g., lexicographical. Note that this trick allows us to bypass the use of the distance function an yields a fully correct TDH with optimal rate, which is an improvement over the DDH-based scheme.

Comparison Operator. We define the function $\mathsf{LEq} : \mathbb{J}_N \times \mathbb{J}_N \to \{0,1\}$ to return 1 if the bit representation of the first input is smaller or equal than the bit representation of the second input according to some order (e.g., lexicographical).

Construction 4.3 (Rate-1 TDH for $\mathcal{L}$ from QR)**.** *Our QR-based* TDH *scheme consists of the following algorithms.*

- $\mathsf{S}(1^\lambda, 1^n)$:
  1. *Sample a Blum integer* $N := p \cdot q$*, where* $p = q = 3 \ (mod\ 4)$.
  2. *Sample* $g \xleftarrow{\$} \mathbb{QR}_N$.
  3. *Sample a matrix*
$$\mathbf{A} := \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix} \xleftarrow{\$} \mathbb{QR}^{2\times n}$$
  4. *Output*
$$\mathsf{hk} := ((N, g), \mathbf{A}) \tag{4.12}$$
- $\mathsf{G}(\mathsf{hk}, f_\alpha)$: *parse* hk *as in Equation 4.12,* $\mathbf{A} = (g_{j,b})_{j\in[n],b\in\{0,1\}}$ *and proceed as follows.*
  1. *Sample* $s \xleftarrow{\$} \left[\frac{N-1}{2}\right]$.
  2. *Set*
$$u := g^s$$
  and
$$\mathbf{B} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{n,0} \\ u_{1,1}, u_{2,1}, \ldots, u_{n,1} \end{pmatrix} \qquad \textit{where} \qquad u_{j,b} := \begin{cases} g_{j,b}^s \cdot (-1)^{\alpha_j} & \textit{if } b = 1 \\ g_{j,b}^s & \textit{otherwise} \end{cases}$$
  3. *Output*
$$\mathsf{ek} := (u, \mathbf{B}) \qquad\qquad \mathsf{td} := s \tag{4.13}$$
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *parse* hk *as in Equation 4.12,* $\mathbf{A} = (g_{j,b})_{j\in[n],b\in\{0,1\}}$ *and* ρ *as* $r \in \left[\frac{N-1}{2}\right]$*, and output*
$$\mathsf{h} := g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]} \tag{4.14}$$

- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$: *parse* ek *as* $(u, \mathbf{B})$, $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$ *and* ρ *as* $r \in \left[\frac{N-1}{2}\right]$, *set*

$$e := u^r \cdot \prod_{j=1}^{n} u_{j,\mathsf{x}[j]} \tag{4.15}$$

and output

$$\mathsf{e} := \mathsf{LEq}(e, (-1) \cdot e) \tag{4.16}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$: *parse* $\mathsf{h} \in \mathbb{QR}$ *and* td *as in Equation 4.13 and output*

$$\mathsf{e}_0 := \mathsf{LEq}(\mathsf{h}^s, (-1) \cdot \mathsf{h}^s) \qquad \mathsf{e}_1 := \mathsf{LEq}((-1) \cdot \mathsf{h}^s, \mathsf{h}^s) \tag{4.17}$$

Analysis. In the following we show that the scheme satisfies all of the properties for a TDH.

Theorem 4.4 (Correctness of QR-based construction)**.** *The* TDH *scheme from Construction 4.3 is a fully correct* TDH*.*

Proof. To show that the scheme is correct, it is enough to observe that

$$\begin{aligned}
\mathsf{e} &= \mathsf{LEq}(e, (-1) \cdot e) \\
&= \mathsf{LEq}\left(u^r \cdot \prod_{j=1}^{n} u_{j,\mathsf{x}[j]}, (-1) \cdot u^r \cdot \prod_{j=1}^{n} u_{j,\mathsf{x}[j]}\right) \\
&= \mathsf{LEq}\left((-1)^{f_\alpha(\mathsf{x})} \cdot \left(g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right)^s, (-1)^{f_\alpha(\mathsf{x})\oplus 1} \cdot \left(g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right)^s\right) \\
&= \mathsf{LEq}\left((-1)^{f_\alpha(\mathsf{x})} \cdot \mathsf{h}^s, (-1)^{f_\alpha(\mathsf{x})\oplus 1} \cdot \mathsf{h}^s\right) \\
&= \mathsf{e}_{f_\alpha(\mathsf{x})}
\end{aligned}$$

with probability 1. Also note that $\mathsf{e}_0 \neq \mathsf{e}_1$ for all $\mathsf{h}^s \in \mathbb{QR}_N$. □

Theorem 4.5 (Input privacy of QR-based construction)**.** *The* TDH *scheme from Construction 4.3 provides statistical input security.*

Proof. Let $r \xleftarrow{\$} \left[\frac{N-1}{2}\right]$ and $\tilde{r} \xleftarrow{\$} \left[\frac{\varphi(N)}{2}\right]$, then it holds that

$$\left(\mathsf{hk}, \mathsf{h} := g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right) \overset{s}{\equiv} \left(\mathsf{hk}, \tilde{\mathsf{h}} := g^{\tilde{r}} \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right)$$

since $(N-1)$ and $\varphi(N)$ are statistically close. Note that $g^{\tilde{r}}$ is a uniformly sampled element of $\mathbb{QR}_N$, thus so is $\tilde{\mathsf{h}}$, for all $\mathsf{x} \in \{0,1\}^n$. This concludes our proof. □

Theorem 4.6 (Function privacy of QR-based construction)**.** *The* TDH *scheme from Construction 4.3 provides function privacy under the QR assumption.*

Proof. Fix some $(\lambda, n) \in \mathbb{N}$. Let us rewrite

$$\begin{aligned}
&(\mathsf{hk}, \mathsf{ek}) \\
&= ((N, g, \mathbf{A}), (u, \mathbf{B})) \\
&= \left(\left(N, g, \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix}\right), \left(g^s, \begin{pmatrix} g_{1,0}^s, & g_{2,0}^s, & \ldots, & g_{n,0}^s \\ g_{1,1}^s \cdot (-1)^{\alpha_1}, & g_{2,1}^s \cdot (-1)^{\alpha_2}, & \ldots, & g_{n,1}^s \cdot (-1)^{\alpha_n} \end{pmatrix}\right)\right).
\end{aligned}$$

Next we define ek' as

$$\mathsf{ek}' := \left(g^s, \begin{pmatrix} g_{1,0}^s, & g_{2,0}^s, & \ldots, & g_{n,0}^s \\ g_{1,1}^s, & g_{2,1}^s, & \ldots, & g_{n,1}^s \end{pmatrix}\right).$$

By an invocation of Proposition 3.1 it holds that

$$(\mathsf{hk}, \mathsf{ek}) \stackrel{c}{\equiv} (\mathsf{hk}, \mathsf{ek}').$$

Observe that in $(\mathsf{hk}, \mathsf{ek}')$ the variable f_α is hidden in an information theoretic sense. □

Trapdoor Hash from DCR. The above approach can be easily generalized to be instantiated with the subgroup indistinguishability assumption (as defined in [BG10]), which captures the quadratic residuosity and the decisional composite residuosity (DCR) assumption as special cases. For the latter case however, the "message" group (borrowing the terminology from [BG10]) has order N as opposed to 2, which is exponentially large. Therefore we need a different strategy for the encoding algorithm: We suggest using the same mechanism as in our DDH-based construction, which leverages the distance function to establish a noisy channel with a $\left(1 - \frac{1}{\lambda}\right)$-fraction of erasures. Since it is a trivial extension of what already shown, we defer the description of the DCR-based construction to Appendix A.

4.4 Trapdoor Hash for Linear Predicates from LWE

In the following we present our lattice-based construction.

Overview. We briefly explain how to adapt the ideas from the previous schemes to the lattice settings. The first step is to replace the subset product function with a subset sum, defined in a natural way: The hash function is set to be $\mathsf{h} := \sum_{j=1}^{\tilde{n}} \mathbf{a}_{j,\tilde{\mathsf{x}}[j]}$ for uniformly random $\mathbf{a}_{j,b} \stackrel{\$}{\leftarrow} \mathbb{Z}^k$. We then embed a linear function f_α in the encoding key by simply adding $(q/2) \cdot \alpha_j$ to the i-th element $u_{j,1}$, where each $u_{j,b}$ is an LWE sample of $\mathbf{a}_{j,b}$ with secret vector $\mathbf{s}$. The effect of this is that the subset sum encoding effectively evaluates f_α over the input x, since (ignoring the error) it holds that $e := \sum_{j=1}^{n} u_{j,\tilde{\mathsf{x}}[j]} = \mathbf{s}^\top \mathsf{h} + f_\alpha(\mathsf{x}) \cdot q/2$.

Observe that e is either equal to $\mathbf{s}^\top \mathsf{h}$ or an additive factor $(q/2)$ off, depending on the function output. This information can be compressed to a single bit rounding e to the nearest multiple of $q/2$. An important property of this encoding is that it is insensitive to small perturbations, which allows us to establish correctness even in the presence of noise.

The Rounding Function. Let $q = 2p$ be an even integer modulus. Define the rounding function $\lfloor \cdot \rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ by

$$\lfloor x \rceil_2 = \lfloor \bar{x} \cdot 2/q \rceil \mod 2$$

where $\bar{x} \in \mathbb{Z}$ is an arbitrary residue-class representative of $x \in \mathbb{Z}_q$. In our instantiations. We are going to use the following simple lemma about the rounding function $\lfloor \cdot \rceil_2$, implicitly proven in [BPR12].

Lemma 4.1. *Let $q = 2p$ be an even integer modulus. Let $x \xleftarrow{\$} \mathbb{Z}_q$ be distributed uniformly random. Then it holds for all $v \in [-B, B]$ that $\lfloor x + v \rceil_2 = \lfloor x \rceil_2$, except with probability $(2B+1) \cdot 2/q$ over the choice of x.*

Proof. There are exactly 2 multiples of $q/2$ in $\{0, \ldots, q-1\}$, namely 0 and $q/2$. Thus, the probability that a uniform $x \in \mathbb{Z}_q$ lands B-close from the nearest multiple of $q/2$ is exactly

$$(2B+1) \cdot 2/q.$$

□

Construction 4.4 (Rate-1 trapdoor hash for $\mathcal{L}$ from LWE)**.** *Set $q := 2\tilde{q}$ be an even modulus, let k, n, ℓ, B be positive integers and let χ be a B-bounded error distribution over $\mathbb{Z}$.*

- $\mathsf{S}(1^\lambda, 1^n)$:
  1. *For $j = 1, \ldots, n+\ell$ choose a random $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^k$*
  2. *Set*
$$\mathbf{A} := \begin{pmatrix} \mathbf{a}_{1,0}, \mathbf{a}_{2,0}, \ldots, \mathbf{a}_{n+\ell,0} \\ \mathbf{a}_{1,1}, \mathbf{a}_{2,1}, \ldots, \mathbf{a}_{n+\ell,1} \end{pmatrix}$$
  3. *Set $\mathbf{A} := (\hat{\mathbf{A}}, \tilde{\mathbf{A}})$ and output*
$$\mathsf{hk} := \mathbf{A} \tag{4.18}$$
- $\mathsf{G}(\mathsf{hk}, f_\alpha)$: *parse* hk *as in Equation 4.18 and proceed as follows.*
  1. *Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$ and for $j = 1, \ldots, n+\ell$ and $b \in \{0,1\}$ $e_{j,b} \xleftarrow{\$} \chi$*
  2. *For $j = 1, \ldots, n+\ell$ and $b \in \{0,1\}$ set $u_{j,b} \leftarrow \mathbf{s}^\top \mathbf{a}_{j,b} + e_{j,b} + \alpha_j \cdot b \cdot (q/2)$ (where $\alpha_j = 0$ for $j > n$)*
  3. *Set*
$$\mathbf{B} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{n+\ell,0} \\ u_{1,1}, u_{2,1}, \ldots, u_{n+\ell,1} \end{pmatrix}$$
  4. *Choose a PRF-key $K \xleftarrow{\$} \{0,1\}^\lambda$*
  5. *Output*
$$\mathsf{ek} := (\mathsf{hk}, \mathbf{B}, K) \qquad \mathsf{td} := (\mathbf{s}, K) \tag{4.19}$$
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *parse* hk *as in Equation 4.18, ρ as $\mathbf{r} \in \{0,1\}^\ell$, set $\tilde{\mathsf{x}} := \mathsf{x} \| \mathbf{r}$ and proceed as follows.*
  1. *Output*
$$\mathsf{h} := \sum_{j=1}^{n+\ell} \mathbf{a}_{j,\tilde{\mathsf{x}}[j]} \tag{4.20}$$

- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: parse* ek *as in Equation 4.19 and* ρ *as* $\mathbf{r} \in \{0,1\}^\ell$*, set* $\tilde{\mathsf{x}} := \mathsf{x} \| \mathbf{r}$*, define*

$$e := \sum_{j=1}^{n+\ell} u_{j,\tilde{\mathsf{x}}[j]} \tag{4.21}$$

and output

$$\mathsf{e} := \lfloor e + \mathsf{PRF}_K(\mathsf{h}) \rceil_2 \tag{4.22}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$: *parse* h *as in Equation 4.20 and* td *as in Equation 4.19, compute* $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{hk}, \mathsf{x}, \mathsf{r})$ *and output*

$$\mathsf{e}_0 := \left\lfloor \mathbf{s}^\top \mathsf{h} + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 \qquad \mathsf{e}_1 := \left\lfloor \mathbf{s}^\top \mathsf{h} + \mathsf{PRF}_K(\mathsf{h}) + q/2 \right\rceil_2 \tag{4.23}$$

Analysis. We start by showing that Construction 4.4 satisfies the notion of weak correctness. By choosing the modulus-to-noise ratio sufficienly large (e.g. superpolynomial), we can achieve a negligble correctness error.

Theorem 4.7 (Correctness of LWE-based Construction)**.** *The* TDH *scheme from Construction 4.4 is a* $((2B(n+\ell)+1) \cdot 2/q)$*-weakly correct* TDH*, given that* PRF *is a pseudorandom function.*

Proof. Let us rewrite

$$\begin{aligned}
\mathsf{e} &= \lfloor e + \mathsf{PRF}_K(\mathsf{h}) \rceil_2 \\
&= \left\lfloor \sum_{j=1}^{n+\ell} u_{j,\tilde{\mathsf{x}}[j]} + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 \\
&= \left\lfloor \sum_{j=1}^{n+\ell} (\mathsf{s}^\top \mathbf{a}_{j,\tilde{\mathsf{x}}[j]} + e_{j,\tilde{\mathsf{x}}[j]} + \alpha_j \cdot \mathsf{x}[j] \cdot (q/2)) + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 \\
&= \left\lfloor \mathsf{s}^\top \sum_{j=1}^{n+\ell} \mathbf{a}_{j,\tilde{\mathsf{x}}[j]} + \sum_{j=1}^{n+\ell} e_{j,\tilde{\mathsf{x}}[j]} + f_\alpha(\mathsf{x}) \cdot (q/2) + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 \\
&= \left\lfloor \mathbf{s}^\top \mathsf{h} + e^* + f_\alpha(\mathsf{x}) \cdot (q/2) + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2,
\end{aligned}$$

where $e^* = \sum_{j=1}^{n+\ell} e_{j,\tilde{\mathsf{x}}[j]}$. Note that since all $e_{j,b}$ are B-bounded, it holds that e^* is $(n+\ell)B$-bounded via the triangle inequality. Now, as PRF_K is pseudorandom, we can switch it to a truly random function while only incurring a negligible difference in correctness. Thus assume that $\mathsf{PRF}_K(\mathsf{h})$ is truly random. By Lemma 4.1 we have that

$$\mathsf{e} = \left\lfloor \mathsf{s}^\top \mathsf{h} + (q/2) \cdot f_\alpha(\mathsf{x}) + e^* + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 = \left\lfloor \mathsf{s}^\top \mathsf{h} + (q/2) \cdot f_\alpha(\mathsf{x}) + \mathsf{PRF}_K(\mathsf{h}) \right\rceil_2 = \mathsf{e}_{f_\alpha(\mathsf{x})}$$

except with probability $(2B(n+\ell)+1) \cdot 2/q$ over they choice of $\mathsf{PRF}_K(\mathsf{h})$. To conclude the proof it is sufficient to observe that, for all $z \in \mathbb{Z}_q$, it holds that

$$\lfloor z \rceil_2 \neq \lfloor z + q/2 \rceil_2 .$$

□

We will briefly discuss different parameter instantiations of Theorem 4.7. By choosing q/B to be superpolynomial, we get that $((2B(n+\ell)+1)\cdot 2/q)$ is negligible and thus there is a negligible correctness error. On the other hand, by choosing $q \geq 2\cdot(2Bn+1)\lambda = O(Bn\lambda)$, we get a $1/\lambda$-weakly correct TDH. With this choice of parameters however, we get a polynomial modulus-to-noise ratio of $q/B = O(n\cdot\lambda)$. Towards establishing input privacy, we will show that the function H is in fact a universal hash function.

Lemma 4.2. *Let* hk *be chosen uniformly at random as above, then the function* $\mathsf{H}(\mathsf{hk},\mathsf{x},\rho)$ *is a universal hash function in the input* $\tilde{x} = (\mathsf{x}\|\mathsf{r})$.

Proof. Let $\tilde{\mathsf{x}} \neq \tilde{\mathsf{x}}' \in \mathbb{Z}_q^{n+\ell}$ and let $j^* \in [n+\ell]$ be the index such that $\tilde{\mathsf{x}}[j^*] \neq \tilde{\mathsf{x}}'[j^*]$. It holds that

$$\mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}}) = \sum_{j=1}^{n+\ell} \mathbf{a}_{j,\tilde{\mathsf{x}}[j]} \qquad \mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}}') = \sum_{j=1}^{n+\ell} \mathbf{a}_{j,\tilde{\mathsf{x}}'[j]} = \sum_{j\in[n+\ell]\setminus j^*} \mathbf{a}_{j,\tilde{\mathsf{x}}'[j]} + \mathbf{a}_{\tilde{\mathsf{x}}'[j^*],j^*}.$$

Observe that $\mathbf{a}_{\tilde{\mathsf{x}}[j^*],j^*}$ is uniform and independent of all the other $\mathsf{a}_{j,b}$. Moreover, it does not appear in $\mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}})$. It follows that

$$\Pr\left[\mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}}) = \mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}}')\right] = \frac{1}{q^k}.$$

□

We are now in the position of proving the following theorem.

Theorem 4.8 (Input privacy of LWE-based construction)**.** *The* TDH *scheme from Construction 4.4 provides statistical input security.*

Proof. Fix any $\mathsf{x} \in \{0,1\}^n$ and let $\mathsf{r} \xleftarrow{\$} \{0,1\}^\ell$, where $\ell \geq k\cdot\log(q) + 2\lambda$. Set $\tilde{\mathsf{x}} := \mathsf{x}\|\mathsf{r}$. As r is uniformly random on $\{0,1\}^\ell$, it holds that $H_\infty(\tilde{\mathsf{x}}) \geq \ell$. The Leftover Hash Lemma (Lemma 3.1) yields that

$$\left(\mathsf{hk} := \mathbf{A}, \mathsf{h} := \sum_{j=1}^{n+\ell} \mathbf{a}_{j,\tilde{\mathsf{x}}[j]} = \mathsf{H}(\mathsf{hk},\tilde{\mathsf{x}})\right) \overset{s}{\equiv} (\mathbf{A},\mathsf{u})$$

where u is uniformly chosen from $\mathbb{Z}_q^k$. □

Theorem 4.9 (Function privacy of LWE-based construction)**.** *The* TDH *scheme from Construction 4.4 provides function privacy under the* $\mathsf{LWE}_{k,q,\chi}$ *assumption.*

Proof. As the $\mathbf{a}_{j,b} \in \mathbb{Z}_q^k$ are chosen uniformly at random, we can replace the $\mathbf{s}^\top\mathbf{a}_{j,b} + e_{j,b}$ with uniformly random values under the $\mathsf{LWE}_{k,q,\chi}$ assumption. Consequently, we can replace the

$$u_{j,b} = \mathbf{s}^\top\mathbf{a}_{j,b} + e_{j,b} + \alpha_j\cdot b\cdot(q/2)$$

with uniformly random values and conclude

$$(\mathsf{hk},\mathsf{ek}) = (\mathbf{A},\mathbf{B}) \overset{c}{\equiv} (\mathbf{A},\tilde{\mathbf{B}})$$

where $\mathbf{A}$ and $\mathbf{B}$ are sampled as in Construction 4.4 and $\tilde{\mathbf{B}}$ is chosen uniformly from $\mathbb{Z}_q^{2\times(n+\ell)}$. Note that in the RHS distribution is independent of the function f_α, which concludes the proof. □

5 Rate-1 Oblivious Transfer and More

In this section, we present the first family of applications of trapdoor hash. We show how to use rate-1 trapdoor hash to securely realize basic sender-receiver functionalities through single-round protocols with optimal sender-receiver communication, i.e. optimal *download rate*.

The first fundamental functionality we investigate is *oblivious transfer* (OT), where a receiver with private input $i \in [k]$ communicates with a sender with k secrets in order to obtain the i^{th} secret. We consider two scenarios where download-rate-1 can be achieved. The first is *batch OT*, where a batch of OT instances are invoked in parallel, and the second is *string OT*, which consists of a single OT instance with secrets that are assumed to be sufficiently long. We also discuss a couple of related primitives: *oblivious linear function evaluation* (OLE), where the sender has a linear function $f(x) = ax + b$ and the goal is to evaluate f on the receiver's private input x, and the more general *matrix-vector product* where the sender has a matrix M, the receiver has a vector v, and the goal is to compute the product $Mv^\top$.

We begin by setting a general formal framework for our model and defining the functionalities of focus. We then show how to construct batch OT with optimal download rate using rate-1 TDH for index predicates, and further obtain a string OT protocol with optimal overall rate. Moreover, using trapdoor hash for linear predicates, we extend our constructions to achieve optimal-rate protocols for both OLE and oblivious matrix-vector product (OMV). For completeness, we prove that rate-1 OT is impossible when the sender's input is small, and, in particular, that *exact* rate-1 OT (rather than asymptotic) is unachievable, even when pre-processing is allowed, thus justifying the focus on large input scenarios.

5.1 Model and Definitions

We consider a general sender-receiver setting, where a receiver with input $x \in \{0,1\}^*$ and a sender with input $y \in \{0,1\}^*$ invoke a two-message protocol, at the end of which the receiver is able to recover a value $f(x, y)$ but otherwise learns nothing about y. We also require that x is kept hidden from the sender.

Thus, we formally define a *sender-receiver functionality* as a two-input function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, which is defined over an input domain $\mathcal{D}_f \subseteq \{0,1\}^* \times \{0,1\}^*$. We next formalize our requirements from a two-message protocol for such a functionality f. We define standard notions of correctness and privacy, both for sender and receiver. We also define a stronger notion of sender privacy, which was first proposed in [IP07], and is necessary for some applications.

Definition 5.1 (Two-message protocol for sender-receiver functionality)**.** *A* two-message protocol *(a protocol for short) is a triple of PPT algorithms* $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ *with the following syntax:*

- $(\mathsf{st}, \mathsf{msg}_1) \leftarrow \Pi_1(1^\lambda, x)$. Π_1 *takes as input a security parameter* λ *and a receiver input* $x \in \{0,1\}^*$*, and outputs a receiver message* msg_1 *and a receiver state* st.
- $\mathsf{msg}_2 \leftarrow \Pi_2(\mathsf{msg}_1, y)$. Π_2 *takes as input a receiver message* msg_1 *and a sender input* $y \in \{0,1\}^*$*, and outputs a sender message* msg_2.
- $z \leftarrow \Pi_3(\mathsf{st}, \mathsf{msg}_2)$. Π_3 *takes as input a receiver state* st *and a sender message* msg_2*, and outputs a receiver output* $z \in \{0,1\}^*$.

We say that such a two-message protocol Π realizes *a sender-receive functionality* f *if the following properties hold.*

- ***Correctness:*** Π *is* correct *if there exists a negligible function* $\epsilon(\lambda)$ *such that following holds for all* $\lambda \in \mathbb{N}$ *and all* $(x, y) \in \mathcal{D}_f$.

$$\Pr\left[z = f(x,y) \middle| \begin{array}{ccc} (\mathsf{st}, \mathsf{msg}_1) & \leftarrow & \Pi_1(1^\lambda, x) \\ \mathsf{msg}_2 & \leftarrow & \Pi_2(\mathsf{msg}_1, y) \\ z & \leftarrow & \Pi_3(\mathsf{st}, \mathsf{msg}_2) \end{array}\right] \geq 1 - \epsilon(\lambda)$$

  We say that Π *is* weakly correct *if the above holds for* $\epsilon(\lambda)$ *that is vanishing (but possibly non-negligible).*

- ***Receiver Privacy:*** Π *is (computationally)* receiver-private *if for any (non-uniform, stateful) polynomial time environment* Env*, there exists a negligible function* $\epsilon(\lambda)$ *such that for all* $\lambda \in \mathbb{N}$*,* $\Pr[\mathsf{RPriv}_{\Pi,\mathsf{Env}}(1^\lambda) = 1] \leq \frac{1}{2} + \epsilon(\lambda)$*, where* $\mathsf{RPriv}_{\Pi,\mathsf{Env}}(1^\lambda)$ *is defined as the following experiment:*

  1. $(x_0, x_1, y) \leftarrow \mathsf{Env}(1^\lambda)$
  2. $b \xleftarrow{\$} \{0,1\}; (\cdot, \mathsf{msg}_1) \leftarrow \Pi_1(1^\lambda, x_b)$
  3. $b' \leftarrow \mathsf{Env}(\mathsf{msg}_1)$
  4. *Output 1 if* $b = b'$ *and* $(x_0, y) \in \mathcal{D}_f$ *and* $(x_1, y) \in \mathcal{D}_f$*, otherwise output 0.*

- ***Sender Privacy:*** Π *is statistically, resp. computationally,* sender-private *if there exists a PPT algorithm* Sim *such that for any polynomial-length* $\{x_\lambda, y_\lambda\}_{\lambda\in\mathbb{N}}$*, such that* $(x_\lambda, y_\lambda) \in \mathcal{D}_f$ *for all* λ*, the distribution ensembles* $\{\mathsf{Real}_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathsf{Ideal}_\lambda\}_{\lambda\in\mathbb{N}}$*, where*

$$\mathsf{Real}_\lambda = (\mathsf{st}, \Pi_2(\mathsf{msg}_1, y_\lambda)) \qquad \mathsf{Ideal}_\lambda = (\mathsf{Sim}(1^\lambda, x_\lambda, f(x_\lambda, y_\lambda))$$

  for $(\mathsf{st}, \mathsf{msg}_1) \xleftarrow{\$} \Pi_1(1^\lambda, x_\lambda)$*, are statistically, resp. computationally, indistinguishable.*

  We say that strong sender privacy *holds if the above holds with* Sim *that is not given* x_λ *as input.*

Download Rate. At a high level, the download rate of a two-message protocol measures the multiplicative blow-up in the sender-receiver communication relative to a trusted setting, where the sender can simply send $z = f(x, y)$ given x. That is, the download rate is defined as the asymptotic ratio between the length of $f(x, y)$ and the length of the sender's message msg_2 for inputs $(x, y) \in \mathcal{D}_f$. Since the *difference* between the latter two quantities should grow with the security parameter λ (see Section 5.6), we need to use an asymptotic notion of rate where the output length also grows with λ parameter.

Definition 5.2 (Download rate)**.** *Let* $0 \leq \omega \leq 1$*. We say that a protocol* $\Pi_f = (\Pi_1, \Pi_2, \Pi_3)$ *for a sender-receiver functionality* f *has download rate* ω *if there exists a polynomial* $B(\lambda)$ *such for all polynomial-length input sequences* $\{(x_\lambda, y_\lambda)\}_{\lambda\in\mathbb{N}}$ *such that* $(x_\lambda, y_\lambda) \in \mathcal{D}_f$ *and* $|f(x_\lambda, y_\lambda)| \geq B(\lambda)$ *for all* λ*, we have*

$$\liminf_{\lambda\to\infty} \frac{|f(x_\lambda, y_\lambda)|}{m_\lambda} = \omega$$

where m_λ *is the maximal length of* msg_2 *when* Π_f *runs on inputs* (x_λ, y_λ) *and security parameter* λ*.*

In this work we consider protocols whose download rate is 1, which is clearly optimal. This intuitively means that whenever the output length is a sufficiently big polynomial in the security parameter, the ratio between the output length and the sender's message length tends to 1. When it is important, we will explicitly specify the convergence rate. The convergence rate of interest will mostly be $1 - O(1/\lambda)$.

5.2 Useful Functionalities

Next, we define several useful sender-receiver functionalities, for which we construct two-message protocols with optimal download rate using trapdoor hash.

Oblivious Transfer. In standard (single-bit) OT, the sender has k secret bits $(\mathsf{s}_1, \ldots, \mathsf{s}_k)$, for some $k \in \mathbb{N}$, and the receiver, with input $i \in [k]$, wishes to retrieve the i^{th} secret, s_i. As will be shown in Section 5.6, achieving rate-1 is impossible in the single-bit OT setting, and therefore, we consider scenarios with longer outputs. Most generally, we may consider *batch string OT* where the parties wish to perform n independent instances of OT: in the j^{th} instance the sender has k secret *strings* $(\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k})$ and the receiver has an index $i_j \in [k]$. The receiver's output is defined to consist of s_{j,i_j} for every $j \in [n]$. It is easy to see that optimal-rate construction to this general case can be derived from optimal-rate constructions for the following two special cases that we consider:

- *Batch Oblivious Transfer.* In batch (single-bit) OT, we consider a setting where a batch of independent single-bit OT instances are carried out in parallel. The sender's input consists of n k-tuples of single-bit secrets $\langle(\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k})\rangle_{j\in[n]}$, each corresponding to a single-bit OT instance, and the receiver's input is a tuple of n indices $\langle i_j\rangle_{j\in[n]}$, where $i_j \in [k]$ for every j. We require that the receiver's output contains the i_j^{th} secret of every OT instance j.

$$\mathcal{D}_{\mathsf{bOT}} = \{(\langle i_j\rangle_{j\in[n]}, \langle(\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k})\rangle_{j\in[n]}) \mid n, k \in \mathbb{N} \text{ and } \forall j, \ell\ i_j \in [k], \mathsf{s}_{j,\ell} \in \{0,1\}\}$$
$$\mathsf{bOT}(\langle i_j\rangle_{j\in[n]}, \langle \mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k}\rangle_{j\in[n]}) = \langle \mathsf{s}_{j,i_j}\rangle_{j\in[n]}$$

- *String Oblivious Transfer.* Another OT variant we consider is string OT (or just OT), which is a generalization of the standard setting where the sender has k secret n-bit strings (rather than single bits). In fact, string OT can be seen as a special case of batch OT, where all OT instances share a common receiver's input.

$$\mathcal{D}_{\mathsf{OT}} = \{((1^k, i), (\mathsf{s}_1, \ldots, \mathsf{s}_k)) \mid n, k \in \mathbb{N} \text{ and } i \in [k], \forall j\ \mathsf{s}_j \in \{0,1\}^n\}$$
$$\mathsf{OT}((1^k, i), (\mathsf{s}_1, \ldots, \mathsf{s}_k)) = \mathsf{s}_i$$

We sometimes consider a variant where k is fixed, to which we refer as *(batch) 1-out-of k OT* and denote by $\binom{k}{1}$-OT.

Evaluation of Linear Functions. It is well-known that $\binom{2}{1}$-OT can be also viewed as the evaluation of the linear function $f_{\mathsf{s}_0,\mathsf{s}_1}(i) = \mathsf{s}_0 + (\mathsf{s}_1 - \mathsf{s}_0)i$ over $\mathbb{F}_2$, for sender's input $(\mathsf{s}_0, \mathsf{s}_1) \in \{0,1\}^2$ and receiver's input $i \in \{0,1\}$. In *oblivious linear function evaluation* (OLE for short) [NP99], we consider a generalization of OT to evaluation of any linear function of the form $f(x) = ax + b$ over a finite ring $\mathcal{R}$. In fact, we consider a batch variant of OLE, which is formalized as follows.

– *Batch OLE.* In batch OLE over a finite ring $\mathcal{R}$, the sender's input consists of a batch of n linear functions over $\mathcal{R}$, each defined by a pair of coefficients $a_j, b_j \in \mathcal{R}$, where $f_j(x) = a_j x + b_j$ for every $j \in [n]$. The receiver's input is a batch of n ring elements $\langle x_j \rangle_{j\in[n]} \in \mathcal{R}^n$, and his output is defined as $\langle f_j(x_j) \rangle_{j\in[n]}$.

$$\mathcal{D}_{\mathsf{bOLE}_\mathcal{R}} = \{(\langle x_j \rangle_{j\in[n]}, \langle (a_j, b_j) \rangle_{j\in[n]}) \mid n \in \mathbb{N} \text{ and } \forall j\ x_j, a_j, b_j \in \mathcal{R}\}$$
$$\mathsf{bOLE}_\mathcal{R}(\langle x_j \rangle_{j\in[n]}, \langle (a_j, b_j) \rangle_{j\in[n]}) = \langle a_j x_j + b_j \rangle_{j\in[n]}$$

Generalizing to Linear Maps. We take a further step and generalize OLE to evaluation of linear maps as follows.

– *Oblivious Matrix-Vector Product.* In oblivious matrix-vector product (or OMV for short) over a finite ring $\mathcal{R}$, the sender has as input a matrix $M \in \mathcal{R}^{n\times k}$, which defines a linear map $\mathcal{R}^k \to \mathcal{R}^n$ for some $n, k \in \mathbb{N}$, and the receiver has as input a row vector $v \in \mathcal{R}^k$. The *matrix-vector product* functionality is simply defined as the product $Mv^\top$.

$$\mathcal{D}_{\mathsf{OMV}_\mathcal{R}} = \{(v, M) \mid n, k \in \mathbb{N} \text{ and } v \in \mathcal{R}^k, M \in \mathcal{R}^{n\times k}\}$$
$$\mathsf{OMV}_\mathcal{R}(v, M) = Mv^\top$$

5.3 Rate-1 Batch Oblivious Transfer from Trapdoor Hash

We now show our first TDH-based construction in the sender-receiver setting: a batch OT protocol. In fact, when the underlying TDH is $(1-\epsilon)$-correct for a non-negligible ϵ, we obtain a weakly correct batch OT protocol, where a failure in each of the instances in the batch occurs independently and with probability ϵ, in which case the protocol outputs $\perp$.

Overview. The idea is very straight forward. For a batch 1-out-of-k OT with batch size n, we use TDH for inputs of length $n \cdot k$ bits. Speaking ahead, the input for the trapdoor hash function consists of all the secrets of the sender, i.e. $\langle \mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,n} \rangle_{j\in[n]}$. The receiver, with input $\langle i_j \rangle_{j\in[n]}$, samples a hash key hk and a pair of encoding key and a trapdoor $(\mathsf{ek}_j, \mathsf{td}_j)$, for every $j \in [n]$, corresponding to the index predicate for s_{j,i_j}, i.e. $f_{[(j-1)k+i_j]}(x) = x[(j-1)k + i_j]$. He then sends the hash key and each of the encoding keys to the sender. The sender, in return, computes the hash of his input $\langle (\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k}) \rangle_{j\in[n]}$, and encodes, using ek_j, the secret s_{j,i_j}. He then sends the hash value and the single-bit encodings to the receiver who uses the decoding algorithm of the TDH to recover s_{j,i_j} for every $j \in [n]$. Below, we give a formal description of the construction.

Construction 5.1 (Download-rate-1 weakly correct batch OT from rate-1 TDH)**.** *Let $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ be a rate-1 (weakly) correct TDH for index predicates. Our batch OT protocol $\mathsf{bOT} = (\mathsf{bOT}_1, \mathsf{bOT}_2, \mathsf{bOT}_3)$ with download rate-1, and error probability ϵ per instance, consists of the following algorithms.*

- $\mathsf{bOT}_1(1^\lambda, 1^k, \langle i_j \rangle_{j\in[n]})$*:*
  1. *Sample* $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^{n\cdot k})$.
  2. *For every* $j = 1, \ldots, n$*, sample* $(\mathsf{ek}_j, \mathsf{td}_j) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_{[(j-1)\cdot k+i_j]})$.
  3. *Output*

$$\mathsf{st} := (\langle i_1, \ldots, i_n \rangle, \mathsf{hk}, \mathsf{td}_1, \ldots, \mathsf{td}_n) \qquad \mathsf{msg}_1 := (\mathsf{hk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_n) \tag{5.1}$$

- $\mathsf{bOT}_2(\mathsf{msg}_1, \langle(\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k})\rangle_{j\in[n]})$*: parse* msg_1 *as in Equation 5.1 and proceed as follows.*
  1. *Sample* $\rho \in \{0,1\}^*$ *and compute* $h \leftarrow \mathsf{H}(\mathsf{hk}, \langle(\mathsf{s}_{1,1}, \ldots, \mathsf{s}_{1,k}), \ldots, (\mathsf{s}_{n,1}, \ldots, \mathsf{s}_{n,k})\rangle; \rho)$.
  2. *For every* $j = 1, \ldots, n$*, set* $\mathsf{e}_j \leftarrow \mathsf{E}(\mathsf{ek}_j, \langle(\mathsf{s}_{1,1}, \ldots, \mathsf{s}_{1,k}), \ldots, (\mathsf{s}_{n,1}, \ldots, \mathsf{s}_{n,k})\rangle; \rho)$.
  3. *Output*

$$\mathsf{msg}_2 := (\mathsf{h}, \mathsf{e}_1, \ldots, \mathsf{e}_n) \tag{5.2}$$

- $\mathsf{bOT}_3(\mathsf{st}, \mathsf{msg}_2)$*: parse* st *and* msg_2 *as in Equations 5.1 and 5.2 (resp.), and proceed as follows.*
  1. *For every* $j = 1, \ldots, n$*,*
     1.1. *Compute* $(\mathsf{e}_{j,0}, \mathsf{e}_{j,1}) \leftarrow \mathsf{D}(\mathsf{td}_j, \mathsf{h})$.
     1.2. *If* $\mathsf{e}_{j,0} = \mathsf{e}_{j,1}$ *set* $\tilde{\mathsf{s}}_j := \perp$.
     1.3. *Otherwise, if* $\mathsf{e}_j = \mathsf{e}_{j,0}$*, set* $\tilde{\mathsf{s}}_j := 0$.
     1.4. *Otherwise, if* $\mathsf{e}_j = \mathsf{e}_{j,1}$*, set* $\tilde{\mathsf{s}}_j := 1$.
  2. *Output* $\langle\tilde{\mathsf{s}}_1, \ldots, \tilde{\mathsf{s}}_n\rangle$

Analysis. (Weak) correctness follows immediately from the correctness of the underlying TDH scheme. Notice that since the TDH has one-sided error probability of ϵ, then any output bit of the OT equals $\perp$ with probability ϵ. Since the bits are generated using independent TDH encodings, the error is independent over the different output bits.

In the two theorems below, we show that the privacy properties of the scheme follow from the respective privacy properties of the underlying TDH.

Theorem 5.1 (Receiver Privacy of TDH-based batch OT)**.** *Let* $\mathcal{H}$ *be a function-private* TDH*. Then, the batch OT protocol from Construction 5.1 is receiver-private.*

Proof. Let $(\langle i_1, \ldots, i_n\rangle, \langle i'_1, \ldots, i'_n\rangle)$ be the challenge indices chosen by Env_1, then, via a standard hybrid argument against the function privacy of TDH, one can see that

$$(\mathsf{hk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_n) \stackrel{c}{\equiv} (\mathsf{hk}, \mathsf{ek}'_1, \ldots, \mathsf{ek}'_n)$$

where $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{S}(1^\lambda, 1^{n\cdot k})$, and $(\mathsf{ek}_j, \mathsf{td}_j) \stackrel{\$}{\leftarrow} \mathsf{G}(\mathsf{hk}, f_{[(i_j-1)\cdot n+j]})$ and $(\mathsf{ek}'_j, \mathsf{td}'_j) \stackrel{\$}{\leftarrow} \mathsf{G}(\mathsf{hk}, f_{[(i^*_j-1)\cdot n+j]})$ for all $j = 1, \ldots, n$. Therefore, a polynomial-time Env_2 cannot distinguish between the two distributions with non-negligible advantage. □

For the following theorem we need a TDH with standard correctness, that is the theorem is not applicable for schemes that only satisfy weak correctness.

Theorem 5.2 (Sender Privacy of TDH-based batch OT)**.** *Let* $\mathcal{H}$ *be a statistically (resp., computationally) input-private* TDH *with (standard) correctness. Then, the batch OT scheme from Construction 5.1 provides statistical (resp. computational) sender privacy.*

Proof. We construct a simulator Sim which, on inputs 1^λ, $\langle i_1, \ldots, i_n\rangle$ and $\langle \mathsf{s}_{1,i_1}, \ldots, \mathsf{s}_{n,i_n}\rangle$, samples a hash key $\mathsf{hk}' \stackrel{\$}{\leftarrow} \mathsf{S}(1^\lambda, 1^{n\cdot k})$ and a pair $(\mathsf{ek}'_j, \mathsf{td}'_j) \stackrel{\$}{\leftarrow} \mathsf{G}(\mathsf{hk}, f_{[(i_j-1)\cdot n+j]})$ for every $j \in [n]$, then computes

$$\mathsf{h}' \leftarrow \mathsf{H}(\mathsf{hk}', \mathsf{s}'; \rho') \qquad (\mathsf{e}'_{j,0}, \mathsf{e}'_{j,1}) \leftarrow \mathsf{D}(\mathsf{td}'_j, \mathsf{h}') \quad \text{for all } j = 1, \ldots, n$$

where $\mathsf{s}' = \langle (0^{i_j-1}, \mathsf{s}_{1,i_j}, 0^{n-i_j}) \rangle_{j\in[n]}$ and $\rho' \xleftarrow{\$} \{0,1\}^*$. The output of Sim is defined as

$$\mathsf{st}' := (\langle i_j \rangle_{j\in[n]}, \mathsf{hk}', \mathsf{td}'_1, \ldots, \mathsf{td}'_n) \qquad \mathsf{msg}'_2 := (\mathsf{h}', \mathsf{e}'_{1,\mathsf{s}_{1,i_1}}, \ldots, \mathsf{e}'_{n,\mathsf{s}_{n,i_n}})$$

Now, let $\mathsf{Ideal} = (\mathsf{st}', \mathsf{msg}'_2)$ and let $\mathsf{Real} = (\mathsf{st}, \mathsf{msg}_2)$ be the distribution obtained from a real execution of bOT with inputs $\langle i_j \rangle_{j\in[n]}$ and $\mathsf{s} = \langle (\mathsf{s}_{j,1}, \ldots, \mathsf{s}_{j,k}) \rangle_{j\in[n]}$, where

$$\mathsf{st} = (\langle i_j \rangle_{j\in[n]}, \mathsf{hk}, \mathsf{td}_1, \ldots, \mathsf{td}_n) \qquad \mathsf{msg}_2 = (\mathsf{h}, \mathsf{e}_{1,\mathsf{s}_{1,i_1}}, \ldots, \mathsf{e}_{n,\mathsf{s}_{n,i_n}})$$

It is obvious that $\mathsf{st} \equiv \mathsf{st}'$, and therefore, $\mathsf{Real} \equiv \mathsf{Real}'$, where $\mathsf{Real}' = (\mathsf{st}', \mathsf{msg}''_2)$ is a hybrid distribution with $\mathsf{msg}''_2 = (\mathsf{h}'', \mathsf{e}''_1, \ldots, \mathsf{e}''_n)$ where

$$\mathsf{h}'' \leftarrow \mathsf{H}(\mathsf{hk}', \mathsf{s}; \rho) \qquad \mathsf{e}''_j \leftarrow \mathsf{E}(\mathsf{ek}'_j, \mathsf{s}; \rho)$$

Next, since the underlying TDH scheme is statistically input-private, we have that $(\mathsf{hk}', \mathsf{h}') \overset{s}{\equiv} (\mathsf{hk}', \mathsf{h}'')$ (computational case is similar), and therefore we can replace h' in Ideal with h''. Further, by the correctness of the TDH, $\mathsf{D}(\mathsf{td}'_j, \mathsf{h}'') = \mathsf{e}''_j$, for all $j = 1, \ldots, n$, except with negligible probability over the random choice of ρ. It follows that the simulated view is statistically (resp., computationally) indistinguishable from Real', and therefore, from Real. □

5.4 From Weakly Correct Batch OT to String OT

In the previous section, we show how to construct batch OT from trapdoor hash. If we instantiate the construction using the QR- and LWE-based TDH schemes from Sections 4, which have a negligible error probability, then we get batch OT with full correctness as defined in Definition 5.1. Thus, since string OT is a special case of batch OT as mentioned earlier, then we immediately get string OT schemes with optimal download rate based on QR and LWE. However, when the underlying TDH has non-negligible error, as it is with the DDH-based scheme, then the obtained OT has non-negligible error as well.

In this section, we show how to overcome the non-negligible error in the string OT from the DDH-based TDH through a generic error correction technique that preserves the download rate. We also show how to bootstrap any OT protocol with optimal download rate to a protocol with optimal *overall* rate, and thus achieving rate-1 even when we consider the first message as well.

5.4.1 Correcting the Errors

In the following, we show how to use rate 1 error-correcting codes to construct a string OT protocol with download-rate 1 and negligible error from any batch OT with non-negligible but "small" error probability. We then show how to instantiate the construction with the TDH schemes from Section 4 and appropriate codes.

Overview. Again, when the underlying batch OT has negligible error, then, to invoke string OT with sender's inputs $\mathsf{s}_1, \ldots, \mathsf{s}_k \in \{0,1\}^n$ and receiver's selection $i \in [k]$, we simply perform batch (1-out-of-k) OT for a batch of size n, where the j^{th} input tuple consists of the j^{th} bit of every secret, i.e. $(\mathsf{s}_1[j], \ldots, \mathsf{s}_k[j])$ and $i_j = i$ for every j. When the TDH induces a non-negligible error probability, the receiver might not be able to recover s_i entirely. More specifically, if the TDH is $(1-\epsilon)$-correct, then we expect to have ϵ fraction of errors (or erasures) in the decoded string (for

one-sided error we get erasures). In such a case, we use error-correcting codes to ensure successful recovery, and apply the described protocol over encodings of the secrets using sufficiently good code. To maintain the optimal rate of the OT, we require that the code has rate $1 - O(1/\lambda)$.

We now describe the construction in details.

Construction 5.2 (String OT from weakly correct batch OT)**.** *Let* $\mathsf{bOT} = (\mathsf{bOT}_1, \mathsf{bOT}_2, \mathsf{bOT}_3)$ *be a batch OT scheme with download-rate 1, where an independent error (*$\perp$ *output) occurs at every OT instance with probability at most* $\epsilon(\lambda)$ *for security parameter* λ*. Let* $\{(\mathsf{Encode}_{\lambda,n}, \mathsf{Decode}_{\lambda,n})\}_{\lambda,n\in\mathbb{N}}$ *be a family of error-correcting codes which can efficiently correct* $2\epsilon N$ *errors.*

Our download-rate-1 OT protocol $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *consists of the following algorithms.*

- $\mathsf{OT}_1(1^\lambda, 1^n, 1^k, i)$*: output* $(\mathsf{st}, \mathsf{msg}_1) \leftarrow \mathsf{bOT}_1(1^\lambda, 1^k, \langle i^{N_n}\rangle)$.
- $\mathsf{OT}_2(\mathsf{msg}_1, (\mathsf{s}_1, \ldots, \mathsf{s}_k))$:
  1. *For every* $j = 1, \ldots, k$*, compute*
  $$(z_{1,j}, \ldots, z_{N,j}) \leftarrow \mathsf{Encode}_{\lambda,n}(\mathsf{s}_j)$$
  2. *Output* $\mathsf{msg}_2 \leftarrow \mathsf{bOT}_2(\mathsf{msg}_1, \langle(z_{1,1}, \ldots, z_{1,k}), \ldots, (z_{N,1}, \ldots, z_{N,k})\rangle)$.
- $\mathsf{OT}_3(\mathsf{st}, \mathsf{msg}_2)$: *compute* $z \leftarrow \mathsf{bOT}_2(\mathsf{st}, \mathsf{msg}_2)$ *and output*
$$\tilde{\mathsf{s}} \leftarrow \mathsf{Decode}_{\lambda,n}(z) \tag{5.3}$$

Analysis. We start with correctness, and show that as long as $\delta(\lambda)$ is sufficiently large, then the erasure code is able to recover the message correctly with an overwhelming probability.

Theorem 5.3 (Correctness of string OT)**.** *Given that* $\epsilon N = \Omega(\lambda)$*, the OT scheme from Construction 5.2 is correct.*

Proof. Let $y := \mathsf{Encode}_{\lambda,n}(\mathsf{s}_i)$. We have that for every index j it holds that $z_j = y_j$, except with probability ϵ and independently of j. Thus, in expectation the number of errors among the $\{z_j\}$ is ϵN. Using the Chernoff bound, we get that the probability of having more than $2\epsilon N$ errors is at most $e^{-\epsilon N/3}$ which is negligible in λ. Conditioned that we have less than $2\epsilon N$ errors, decoding will succeed and we get $\tilde{\mathsf{s}} = \mathsf{s}_i$. □

Instantiations. Using a TDH with negligible error would result in a batch OT with negligible error, and therefore, an OT scheme that satisfies correctness (see Definition 5.1). This means that with the superpoly LWE-, QR- and DCR-based schemes as the underlying TDH, we can set the error correcting code to simply be the identity function. In the case where we use the DDH-based TDH, we can set the parameters such that the correctness error is at most $1/\lambda$. Consequently, we need a code of rate $1 - O(1/\lambda)$ which can correct a $1/\lambda$ fraction of errors. Such a code is provided by Theorem 3.3. The same holds for our construction from LWE with polynomial modulus, i.e. by choosing $n = O(Bn\lambda)$, we get a correctness error of at most $1/\lambda$. Consequently, we can also use the codes of Theorem 3.3 in this case.

Recall that the second message in the OT protocol is of the form $\mathsf{msg}_2 := (\mathsf{h}, \mathsf{e}_1, \ldots, \mathsf{e}_N)$. Consequently, we also have to account for the size of the hash value h. But since the representation size of h is a fixed polynomial in λ, we can amortize it by choosing the parameter n sufficiently large, e.g. by choosing n such that $\mathsf{size}(\mathsf{h}) \leq n/\lambda$. Once this is given, we can bound the overall rate of the construction by $1 - O(1/\lambda)$.

Remark 5.1. *There is a gap between the syntax of Construction 5.2 and the definition of the OT functionality from Section 5.2. In particular, the receiver is given 1^n as input, in contrary to the definition where receiver's input is defined to be just $i \in [k]$ (in addition to the security parameter). This is fixed in the final OT construction, which is obtained from Theorem 5.4 in the next section.*

5.4.2 Bootstrapping to Optimal Overall Rate

Recall that our two-message string OT protocol achieves a download rate of $1 - O(1/\lambda)$, but the overall rate (when considering also the first message) is inverse in the security parameter. Fortunately, any 2-message string OT with a download rate of $1-O(1/\lambda)$ can be generically bootstrapped to a 2-message OT with overall rate of $1 - O(1/\lambda)$. The transformation is very simple and it stems for the observation that for any two-message OT, the first message is always reusable, thus its rate can be amortized by running independent OTs on small message blocks. Let the upload rate be defined just like the download rate (Definition 5.2) except that m_λ is a bound on the length of the output of Π_1. We give the following theorem of two-message string OT.

Theorem 5.4 (OT with overall rate-$1-O(1/\lambda)$)**.** *Let $\mathsf{OT} = (\mathsf{OT_1}, \mathsf{OT_2}, \mathsf{OT_3})$ be a 1-out-of-k string OT protocol with upload rate of $1/q_0(\lambda, n)$, for some polynomial $q_0(\lambda, n)$ in the security parameter λ and in the message length n, and with download rate of $1 - O(1/\lambda)$. Then there exists a 1-out-of-k two-message string OT protocol with overall rate 1.*

Proof. Fix $\tilde{n} = q_0(\lambda, n) \cdot n$, for some $n \in \mathbb{N}$. The OT protocol is modified as follows: The $\mathsf{OT_1}$ algorithm is executed on input a length parameter 1^n and the output $\mathsf{msg_1}$ is defined to be whatever the algorithm outputs. Let s be a string of length $\tilde{n}$ and denote by $(\mathsf{s}^{(1)}, \ldots, \mathsf{s}^{(q_0(\lambda,n))})$ its n-size blocks. The algorithm $\mathsf{OT_2}$ is executed on input $\mathsf{msg_1}$ independently on each set of inputs $(\mathsf{s}_1^{(i)}, \ldots, \mathsf{s}_k^{(i)})$ together with the message $\mathsf{msg_1}$. Denote the set of outputs by $(\mathsf{msg}_2^{(1)}, \ldots, \mathsf{msg}_2^{(q_0(\lambda,n))})$ and let this set be the output of $\mathsf{OT_2}$. Then $\mathsf{OT_3}$ is executed independently on each element and the output is defined as the concatenation of all resulting strings.

It is not hard to see that correctness and security are preserved under parallel composition. Since the download rate of the underlying OT protocol is $1 - O(1/\lambda)$ (for a growing n), then so is the second-message rate of the modified protocol (for a growing $\tilde{n}$). Then the upload rate of the resulting OT (for a growing $\tilde{n}$) is

$$\frac{\tilde{n}}{q_0(\lambda, n)} = \frac{q_0(\lambda, n) \cdot n}{q_0(\lambda, n)} = n$$

which approaches infinity as n grows. □

5.4.3 Malicious Security

Any semi-honest two-message OT is also secure (at least for an indistinguishability notion of security) against a fully corrupted sender. While the same does not necessarily hold for a corrupted receiver, Badrinarayanan et al. [BGI+17a] showed how to compile any semi-honest two-message OT with rate $> \frac{1}{2}$ into statistically sender private (SSP) OT with constant rate (although lower than 1).

Theorem 5.5 ([BGI+17a])**.** *If $\mathsf{OT} = (\mathsf{OT_1}, \mathsf{OT_2}, \mathsf{OT_3})$ is a high rate ($> 1/2$) two-message semi-honest OT protocol, then there exists a constant rate two-message SSP OT protocol.*

Prior to our work, the only SSP OT from LWE was presented in a recent work of Brakerski and Döttling [BD18] (with inverse logarithmic rate in the security parameter). Our work directly improves upon their result as it yields the first constant-rate SSP OT from LWE.

It is worth mentioning that one can generically upgrade passive security to active security, without affecting the asymptotic rate, by using succinct zero-knowledge arguments [Kil92]. This comes at the cost of additional interaction (or reliance of non-falsifiable assumptions [GW11]) and settling for computational security. Interactive arguments can be based on collision resistance hash function, which can in turn be instantiated from the DDH, QR, or LWE assumption [NN01]. We stress that, even with interaction and computational security, rate-1 OT from DDH, QR, or LWE was not known prior to our work. In fact, even under DCR, achieving rate-1 required the group size to grow with the rate.

5.5 OLE, Vector-Matrix Product, and Other Generalizations

In the previous sections, we were able to utilize rate-1 trapdoor hash functions to realize the first rate-1 OT protocols from various assumptions. However, we have not yet exploited all the possibilities that trapdoor hash has to offer in the sender-receiver setting. In this section, we describe a few directions for extending the use of trapdoor hash in order to realize more general functionalities, such as oblivious linear function evaluation (OLE) and oblivious matrix-vector product (OMV). The latter can be viewed as a relaxed form of additively homomorphic encryption, allowing the client to encrypt a vector v over a ring, while supporting multiple evaluations of inner products $uv^\top$ over the encrypted v, with download rate 1. Concretely, we can support this form of encryption over rings of aribitrary size, provided that they are either of the form $\mathbb{Z}_N$ for a *smooth* integer N or are fields of small characteristic (namely, $\mathbb{F}_q$ where q is a power of a small prime). In most of these cases, even under the DCR assumption it was not known how to achieve rate 1, let alone under other assumptions. See [JVC18b] and references therein for constructions and applications of additively homomorphic encryption.

Oblivious Evaluation of Small Classes of Functions. As discussed in Remark 4.1, trapdoor hash for index predicates can be transformed into trapdoor hash for any class of predicates $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell\in\mathbb{N}}$. The transformation is efficient when $|\mathcal{F}_\ell|$ is bounded by some polynomial in ℓ, in which case we say that the class of predicates is "small". By deploying trapdoor hash for such a class $\mathcal{F}$ to Construction 5.1, we obtain a two-message protocol that realizes batch oblivious evaluation of $\mathcal{F}$ where, roughly speaking, receiver has as input a batch of private predicates $x = \langle f_1, \ldots, f_n\rangle \in \mathcal{F}_\ell^n$, sender has a private input $y \in \{0,1\}^\ell$, and the goal is to give $\langle f_1(y), \ldots, f_n(y)\rangle \in \{0,1\}^n$ to the receiver. One can see that the batch OT protocol from Construction 5.1 is a special case for $\mathcal{F} = \mathcal{I}$.

Even more generally, we can realize the above functionality for general "small" classes of functions $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell\in\mathbb{N}}$, where $\mathcal{F}_\ell$ is a polynomially large set of functions from $\{0,1\}^\ell$ to $\{0,1\}^{p(\ell)}$ for some polynomial $p(\ell)$, by applying the above $p(\ell)$ times in parallel, once for every output bit.

Since the transformation from Remark 4.1 preserves the rate of the trapdoor hash, we get protocols with optimal asymptotic rate.

Special Case: OLE over Small Fields. It appears that one can also describe the batch OLE functionality over a "small" field $\mathbb{F}_p$ (formalized as bOLE in Section 5.2), as a special case of oblivious evaluation of a "small" function class. More specifically, recall that a sender's input in bOLE is defined

as $y = \langle (a_j, b_j) \rangle_{j \in [n]} \in \mathbb{F}_p^{n \times 2}$. Consequently, for any bOLE receiver input $x = \langle x_1, \ldots, x_n \rangle$, we define a corresponding $\hat{x} = \langle f_1, \ldots, f_n \rangle$, where for any $i \in [n]$, the function $f_i : \mathbb{F}_p^{n \times 2} \to \mathbb{F}_p$ is defined as $f_i(\langle (a_j, b_j) \rangle_{j \in [n]}) = a_i x_i + b_i$ for all $i \in [n]$. Now, if we take the function class $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell \in \mathbb{N}}$ where $\mathcal{F}_{2n\lceil \log p \rceil} = \{f_{i,x}(\langle (a_1, b_1), \ldots, (a_n, b_n) \rangle) = a_i x + b_i \mid i \in [n], x \in \mathbb{F}_p\}$, then it clearly holds that realizing bOLE for inputs x and y is equivalent to oblivious batch evaluation of $\mathcal{F}$ with inputs $\hat{x}$ and y. Further, when we fix p as a constant, then $\mathcal{F}$ is a small function class, and therefore, following the outline from above, we obtain an efficient download-rate-1 bOLE protocol based on DDH, QR, or LWE (more specifically, with communication growing polynomially in n and p).

Oblivious Matrix-Vector Product. Lastly, we explain how to use trapdoor hash for the more general class of linear predicates in order to realize the oblivious matrix-vector product functionality over $\mathbb{F}_2$ (formalized as OMV in Section 5.2). Recall that in OMV over $\mathbb{F}_2$, the receiver has as input a vector $v \in \mathbb{F}_2^k$ and the sender has a matrix $M \in \mathbb{F}_2^{n \times k}$ and the goal is for the receiver to learn the product $Mv^\top$. It would be convenient to look at OMV as an oblivious batch evaluation of inner products between vectors: if we let $u_1, \ldots u_n \in \mathbb{F}_2^k$ be the row vectors of M, then our goal becomes to provide the inner products $\langle u_i v^\top \rangle_{i \in [n]}$ to the receiver while keeping v and the u_i's private.

Again, we refer to Construction 5.1 as a starting point, and replace the trapdoor hash for index predicates with trapdoor hash for linear predicates (to which we have constructions from QR and LWE, see Section 4). The receiver generates a hashing key hk for input length $n \cdot k$ and n pairs of an encoding key and a trapdoor $(\mathsf{ek}_1, \mathsf{td}_1), \ldots, (\mathsf{ek}_n, \mathsf{td}_n)$, where $(\mathsf{ek}_i, \mathsf{td}_i)$ correspond to the linear predicate $f_{(0^{k(i-1)} || v || 0^{n-i})}(u_1 || \ldots || u_n) = u_i v^\top$. He then sends hk and ek_i, for all $i \in [n]$, to the sender. Using the keys, the sender computes the hash of his matrix M, then encodes $u_i v^\top$ for every $i \in [n]$. The receiver uses the hash and the trapdoor td_i to decode $u_i v^\top$, for all $i \in [n]$, and recover the product $Mv^\top$. From the optimal rate of the trapdoor hash, we get a download-rate-1 protocol for OMV.

Lastly, we observe that the LWE-based TDH constructions from Section 4 can be easily tweaked to support "trapdoor-evaluation" of linear functions over fields $\mathbb{F}_p$ where p is a small constant prime. Roughly speaking, we can choose the modulus q to be a multiple of p and define the rounding function modulo p instead of 2. Furthermore, the encoding keys are computed by replacing the factor $q/2$ with q/p in in the corresponding terms of the matrix $\mathbf{B}$. This defines a subgroup of order p and allows us to compute linear functions over $\mathbb{F}_p$.

Extending Beyond Small Fields. So far we were able to realize algebraic functionalities over small fields only. Fortunately, standard algebraic tricks allow us to extend these results to achieve the same functionalities over two more general and larger types of algebraic structures: 1. rings modulo a smooth integer (an integer that can be written as the multiplication of small primes), and 2. fields with small characteristics (extensions of $\mathbb{F}_p$ where p is a small prime).

More specifically, to get OLE or OMV over smooth moduli, we use the Chinese Remainder Theorem, which defines a homomorphism between such a ring $\mathbb{Z}_N$ for a smooth $N = p_1 \cdot \cdots \cdot p_k$ (where $p_1, \ldots, p_k$ are primes) and $\mathbb{F}_{p_1} \times \cdots \times \mathbb{F}_{p_k}$. Now, to realize the target functionality over $\mathbb{Z}_N$, we simply perform k invocations of the appropriate protocol, each over some small field $\mathbb{F}_{p_i}$, while translating each of the inputs in $\mathbb{Z}_N$ to an input in the appropriate $\mathbb{F}_{p_i}$. As for OLE (or OMV) over extensions fields $\mathbb{F}_{p^k}$, we still rely on executions over $\mathbb{F}_p$, except here, we use the fact that multiplication over $\mathbb{F}_{p^k}$ can be expressed as a linear mapping over the base field, $\mathbb{F}_p$.

5.6 On the Tightness of Our Protocols

Our protocols achieve asymptotic download rate of 1 (as in Definition 5.2), which is clearly optimal. However, the (additive) difference between the sender's message length and the output length grows with the security parameter λ. We argue that this gap is necessary. Moreover, we show that the same holds in the preprocessing model when the sender's input is long.

Let us first consider the case of 1-out-of-2 OT in the plain model where the sender's input is short, e.g., consists of two single-bit secrets. It is not hard to see that in the case where the sender's message msg_2 is also short, specifically - shorter than the security parameter λ, then the OT scheme cannot be secure. In particular, 2-message OT with $|\mathsf{msg}_2| = t$ implies PKE with t-bit ciphertexts [GMR01], which can be easily broken via a $O(2^t)$ brute-force attack.

It follows from the above that when the OT output length is constant, the difference between the output length and the sender's message length should indeed grow with λ. This simple lower bound does not rule out the possibility that when the output length is bigger than λ, the sender's message length is the same. Moreover, it does not allow to the setting of secure computation with a correlated randomness setup (or equivalently, input-independent preprocessing). In this setting, there is a simple protocol that for output length n has sender message of length $2n$ [Bea95b]. The following theorem shows that even in this relaxed setting, there is an inherent gap between the output length and the sender message length.

Theorem 5.6. *Let $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ be a 1-out-of-2 OT protocol with a correlated randomness setup. Suppose that the sender's input strings are of length n and the sender's message is of length $t < 2n$. Then there is a distinguisher running in time $O(2^{(t-n)/2})$ that distinguishes between messages corresponding to the two receiver inputs with constant advantage.*

Proof. We construct a distinguisher $\mathcal{D}$ that takes as input a receiver's message msg_1, which either corresponds to receiver input $i = 1$, in which case $\mathsf{msg}_1 \leftarrow \mathsf{OT}_1(1^\lambda, 1^n, 1)$, or to a receiver input $i = 2$, i.e. $\mathsf{msg}_1 \leftarrow \mathsf{OT}_1(1^\lambda, 1^n, 2)$, and distinguishes between the two cases with advantage $\geq \frac{1}{3} - \mathsf{negl}(\lambda)$.

$\mathcal{D}(\mathsf{msg}_1)$:

1. Sample a uniform $\mathsf{s}_1 \xleftarrow{\$} \{0,1\}^n$.
2. Set $X \leftarrow \emptyset$ and repeat the following T times (T will be specified below).
   2.1. Sample a uniform $\mathsf{s}_2 \xleftarrow{\$} \{0,1\}^n$ that was not sampled before and set
   $$m = \mathsf{OT}_2(\mathsf{msg}_1, (\mathsf{s}_1, \mathsf{s}_2))$$
   2.2. If $m \in M$ output 1, otherwise update $M \leftarrow M \cup \{m\}$.
3. Output 0.

Let $t' := t - n < n$, and denote by $\mathsf{s}_2^{(j)}$ and $m^{(j)}$ the values of s_2 and m of the j^{th} iteration in the attack. From the correctness of the OT, if msg_1 was generated for $i = 2$, then, with probability at least $1 - \mathsf{negl}(\lambda)$, it holds that $\mathsf{OT}_3(\mathsf{st}, m^{(j)}) = \mathsf{s}_2^{(j)}$ for every $j \in [T]$. Since all the $\mathsf{s}_2^{(j)}$ are distinct, this implies that M contains no collisions, and therefore, $\Pr[\mathcal{D}(\mathsf{OT}_1(1^\lambda, 1^n, 2)) = 1] \geq 1 - \mathsf{negl}(\lambda)$.

On the other hand, if msg_1 corresponds to $i = 1$ then, $\mathsf{OT}_3(\mathsf{st}, m^{(j)}) = \mathsf{s}_1$ for every j. Denote $\mathsf{msg}_2(\mathsf{s}_1) := \{\mathsf{OT}_3(\mathsf{st}, m) = \mathsf{s}_1 \mid m \in \{0,1\}^t\}$. Now observe that $\mathsf{OT}_3(\mathsf{st}, \cdot)$ maps $\{0,1\}^t$ to $\{0,1\}^n$,

and therefore, $|\mathsf{msg}_2(\mathsf{s}_1)|$ has expected value of $2^{t'}$ for a uniform s_1. Using Markov's inequality, we get that $|\mathsf{msg}_2(\mathsf{s}_1)| \leq 2^{t'+1}$ with non-negligible probability at least $\frac{1}{2}$.

Assume for now that $|\mathsf{msg}_2(\mathsf{s}_1)| \leq 2^{t'+1}$ indeed. Notice that the $m^{(j)}$'s are sampled from $\mathsf{msg}_2(\mathsf{s}_1)$ according to some distribution. $\mathcal{D}$ outputs 1 when it detects a collision, i.e. $\mathsf{s}^{(j)} = \mathsf{s}^{(j')}$, and therefore, what we essentially have is a variant of the Birthday Paradox where the items sampled are not replaced (as we never pick the same $\mathsf{s}^{(j)}$ twice). It is well-known that the case where the values are sampled uniformly is the case where a collision is least probable (one way to show that is using Schur convexity), and therefore we assume that for every $m \in \mathsf{msg}_2(\mathsf{s}_1)$ there are exactly $Z := 2^n/|\mathsf{msg}_2(\mathsf{s}_1)| \geq 2^{n-t'-1}$ values of s_2 such that $\mathsf{OT}_2(\mathsf{msg}_1, (\mathsf{s}_1, \mathsf{s}_2)) = m$. The probability for finding a collision, and thus outputting 1, can be bound as follows.

$$1 - \left(1 - \frac{Z-1}{2^n-1}\right)\left(1 - \frac{2(Z-1)}{2^n-1}\right)\cdots\left(1 - \frac{T(Z-1)}{2^n-1}\right) > 1 - \prod_{j=1}^{T}\left(1 - \frac{j(Z-1)}{2^n}\right)$$

$$> 1 - \prod_{j=1}^{T} e^{-\frac{j(Z-1)}{2^n}} = 1 - e^{-\frac{Z-1}{2^n}T(T-1)/2}$$

For $T = 2\sqrt{\ln 3 \cdot 2^{n+1}/(Z-1)}$, the above probability is larger than $\frac{2}{3}$, and thus we have that $\Pr[\mathcal{D}(\mathsf{OT}_1(1^\lambda, 1^n, 1)) = 0] \geq \frac{1}{3}$. This concludes the proof. □

A simple corollary of Theorem 5.6 is that 1/2 is a barrier for the download rate of *information-theoretic* 1-out-of-2 OT in the correlated randomness model.

6 Applications of Rate-1 OT

In the following we discuss several new implications that stem from our main theorem.

6.1 Private Information Retrieval

A single-server private information retrieval (PIR) protocol [KO97] allows a client to query a large database of entries, which is stored in a remote server(s), without leaking the index of the entries of interest. The general efficiency of PIR protocols is measured in terms of communication complexity, which is required to be sub-linear in the size of the database. Another important efficiency measure of PIR is *download rate*, which is defined as the asymptotic ratio between the length of the server's answer to a query and the length of a database entry. In fact, definitions of PIR and its download rate are identical to the definitions of string OT and its related notion of download rate given in Section 5 (see Definitions 5.1 and 5.2), except that for PIR we do not require sender privacy.

Ishai and Paskin [IP07] showed that a rate-1 2-message OT implies a round-optimal single-server PIR with poly-logarithmic communication in the size of the database. Thus, PIR protocols from DDH, QR, LWE and DCR follow as immediate corollaries of our results. In fact, all of the resulting protocols have an asymptotically optimal download rate of 1. In the following we recall the construction from [IP07] and prove a somewhat stronger statement, i.e., that any 1-out-of-λ 2-message OT[14] with *constant* rate already suffices to construct PIR with poly-logarithmic

[14] In fact, $k = \lambda$ can be replaced with any $\mathsf{poly}(\lambda)$.

communication overhead. When the underlying OT has download rate 1, the obtained PIR has download rate 1 as well.

Construction 6.1 (PIR from constant-rate OT)**.** *Let* $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *be an OT protocol with download rate* $\frac{1}{c}$ *for some constant* c*. We assume, without loss of generality, that our database* DB *consists of* N *entries, each of size* B*, where* $N = \lambda^d$ *for some* $d \in \mathbb{N}$*. Our download-rate-1 PIR protocol* $\mathsf{PIR} = (\mathsf{PIR}_1, \mathsf{PIR}_2, \mathsf{PIR}_3)$ *consists of the following algorithms.*

- $\mathsf{PIR}_1(1^\lambda, (i \in [N], 1^N))$: *parse* i *as* $(i_1, \ldots, i_d)$*, where* $i_j \in [\lambda]$ *for all* j*, and proceed as follows.*
  1. *For all* $\ell \in [d]$ *compute* $\left(\mathsf{st}^{(\ell)}, \mathsf{msg}_1^{(\ell)}\right) \leftarrow \mathsf{OT}_1\left(1^\lambda, 1^\lambda, i_\ell\right)$
  2. *Output a query* msg_1 *and an internal state* st *defined as follows*
$$\mathsf{msg}_1 = \left(\mathsf{msg}_1^{(1)}, \ldots, \mathsf{msg}_1^{(d)}\right) \qquad \mathsf{st} = \left(\mathsf{st}_1^{(1)}, \ldots, \mathsf{st}_1^{(d)}\right) \tag{6.1}$$
- $\mathsf{PIR}_2(\mathsf{msg}_1, (\mathsf{DB}, 1^N, 1^B))$*: parse* msg_1 *as in Equation 6.1 and proceed as follows.*
  1. *Set* $Y = \mathsf{DB}$ *and* $B^{(1)} = B$*.*
  2. *For every* $\ell = 1, \ldots, d$*,*
     2.1. *Parse* Y *as the concatenation of* N/λ^ℓ *tuples, where the* j^{th} *tuple consists of* λ *entries* $(y_{j,1}, \ldots, y_{j,\lambda})$*, each of length* $B^{(\ell)}$ *bits.*
     2.2. *For all* $j \in \left[N/\lambda^\ell\right]$*, set* $\mathsf{msg}_{2,j}^{(\ell)} \leftarrow \mathsf{OT}_2\left(\mathsf{msg}_1^{(\ell)}, (y_{j,1}, \ldots, y_{j,\lambda})\right)$
     2.3. *Update* $Y \leftarrow \left(\mathsf{msg}_{2,1}^{(\ell)}, \ldots, \mathsf{msg}_{2,N/\lambda^\ell}^{(\ell)}\right)$ *and* $B^{(\ell+1)} = |\mathsf{msg}_{2,\cdot}^{(\ell)}|$*.*
  3. *Output* $\mathsf{msg}_2 = \mathsf{msg}_{2,1}^{(d)}$*.*
- $\mathsf{PIR}_3(\mathsf{msg}_2, \mathsf{st})$*: parse* st *as in Equation 6.1 and proceed as follows.*
  1. *Set* $\mathsf{msg}_2^{(d)} = \mathsf{msg}_2$*.*
  2. *For every* $\ell = d, \ldots, 1$*, set* $\mathsf{msg}_2^{(\ell-1)} \leftarrow \mathsf{OT}_3(\mathsf{msg}_2^{(\ell)}, \mathsf{st}^{(\ell)})$*.*
  3. *Output* $\mathsf{msg}_2^{(0)}$*.*

Analysis. We begin by discussing the communication complexity of the above protocol, and in particular, its download rate. We claim that for a database of length $\mathsf{poly}(\lambda)$ and large enough block size, the asymptotic download rate approaches 1. More specifically, let $B(\lambda)$ be the polynomial bound on the sender's secret length for which we achieve asymptotic rate $\frac{1}{c}$ in OT (see Definition 5.2). Then, if N is polynomial in λ, i.e. d is constant, and the length of the database entries, i.e. B, is larger than $B(\lambda)$, we have that $B^{(\ell+1)}/B^{(\ell)} \to 1/c$ with $\lambda \to \infty$, for any ℓ. From induction, we get that $|\mathsf{msg}_2|/B \to 1/c^d$, which is constant, and equals 1 when $c = 1$.

Regarding client-server communication, then, the client sends $d = \log_\lambda N$ OT messages, each is clearly of length at most polynomial in λ, which makes it $O(\mathsf{poly}(\lambda) \log N)$ communication complexity overall. The security of the construction is established by the following theorem.

Theorem 6.1 (PIR from Constant-Rate OT)**.** *Let* OT *be a receiver-private* 2*-message OT protocol, then the protocol* PIR *from Construction 6.1 is receiver-private.*

The proof follows from a standard hybrid argument over the receiver privacy of the OT. We refer the reader to [IP07] for further details.

6.2 Evaluating Branching Programs over Encrypted Data

Another result in the work of Ishai and Paskin [IP07] is a compiler that takes any 2-message rate-1 OT with *strong* sender privacy into a homomorphic encryption scheme that allows anyone to evaluate branching programs (a superclass of NC^1) over encrypted data. The crucial property of the scheme is that the size of the evaluated ciphertexts depends only on the length of the branching program but not on its *size*. This immediately yields a sublinear secure function evaluation protocol where the client's work is independent of the size of the branching program (which is in fact hidden to its eyes). The scheme can be seen as a generalized version of the PIR construction (see Section 6.1) where the answer of the server is determined by a polynomial-width unbounded length branching program. Several interesting applications of this scheme are discussed in [IP07], such as a keyword search protocol which hides from the client the size of the database held by the server.

We note that the rate-1 OT scheme from Construction 5.2 does not generically provide strong sender privacy from any trapdoor hash. However, it can be easily verified that by using the trapdoor hash schemes from Section 4 (based on DDH, QR or LWE) we do obtain strong sender privacy as defined in Definition 5.1. Hence, combining their approach with our rate-1 OT protocols, we obtain the following new implications.

Corollary 6.1. *There exists a homomorphic encryption scheme that supports the evaluation of branching programs of* unbounded *size s and depth d, where the size of the evaluated ciphertext is $\mathsf{poly}(d, \lambda)$ (and in particular independent of s), from the {DDH, QR} assumption.*

Corollary 6.2. *There exists a homomorphic encryption scheme that supports the (semi-)compact evaluation of branching programs of unbounded size with message-ciphertext rate approaching 1 (for a growing message length), from the {DDH, QR, LWE} assumption.*

Prior to our work, compact homomorphic encryption schemes for branching programs were only known under the LWE or the DCR assumption, and only the latter class of schemes achieved an (asymptotically) optimal rate.

6.3 Lossy Trapdoor Functions

A lossy trapdoor function [PW08] $(\mathsf{InjectiveSetup}, \mathsf{LossySetup}, \mathsf{Eval}, \mathsf{Invert})$ is an augmented trapdoor function where the setup comes in two different (but computationally indistinguishable) flavors: In the *injective* mode ($\mathsf{InjectiveSetup}$), the function (Eval) can be efficiently inverted (Invert), given the random coins of the setup. In the *lossy* mode ($\mathsf{LossySetup}$), the input of the function is lost in an information theoretic sense. Lossy trapdoor functions were originally introduced as a building block in the construction of CCA-secure encryption. Constructing lossy trapdoor functions from OT was first considered in [HO12]. There is a very simple rate-preserving transformation from rate-1 OT to lossy trapdoor function, which we show below.

Construction 6.2 (Lossy Trapdoor Functions from OT)**.** *Let $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ be a 2-message 1-out-of-2 OT with rate 1. The algorithms for a lossy trapdoor function are described in the following.*

- $\mathsf{InjectiveSetup}(1^\lambda, 1^n)$:

  1. *Sample $r \xleftarrow{\$} \{0,1\}^n$.*

2. *Compute* $(\mathsf{st}, \mathsf{msg}_1) \leftarrow \mathsf{OT}_1\left(1^\lambda, 1^2, 1\right)$.
3. *Output a function key* k *and a trapdoor* td *defined as follows*

$$\mathsf{k} = (\mathsf{msg}_1, r) \qquad\qquad \mathsf{td} = \mathsf{st} \tag{6.2}$$

- $\mathsf{LossySetup}(1^\lambda, 1^n)$:
  1. *Sample* $r \xleftarrow{\$} \{0,1\}^n$.
  2. *Compute* $(\mathsf{st}, \mathsf{msg}_1) \leftarrow \mathsf{OT}_1\left(1^\lambda, 1^2, 2\right)$.
  3. *Output* $\mathsf{k} = (\mathsf{msg}_1, r)$ *as the function key.*
- $\mathsf{Eval}(\mathsf{k}, \mathsf{x})$: *parse* k *as in Equation 6.2 and output* $\mathsf{y} = \mathsf{OT}_2\left(\mathsf{msg}_1, (\mathsf{x}, r)\right)$ *as the image of* x.
- $\mathsf{Invert}(\mathsf{t}, \mathsf{y})$: *parse* td *as in Equation 6.2 and output* $\mathsf{x} = \mathsf{OT}_3\left(\mathsf{st}, \mathsf{msg}_2\right)$ *as the pre-image of* y.

The rate of the lossy trapdoor function (defined as the asymptotic ratio between length of the input and length of the output of Eval) is clearly identical to the download rate of the underlying OT. We say that a function is L-lossy if the size of the lossy range is by a factor L smaller than the domain.

Theorem 6.2 (Lossiness)**.** *Let* OT *be a rate-1 OT and let* n' *be the size of the output of* $\mathsf{Eval}(\mathsf{k}, \cdot)$*, where* $\mathsf{k} \leftarrow \mathsf{LossySetup}(1^\lambda, 1^n)$*. Then Construction 6.2 is a* $\left(\frac{2^n}{2^{n'-n}}\right)$*-lossy trapdoor function.*

Proof. First, note that $r \in \{0,1\}^n$ is sampled uniformly at random and, in particular, is chosen independently of the random coins of the OT. Furthermore, by the correctness of the OT it holds that

$$r = \mathsf{OT}_3\left(\mathsf{st}, \mathsf{OT}_2\left(\mathsf{msg}_1, (\mathsf{x}, r)\right)\right)$$

with all but negligible probability. Since r is fixed and does not depend on x, at most $n' - n$ bits of x are leaked by the output of Eval, on input a lossy key k. Thus the size of the range of $\mathsf{Eval}(\mathsf{k}, \cdot)$ is bounded by $2^{n'-n}$. □

Observe that, since the download rate of the OT approaches 1, then the term $2^{n'-n}$ approaches 1 as n grows. This means that the lossiness of the function increases with n. In the following we argue that the output of the injective setup and the lossy setup are computationally indistinguishable.

Theorem 6.3 (Mode Indistinguishability)**.** *Let* OT *be a receiver private OT, then Construction 6.2 is mode-indistinguishable.*

Proof. The proof follows from a simple reduction against the receiver privacy of the OT. □

Our results imply the existence of a rate-1 lossy trapdoor function under the {DDH, QR, LWE} assumption. Prior to our work, rate optimal schemes were known to exist only under the DCR assumption.

7 Private Laconic Oblivious Transfer

In this section, we discuss another application of trapdoor hash: *private laconic oblivious transfer*, or $p\ell\mathsf{OT}$ for short. As discussed in the introduction, $p\ell\mathsf{OT}$ has strong applications in secure computation. In particular, following the outline presented in [CDG+17] to utilize laconic OT for non-interactive secure RAM computation with unprotected memory access, we can use private laconic OT to obtain secure RAM computation where the access pattern to the memory is also hidden, and therefore achieve a stronger notion of security.

We start by defining $p\ell\mathsf{OT}$, as well as a useful associated *reusability* notion. We then show how to realize $p\ell\mathsf{OT}$ using TDH for the class of index predicates. In fact, to obtain $p\ell\mathsf{OT}$ schemes that satisfy our security requirements, it is not sufficient to rely on the basic security notion of TDH, and we need a stronger one. Therefore, we first define a TDH scheme *with secret encoding*, and then show how to construct $p\ell\mathsf{OT}$ from any such a scheme. We also achieve reusable $p\ell\mathsf{OT}$ using a generalization of the secret encoding property.

To instantiate these constructions, we show that the *basic* DDH-based TDH from Section 4.2.1 has secret encodings, and further, in Appendix C, we describe a more efficient construction of a TDH with secret encodings based on the SXDH assumption over bilinear groups. We also show how to upgrade both schemes to have *reusable* secret encodings in Appendix B.

Lastly, we observe that the $p\ell\mathsf{OT}$ schemes we obtain are somewhat "unbalanced" in their parameters, and that their efficiency can be enhanced by applying a generic balancing technique, that takes any such "unbalanced" reusable $p\ell\mathsf{OT}$, and transforms it into a more efficient $p\ell\mathsf{OT}$ scheme. By applying the balancing technique to our DDH-based and pairing-based constructions, we achieve $p\ell\mathsf{OT}$ schemes with overall communication complexity proportional to $\sqrt{n}$ and, respectively, $n^{1/3}$.

7.1 Formal Definitions

Formal definitions of $p\ell\mathsf{OT}$ and reusable $p\ell\mathsf{OT}$ are provided below. We build over the formalization of laconic OT from [CDG+17] and introduce the new notions of privacy, then reusability.

Private Laconic OT. In Oblivious Transfer (OT), a sender and a receiver want to perform the following functionality: the sender has two secrets $(\mathsf{s}_0, \mathsf{s}_1)$ and the receiver has a bit $\mathsf{b} \in \{0,1\}$. The sender wishes to transfer secret s_b to the receiver. The bit b must not be leaked to the sender, and the receiver must not learn the other secret $\mathsf{s}_{1-\mathsf{b}}$. Clearly, in order to realize OT, the sender has to have some information that depends on b, and therefore, a first round, were the receiver sends an encoding of b, is necessary.

In laconic OT [CDG+17], the receiver has a large database $\mathsf{x} \in \{0,1\}^n$, and the sender wishes to transfer message $\mathsf{s}_{\mathsf{x}[i]}$ for a "selector" index $i \in [n]$ of his choice. Again, x must not be leaked to the sender, and the receiver must not learn the other message $\mathsf{s}_{1-\mathsf{x}[i]}$. Clearly, laconic OT can be realized by letting the receiver send an "OT encoding" of every $\mathsf{x}[i]$ in the first round. However, we require that the encoding of x, namely its *hash*, is laconic (hence the name).

Private laconic OT is a variant of laconic OT, where we additionally require that the index i remains private, and is not leaked to the receiver. We formalize $p\ell\mathsf{OT}$ as follows.

Definition 7.1 (Private Laconic OT)**.** *A* private laconic OT *scheme ($p\ell\mathsf{OT}$ for short) is a tuple of four PPT algorithms* $\Pi_{p\ell\mathsf{OT}} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *with the following properties.*

- ***Syntax:***

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^n)$. *The generating algorithm takes as input the security parameter* 1^λ*, and the size of the database* n*, and outputs public parameters* $\mathsf{pp} \in \{0,1\}^*$.
- $\mathsf{h} \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho)$. *The hashing algorithm takes as input the public parameters* pp*, a database* $\mathsf{x} \in \{0,1\}^n$*, and randomness* $\rho \in \{0,1\}^*$*, and deterministically outputs a hash value* $\mathsf{h} \in \{0,1\}^\eta$.
- $\mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1))$. *The sending algorithm takes as input the public parameters* pp*, a hash value* h*, an index* $i \in [n]$*, and a pair of secrets* $(\mathsf{s}_0, \mathsf{s}_1) \in \{0,1\} \times \{0,1\}$*, and outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^*$.
- $\mathsf{s} \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{ct}, \mathsf{x}; \rho)$. *The receiving algorithm takes as input the public parameters* pp*, a ciphertext* ct*, a database* $\mathsf{x} \in \{0,1\}^n$*, and randomness* $\rho \in \{0,1\}^*$*, and deterministically outputs a secret* $\mathsf{s} \in \{0,1\}$.

- ***Correctness:*** $\Pi_{p\ell\mathsf{OT}}$ *is* correct *if there exists a negligible function* $\epsilon(\lambda)$ *such that the following holds for all* $\lambda, n \in \mathbb{N}$*, any database* $\mathsf{x} \in \{0,1\}^n$*, any index* $i \in [n]$*, and any pair of secrets* $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}$.

$$\Pr\left[\mathsf{s} = \mathsf{s}_{\mathsf{x}[i]} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, n) \\ \rho \xleftarrow{\$} \{0,1\}^* \\ \mathsf{h} \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1)) \\ \mathsf{s} \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathsf{ct}, \mathsf{x}; \rho) \end{array}\right] \geq 1 - \epsilon(\lambda)$$

- ***Receiver Privacy:*** $\Pi_{p\ell\mathsf{OT}}$ *is statistically, resp., computationally,* receiver-private *if for any polynomial-length* $\{\mathsf{x}_\lambda, \mathsf{x}'_\lambda\}_{\lambda\in\mathbb{N}}$ *where* $n_\lambda := |\mathsf{x}_\lambda| = |\mathsf{x}'_\lambda|$ *for all* $\lambda \in \mathbb{N}$*, the following two distribution ensembles*

$$\{(\mathsf{pp}_\lambda, \mathsf{h}_\lambda)\}_{\lambda\in\mathbb{N}} \qquad\qquad \{(\mathsf{pp}_\lambda, \mathsf{h}'_\lambda)\}_{\lambda\in\mathbb{N}}$$

where $\mathsf{pp}_\lambda \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^{n_\lambda})$ *and* $\mathsf{h}_\lambda := \mathsf{Hash}(\mathsf{pp}_\lambda, \mathsf{x}_\lambda; \rho)$*,* $\mathsf{h}'_\lambda := \mathsf{Hash}(\mathsf{pp}_\lambda, \mathsf{x}'_\lambda; \rho')$ *for* $\rho, \rho' \xleftarrow{\$} \{0,1\}^*$*, are statistically, resp. computationally, indistinguishable.*

- ***Sender Privacy (against a semi-honest receiver):*** $\Pi_{p\ell\mathsf{OT}}$ *is (computationally)* sender-private *if there exists a PPT algorithm* Sim *such that for any* $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}$*, any polynomial-length* $\{\mathsf{x}_\lambda\}_{\lambda\in\mathbb{N}}$ *and any* $\{i_\lambda\}_{\lambda\in\mathbb{N}}$*, where* $n_\lambda := |\mathsf{x}_\lambda|$ *and* $i_\lambda \in [n_\lambda]$ *for all* $\lambda \in \mathbb{N}$*, the distribution ensembles* $\{\mathsf{Real}_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathsf{Ideal}_\lambda\}_{\lambda\in\mathbb{N}}$*, where*

$$\mathsf{Real}_\lambda = (\mathsf{pp}_\lambda, \mathsf{x}_\lambda, (\mathsf{ct}_\lambda, \rho)) \qquad\qquad \mathsf{Ideal}_\lambda = (\mathsf{pp}_\lambda, \mathsf{x}_\lambda, \mathsf{Sim}(1^\lambda, \mathsf{pp}_\lambda, \mathsf{x}_\lambda, \mathsf{s}_{\mathsf{x}_\lambda[i]}))$$

such that $\rho \xleftarrow{\$} \{0,1\}^*$*,* $\mathsf{pp}_\lambda \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^{n_\lambda})$ *and* $\mathsf{ct}_\lambda \xleftarrow{\$} \mathsf{Send}(\mathsf{pp}_\lambda, \mathsf{h}_\lambda, i_\lambda, (\mathsf{s}_0, \mathsf{s}_1))$ *for* $\mathsf{h}_\lambda = \mathsf{Hash}(\mathsf{pp}_\lambda, \mathsf{x}_\lambda; \rho)$*, are computationally indistinguishable.*

- ***Receiver Compactness:*** $\Pi_{p\ell\mathsf{OT}}$ *is* receiver-compact *if the output length of* Hash*,* η*, is independent in* n*, and is bounded by some polynomial in the security parameter* λ.

Reusable Private Laconic OT. Reusable $p\ell\mathsf{OT}$ is a special case of $p\ell\mathsf{OT}$ that allows the sender and receiver to efficiently perform many instances of $p\ell\mathsf{OT}$ that share the same selector index i w.r.t. to many different databases, such that the overhead in the communication cost is only additive in the number of instances. More specifically, in reusable $p\ell\mathsf{OT}$, the sender generates a reusable ciphertext ct that depends only on i (but not on a database) and sends it to the receiver in a preliminary phase. In the "online phase", every time the receiver sends a hash corresponding to some database x, the sender responds by sending a "compact" ciphertext c, specific to x, that allows the receiver to retrieve the $\mathsf{x}[i]^{th}$ message among a fresh pair of s_0 and s_1. Below, we provide a formal definition.

Definition 7.2 (Reusable Private Laconic OT)**.** *We say that a $p\ell\mathsf{OT}$ scheme* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *is* reusable *if it satisfies the following properties.*

- ***Syntax:*** *we require that there exist two PPT algorithms,* Send_1 *and* Send_2*, for which the* Send *procedure can be written as follows:*

  $\mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1))$*:*

  1. *Sample* $\sigma \xleftarrow{\$} \{0,1\}^*$.
  2. *Output* $(\mathsf{Send}_1(\mathsf{pp}, i; \sigma), \mathsf{Send}_2(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1); \sigma))$

  We sometimes write $\Pi_{p\ell\mathsf{OT}} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}_1, \mathsf{Send}_2, \mathsf{Receive})$.

- ***Multi-data Sender Privacy (against a semi-honest receiver):*** *we require that there exists a PPT algorithm* Sim *such that for any polynomial-length* $\{\langle \mathsf{x}_{\lambda,1}, \ldots, \mathsf{x}_{\lambda,\ell}\rangle\}_{\lambda\in\mathbb{N}}$*, any* $\{i_\lambda\}_{\lambda\in\mathbb{N}}$*, such that* $n_\lambda := |\mathsf{x}_{\lambda,j}|$ *and* $i_\lambda \in [n_\lambda]$ *for all* λ, j*, and any* $(\mathsf{s}_{1,0}, \mathsf{s}_{1,1}), \ldots, (\mathsf{s}_{\ell,0}, \mathsf{s}_{\ell,1}) \in \{0,1\}^2$*, the distribution ensembles* $\{\mathsf{Real}_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathsf{Ideal}_\lambda\}_{\lambda\in\mathbb{N}}$*, where*

  $$\mathsf{Real}_\lambda = (\mathsf{pp}_\lambda, \langle \mathsf{x}_{\lambda,j}\rangle_{j\in[\ell]}, (\mathsf{ct}_\lambda, \langle \mathsf{c}_{\lambda,j}\rangle_{j\in[\ell]}, \rho))$$
  $$\mathsf{Ideal}_\lambda = (\mathsf{pp}_\lambda, \langle \mathsf{x}_{\lambda,j}\rangle_{j\in[\ell]}, \mathsf{Sim}(1^\lambda, \mathsf{pp}_\lambda, \langle \mathsf{x}_{\lambda,j}, \mathsf{s}_{j,\mathsf{x}_{\lambda,j}[i]}\rangle_{j\in[\ell]}))$$

  such that $\rho := \langle \rho_j\rangle, \sigma \xleftarrow{\$} \{0,1\}^*$, $\mathsf{pp}_\lambda \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^{n_\lambda})$, $\mathsf{ct}_\lambda = \mathsf{Send}_1(\mathsf{pp}_\lambda, i_\lambda; \sigma)$*, and* $\mathsf{c}_{\lambda,j} = \mathsf{Send}_2(\mathsf{pp}_\lambda, \mathsf{h}_{\lambda,j}, i_\lambda, (\mathsf{s}_{j,0}, \mathsf{s}_{j,1}); \sigma)$ *for* $\mathsf{h}_{\lambda,j} = \mathsf{Hash}(\mathsf{pp}_\lambda, \mathsf{x}_{\lambda,j}; \rho_j)$*, for all* $j \in [\ell]$*, are computationally indistinguishable.*

- ***Sender Compactness:*** $\Pi_{p\ell\mathsf{OT}}$ *is* sender-compact *if the output length of* Send_2 *is independent in* n*, and is bounded by some polynomial in the security parameter* λ.

We conclude with few remarks.

Remark 7.1 (Generically achieving receiver privacy and compactness)**.** *We note that, as mentioned in [CDG+17] regarding laconic OT, the requirement for receiver privacy can be realized using a generic two-round secure computation protocol, and in particular, using standard OT and garbled circuits. More specifically, if we have a scheme that satisfies both correctness and sender privacy, then we can transform it to a $p\ell\mathsf{OT}$ by letting the receiver, in the hashing phase, to send an OT commitment for every bit in the hash value* h*. The sender then sends a garbled circuit for the functionality* $\mathsf{Send}(\mathsf{pp}, \cdot, i, (\mathsf{s}_0, \mathsf{s}_1))$*, and uses OT to transfer garbling labels for* h*. The receiver uses the labels to evaluate the garbled circuit and obtain the corresponding ciphertext* ct*, using which he can compute the secret* $\mathsf{s} := \mathsf{Receive}(\mathsf{pp}, \mathsf{ct}, \mathsf{x}; \rho)$*. Further, by replacing the standard OT in the*

transformation with "receiver-compact" laconic OT[15] *to transfer garbling labels for* h*, we can convert any non-receiver-compact* $p\ell\mathsf{OT}$ *into a receiver-compact scheme in exchange of an additive blow-up in the sender-receiver communication.*

Remark 7.2 (Correctness and receiver privacy in reusable $p\ell\mathsf{OT}$)**.** *We do not need to define new notions of correctness and receiver privacy for reusable* $p\ell\mathsf{OT}$ *since they are implied by the syntax and the corresponding properties of (standard)* $p\ell\mathsf{OT}$*.*

Remark 7.3 (On sender-compactness)**.** *Any* $p\ell\mathsf{OT}$ *scheme* $\Pi_{p\ell\mathsf{OT}}$ *can be transformed to a non-sender-compact reusable* $p\ell\mathsf{OT}$ *by setting* $\mathsf{Send}_2 := \Pi_{p\ell\mathsf{OT}}.\mathsf{Send}$*, and* $\mathsf{Send}_1(\cdot) = \bot$*. In such a case, reusing a scheme is equivalent to multiple independent executions, and thus, without sender-compactness, reusabilitiy is trivial.*

7.2 Private Laconic OT from Trapdoor Hash: The Basic Construction

We now present our basic construction of $p\ell\mathsf{OT}$ from trapdoor hash for index predicates.

Overview. Consider the following outline for constructing $p\ell\mathsf{OT}$ from trapdoor hash for index predicates. The public parameters contain a hash key hk which is generated using the generating algorithm of the underlying TDH scheme and defines a trapdoor hash function h_{hk} over inputs of size n. The receiver computes the hash of his input x as $\mathsf{h} := h_{\mathsf{hk}}(\mathsf{x})$, and sends it to the sender. The sender, in his turn, generates a pair $(\mathsf{ek}, \mathsf{td})$ corresponding to his private selector index, i (i.e. to $f_i(x) = x_i$), and computes the 0-encoding e_0 and 1-encoding e_1 that correspond to h under ek. The sender encrypts each of the secrets, s_0 and s_1, under the corresponding encoding, e_0 and respectively e_1, and sends both ciphertexts to the receiver, together with the encoding key ek.

From the correctness of the underlying TDH, the receiver is able to compute the encoding $\mathsf{e}_{\mathsf{x}[i]}$ using the encoding algorithm of the TDH, and therefore, he can decrypt the appropriate ciphertext and obtain $\mathsf{s}_{\mathsf{x}[i]}$. Moreover, the encoding key ek does not reveal any information about i, and therefore i remains private. We notice, however, that the security notion of TDH, from Definition 4.1, does not guarantee that $\mathsf{e}_{1-\mathsf{x}[i]}$ is secret in any sense. Thus, the receiver can potentially learn information about $\mathsf{e}_{1-\mathsf{x}[i]}$ and, therefore, about $\mathsf{s}_{1-\mathsf{x}[i]}$. For instance, in any rate-1 TDH, correctness implies that $\mathsf{e}_{1-\mathsf{x}[i]} = 1 - \mathsf{e}_{\mathsf{x}[i]}$ with a high probability.

Secret Encoding to the Rescue. For the above approach to work, we introduce a supplementary notion of security for TDH, which we call *secret encoding*. Roughly speaking, in TDH with secret encoding, we require that the encoding $\mathsf{e}_{1-f(\mathsf{x})}$ is secret, even given the public hk and ek. It is not hard to see that using a TDH with secret encoding, the construction outlined above satisfies full sender privacy, and moreover, that a natural generalization of secret encoding for when the encoding key is reused, yields a reusable $p\ell\mathsf{OT}$ construction.

We hereby provide formal definitions of the secret encoding property of a TDH and its generalization: reusable secret encoding.

Definition 7.3 (Trapdoor Hash with Secret Encoding)**.** *We say that a* TDH *scheme* $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ *for predicates* $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ *has* secret encoding *if for any polynomial-length* $\{\mathsf{x}_\lambda\}_{\lambda\in\mathbb{N}}$ *such that* $n_\lambda := |\mathsf{x}_\lambda|$ *for all* λ*, and any* $\{f_n\}_{n\in\mathbb{N}}$ *such that* $f_n \in \mathcal{F}_n$ *for all* n*, it holds that*

$$\{(\mathsf{x}_\lambda, \mathsf{hk}_\lambda, \mathsf{ek}_\lambda, \mathsf{e}_\lambda, \rho)\}_{\lambda\in\mathbb{N}} \stackrel{c}{\equiv} \{(\mathsf{x}_\lambda, \mathsf{hk}_\lambda, \mathsf{ek}_\lambda, \mathsf{e}'_\lambda, \rho)\}_{\lambda\in\mathbb{N}}$$

[15] Receiver-compactness is presented as an efficiency requirement for laconic OT in [CDG+17].

where $\rho \xleftarrow{\$} \{0,1\}^*$, $\mathsf{hk}_\lambda \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^{n_\lambda})$, $(\mathsf{ek}_\lambda, \mathsf{td}_\lambda) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}_\lambda, f_{n_\lambda})$, *and* $\mathsf{e}_\lambda = \mathsf{e}_{\lambda, 1-f_{n_\lambda}(\mathsf{x}_\lambda)}$ *where* $(\mathsf{e}_{\lambda,0}, \mathsf{e}_{1,\lambda}) = \mathsf{D}(\mathsf{td}_\lambda, \mathsf{H}(\mathsf{hk}_\lambda, \mathsf{x}_\lambda; \rho))$, *and* $\mathsf{e}'_\lambda \xleftarrow{\$} \{0,1\}^\omega$.

Definition 7.4 (Trapdoor Hash with Reusable Secret Encoding)**.** *We say that a* TDH *scheme* $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ *for predicates* $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ *has* reusable secret encoding *if for any polynomial-length* $\{\langle \mathsf{x}_{\lambda,1}, \ldots, \mathsf{x}_{\lambda,\ell}\rangle\}_{\lambda\in\mathbb{N}}$ *such that* $|\mathsf{x}_{\lambda,j}|$ *for all* λ, j*, and any* $\{f_n\}_{n\in\mathbb{N}}$ *such that* $f_n \in \mathcal{F}_n$ *for all* n*, it holds that*

$$\{(\langle \mathsf{x}_{\lambda,1}, \ldots, \mathsf{x}_{\lambda,\ell}\rangle, \mathsf{hk}_\lambda, \mathsf{ek}_\lambda, \langle \mathsf{e}_{\lambda,1}, \ldots, \mathsf{e}_{\lambda,\ell}\rangle, \rho)\} \stackrel{c}{\equiv} \{(\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell\rangle, \mathsf{hk}_\lambda, \mathsf{ek}_\lambda, \langle \mathsf{e}'_{\lambda,1}, \ldots, \mathsf{e}'_{\lambda,\ell}\rangle, \rho)\}$$

where $\rho := \langle \rho_1, \ldots, \rho_\ell\rangle \xleftarrow{\$} \{0,1\}^*$, $\mathsf{hk}_\lambda \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^{n_\lambda})$, $(\mathsf{ek}_\lambda, \mathsf{td}_\lambda) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}_\lambda, f_{n_\lambda})$, *and* $\mathsf{e}_{\lambda,j} = \mathsf{e}_{\lambda,j,1-f_{n_\lambda}(\mathsf{x}_{\lambda,j})}$ *where* $(\mathsf{e}_{\lambda,j,0}, \mathsf{e}_{\lambda,j,1}) = \mathsf{D}(\mathsf{td}_\lambda, \mathsf{H}(\mathsf{hk}_\lambda, \mathsf{x}_{\lambda,j}; \rho_j))$, *and* $\mathsf{e}'_{\lambda,j} \xleftarrow{\$} \{0,1\}^\omega$, *for* $j \in [\ell]$.

The Scheme. We now formally describe how to use any TDH with secret encoding to obtain private laconic OT. The construction uses TDH in a very straight-forward manner, and consequently obtains a $p\ell$OT scheme that is as efficient as the underlying TDH. Further, we show that when the underlying TDH has reusable secret encoding, then the obtained $p\ell$OT is reusable.

Construction 7.1 (Basic $p\ell$OT from TDH)**.** *Let* $\mathcal{H} = (\mathsf{S}, \mathsf{G}, \mathsf{H}, \mathsf{E}, \mathsf{D})$ *be a* $(1-\epsilon(\lambda))$*-correct rate-*$\frac{1}{p(\lambda)}$ TDH *scheme for some negligible function* $\epsilon(\lambda)$ *and polynomial* $p(\lambda)$*. Our* $p\ell$OT *construction builds over such a* TDH *and consists of the following algorithms:*

- $\mathsf{Gen}(1^\lambda, 1^n)$*: output* $\mathsf{pp} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$.
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho)$*: output* $\mathsf{H}(\mathsf{pp}, \mathsf{x}; \rho)$.
- $\mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1))$*:*

  1. *Sample* $\sigma \xleftarrow{\$} \{0,1\}^*$.

  $\mathsf{Send}_1(\mathsf{pp}, i; \sigma)$:

  2. *Set* $(\mathsf{ek}, \mathsf{td}) := \mathsf{G}(\mathsf{pp}, i; \sigma)$.

  $\mathsf{Send}_2(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1); \sigma)$:

  3. *Set* $(\mathsf{e}_0, \mathsf{e}_1) := \mathsf{D}(\mathsf{td}, \mathsf{h})$.
  4. *Sample* $\beta \xleftarrow{\$} \{0,1\}$*, and set*

  $$\mathsf{c}_0 := (0^{p(\lambda)-1} || \mathsf{s}_\beta) \oplus \mathsf{e}_\beta \qquad \mathsf{c}_1 := (0^{p(\lambda)-1} || \mathsf{s}_{1-\beta}) \oplus \mathsf{e}_{1-\beta}$$

  5. *Output*

  $$\mathsf{ct} := (\mathsf{ek}, \mathsf{c}_0, \mathsf{c}_1) \tag{7.1}$$

- $\mathsf{Receive}(\mathsf{pp}, \mathsf{ct}, \mathsf{x}; \rho)$*: parse* ct *as in Equation 7.1.*

  1. *Set* $\mathsf{e} := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$.
  2. *Compute*

  $$\mathsf{c}_0 \oplus \mathsf{e} \qquad \mathsf{c}_1 \oplus \mathsf{e}$$

  and output the LSB bit for whichever has $p(\lambda) - 1$ *leading zeros.*

Analysis. Receiver-compactness and correctness of the construction follow directly from the compactness and correctness of the underlying TDH (resp.). Notice, however, that in $\mathsf{Receive}$, there is a negligible probability that both $\mathsf{ct}_0 \oplus \mathsf{e}$ and $\mathsf{ct}_1 \oplus \mathsf{e}$ will have $p(\lambda) - 1$ leading zeros, in which case the receiver might obtain an incorrect value of s.

It is also easy to see that receiver privacy is immediately implied by the input privacy provided by the underlying TDH. We thus focus on showing that the scheme satisfies sender privacy as well, assuming the underlying TDH has secret encoding. In fact, we prove the more general notion of multi-data privacy, which is satisfied when the TDH has reusable secret encoding (since the TDH has inverse-polynomial rate, the resulting reusable scheme is sender-compact).

Theorem 7.1. *Let $\mathcal{H}$ be a TDH with secret encoding. Then, the $p\ell\mathsf{OT}$ scheme from Construction 7.1 satisfies sender privacy. Further, if $\mathcal{H}$ has reusable secret encoding, then the $p\ell\mathsf{OT}$ scheme provides multi-data sender privacy.*

Proof. Again, we consider the case where $\mathcal{H}$ has reusable secret encoding, and show that, the corresponding $p\ell\mathsf{OT}$ provides multi-data privacy. The proof for regular (one-time) privacy from regular (one-time) secret encoding is a special case. We construct a simulator Sim which, on inputs 1^λ, pp, $\langle \mathsf{x}_j, \mathsf{s}_{j,\mathsf{x}_j[i]} \rangle_{j\in[\ell]}$, generates $\mathsf{ek}' = \mathsf{G}(\mathsf{pp}, i'; \sigma')$ for an arbitrary $i' \in [n]$ and $\sigma' \xleftarrow{\$} \{0,1\}^*$, and sets $\mathsf{ct}' = \mathsf{ek}'$. Then, for every $j \in [\ell]$, he samples $\rho'_j \xleftarrow{\$} \{0,1\}^*$, computes encoding $\mathsf{e}'_j = \mathsf{E}(\mathsf{ek}', \mathsf{x}_j; \rho'_j)$, and sets $\mathsf{c}'_{j,\beta_j} = (0^{p(\lambda)-1} || \mathsf{s}_{j,\mathsf{x}_j[i]}) \oplus \mathsf{e}'_j$ and $\mathsf{c}'_{j,1-\beta_j} \xleftarrow{\$} \{0,1\}^{p(\lambda)}$, where $\beta'_j \xleftarrow{\$} \{0,1\}$ is a uniform bit. We define the output of Sim to be $((\mathsf{ct}, \langle(\mathsf{c}'_{j,0}, \mathsf{c}'_{j,1})\rangle_{j\in[\ell]}), \langle \rho'_j \rangle_{j\in[\ell]})$.

Now, let $\mathsf{Ideal} = (\mathsf{pp}, \langle \mathsf{x}_j \rangle_{j\in[\ell]}, (\mathsf{ct}', \langle(\mathsf{c}'_{j,0}, \mathsf{c}'_{j,1})\rangle_{j\in[\ell]}, \langle \rho'_j \rangle_{j\in[\ell]}))$ and let $\mathsf{Real} = (\mathsf{pp}, \langle \mathsf{x}_j \rangle_{j\in[\ell]}, (\mathsf{ct}, \langle(\mathsf{c}_{j,0}, \mathsf{c}_{j,1})\rangle_{j\in[\ell]}, \langle \rho_j \rangle_{j\in[\ell]}))$ be the distribution obtained from a real execution of $\Pi_{p\ell\mathsf{OT}}$ with public parameters pp and inputs $\langle \mathsf{x}_j \rangle_{j\in[\ell]}$ and $i \in [n]$, where $\langle \rho_j \rangle_{j\in[\ell]} \xleftarrow{\$} \{0,1\}^*$ and

$$\mathsf{ct} = \mathsf{Send}_1(\mathsf{pp}, i; \sigma) \qquad (\mathsf{c}_{j,0}, \mathsf{c}_{j,1}) = \mathsf{Send}_2(\mathsf{pp}, \mathsf{Hash}(\mathsf{pp}, \mathsf{x}_j; \rho_j), i, (\mathsf{s}_{j,0}, \mathsf{s}_{j,1}); \sigma)$$

for $\mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda, 1^n)$ and $\sigma \xleftarrow{\$} \{0,1\}^*$.

Assume towards contradiction that there exists a distinguisher $\mathcal{A}$ that distinguishes between Real and Ideal for some specific $\langle \mathsf{x}^*_j \rangle_{1\le j\le \ell}$, i^*, and $\langle(\mathsf{s}^*_{j,0}, \mathsf{s}^*_{j,1})\rangle_{1\le j\le \ell}$. We construct a distinguisher $\mathcal{A}'$, that breaks the reusable secret encoding property of $\mathcal{H}$ w.r.t. $\langle \mathsf{x}^*_j \rangle_{1\le j\le \ell}$. We define $\mathcal{A}'$ that takes as input a tuple $(\langle \mathsf{x}_j \rangle_{1\le j\le \ell}, \mathsf{hk}, \mathsf{ek}, \langle \mathsf{e}_j \rangle_{j\in[\ell]}, \langle \rho_j \rangle_{j\in[\ell]})$ (see Definition 7.4), and calls $\mathcal{A}$ on $\mathsf{View} = (\mathsf{hk}, \langle \mathsf{x}_j \rangle_{j\in[\ell]}, (\mathsf{ek}, \langle(\mathsf{c}''_{j,0}, \mathsf{c}''_{j,1})\rangle_{j\in[\ell]}, \langle \rho_j \rangle_{j\in[\ell]}))$ where, for every $j \in [\ell]$,

$$\mathsf{c}''_{j,\beta_j} = (0^{p(\lambda)-1} || \mathsf{s}^*_{j,\beta_j}) \oplus \mathsf{e}''_j \qquad \mathsf{c}''_{j,1-\beta_j} = (0^{p(\lambda)-1} || \mathsf{s}^*_{j,\beta_j}) \oplus \mathsf{e}_j$$

for $\mathsf{e}''_j = \mathsf{E}(\mathsf{ek}, \mathsf{x}_j; \rho_j)$ and $\beta_j \xleftarrow{\$} \{0,1\}$.

Consider the distribution View with $\langle \mathsf{x}^*_j \rangle_{j\in[\ell]}$, $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_{i^*})$ and $\langle \rho_j \rangle_{j\in[\ell]} \xleftarrow{\$} \{0,1\}^*$. It suffices to show that: (i) if $\mathsf{e}_j = \mathsf{e}_{j,1-\mathsf{x}_j[i]}$ for all $j \in [\ell]$, where $(\mathsf{e}_{j,0}, \mathsf{e}_{j,1}) = \mathsf{D}(\mathsf{td}, \mathsf{H}(\mathsf{hk}, \mathsf{x}_j; \rho_j))$, then $\mathsf{View} \stackrel{c}{\equiv} \mathsf{Real}$, and, on the other hand, (ii) if $\mathsf{e}_j \xleftarrow{\$} \{0,1\}^{p(\lambda)}$ for $j \in [\ell]$, then $\mathsf{View} \stackrel{c}{\equiv} \mathsf{Ideal}$.

We start with (i) From inspection, we see that $(\mathsf{pp}, \mathsf{ct}) \equiv (\mathsf{hk}, \mathsf{ek})$. Further, from the correctness property of $\mathcal{H}$, we have that $\mathsf{e}''_j = \mathsf{e}_{j,\mathsf{x}_j[i]}$ except with negligible probability, which implies, together with the fact that $\mathsf{e}_j = \mathsf{e}_{j,1-\mathsf{x}_j[i]}$, that $\langle(\mathsf{c}''_{j,0}, \mathsf{c}''_{j,1})\rangle_{j\in[\ell]} \stackrel{c}{\equiv} \langle(\mathsf{c}_{j,0}, \mathsf{c}_{j,1})\rangle_{j\in[\ell]}$ given $(\mathsf{pp}, \mathsf{ct}) \equiv (\mathsf{hk}, \mathsf{ek})$,

For (ii), we again see that $\mathsf{pp} \equiv \mathsf{hk}$. This does not immediately imply that $\mathsf{ek} \equiv \mathsf{ct}'$, since $\mathsf{ct}' = \mathsf{ek}'$ is computed using an arbitrary i'. Regardless, we can use the function privacy property of $\mathcal{E}$, which states that $(\mathsf{hk}, \mathsf{ek}) \stackrel{c}{\equiv} (\mathsf{hk}, \mathsf{ek}')$ for such ek and ek'. The rest is immediate since when $\langle \mathsf{e}_j \rangle_{j\in[\ell]}$ are uniform, so are $\langle \mathsf{c}''_{j,1-\mathsf{x}_j[i]} \rangle_{j\in[\ell]}$, and therefore, they distribute equivalently to $\langle \mathsf{c}'_{j,1-\mathsf{x}_j[i]} \rangle_{j\in[\ell]}$. □

Instantiation I: (Reusable) $p\ell\mathsf{OT}$ from DDH-based Trapdoor Hash. We instantiate the construction from above using our basic DDH-based TDH from Section 4.2.1. Fortunately, as stated and proven in the theorem below, the construction already satisfies the secret encoding property and therefore we get $p\ell\mathsf{OT}$ for free.

Theorem 7.2. *The TDH scheme from Construction 4.1 has secret encoding under DDH.*

Proof. Fix some $\lambda, n \in \mathbb{N}$, $\mathsf{x} \in \{0,1\}^n$ and $i \in [n]$, and assume w.l.o.g. that $\mathsf{x}[i] = 0$ (the proof for $\mathsf{x}[i] = 1$ is similar). Let $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_i)$, $\mathsf{h} := \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$, where ρ is parsed as $r \xleftarrow{\$} \mathbb{Z}_p$, and $(\mathsf{e}_0, \mathsf{e}_1) := \mathsf{D}(\mathsf{td}, \mathsf{h})$. Our proof is based on reductions to DDH similar to those in the proof of Theorem 4.2. We first show that

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}_1, r) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}', r) \tag{7.2}$$

where ek' is defined as follows

$$\mathsf{ek}' := \left(u, \begin{pmatrix} u_{1,0}, \ldots, u_{i,0}, \ldots, u_{n,0} \\ u_{1,1}, \ldots, u'_{i,1}, \ldots, u_{n,1} \end{pmatrix}\right)$$

for a uniform $u'_{i,1}$ and $u, \{u_{j,b}\}$ as defined in Equation 4.4, and then complete the proof by showing that

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}', r) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}', r) \tag{7.3}$$

Let $\mathcal{D}$ be a PPT distinguisher that breaks (7.2), i.e.,

$$|\Pr[\mathcal{D}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}_1, r) = 1] - \Pr[\mathcal{D}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}', r) = 1]| > \mathsf{negl}(\lambda)$$

for any negligible function $\mathsf{negl}(\cdot)$. We construct an adversary $\mathcal{A}$ that uses $\mathcal{D}$ to break the DDH assumption. $\mathcal{A}$ takes as input a tuple $((\mathbb{G}, p, g), (R, S, T))$, and proceeds as follows.

1. Sample $r \xleftarrow{\$} \mathbb{Z}_p$ and $r_{j,b} \xleftarrow{\$} \mathbb{Z}_p$ for every $j \in [n]$ and $b \in \{0,1\}$.
2. For every $j \in [n] \backslash \{i\}$, set

$$\tilde{g}_{j,0} := g^{r_{j,0}} \qquad \tilde{g}_{j,1} := g^{r_{j,1}}$$
$$\tilde{u}_{j,0} := S^{r_{j,0}} \qquad \tilde{u}_{j,1} := S^{r_{j,1}}$$

3. Sample $t \xleftarrow{\$} \mathbb{Z}_p$, and set

$$\tilde{g}_{i,0} := g^{r_{j,0}} \qquad \tilde{g}_{i,1} := R$$
$$\tilde{u}_{i,0} := S^{r_{j,0}} \qquad \tilde{u}_{i,1} := T \cdot g^t$$

4. Set

$$\widetilde{\mathsf{hk}} := \left((\mathbb{G}, p, g), \begin{pmatrix} \tilde{g}_{1,0}, \tilde{g}_{2,0}, \ldots, \tilde{g}_{n,0} \\ \tilde{g}_{1,1}, \tilde{g}_{2,1}, \ldots, \tilde{g}_{n,1} \end{pmatrix}\right) \qquad \widetilde{\mathsf{ek}} := \left(S, \begin{pmatrix} \tilde{u}_{1,0}, \tilde{u}_{2,0}, \ldots, \tilde{u}_{n,0} \\ \tilde{u}_{1,1}, \tilde{u}_{2,1}, \ldots, \tilde{u}_{n,1} \end{pmatrix}\right)$$

$$\tilde{\mathsf{e}} := S^r \prod_{j \in [n]} \tilde{u}_{j,\mathsf{x}[j]} g^t$$

and output $\mathcal{D}(\mathsf{x}, \widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}, \tilde{\mathsf{e}}, r)$.

Simple calculations show that if $\mathcal{D}$ breaks 7.2 then $\mathcal{A}$ breaks the DDH assumption. We skip further details as they are similar to the proof of Theorem 4.2. Moreover, a similar reduction, where the distinguisher always outputs a uniform $\tilde{\mathsf{e}}$ shows that 7.3 holds true as well. □

Theorem 7.2 allows us to instantiate Construction 7.1 with the TDH scheme from Construction 4.1. Further, in Appendix B, we show how to use *correlated-input secure hash* (CIH) [GOR11, AMN+18] to further extend the DDH-based TDH to a scheme with reusable secret encoding, and thus, when used for the $p\ell\mathsf{OT}$ from Construction 7.1, gives a compact reusable $p\ell\mathsf{OT}$ scheme.

Corollary 7.1. *There exists a reusable $p\ell\mathsf{OT}$ scheme, with statistical receiver privacy and sender privacy under the DDH assumption, that has communication complexity $O(\mathsf{poly}(\lambda)n)$.*

Instantiation II: Sublinear $p\ell\mathsf{OT}$ from Pairing-based Trapdoor Hash. In Appendix C, we present another construction of trapdoor hash based on the SXDH assumption over bilinear groups, which is the analog of DDH over groups with bilinear pairings. The construction is similar to the DDH-based scheme, however, using pairings, we are able to shrink the public parameters and the ciphertext to length proportional to $\sqrt{n}$. Reusable secret encoding can be also achieved for the pairing-based scheme under ad hoc difficulty assumptions (or in the random oracle model). We elaborate in Appendix C. Using the pairing-based TDH to construct $p\ell\mathsf{OT}$ obtains the following.

Corollary 7.2. *There exists a $p\ell\mathsf{OT}$ scheme, with statistical receiver privacy and sender privacy under the SXDH assumption, that has communication complexity $O(\mathsf{poly}(\lambda)\sqrt{n})$. If we also assume the existence of a random oracle, or a correlated-input secure hash[16] over bilinear groups, then the scheme can be extended to a reusable $p\ell\mathsf{OT}$.*

7.3 The Balancing Technique

We observe that in our DDH-based and pairing-based $p\ell\mathsf{OT}$ schemes (or any instantiation of Construction 7.1 for the matter), and their reusable variants, the receiver-sender communication, which consists only of the compact hash, is small compared to the sender-receiver communication, which contains the ciphertext and the public parameters that are of size $O(\mathsf{poly}(\lambda)n)$ or $O(\mathsf{poly}(\lambda)\sqrt{n})$ (resp.). In this section, we propose a "balancing technique", which takes any such unbalanced *reusable* $p\ell\mathsf{OT}$ scheme and transforms it into a balanced scheme, which is although not reusable, has better overall efficiency, and may be useful especially in applications where we seek to minimize the non-amortized communication complexity of a single $p\ell\mathsf{OT}$ invocation (see introduction).

[16] In fact, we need a much weaker notion of CIH than the one used in prior work [GOR11, AMN+18].

Overview. For presentation, let us consider the reusable DDH-based $p\ell\mathsf{OT}$, which is both receiver- and sender-compact, and has public parameters and ciphertext of length $O(\mathsf{poly}(\lambda)n)$. We balance the scheme as follows. We write the database x as $\sqrt{n}$ databases $\mathsf{x}_1, \ldots, \mathsf{x}_{\sqrt{n}}$, each of size $\sqrt{n}$, and interpret every index $j \in [n]$ as a pair $(j_1, j_2) \in [\sqrt{n}]^2$. We then use the underlying reusable $p\ell\mathsf{OT}$ w.r.t. index i_2 over each of the $\sqrt{n}$ small databases in order to transfer $\sqrt{n}$ bits. That is, the receiver receives $\sqrt{n}$ bits conditioned on values $\mathsf{x}_1[i_2], \ldots, \mathsf{x}_{\sqrt{n}}[i_2]$. The bits are chosen so that the receiver is able to recover s_b if and only if $\mathsf{x}_{i_1}[i_2] = b$. More specifically, for every $j \neq i_1$, the sender chooses a random bit, m_j, which is transferred to the receiver both in the case where $\mathsf{x}_j[i_2] = 0$ and where $\mathsf{x}_j[i_2] = 1$. Intuitively, this cancels out the effect of $\mathsf{x}_j[i_2]$ for such j's. The only element in the database which should make a difference is $\mathsf{x}_{i_1}[i_2]$, and therefore, in the $p\ell\mathsf{OT}$ over x_{i_1}, the sender transfers $\mathsf{m}^0_{i_1} := \mathsf{s}_0 - \sum_{j \neq i_1} \mathsf{m}_j$ if $\mathsf{x}_{i_1}[i_2] = 0$, and $\mathsf{m}^1_{i_1} := \mathsf{s}_1 - \sum_{j \neq i_1} \mathsf{m}_j$ if $\mathsf{x}_{i_1}[i_2] = 1$. If $\mathsf{x}_{i_1}[i_2] = b$, then the receiver can sum up the $\sqrt{n}$ bits from the $\sqrt{n}$ $p\ell\mathsf{OT}$'s, and obtain s_b. The security of the scheme is derived from the security of the underlying $p\ell\mathsf{OT}$.

Although the described scheme is not receiver-compact, following Remark 7.1 we can obtain a compact scheme by using laconic OT and garbled circuits to delegate some of the sender's computation to the receiver-side.

The Scheme. In Construction 7.2 below, we generalize the above idea to work over any underlying compact and reusable $p\ell\mathsf{OT}$, with possibly different parameters, and therefore parametrize the balancing as well. We then provide a formal proof for the construction.

Construction 7.2. *Let $\Pi'_{p\ell\mathsf{OT}} = (\mathsf{Gen}', \mathsf{Hash}', \mathsf{Send}'_1, \mathsf{Send}'_2, \mathsf{Receive}')$ be a receiver-compact reusable $p\ell\mathsf{OT}$ scheme where the outputs length of Gen' and Send'_1 are $\psi(\lambda, n)$ and $\gamma(\lambda, n)$, resp., for security parameter λ and database size n. For any $0 < \alpha < 1$, we construct a non-compact $p\ell\mathsf{OT}$ scheme $\Pi_{p\ell\mathsf{OT}} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ where the outputs length of Gen, Send and Hash are $\psi(\lambda, n^{1-\alpha})$, $\gamma(\lambda, n^{1-\alpha}) + \mathsf{poly}(\lambda)n^\alpha$, and $\mathsf{poly}(\lambda)n^{1-\alpha}$, resp., for parameters λ and n.*

- $\mathsf{Gen}(1^\lambda, 1^n)$*: output* $\mathsf{Gen}'(1^\lambda, 1^{n^{1-\alpha}})$.
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho)$*: parse ρ as $(\rho_1, \ldots, \rho_{n^\alpha})$, and x as $(\mathsf{x}_1, \ldots, \mathsf{x}_{n^\alpha})$, where $\mathsf{x}_j \in \{0,1\}^{n^{1-\alpha}}$ for every $j \in [n^\alpha]$ (pad if necessary).*
  1. *For every $j \in [n^\alpha]$, set $\mathsf{h}_j := \mathsf{Hash}'(\mathsf{pp}, \mathsf{x}_j; \rho_j)$.*
  2. *Output*
  $$\mathsf{h} := (\mathsf{h}_1, \ldots, \mathsf{h}_{n^\alpha}) \tag{7.4}$$
- $\mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1))$*: parse h as in Equation 7.4, and i as $(i_1, i_2) \in [n^\alpha] \times [n^{1-\alpha}]$.*
  1. *Sample $\rho' \xleftarrow{\$} \{0,1\}^*$.*
  2. *Set $\mathsf{ct}' := \mathsf{Send}'_1(\mathsf{pp}, i; \rho')$.*
  3. *For every $j \in [n^\alpha] \backslash \{i_1\}$, sample $\mathsf{m}_j \xleftarrow{\$} \{0,1\}$, and set $\mathsf{c}_j := \mathsf{Send}_2(\mathsf{pp}, \mathsf{h}_j, i_2, (\mathsf{m}_j, \mathsf{m}_j); \rho')$.*
  4. *Set*
  $$\mathsf{m}^0_{i_1} := \mathsf{s}_0 - \sum_{j \in [n^\alpha] \backslash \{i_1\}} \mathsf{m}_j \qquad \mathsf{m}^1_{i_1} := \mathsf{s}_1 - \sum_{j \in [n^\alpha] \backslash \{i_1\}} \mathsf{m}_j$$
  and
  $$\mathsf{c}_{i_1} := \mathsf{Send}_2(\mathsf{pp}, \mathsf{h}_j, i_2, (\mathsf{m}^0_{i_1}, \mathsf{m}^1_{i_1}); \rho')$$

5. *Output*

$$\mathsf{ct} := (\mathsf{ct}', \mathsf{c}_1, \ldots, \mathsf{c}_{n^\alpha}) \tag{7.5}$$

- $\mathsf{Receive}(\mathsf{pp}, \mathsf{ct}, \mathsf{x}; \rho)$*: parse* ct *as in Equation 7.5.*

  1. *For* $j \in [n^\alpha]$*, set* $\mathsf{m}_j := \mathsf{Receive}(\mathsf{pp}, (\mathsf{ct}', \mathsf{c}_j), \mathsf{x}_j; \rho)$.
  2. *Output*

$$\mathsf{s} := \sum_{j \in [n]} \mathsf{m}_j$$

Analysis. Correctness is immediate. In the following two theorems, we derive the privacy of the balanced construction on the privacy of the underlying $p\ell\mathsf{OT}$. We start with receiver privacy.

Theorem 7.3. *Let* $\Pi'_{p\ell\mathsf{OT}}$ *be a* $p\ell\mathsf{OT}$ *scheme with statistical (resp., computational) receiver privacy. Then, the scheme* $\Pi_{p\ell\mathsf{OT}}$ *from Construction 7.2, built over* $\Pi'_{p\ell\mathsf{OT}}$ *with any* $\alpha \in (0,1)$*, is statistically (computationally) receiver-private.*

Proof. We show that for any $\lambda, n \in \mathbb{N}$ and any two databases $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$, it holds that

$$(\mathsf{pp}, \mathsf{h}) \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{h}')$$

where $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Gen}(1^\lambda, 1^n)$, $\rho, \rho' \stackrel{\$}{\leftarrow} \{0,1\}^*$, $\mathsf{h} = \mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho)$ and $\mathsf{h}' = \mathsf{Hash}(\mathsf{pp}, \mathsf{x}'; \rho')$.

Parse x and x', as $(\mathsf{x}_1, \ldots, \mathsf{x}_{n^\alpha})$ and $(\mathsf{x}'_1, \ldots, \mathsf{x}'_{n^\alpha})$, where $\mathsf{x}_j, \mathsf{x}'_j \in \{0,1\}^{n^{1-\alpha}}$ for every $j \in [n^\alpha]$. Our goal is to prove that

$$(\mathsf{pp}, (\mathsf{h}_1, \ldots, \mathsf{h}_{n^\alpha})) \stackrel{c}{\equiv} (\mathsf{pp}, (\mathsf{h}'_1, \ldots, \mathsf{h}'_{n^\alpha}))$$

where $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Gen}'(1^\lambda, 1^{n^{1-\alpha}})$ and, for any j, $\mathsf{h} := \mathsf{Hash}'(\mathsf{pp}, \mathsf{x}_j; \rho_j)$ and $\mathsf{h}' := \mathsf{Hash}'(\mathsf{pp}, \mathsf{x}'_j; \rho'_j)$ where $\rho_j, \rho'_j \stackrel{\$}{\leftarrow} \{0,1\}^*$.

We prove the indistinguishability using a hybrids argument with $n^\alpha + 1$ distributions Hybrid_0, Hybrid_1, $\ldots$, $\mathsf{Hybrid}_{n^\alpha}$, where $\mathsf{Hybrid}_0 := (\mathsf{pp}, (\mathsf{h}_1, \ldots, \mathsf{h}_{n^\alpha}))$, $\mathsf{Hybrid}_{n^\alpha} := (\mathsf{pp}, (\mathsf{h}'_1, \ldots, \mathsf{h}'_{n^\alpha}))$, and every hybrid Hybrid_k in between is defined as

$$\mathsf{Hybrid}_k := (\mathsf{pp}, (\mathsf{h}'_1, \ldots, \mathsf{h}'_{k-1}, \mathbf{h'_k}, \mathsf{h}'_{k+1}, \ldots, h_{n^\alpha}))$$

.

We now prove that $\mathsf{Hybrid}_k \stackrel{c}{\equiv} \mathsf{Hybrid}_{k-1}$ for every $k \in [n^\alpha]$ and by this we finish. We do that using a reduction to the receiver privacy of the underling $p\ell\mathsf{OT}$ scheme, $\Pi'_{p\ell\mathsf{OT}}$. Let $\mathcal{D}$ be a distinguisher for which

$$|\Pr[\mathcal{D}(\mathsf{Hybrid}_{k-1}) = 1] - \Pr[\mathcal{D}(\mathsf{Hybrid}_k) = 1]| > \mathsf{negl}(\lambda)$$

for any negligible function $\mathsf{negl}(\cdot)$.

We use $\mathcal{D}$ to construct an adversary $\mathcal{A}$ that breaks the receiver privacy of $\Pi_{p\ell\mathsf{OT}}$. $\mathcal{A}$ receives as input (PP, H) and outputs

$$\mathcal{D}(\tilde{\mathsf{pp}} := PP, (\tilde{h}_1, \ldots, \tilde{h}_{k-1}, \tilde{h}_k := H, \tilde{h}_{k+1}, \ldots, \tilde{h}_{n^\alpha}))$$

where $\tilde{h}_j := \mathsf{Hash}'(PP, \mathsf{x}_j; \rho_j)$ for $1 \leq j < k$ and $\tilde{h}_j := \mathsf{Hash}'(PP, \mathsf{x}'_j; \rho_j)$ for $k < j \leq n^\alpha$, where $\rho_1, \ldots, \rho_{n^\alpha} \stackrel{\$}{\leftarrow} \{0,1\}^*$. It is easy to verify that if $(PP, H) \equiv (\mathsf{pp}, \mathsf{h}_k)$, then the input to $\mathcal{D}$ distributes like Hybrid_{k-1}, and that if $(PP, H) \equiv (\mathsf{pp}, \mathsf{h}'_k)$, then it distributes like Hybrid_k. By this we finish. □

Next, we show sender privacy.

Theorem 7.4. *Let* $\Pi'_{p\ell\mathsf{OT}} := (\mathsf{Gen}', \mathsf{Hash}', \mathsf{Send}'_1, \mathsf{Send}'_2, \mathsf{Receive}')$ *be a reusable* $p\ell\mathsf{OT}$ *scheme with multi-data sender privacy. Then, the* $p\ell\mathsf{OT}$ *scheme from Construction 7.2, built over* $\Pi_{p\ell\mathsf{OT}}$ *with any* $\epsilon \in (0,1)$*, provides (regular) sender privacy.*

Proof. We show that there exists a PPT simulator Sim such that for any $\lambda, n \in \mathbb{N}$, any database $\mathsf{x} \in \{0,1\}^n$, any index $i \in [n]$, and any pair of secrets $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}$, it holds that

$$(\mathsf{pp}, \mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1))) \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{Sim}(\mathsf{pp}, \mathsf{x}, \mathsf{s}_{\mathsf{x}[i]}))$$

where $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Gen}(1^\lambda, 1^n)$, $\rho \stackrel{\$}{\leftarrow} \{0,1\}^*$, and $\mathsf{h} := \mathsf{Hash}(\mathsf{pp}, \mathsf{x}; \rho)$.

From the multi-data receiver privacy of $\Pi_{p\ell\mathsf{OT}}$, we know that there exists a simulator, Sim', such that for any $\langle \mathsf{x}_j \rangle_{1 \le j \le \ell}$, any index $i \in [n]$, and any $\langle (\mathsf{s}_{j,0}, \mathsf{s}_{j,1}) \rangle_{1 \le j \le \ell}$, it holds that

$$\begin{aligned}(\mathsf{pp}, (\mathsf{Send}'_1(\mathsf{pp}, i; \rho'), \langle \mathsf{Send}'_2(\mathsf{pp}, \mathsf{h}_j, i, (\mathsf{s}_{j,0}, \mathsf{s}_{j,1}); \rho') \rangle_{1 \le j \le \ell})) \\ \stackrel{c}{\equiv} (\mathsf{pp}, \mathsf{Sim}'(\mathsf{pp}, \langle \mathsf{x}_j, \mathsf{s}_{j,\mathsf{x}_j[i]} \rangle_{1 \le j \le \ell})) \end{aligned} \tag{7.6}$$

where $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Gen}'(1^\lambda, 1^n)$, $\rho, \rho' \stackrel{\$}{\leftarrow} \{0,1\}^*$, and $\mathsf{h}_j := \mathsf{Hash}'(\mathsf{pp}, \mathsf{x}_j; \rho)$ for $j = 1, \ldots, \ell$.

We use such a simulator to construct a simulator Sim, which takes as input a tuple $(\mathsf{pp}, \mathsf{x}, \mathsf{s}_{\mathsf{x}[i]})$, and proceeds as follows.

1. Parse x as $(\mathsf{x}_1, \ldots, \mathsf{x}_{n^\epsilon})$, where $\mathsf{x}_j \in \{0,1\}^{n^{1-\epsilon}}$ for every j (pad if necessary).
2. Sample $\mathsf{m}_1, \ldots, \mathsf{m}_{n^\epsilon} \stackrel{\$}{\leftarrow} \{0,1\}$ such that $\sum_j \mathsf{m}_j = \mathsf{s}_{\mathsf{x}[i]}$.
3. Output $\mathsf{Sim}'(\mathsf{pp}, \langle \mathsf{x}_j, \mathsf{m}_j \rangle_{1 \le j \le n^\epsilon})$.

From Construction 7.2, it can be seen that for any x, i, and $(\mathsf{s}_0, \mathsf{s}_1)$,

$$\mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1)) \equiv (\mathsf{Send}'_1(\mathsf{pp}, i; \rho'), \langle \mathsf{Send}'_2(\mathsf{pp}, \mathsf{h}_j, i_2, (\mathsf{m}_{j,0}, \mathsf{m}_{j,1})) \rangle_{1 \le j \le n^\epsilon}) \tag{7.7}$$

where $\langle \mathsf{m}_{j,\mathsf{x}_j[i]} \rangle_{1 \le j \le n^\epsilon}$ are uniform under the constraint that $\sum_j \mathsf{m}_{j,\mathsf{x}_j[i]} = \mathsf{s}_{\mathsf{x}[i]}$, $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{Gen}'(1^\lambda, 1^{n^{1-\epsilon}})$, $\rho, \rho' \stackrel{\$}{\leftarrow} \{0,1\}^*$, and $\mathsf{h}_j := \mathsf{Hash}'(\mathsf{pp}, \mathsf{x}_j; \rho)$ for $j = 1, \ldots, n^\epsilon$. By combining Equations 7.6 and 7.7, we get

$$\mathsf{Send}(\mathsf{pp}, \mathsf{h}, i, (\mathsf{s}_0, \mathsf{s}_1)) \stackrel{c}{\equiv} \mathsf{Sim}(\mathsf{pp}, \langle \mathsf{x}_j, \mathsf{m}_j \rangle_{1 \le j \le n^\epsilon})$$

which concludes the proof. □

Applying the Balancing over our Instantiations. Recall the reusable $p\ell\mathsf{OT}$ schemes based on the DDH and SXDH assumption. The two constructions are unbalanced: the hash and output of Send_2 are compact while the public parameters and output of Send_1 are linear and square-root, respectively. Since the two constructions have different parameters, the optimal parameterization of the balancing is different. For the DDH-based scheme, we use balancing parameter $\alpha = \frac{1}{2}$, whereas for the pairing-based TDH we use $\alpha = \frac{1}{3}$, and obtain the following corollaries.

Corollary 7.3. *There exists a* $p\ell\mathsf{OT}$ *scheme, with statistical receiver privacy and sender privacy under the DDH assumption, that has overall communication complexity of* $O(\mathsf{poly}(\lambda)\sqrt{n})$.

Corollary 7.4. *There exists a* $p\ell\mathsf{OT}$ *scheme, with statistical receiver privacy and sender privacy assuming SXDH and correlated-input secure hash over bilinear groups (or a random oracle), that has overall communication complexity of* $O(\mathsf{poly}(\lambda)n^{\frac{1}{3}})$.

8 Acknowledgements

We thank Craig Gentry, Shai Halevi, Srinath Setty, and Vinod Vaikuntanathan for helpful discussions and pointers.

S. Garg supported by DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

Y. Ishai supported by ERC Project NTSC (742754), ISF grant 1709/14, NSF-BSF grant 2015782, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.

G. Malavolta supported by a gift from Ripple, a gift from DoS Networks, a grant from Northrop Grumman, a Cylab seed funding award, and a JP Morgan Faculty Fellowship.

T. Mour supported by BSF grant 2012378, and NSF-BSF grant 2015782.

R. Ostrovsky supported by NSF grant 1619348, BSF grant 2015782, DARPA SafeWare subcontract to Galois Inc., DARPA SPAWAR contract N66001-15-C-4065, JP Morgan Faculty Research Award, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government.

References

[ADI+17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 223–254, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 8

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, pages 119–135, 2001. 8

[AMN+18] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 543–574, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 9, 20, 61, 75, 76

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 9, 75

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240

of *Lecture Notes in Computer Science*, pages 370–390, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 6, 8, 47

[Bea95a] Donald Beaver. Precomputing oblivious transfer. In *CRYPTO '95*, pages 97–109, 1995. 5

[Bea95b] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany. 49

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 13, 22, 35, 73

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 4, 7, 12, 31

[BGI+17a] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 275–303, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 6, 46

[BGI17b] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT 2017, Part II*, pages 163–193, 2017. 5, 8

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 22

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 6

[BMN18] Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. In *INDOCRYPT 2018*, pages 375–398, 2018. 8

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 14, 36

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 5, 9, 17, 18, 19, 27, 54, 56, 57

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. 4

[Cha04] Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04: 9th Australasian Conference on Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 50–61, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany. 7

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 7

[CRVW02] Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *34th Annual ACM Symposium on Theory of Computing*, pages 659–668, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 24

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 6, 7, 11, 27

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. 6

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 21

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO 2016, Part III*, pages 93–122, 2016. 8

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany. 4, 5, 7

[DKK18] Itai Dinur, Nathan Keller, and Ohad Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 213–242, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 31

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008. 23

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 23

[FGJI17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from paillier encryption. In *ProvSec 2017*, pages 381–399, 2017. 4

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 81

[Gal63] Robert G. Gallager. Low-density parity-check codes, 1963. 25

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 4

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. 4

[GGH18] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. Cryptology ePrint Archive, Report 2018/872, 2018. https://eprint.iacr.org/2018/872. 7, 11, 31

[GGI+15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *J. Cryptology*, 28(4):820–843, 2015. 4

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 464–479. IEEE, 1984. 31

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 362–391, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 6, 7, 11, 27

[GH19] Craig Gentry and Shai Halevi. Compressible fhe with applications to pir. Technical report, 2019. (personal communication). 21

[GKL10] Jens Groth, Aggelos Kiayias, and Helger Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In *PKC 2010*, pages 107–123, 2010. 4

[GL10] David Goldenberg and Moses Liskov. On related-secret pseudorandomness. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 255–272, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany. 9, 75

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press. 22

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, pages 126–135, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 49

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 4

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 182–200, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 9, 61, 75

[GUV07] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *IEEE Conference on Computational Complexity*, pages 96–108. IEEE Computer Society, 2007. 24

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press. 47

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012. 8

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-ddh and lossy trapdoor functions. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 627–643, 2012. 17, 52

[HW14] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. Cryptology ePrint Archive, Report 2014/669, 2014. http://eprint.iacr.org/2014/669. 6

[IKM+13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 600–620, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany. 5

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 9, 75

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC 2008*, pages 433–442, 2008. 8

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 6, 7, 10, 16, 39, 50, 51, 52

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC 2009*, pages 294–314, 2009. 8

[JVC18a] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, pages 1651–1669, 2018. 5, 8

[JVC18b] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, Baltimore, MD, 2018. USENIX Association. 47

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, British Columbia, Canada, May 4–6, 1992. ACM Press. 47

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press. 6

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 4, 6, 7, 16, 50

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *ISC 2005*, pages 314–328, 2005. 4, 7

[Mac02] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. 25

[NN01] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd Annual ACM Symposium on Theory of Computing*, pages 590–599, Crete, Greece, July 6–8, 2001. ACM Press. 5, 6, 9, 47

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st Annual ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, GA, USA, May 1–4, 1999. ACM Press. 8, 17, 41

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA 2001*, pages 448–457, 2001. 8

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 77

[OI07] Rafail Ostrovsky and William E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2007. 7

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 121–145, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany. 6

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 72

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 22

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press. 22

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press. 6, 7, 10, 17, 52

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 859–870, 2018. 5, 6, 9

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 22, 23

[SS94] Michael Sipser and Daniel A. Spielman. Expander codes. In *35th Annual Symposium on Foundations of Computer Science*, pages 566–576, Santa Fe, New Mexico, November 20–22, 1994. IEEE Computer Society Press. 24

[Ste98] Julien P. Stern. A new efficient all-or-nothing disclosure of secrets protocol. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371, Beijing, China, October 18–22, 1998. Springer, Heidelberg, Germany. 5, 7

[WW10] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 707–723, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 5

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 4

A Trapdoor Hash for Index Predicates from DCR

In the following we present our TDH construction against the decisional composite residuosity (DCR) assumption over $\mathbb{Z}_{N^2}^*$. Note that the group $\mathbb{Z}_{N^2}^*$ can be rewritten as the product of the subgroup $\mathbb{H} := \{(1+N)^i : i \in [N]\}$, generated by $(1+N)$, and the group of N^{th} residues $\mathbb{NR}_N := \{x^N : x \in \mathbb{Z}_{N^2}^*\}$ of order $\varphi(N)$. We recall the DCR assumption from [Pai99].

Definition A.1 (Decisional Composite Residuosity)**.** *Let N be a uniformly sampled Blum integer and let $\mathbb{NR}_N$ be the multiplicative group of N^{th} residues. Then for any PPT adversary $\mathcal{A}$ it holds that*

$$|\Pr[\mathcal{A}(N,a)=1]-\Pr[\mathcal{A}(N,b)=1]| = \mathsf{negl}(\lambda)$$

where $a \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ and $b \xleftarrow{\$} \mathbb{NR}_N$.

It is useful to recall the following proposition from [BG10].

Proposition A.1 ([BG10])**.** *Let N be a uniformly sampled Blum integer, let $\mathbb{NR}_N$ be the multiplicative group of N^{th} residues, and let $\ell = \ell(\lambda)$ be a polynomial. If the decisional composite residuosity assumption holds then for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ it holds that*

$$\left|\ \Pr[\mathcal{A}_1(\tau, \mathbf{a}\cdot(\mathbf{g})^r)=1 \mid \mathcal{A}_0(N,\mathbf{g})=(\tau,\mathbf{a})] - \Pr[\mathcal{A}_1(\tau,(\mathbf{g})^r)=1 \mid \mathcal{A}_0(N,\mathbf{g})=(\tau,\mathbf{a})]\ \right| = \mathsf{negl}(\lambda)$$

where $a \in \mathbb{H}^\ell$, $\mathbf{g} \xleftarrow{\$} \mathbb{NR}^\ell_N$, and $r \xleftarrow{\$} [N-1]$.

Construction A.1 (Trapdoor hash for $\mathcal{I}^n$ from DCR)**.** *Our DCR-based* TDH *scheme consists of the following algorithms.*

- $\mathsf{S}(1^\lambda, 1^n)$:
  1. *Sample a Blum integer $N := p \cdot q$, where $p = q = 3 \ (mod\ 4)$*
  2. *Sample $g \xleftarrow{\$} \mathbb{NR}_N$*
  3. *Sample a matrix*
$$\mathbf{A} := \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{n,0} \\ g_{1,1}, g_{2,1}, \ldots, g_{n,1} \end{pmatrix} \xleftarrow{\$} \mathbb{NR}^{2\times n}_N$$
  4. *Output*
$$\mathsf{hk} := (N, g, \mathbf{A}) \tag{A.1}$$
- $\mathsf{G}(\mathsf{hk}, f_i)$: *parse* hk *as in Equation A.1 and proceed as follows.*
  1. *Sample $s, t \xleftarrow{\$} [N-1]$.*
  2. *Set*
$$u := g^s$$
  and
$$\mathbf{B} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{n,0} \\ u_{1,1}, u_{2,1}, \ldots, u_{n,1} \end{pmatrix} \qquad \text{where} \qquad u_{j,b} := \begin{cases} g^s_{j,b} \cdot (1+N)^t & \text{if } (j,b) = (i,1) \\ g^s_{j,b} & \text{otherwise} \end{cases}$$
  3. *Output*
$$\mathsf{ek} := (u, \mathbf{B}) \qquad\qquad \mathsf{td} := (s,t) \tag{A.2}$$
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *parse* hk *as in Equation A.1,* $\mathbf{A} = (g_{j,b})_{j\in[n], b\in\{0,1\}}$*, and ρ as $r \in \mathbb{Z}_N$, and output*
$$\mathsf{h} := g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]} \tag{A.3}$$

- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: parse* ek *as* $(u, \mathbf{B})$, $\mathbf{B} = (u_{j,b})_{j\in[n],b\in\{0,1\}}$, *and* ρ *as* $r \in \mathbb{Z}_N$ *and output*

$$\mathsf{e} := u^r \cdot \prod_{j=1}^{n} u_{j,\mathsf{x}[j]} \tag{A.4}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$: *parse* $\mathsf{h} \in \mathbb{G}$ *and* td *as in Equation A.2 and output*

$$\mathsf{e}_0 := \mathsf{h}^s \qquad\qquad \mathsf{e}_1 := \mathsf{h}^s g^t \tag{A.5}$$

Analysis. The full correctness of the construction is trivial. In the following we show that the scheme is stastically input-private and computationally function-private under the DCR assumption.

Theorem A.1 (Input privacy of DCR-based construction)**.** *The* TDH *scheme from Construction A.1 provides statistical input security.*

Proof. Let $r \xleftarrow{\$} [N-1]$ and $\tilde{r} \xleftarrow{\$} \varphi(N)$, then it holds that

$$\left(\mathsf{hk}, \mathsf{h} := g^r \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right) \stackrel{s}{\equiv} \left(\mathsf{hk}, \tilde{\mathsf{h}} := g^{\tilde{r}} \cdot \prod_{j=1}^{n} g_{j,\mathsf{x}[j]}\right)$$

since $(N-1)$ and $\varphi(N)$ are statistically close. Note that $g^{\tilde{r}}$ is a uniformly sampled element of $\mathbb{NR}_N$, thus so is $\tilde{\mathsf{h}}$, for all $\mathsf{x} \in \{0,1\}^n$. This concludes our proof. □

Theorem A.2 (Function privacy of DCR-based construction)**.** *The* TDH *scheme from Construction A.1 provides function privacy under the DCR assumption.*

Proof. Follows from a simple invocation of Proposition A.1. □

From Rate-$1/\lambda$ to Rate-1. Although the rate of our DCR-based construction is linear in λ, we can easily apply the compiler described in Section 4.2.2 (with the distance function defined over $\mathbb{Z}^*_{N^2}$) to achieve rate 1, for a growing n.

B Trapdoor Hash with Reusable Secret Encoding under DDH

Although in our basic DDH-based TDH the encoding $\mathsf{e}_{1-\mathsf{x}[i]}$ distributes uniformly and therefore remains secret, when the same encoding key is reused for multiple inputs one can see that the corresponding $\mathsf{e}_{1-\mathsf{x}[i]}$'s are not jointly uniform, and are in fact, strongly correlated. Thus, the construction does not have reusable secret encoding. In this section, we address the encoding correlation issue, then suggest a method to solve it to obtain DDH-based TDH with reusable secret encoding for index predicates.

B.1 Correlation in the Basic Construction

Recall that for sequence of encodings $(\mathsf{e}_{1,0}, \mathsf{e}_{1,1}), \ldots, (\mathsf{e}_{\ell,0}, \mathsf{e}_{\ell,1})$ computed w.r.t. inputs $\mathsf{x}_1, \ldots, \mathsf{x}_\ell$, it holds that

$$(\mathsf{e}_{1,1-\mathsf{x}_1[i]}, \ldots, \mathsf{e}_{\ell,1-\mathsf{x}_\ell[i]}) = (\mathsf{e}_{1,\mathsf{x}_1[i]} \cdot g^{(1-2\mathsf{x}_1[i])t}, \ldots, \mathsf{e}_{\ell,\mathsf{x}_\ell[i]} \cdot g^{(1-2\mathsf{x}_\ell[i])t})$$

From correctness, an adversary who has $\mathsf{x}_1, \ldots, \mathsf{x}_\ell$, and is given the keys hk and ek, is able to compute the values $\mathsf{e}_{k,\mathsf{x}_k[i]}$ for every $k \in [\ell]$, and therefore every key $\mathsf{e}_{k,1-\mathsf{x}_k[i]}$ in the sequence is the product of a known value and another value which, despite indistinguishable from uniform, depends only on $\mathsf{x}_k[i]$ and, therefore, is used in the computation of any other encoding $\mathsf{e}_{k',1-\mathsf{x}_{k'}[i]}$ for which $\mathsf{x}_{k'}[i] = \mathsf{x}_k[i]$. Thus,

$$((\mathsf{x}_1, \ldots, \mathsf{x}_\ell), \mathsf{hk}, \mathsf{ek}, (\mathsf{e}_{1,1-\mathsf{x}_1[i]}, \ldots, \mathsf{e}_{\ell,1-\mathsf{x}_\ell[i]}), \rho) \stackrel{c}{\equiv} ((\mathsf{x}_1, \ldots, \mathsf{x}_\ell), \mathsf{hk}, \mathsf{ek}, (\alpha_{\mathsf{x}_1[i]} e_1, \ldots, \alpha_{\mathsf{x}_\ell[i]} e_\ell), \rho) \quad \text{(B.1)}$$

where $\rho := \langle \rho_1, \ldots, \rho_\ell \rangle \stackrel{\$}{\leftarrow} \{0,1\}^*$, $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \cdot) \stackrel{\$}{\leftarrow} \mathsf{G}(\mathsf{hk}, f_i)$, $e_k := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho_k)$, and $\alpha_0, \alpha_1 \stackrel{\$}{\leftarrow} \mathbb{G}$.

B.2 Warm-up: Breaking the Correlation with Random Oracle

A simple solution would be to use a random oracle to break the correlation between the encodings. We change Construction 4.1 so that the output of E and, consequently, D is the random oracle image of e and, respectively, e_0 and e_1, that are produced by the algorithms. Clearly, neither correctness nor privacy are affected by this change. By the definition of a random oracle, the images of distinct, however correlated, values are jointly uniform. Thus, we obtain resuable secret encoding.

B.3 Replacing the Random Oracle with CIH

It turns out that the random oracle model is an overkill for our task, and that we can achieve multi-data security using much weaker cryptographic tools. Specifically, we can use *correlated-input secure hash functions*, or CIH [IKNP03, GL10, BC10, GOR11, AMN+18].

What is CIH? At a high level, CIH is a publicly parameterized function $h_{\mathsf{hk}} : X_{\mathsf{hk}} \to Y_{\mathsf{hk}}$, where $h_{\mathsf{hk}}(x_1), \ldots, h_{\mathsf{hk}}(x_\ell)$ look random and independent, even when $x_1, \ldots, x_\ell$ are correlated. Of course, we have to specify which types of correlation we allow. We define a correlation through a set of functions $\mathcal{F}_{\mathsf{hk}} := \{f : X_{\mathsf{hk}} \to Y_{\mathsf{hk}}\}$. Correlated inputs are inputs $x_1, \ldots, x_\ell$ that are generated as $f_1(x), \ldots, f_\ell(x)$ for functions $f_1, \ldots, f_\ell \in \mathcal{F}_{\mathsf{hk}}$ chosen by the adversary, and randomness $x \stackrel{\$}{\leftarrow} X_{\mathsf{hk}}$.

We follow the formalization of CIH given in [AMN+18]. The security definition they give is a close generalization of the correlated-input pseudorandomness notion from [GOR11]. We stress, however, that for our purpose, we need a much weaker security notion, and in particular, we do not require any adaptiveness and we can strict the correlation functions $f_1, \ldots, f_\ell$ to be chosen uniformly rather than adversarially. Clearly, any CIH that satisfies the security definition given below is applicable to our construction.

Definition B.1 (Correlated-Input Secure Hash Function (CIH) [AMN+18])**.** *A* correlated-input secure hash function*, or a* CIH*, is a publicly parametrized function that is described by two PPT algorithms* $\mathcal{H} = (\mathsf{S}, \mathsf{H})$ *with the following properties.*

- ***Syntax:***
  - $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda)$. *The sampling algorithm takes as input the security parameter* λ *and outputs a hash key* hk *that defines a domain* X_{hk} *and a range* Y_{hk} *where* $|X_{\mathsf{hk}}|, |Y_{\mathsf{hk}}| = \Omega(2^\lambda)$.
  - $\mathsf{y} \leftarrow \mathsf{H}(\mathsf{hk}, \mathsf{x})$. *The hashing algorithm takes as input a hash key* hk *and an input* $\mathsf{x} \in X_{\mathsf{hk}}$ *and deterministically outputs an image* $\mathsf{y} := h_{\mathsf{hk}}(x) \in Y_{\mathsf{hk}}$.
- ***Correlated-Input Security:*** *let* $\mathcal{F} := \{\mathcal{F}_{\lambda,\mathsf{hk}}\}_{\lambda\in\mathbb{N},\mathsf{hk}\in\{0,1\}^*}$ *be a class of functions, where each* $\mathcal{F}_{\lambda,\mathsf{hk}}$ *is a set of functions parametrized by* λ *and* hk*, and for every* $\lambda \in \mathbb{N}$*, if* $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda)$*, then functions in* $\mathcal{F}_{\lambda,\mathsf{hk}}$ *have domain and range* X_{hk}*. For a* CIH*,* $\mathcal{H} := (\mathsf{S}, \mathsf{H})$*, such a class of functions* $\mathcal{F}$*, and an adversary* $\mathcal{A}$*, we define the experiment* $\mathsf{Expt}_{\mathcal{H},\mathcal{F},\mathcal{A}}(\lambda)$ *as follows:*

  1. $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda)$
  2. $b \xleftarrow{\$} \{0,1\}$
  3. $\mathsf{RF}(\cdot) \xleftarrow{\$} \{f : X_{\mathsf{hk}} \to X_{\mathsf{hk}}\}$
  4. $\alpha \xleftarrow{\$} X_{\mathsf{hk}}$
  5. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RF},\alpha}(b,\cdot)}(\mathsf{hk})$
  6. *Output 1 if and only if* $b = b'$.

  where $\mathcal{O}_{\mathsf{RF},\alpha}(b \in \{0,1\}, f \in \mathcal{F}_{\lambda,\mathsf{pp}})$ *outputs* $\mathsf{H}(\mathsf{pp}, f(\alpha))$ *if* $b = 1$*, and* $\mathsf{RF}(f(\alpha))$ *if* $b = 0$.

  We say that $\mathcal{H}$ *provides correlated-input security w.r.t.* $\mathcal{F}$ *if, for all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathsf{Expt}_{\mathcal{H},\mathcal{F},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)$$

Solution using CIH: Overview. Using a CIH that provides security w.r.t. inputs correlated as $\mathsf{e}_{1,1-\mathsf{x}_1[i]}, \ldots, \mathsf{e}_{\ell,1-\mathsf{x}_\ell[i]}$ in Equation B.1, we can replace the random oracle in the warm-up construction above, and achieve reusable secret encoding. Following the formalization from Definition B.1, what we seek is a CIH where the hash key hk consist of a description of a DDH-hard group $\mathbb{G}$ of prime order p, and $h_{\mathsf{hk}} : \mathbb{G} \to Y_{\mathsf{hk}}$ for every such hk. We want the scheme to provide security w.r.t. the correlation from Equation B.1, which can be defined by a class $\mathcal{F} := \{\mathcal{F}_{\lambda,\mathsf{hk}}\}$, where we set $\mathcal{F}_{\mathbb{G}} := \{f_\alpha(x) = \alpha x \mid \alpha \in \mathbb{G}\}$ (notice that, despite in Equation B.1 there are two α values, they are independent and uncorrelated, therefore it suffices to break the correlation between values sharing the same α).

Fortunately, although we are not aware of a DDH-based CIH with these exact specifications, Attrapadung et al. [AMN+18] construct a CIH scheme that work with slightly different settings and that we can use over our TDH to achieve reusable secret encoding. We recall their construction below.

Construction B.1 (CIH construction from Section 3.2 in [AMN+18])**.** *The* CIH *construction from [AMN+18] consists of the following algorithms:*

- $\mathsf{S}(1^\lambda)$: *sample a cyclic group* $\mathbb{H}$ *of prime order* $q = \Omega(2^\lambda)$*, such that* $q = 2p+1$ *for a prime* p*, and output* $\mathsf{hk} := (\mathbb{H}, q)$*, that defines the following domain and range for* h_{hk}*:*

$$X_{\mathsf{hk}} := \mathbb{QR}_q^{m+1} \qquad\qquad Y_{\mathsf{hk}} := \mathbb{H}$$

where $\mathbb{QR}_q$ *is the group of quadratic residues in* $\mathbb{Z}_q^*$ *and* $m = \mathsf{poly}(\lambda)$*.*

- $\mathsf{H}(\mathsf{hk}, \mathsf{x})$: *output*

$$h_{\mathsf{hk}}(\mathsf{x}) := \mathsf{NR}(\mathsf{x}, 11 || \mathsf{H}(\mathsf{NR}(\mathsf{x}, \mathbf{e}_0), \ldots, \mathsf{NR}(\mathsf{x}, \mathbf{e}_m)))$$

where $\mathbf{e}_0 = 0^m$*,* $\mathbf{e}_j$ *is the* j^{th} *m-bit unit vector,* $\mathsf{H} : \mathbb{H}^{m+1} \to \{0,1\}^{m-2}$ *is a collision-resistant hash function (CRHF), and* $\mathsf{NR} : (\mathbb{Z}_q^*)^{n+1} \times \{0,1\}^n \to \mathbb{H}$ *is the Noar-Reingold PRF [NR97] defined as*

$$\mathsf{NR}(x, y) := h^{(x_0 \prod_{i \in [m]} x_j^{y_j})}$$

for a generator $h \in \mathbb{H}$*.*

Notice that when q is chosen as described above, then $\mathbb{QR}_q$ is a prime-order group. This allows us to base the security of the construction on the DDH assumption as follows.

Proposition B.1. *Under the DDH assumption over* $\mathbb{QR}_q$*, and assuming* H *is a CRHF, the* CIH *scheme from Construction B.1 provides correlated-input security w.r.t.* $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{hk}}\}$*, where*

$$\mathcal{F}_{\lambda,\mathsf{hk}} := \{f_{\mathbf{z}}(\alpha) = \alpha \circ \mathbf{z} \mid \mathbf{z} \in \mathbb{QR}_q^{m+1}\}$$

and $\circ$ *is the Hadamard product (element-by-element multiplication).*

One issue still has to be resolved in order to use the above CIH for our DDH-based TDH: while the enocdings generated by the TDH scheme lay in a prime-order group $\mathbb{G}$, the CIH can be used over inputs that are in $\mathbb{QR}_q^{m+1}$. We solve this by repeating the execution of the TDH independently $m+1$ times, to obtain a vector of $m+1$ independent encodings, rather than a single encoding, for every input. If we choose $\mathbb{G} := \mathbb{QR}_q$ to be the DDH-hard group underlying the TDH, then we obtain correlated encodings that perfectly match the CIH domain and correlation class $\mathcal{F}$.

The Scheme. We formalize the described construction as follows, then proof its security.

Construction B.2 (DDH-based TDH with reusable secret encoding)**.** *Let* $(\mathsf{S}', \mathsf{G}', \mathsf{H}', \mathsf{E}', \mathsf{D}')$ *be the* TDH *scheme from Construction 4.1, and let* $(\mathsf{S}_{\mathsf{CIH}}, \mathsf{H}_{\mathsf{CIH}})$ *be the* CIH *from Construction B.1. Our DDH-based* TDH *scheme for index predicates with reusable secret encoding consists of the following algorithms:*

- $\mathsf{S}(1^\lambda, 1^n)$: *sample a cyclic group* $\mathbb{H}$ *of prime order* $q = \Omega(2^\lambda)$*, such that* $q = 2p+1$ *for a prime* p*, set* $\mathbb{G} := \mathbb{QR}_q$*, and proceed as in* S' *with* $(\mathbb{G}, p, g \xleftarrow{\$} \mathbb{G})$ *to obtain* hk'*. Also define* $\mathsf{hk}_{\mathsf{CIH}} := (\mathbb{H}, q)$*, and output* $\mathsf{hk} := (\mathsf{hk}', \mathsf{hk}_{\mathsf{CIH}})$*.*
- $\mathsf{G}(\mathsf{hk}, f_i)$: *output* $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}'(\mathsf{hk}', f_i)$*, where* G' *denotes* $m+1$ *independent executions of* G'*.*
- $\mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho)$: *output* $\mathsf{h} := \mathsf{H}'(\mathsf{hk}', \mathsf{x}; \rho)$*.*
- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$:

1. *Compute* $\mathsf{e} \leftarrow \mathsf{E}'(\mathsf{ek}, \mathsf{x})$ *where* E' *denotes* $m+1$ *executions of* E'.
2. *Output*

$$\mathsf{e} := \mathsf{H}_{\mathsf{CIH}}(\mathsf{hk}_{\mathsf{CIH}}, \mathsf{e})$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$*:*
  1. *Compute* $(\mathsf{e}_0, \mathsf{e}_1) := \mathsf{D}'(\mathsf{td}', \mathsf{h})$*, where* D' *denotes* $m+1$ *executions of* D'.
  2. *Output*

$$\mathsf{e}_0 := \mathsf{H}_{\mathsf{CIH}}(\mathsf{hk}_{\mathsf{CIH}}, \mathsf{e}_0) \qquad \mathsf{e}_1 := \mathsf{H}_{\mathsf{CIH}}(\mathsf{hk}_{\mathsf{CIH}}, \mathsf{e}_1)$$

Analysis. Correctness, statistical input-privacy and function-privacy are easily implied from the corresponding properties of the underlying TDH from Construction 4.1. We now show that the scheme also satisfies the reusable secret encoding property.

Theorem B.1. *The* TDH *scheme from Construction B.2 has reusable secret encoding under the DDH assumption.*

Proof. We prove the theorem using reduction to the security of the underlying CIH scheme. But first, we show, assuming DDH, that the correlation from Equation B.1 holds for α_0, α_1 that are indistinguishable from uniform. We notice that there are two independent sets of correlated encodings: encodings $\mathsf{e}_{k,1-\mathsf{x}_k[i]}$ for which $\mathsf{x}_k[i] = 0$, and those for which $\mathsf{x}_k[i] = 1$. We assume, w.l.o.g. that the first set corresponds to $k = 1, \ldots, \ell'$, and the second to $k = \ell' + 1, \ldots, \ell$. Thus, we use the following hybrid distributions to prove privacy:

- $\mathsf{Hybrid}_0 := (\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \mathsf{hk}, \mathsf{ek}, \langle \mathsf{e}_{1,1-\mathsf{x}_1[i]}, \ldots, \mathsf{e}_{\ell,1-\mathsf{x}_\ell[i]} \rangle, \rho)$, where $\rho := \langle \rho_1, \ldots, \rho_\ell \rangle \xleftarrow{\$} \{0,1\}^*$ $\mathsf{hk} \leftarrow \mathsf{S}(1^\lambda, 1^n)$, $\mathsf{h}_k := \mathsf{H}(\mathsf{hk}, \mathsf{x}_k; \rho_k)$ for $k = 1, \ldots, \ell$, $(\mathsf{ek}, \mathsf{td}) := \mathsf{G}(\mathsf{hk}, f_i)$, and $(\mathbf{e_0}, \mathbf{e_1}) := \mathsf{D}(\mathsf{td}, \mathsf{h})$.
- $\mathsf{Hybrid}_1 := (\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \mathsf{pp}, \mathsf{ct}, \langle \mathsf{k}'_1, \ldots, \mathsf{k}'_\ell \rangle, \rho)$, where ρ, hk and $\mathsf{ek} := \langle \mathsf{ek}^{(1)}, \ldots, \mathsf{ek}^{(m+1)} \rangle$ are defined as above, and, for $k = 1, \ldots, \ell$, $\mathsf{e}'_k := h_{\mathsf{hk}_{\mathsf{CIH}}}(\alpha_{\mathsf{x}_k[i]} \circ \mathbf{e}_k)$, where $\alpha_0, \alpha_1 \xleftarrow{\$} \mathbb{G}^{m+1}$and $\mathbf{e}_k := \mathsf{E}'(\mathsf{ek}, \mathsf{x}; \rho)$.
- $\mathsf{Hybrid}_2 := (\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \mathsf{hk}, \mathsf{ek}, \langle \mathsf{e}''_1, \ldots, \mathsf{e}''_{\ell'}, \mathsf{e}'_{\ell'+1}, \ldots, \mathsf{e}'_\ell \rangle, \rho)$, where $\rho, \mathsf{hk}, \mathsf{ek}$ and e'_k, for any k, are as defined above, and $\mathsf{e}''_k \xleftarrow{\$} \mathbb{H}$ $k = 1, \ldots, \ell'$.
- $\mathsf{Hybrid}_3 := (\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \mathsf{hk}, \mathsf{ek}, \langle \mathsf{e}''_1, \ldots, \mathsf{e}''_\ell \rangle, \rho)$, where ρ, hk and ek are as defined above, and $\mathsf{e}''_k \xleftarrow{\$} \mathbb{H}$ for $k = 1, \ldots, \ell$.

From definition, to show security it suffices to show that every two consequent hybrids of the above are computationally indistinguishable, and therefore $\mathsf{Hybrid}_0 \overset{c}{\equiv} \mathsf{Hybrid}_3$.

For showing $\mathsf{Hybrid}_0 \overset{c}{\equiv} \mathsf{Hybrid}_1$, we can follow the steps taken in the proof of Theorem 7.2, only here we apply the reduction $m+1$ independent times. We skip the details for brevity.

We now prove that $\mathsf{Hybrid}_1 \overset{c}{\equiv} \mathsf{Hybrid}_2$. Showing that $\mathsf{Hybrid}_2 \overset{c}{\equiv} \mathsf{Hybrid}_3$ is identical up to technicalities. Let $\mathcal{D}$ be a PPT distinguisher that distinguishes between Hybrid_1 and Hybrid_2,

$$| \Pr[\mathcal{D}(\mathsf{Hybrid}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Hybrid}_2) = 1] | > \mathsf{negl}(\lambda)$$

We construct an adversary $\mathcal{A}$ that uses $\mathcal{D}$ to win the security experiment $\mathsf{Expt}_{\mathcal{H},\mathcal{F},\mathcal{A}}$ from Definition B.1, where $\mathcal{F}$ is the correlation class from Proposition B.1. $\mathcal{A}$, which has an oracle access to $\mathcal{O} : \mathcal{F}_{\lambda,\mathsf{hk}_{\mathsf{CIH}}} \to Y_{\mathsf{hk}_{\mathsf{CIH}}}$, takes as input public parameters $\mathsf{hk}_{\mathsf{CIH}}$ and proceeds as follows:

1. Parse $\mathsf{hk}_{\mathsf{CIH}}$ as $(\mathbb{H}, q)$, set $\mathbb{G} := \mathbb{QR}_q$, and proceed as in G' with $(\mathbb{G}, p, g \xleftarrow{\$} \mathbb{G})$ to obtain $\widetilde{\mathsf{hk}}'$. Set $\widetilde{\mathsf{hk}} := (\widetilde{\mathsf{hk}}', \mathsf{hk}_{\mathsf{CIH}})$ and $(\widetilde{\mathsf{ek}}, \cdot) := \mathsf{G}(\widetilde{\mathsf{hk}}, f_i)$.

2. Sample $\rho := \langle \rho_1, \ldots, \rho_\ell \rangle \xleftarrow{\$} \{0,1\}^*$, and, for $k \in [\ell]$, set $\tilde{\mathsf{h}}_k := \mathsf{H}(\tilde{\mathsf{hk}}, \mathsf{x}_k; \rho_k)$ and
$$\mathbf{e}_k := \mathsf{E}'(\widetilde{\mathsf{ek}}, \mathsf{x}_k; \rho_k)$$

3. For $k = 1, \ldots, \ell'$,
$$\tilde{\mathsf{e}}_k \leftarrow \mathcal{O}(f_k)$$
where $f_k(\alpha) := \alpha \circ \mathbf{e}_k$.

4. For $k = \ell' + 1, \ldots, \ell$,
$$\tilde{\mathsf{e}}_k := \mathsf{H}_{\mathsf{CIH}}(\mathsf{hk}_{\mathsf{CIH}}, \alpha_{\mathbf{1}} \circ \mathbf{e}_k)$$
where $\alpha_{\mathbf{1}} \xleftarrow{\$} \mathbb{G}^{m+1}$.

5. Output $\mathcal{D}(\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \widetilde{\mathsf{hk}}, \tilde{\mathsf{ek}}, \langle \tilde{\mathsf{e}}_1, \ldots, \tilde{\mathsf{e}}_\ell \rangle, \rho)$.

Using simple calculations, one can verify that when $\mathcal{O}(f) := \mathsf{H}_{\mathsf{CIH}}(\mathsf{hk}_{\mathsf{CIH}}, f(\alpha_{\mathbf{0}}))$ for a uniform $\alpha_{\mathbf{0}} \in \mathbb{G}^{m+1}$, then $(\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \widetilde{\mathsf{hk}}, \tilde{\mathsf{ek}}, \langle \tilde{\mathsf{e}}_1, \ldots, \tilde{\mathsf{e}}_\ell \rangle, \rho) \stackrel{c}{\equiv} \mathsf{Hybrid}_1$, and that when $\mathcal{O}(f) := \mathsf{RF}(f(\alpha))$ for a random function RF, then $(\langle \mathsf{x}_1, \ldots, \mathsf{x}_\ell \rangle, \widetilde{\mathsf{hk}}, \tilde{\mathsf{ek}}, \langle \tilde{\mathsf{e}}_1, \ldots, \tilde{\mathsf{e}}_\ell \rangle, \rho) \stackrel{c}{\equiv} \mathsf{Hybrid}_2$. Hence, $\mathcal{A}$ wins in the security experiment with non-negligible probability. □

C Sublinear Trapdoor Hash from Pairings

The bottleneck in the efficiency of our DDH construction lays in the size of the underlying TDH keys (hk and ek), which is linear in n. In this section, we show how to use pairings over prime-order groups in order to reduce their size to $O(\mathsf{poly}(\lambda)\sqrt{n})$.

We begin with recalling the definition of bilinear groups and the related SXDH assumption, and then proceed to presenting the construction.

SXDH over Prime-Order Groups with Pairings. The SXDH assumption is the analog of DDH over two prime-order groups, $\mathbb{G}$ and $\mathbb{H}$, that are equipped with a bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_t$, and generated using what is called a *bilinear group generator*. We use such pairings to obtain useful functionality and build efficient schemes with security based on SXDH.

We first define a bilinear group generator and the special case for prime-order groups. We then formalize the SXDH assumption over such structures.

Definition C.1 (Bilinear group generator)**.** *A* bilinear group generator *is an algorithm $\mathcal{G}$ that takes as input a security parameter 1^λ, and outputs $(\mathbb{G}, \mathbb{H}, \mathbb{G}_t, e)$, where $\mathbb{G}, \mathbb{H}, \mathbb{G}_t$ are descriptions of three abelian groups and e is an efficiently computable bilinear and nondegenerate map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_t$.*

Definition C.2 (Prime-order bilinear group generator)**.** *We say that a bilinear group generator $\mathcal{P}$ is* prime-order *if $\mathbb{G}$ and $\mathbb{H}$ both have prime order $p > 2^\lambda$.*

Definition C.3 (The SXDH assumption in prime-order bilinear groups)**.** *We say that a prime-order bilinear group generator $\mathcal{P}$* satisfies the DDH assumption in $\mathbb{G}$ *(resp. $\mathbb{H}$) if for any PPT adversary, $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}((p,\mathbb{G},\mathbb{H},\mathbb{G}_t,e),(g^{a_1},g^{a_2},g^{a_1a_2}))=1]-\Pr[\mathcal{A}((p,\mathbb{G},\mathbb{H},\mathbb{G}_t,e),(g^{a_1},g^{a_2},g^{a_3}))=1]|=\mathsf{negl}(\lambda)$$

where $(p,\mathbb{G},\mathbb{H},\mathbb{G}_t,e)\xleftarrow{\$}\mathcal{P}$, $g\xleftarrow{\$}\mathbb{G}$ (resp. $g\xleftarrow{\$}\mathbb{H}$) and $a_1,a_2,a_3\xleftarrow{\$}\mathbb{Z}_p$.

We say that $\mathcal{P}$ satisfies the symmetric external Diffie-Hellman assumption*, if it satisfies the DDH assumption both in $\mathbb{G}$ and in $\mathbb{H}$.*

Overview. The overall idea is as follows. We replace every row $b\in\{0,1\}$, in the $2\times n$ matrix in hk with $2\sqrt{n}$ group elements: $g_{1,b},\ldots,g_{\sqrt{n},b}\in\mathbb{G}$ and $h_{1,b},\ldots,h_{\sqrt{n},b}\in\mathbb{H}$, for two composite-order groups $\mathbb{G}$ and $\mathbb{H}$. Now, if we represent every index $j\in[n]$ as a pair $(j_1,j_2)\in[\sqrt{n}]^2$, then given such elements, and using a pairing $e:\mathbb{G}\times\mathbb{H}\to\hat{\mathbb{G}}$, one can generate a matrix $\mathbf{A}:=\binom{\hat{g}_{1,0},\hat{g}_{2,0},\ldots,\hat{g}_{n,0}}{\hat{g}_{1,1},\hat{g}_{2,1},\ldots,\hat{g}_{n,1}}$, where we define $\hat{g}_{(j_1,j_2),b}:=e(g_{j_1,b},h_{j_2,b})$. Using such a matrix, we can compute hash values as described in the hashing algorithm of Construction 4.1 above. It remains to show how to generate a corresponding short encoding key ek that would allow to compute encodings that give us both correctness and privacy.

We follow the approach from the above construction, and design an ek, using which we eventually generate a matrix $\mathbf{B}:=\binom{\hat{u}_{1,0},\hat{u}_{2,0},\ldots,\hat{u}_{n,0}}{\hat{u}_{1,1},\hat{u}_{2,1},\ldots,\hat{u}_{n,1}}$, where, roughly speaking, for every $j\in[n]$ and $b\in\{0,1\}$, $\hat{u}_{j,b}=\hat{g}^s_{j,b}$ for a random exponent s, except for $\hat{u}_{i,1}$ which is also multiplied by a random element. As a first attempt, let us define ek to contain $u_{1,b}:=g^{s_1}_{1,b},\ldots,u_{\sqrt{n},b}:=g^{s_1}_{\sqrt{n},b}$ and $v_{1,b}:=h^{s_2}_{1,b},\ldots,v_{\sqrt{n},b}:=h^{s_2}_{\sqrt{n},b}$ for random s_1 and s_2. Consequently, we get $u_{(j_1,j_2),b}:=e(u_{j_1,b},v_{j_2,b})=\hat{g}^{s_1s_2}_{(j_1,j_2),b}$. This is almost what we want, as we still need to "puncture" $u_{i,1}$, and replace it with a random value.

To implement such a functionality, we construct a pairing e that has a special property. More specifically, $\mathbb{G}$, and respectively, $\mathbb{H}$, are each generated as the product of three subgroups, $\mathbb{G}_1\times\mathbb{G}_2\times\mathbb{G}_3=\mathbb{G}$, and resp. $\mathbb{H}_1\times\mathbb{H}_2\times\mathbb{H}_3=\mathbb{H}$, such that for every $i\neq j$, $\mathbb{G}_i$ and $\mathbb{H}_j$ are orthogonal to w.r.t. e, i.e. $e(g,h)=1$ for any $g\in\mathbb{G}_i$ and $h\in\mathbb{H}_j$. Now, for every $j_1,j_2\in[\sqrt{n}]$, we set $u_{j_1,b}:=g^{s_1}_{j_1,b}\gamma_{j_1,b}$ and $v_{j_2,b}:=h^{s_2}_{j_2,b}\delta_{j_2,b}$, where the $\gamma_{j_1,b}$'s are chosen randomly from $\mathbb{G}_1$, and the $\delta_{j_2,b}$'s from $\mathbb{H}_2$. The only exception are $\gamma_{i_1,1}$ and $\delta_{i_2,1}$ which are taken from $\mathbb{G}_3$ and $\mathbb{H}_3$ (resp.).

We get correctness from the fact that, for any $(j,b)\neq(i,1)$, we get $u_{j,b}=\hat{g}^{s_1s_2}_{j,b}e(\gamma_{j_1,b},\delta_{j_2,b})=\hat{g}^{s_1s_2}_{j,b}$ since $\gamma_{j_1,b}$ and $\delta_{j_2,b}$ are sampled from orthogonal subgroups, whereas $e(\gamma_{i_1,1},\delta_{i_2,1})$ does not vanish in $u_{i,1}=\hat{g}^{s_1s_2}_{i,1}e(\gamma_{i_1,1},\delta_{i_2,1})$ since $\mathbb{G}_3$ and $\mathbb{H}_3$ are not orthogonal. To obtain function privacy, we show that the elements $\{g^{s_1}_{j,b}\}$ and $\{h^{s_2}_{j,b}\}$ provide "random masks" and randomize the values in ek which become indistinguishable form uniform. Although $\mathbb{G}$ and $\mathbb{H}$ are composite groups, we are able to derive such a DDH-like argument by defining $\mathbb{G}$ and $\mathbb{H}$ as vector spaces over underlying prime-order groups, G and H, where we assume SXDH.

Below, we give the details of the bilinear group generators that we use, and then proceed to a technical description of our construction.

The Underlying Pairing. Let G and H be two groups of prime order p, equipped with a pairing $e_{\mathcal{P}}:G\times H\to\hat{G}$. We observe that we can define a pairing that maps *vector spaces* of G and H,

$\mathbb{G} := G^m$ and $\mathbb{H} := H^m$ resp., to a corresponding vector space of $\hat{G}$, $\hat{\mathbb{G}} := \mathbb{G}^m$. We chose to work with vector spaces for a reason: this allows us to decompose each of the vector spaces to "parallel", i.e. linearly independent, subgroups. Consequently, we can create such a subgroup-decomposition of $\mathbb{G}$ and $\mathbb{H}$, such that every subgroup in $\mathbb{G}$ is orthogonal to all but one of the subgroups of $\mathbb{H}$, and vice versa. This essentialy enables us to implement the "cancelling" functionality, described above, that we use to obtain correctness, with 3-dimensional vector spaces. Full details are given below.

Construction C.1 (Example 3.7 in [Fre10])**.** *Let $\mathcal{P}$ be a prime-order bilinear group generator that, on input 1^λ, outputs $(p, G, H, \hat{G}, e_\mathcal{P})$ (see Definition C.2). We hereby construct a bilinear group generator algorithm $\mathcal{G}_3$, and an associated subgroup generator $\mathcal{G}_3^{\mathsf{sub}}$ as follows:*

- $\mathcal{G}_3(1^\lambda)$:
  1. *Sample* $(p, G, g, H, h, \hat{G}, e_\mathcal{P}) \xleftarrow{\$} \mathcal{P}(1^\lambda)$.
  2. *Set*
  $$\mathbb{G} := G^3 \qquad \mathbb{H} := H^3 \qquad \hat{\mathbb{G}} := \hat{G}^3$$
  3. *Define pairing* $e : \mathbb{G} \times \mathbb{H} \to \hat{\mathbb{G}}$, *such that*
  $$e((g_1, g_2, g_3), (h_1, h_2, h_3)) = e_\mathcal{P}(g_1, h_1) e_\mathcal{P}(g_2, h_2) e_\mathcal{P}(g_3, h_3) \tag{C.1}$$
  4. *Output* $(\mathbb{G}, g, \mathbb{H}, h, \hat{\mathbb{G}}, e)$.
- $\mathcal{G}_3^{\mathsf{sub}}(\mathbb{G}, g, \mathbb{H}, h)$*:*
  1. *Sample* $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \xleftarrow{\$} \mathbb{Z}_p$ *such that*
  $$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \begin{pmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ z'_1 & z'_2 & z'_3 \end{pmatrix} = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix} \tag{C.2}$$
  for $a_1, a_2, a_3 \neq 0$.
  2. *Set*
  $$\mathbb{G}_1 := \langle g^{(x_1, y_1, z_1)} \rangle \qquad \mathbb{G}_2 := \langle g^{(x_2, y_2, z_2)} \rangle \qquad \mathbb{G}_3 := \langle g^{(x_3, y_3, z_3)} \rangle$$
  $$\mathbb{H}_1 := \langle h^{(x'_1, y'_1, z'_1)} \rangle \qquad \mathbb{H}_2 := \langle h^{(x'_2, y'_2, z'_2)} \rangle \qquad \mathbb{H}_3 := \langle h^{(x'_3, y'_3, z'_3)} \rangle$$
  3. *Output* $((\mathbb{G}_j, g^{(x_j, y_j, z_j)})_{1 \leq j \leq 3}, (\mathbb{H}_j, h^{x'_j, y'_j, z'_j})_{1 \leq j \leq 3})$

Another property we achieve from building over prime-order groups is the ability to assume DDH over G and H (namely, SXDH w.r.t. $e_\mathcal{P}$), and derive a related hardness assumption over the composite-groups $\mathbb{G}$ and $\mathbb{H}$. More specifically, in Proposition C.1, we transform the SXDH assumption to its "vector variant". We skip the proof of the proposition as it is simple and straightforward, and essentially relies on the fact that, if it is hard to solve the DDH problem (or any computational problem for the matter), then it is hard to jointly solve multiple independent instances of DDH.

Proposition C.1. *Let $\mathcal{P}$ be a prime-order bilinear group generator $\mathcal{P}$ that satisfies the SXDH assumption. Then, for any PPT adversary, $\mathcal{A}$, it holds that*

$$\begin{aligned} &|\Pr[\mathcal{A}((p,G,H,\hat{G},e_{\mathcal{P}}),(g^{a_1},g^{a_2},g^{a_1a_2}),(h^{b_1},h^{b_2},h^{b_1b_2}))=1] \\ &-\Pr[\mathcal{A}((p,G,H,\hat{G},e_{\mathcal{P}}),(g^{a_1},g^{a_2},g^{a_3}),(h^{b_1},h^{b_2},h^{b_3}))=1]\,|=\mathsf{negl}(\lambda) \end{aligned} \tag{C.3}$$

where $(p,G,H,\hat{G},e_{\mathcal{P}}) \xleftarrow{\$} \mathcal{P}$, $g \xleftarrow{\$} G^3$, $h \xleftarrow{\$} H^3$, and $a_1,a_2,a_3,b_1,b_2,b_3 \xleftarrow{\$} \mathbb{Z}_p$.

The Scheme. We now give the details of our pairing-based TDH construction, then prove it satisfies the required properties.

Construction C.2 (Sublinear TDH from Pairings)**.** *Let $\mathcal{G}_3$ and $\mathcal{G}_3^{\mathsf{sub}}$ be the algorithms from Construction C.1. Our pairings-based* TDH *scheme consists of the following algorithms:*

- $\mathsf{S}(1^\lambda,1^n)$*:*
  1. *Sample $(\mathbb{G},g,\mathbb{H},h,\hat{\mathbb{G}},e) \xleftarrow{\$} \mathcal{G}_3(1^\lambda)$*
  2. *Sample two matrices*

$$\mathbf{A_1} := \begin{pmatrix} g_{1,0}, g_{2,0}, \ldots, g_{\sqrt{n},0} \\ g_{1,1}, g_{2,1}, \ldots, g_{\sqrt{n},1} \end{pmatrix} \xleftarrow{\$} \mathbb{G}^{2\times\sqrt{n}} \qquad \mathbf{A_2} := \begin{pmatrix} h_{1,0}, h_{2,0}, \ldots, h_{\sqrt{n},0} \\ h_{1,1}, h_{2,1}, \ldots, h_{\sqrt{n},1} \end{pmatrix} \xleftarrow{\$} \mathbb{H}^{2\times\sqrt{n}}$$

  3. *Output*

$$\mathsf{hk} := \Big((\mathbb{G},g,\mathbb{H},h,\hat{\mathbb{G}},e),\mathbf{A_1},\mathbf{A_2}\Big) \tag{C.4}$$

- $\mathsf{G}(\mathsf{hk},f_{[i]})$*: parse* hk *as in Equation C.4, i as $(i_1,i_2) \in [\sqrt{n}]^2$ and proceed as follows.*
  1. *Generate subgroups $((\mathbb{G}_j,g_j)_{1\le j\le 3},(\mathbb{H}_j,h_j)_{1\le j\le 3}) \xleftarrow{\$} \mathcal{G}_3^{\mathsf{sub}}(\mathbb{G},\mathbb{H})$.*
  2. *Sample $s_1,s_2,t_1,t_2 \xleftarrow{\$} \mathbb{Z}_p$ and set*

$$\begin{aligned} &\gamma_{i_1,1} := g_3^{t_1} \in \mathbb{G}_3 && \gamma_{j,b} \xleftarrow{\$} \mathbb{G}_1 \quad \text{for any } (j,b)\neq(i_1,1) \\ &\delta_{i_2,1} := h_3^{t_2} \in \mathbb{H}_3 && \delta_{j,b} \xleftarrow{\$} \mathbb{H}_2 \quad \text{for any } (j,b)\neq(i_2,1) \end{aligned} \tag{C.5}$$

  3. *Set*

$$u := g^{s_1} \qquad\qquad v := h^{s_2}$$

  and

$$\mathbf{B_1} := \begin{pmatrix} u_{1,0}, u_{2,0}, \ldots, u_{\sqrt{n},0} \\ u_{1,1}, u_{2,1}, \ldots, u_{\sqrt{n},1} \end{pmatrix} \quad \text{where} \quad u_{j,b} := g^{s_1}\cdot\gamma_{j,b}$$

$$\mathbf{B_2} := \begin{pmatrix} v_{1,0}, v_{2,0}, \ldots, v_{\sqrt{n},0} \\ v_{1,1}, v_{2,1}, \ldots, v_{\sqrt{n},1} \end{pmatrix} \quad \text{where} \quad v_{j,b} := h_{j,b}^{s_2}\cdot\delta_{j,b}$$

  4. *Output*

$$\mathsf{ek} := (u,\mathbf{B_1},v,\mathbf{B_2}) \qquad\qquad \mathsf{td} := (s_1,s_2,t_1,t_2,g_3,h_3) \tag{C.6}$$

- $\mathsf{H}(\mathsf{pp}, \mathsf{x}; \rho)$*: parse* hk *as in Equation C.4 and* ρ *as* $r_1, r_2 \in \mathbb{Z}_p$*, and output*

$$\mathsf{h} := e(g^{r_1}, h^{r_2}) \prod_{j=1}^{n} e(g_{j_1,\mathsf{x}[j]}, h_{j_2,\mathsf{x}[j]}) \tag{C.7}$$

- $\mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho)$*: parse* ek *as in Equation C.6 and* ρ *as* $r_1, r_2 \in \mathbb{Z}_p$*, and output*

$$\mathsf{e} := e(u^{r_1}, v^{r_2}) \prod_{j=1}^{n} e(u_{j_1,\mathsf{x}[j]}, v_{j_2,\mathsf{x}[j]}) \tag{C.8}$$

- $\mathsf{D}(\mathsf{td}, \mathsf{h})$*: parse* $\mathsf{h} \in \hat{\mathbb{G}}$ *and* td *as in Equation C.6, and output*

$$\mathsf{e}_0 := \mathsf{h}^{s_1 s_2} \qquad\qquad \mathsf{e}_1 := \mathsf{h}^{s_1 s_2} e(g_3^{t_1}, g_3^{t_2}) \tag{C.9}$$

Analysis. The analysis of the pairing-based TDH is ver similar to that of the DDH-based scheme, only here, a more subtle treatment is needed, as the construction is a bit more evolved.

Correctness can be verified by inspection, based on Equation C.2, and input privacy is immediate. To prove both function privacy and secret encoding, we find it useful to first prove Lemma C.1 below, which basically states, that the encoding key ek (Equation C.6) is indistinguishable from uniform, given the hash key hk. We even consider the joint distribution of such an encoding key, attached to an encoding $\mathsf{e}_{1-f_{[i]}(\mathsf{x})}$ and that corresponds to the values in ek.

For simplicity of notation, we associate every such ek distribution with a corresponding distribution of $\mathsf{e}_{1-f_{[i]}(\mathsf{x})}$, as follows. For any $\mathsf{x} \in \{0,1\}^n$, any hk as in Equation C.4, any $\mathsf{ek} = (u, \mathbf{B_1}, v, \mathbf{B_2})$ where $u \in \mathbb{G}$, $\mathbf{B_1} \in \mathbb{G}^{2\times\sqrt{n}}$, $v \in \mathbb{H}$, and $\mathbf{B_2} \in \mathbb{H}^{2\times\sqrt{n}}$, any $\gamma \in \mathbb{G}$ and $\delta \in \mathbb{H}$, and any $\rho \in \{0,1\}^*$, we define the following corresponding encoding

$$\mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \gamma, \delta; \rho) := \mathsf{E}(\mathsf{ek}, \mathsf{x}; \rho) e(\gamma, \delta)^{1-2\mathsf{x}[i]}$$

where e is taken from hk.

We make two simple observations regarding $\mathsf{e}(\cdot)$:

(i) If $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_{[i]})$, and $\gamma := \gamma_{i_1,1}$ and $\delta := \delta_{i_2,1}$ are as generated by $\mathsf{G}(\mathsf{hk}, f_{[i]})$ (Equation C.5) in the computation of ek, then

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \gamma, \delta; \rho)) \equiv (\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}_{1-\mathsf{x}[i]})$$

where $(\mathsf{e}_0, \mathsf{e}_1) = \mathsf{D}(\mathsf{td}, \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho))$

(ii) If ek consist of uniform group elements, and γ and δ are independent uniform group elements in $\mathbb{G}$ and $\mathbb{H}$ (resp.), then

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \gamma, \delta; \rho)) \equiv (\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}')$$

where $\mathsf{e}' \xleftarrow{\$} \hat{\mathbb{G}}$.

We now state and prove the lemma.

Lemma C.1. *Let* $\rho \xleftarrow{\$} \{0,1\}^*$, $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, *and* $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_{[i]})$ *and let* $\gamma_{i_1,1-\mathsf{x}[i]}$ *and* $\delta_{i_2,1-\mathsf{x}[i]}$, *be as defined in Equation C.5 in the computation of* $(\mathsf{ek}, \mathsf{td})$. *Then, under the SXDH assumption over* $\mathcal{P}$, *it holds that*

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \gamma_{i_1,1-\mathsf{x}[i]}, \delta_{i_2,1-\mathsf{x}[i]}; \rho)) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \gamma_{i_1,1-\mathsf{x}[i]}, \delta_{i_2,1-\mathsf{x}[i]}; \rho)))$$

where ek' *consists of uniform independent group elements.*

Proof. Roughly speaking, we follow the steps taken in the proof for Construction 4.1, except here we rely on the hardness of SXDH, rather than DDH. We define $2\sqrt{n}+1$ hybrid distributions: Hybrid_0, $\mathsf{Hybrid}_{1,0}$, $\mathsf{Hybrid}_{1,1}$, …, $\mathsf{Hybrid}_{\sqrt{n},0}$, $\mathsf{Hybrid}_{\sqrt{n},1}$. The first hybrid, Hybrid_0, consists of hk and ek that are produced by an execution of the protocol, together with the corresponding encoding $\mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \gamma_{i_1,1-\mathsf{x}[i]}, \delta_{i_2,1-\mathsf{x}[i]})$. In the following hybrids, we transform every $g_{j,b}^{s_1}$ and $h_{j,b}^{s_2}$ in ek with uniform elements, $g'_{j,b}$ and $h'_{j,b}$, one step at a time. Formally, we define

$$\mathsf{Hybrid}_{k,b} := \left(\mathsf{x}, \mathsf{hk}, \mathsf{ek}_{k,b}, \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}_{k,b}, \gamma_{i_1,1-\mathsf{x}[i]}, \delta_{i_2,1-\mathsf{x}[i]}; \rho)\right)$$

where

$$\mathsf{ek}_{k,0} = \left(\begin{pmatrix} g'_{1,0}\gamma_{1,0}, \ldots, \boldsymbol{g'_{k,0}}\gamma_{k,0}, \ldots, g_{\sqrt{n},0}^{s_1}\gamma_{\sqrt{n},0} \\ g'_{1,1}\gamma_{1,1}, \ldots, g_{k,1}^{s_1}\gamma_{k,1}, \ldots, g_{\sqrt{n},1}^{s_1}\gamma_{\sqrt{n},1} \end{pmatrix}, \begin{pmatrix} h'_{1,0}\delta_{1,0}, \ldots, \boldsymbol{h'_{k,0}}\delta_{k,0}, \ldots, h_{\sqrt{n},0}^{s_2}\delta_{\sqrt{n},0} \\ h'_{1,1}\delta_{1,1}, \ldots, h_{k,1}^{s_2}\delta_{k,1}, \ldots, h_{\sqrt{n},1}^{s_2}\delta_{\sqrt{n},1} \end{pmatrix}\right)$$

$$\mathsf{ek}_{k,1} = \left(\begin{pmatrix} g'_{1,0}\gamma_{1,0}, \ldots, g'_{k,0}\gamma_{k,0}, \ldots, g_{\sqrt{n},0}^{s_1}\gamma_{\sqrt{n},0} \\ g'_{1,1}\gamma_{1,1}, \ldots, \boldsymbol{g'_{k,1}}\gamma_{k,1}, \ldots, g_{\sqrt{n},1}^{s_1}\gamma_{\sqrt{n},1} \end{pmatrix}, \begin{pmatrix} h'_{1,0}\delta_{1,0}, \ldots, h'_{k,0}\delta_{k,0}, \ldots, h_{\sqrt{n},0}^{s_2}\delta_{\sqrt{n},0} \\ h'_{1,1}\delta_{1,1}, \ldots, \boldsymbol{h'_{k,1}}\delta_{k,1}, \ldots, h_{\sqrt{n},1}^{s_2}\delta_{\sqrt{n},1} \end{pmatrix}\right)$$

where $\{g_{j,b}\}$ and $\{h_{j,b}\}$ are as generated in $\mathsf{hk} \xleftarrow{\$} \mathsf{G}(1^\lambda, 1^n)$ (see Equation C.4), $\{\gamma_{j,b}\}$, $\{\delta_{j,b}\}$, s_1 and s_2 are as generated in $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{E}(\mathsf{hk}, f_{[i]})$, and $\{g'_{j,b}\} \xleftarrow{\$} \mathbb{G}$ and $\{h'_{j,b}\} \xleftarrow{\$} \mathbb{H}$ are uniform.

It is easy to see

$$\mathsf{Hybrid}_{\sqrt{n},1} \equiv (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \gamma_{i_1,1-\mathsf{x}[i]}, \delta_{i_2,1-\mathsf{x}[i]}; \rho)))$$

for a uniform ek'. Thus, it suffices to show that every two adjacent hybrids in the sequence are computationally indistinguishable.

We take $\mathsf{Hybrid}_{k-1,1} \stackrel{c}{\equiv} \mathsf{Hybrid}_{k,0}$ for $k > i$. The proofs for the other cases are conceptually identical. Let $\mathcal{D}$ be a PPT distinguisher for which

$$|\Pr[\mathcal{D}(\mathsf{Hybrid}_{k-1,1}) = 1] - \Pr[\mathcal{D}(\mathsf{Hybrid}_{k,0}) = 1]| > \mathsf{negl}(\lambda)$$

for any negligible function $\mathsf{negl}(\cdot)$.

We construct an adversary $\mathcal{A}$ that uses $\mathcal{D}$ to break the indistinguishability in Equation C.3 from Proposition C.1. $\mathcal{A}$ takes as input a tuple $((p, G, H, \hat{G}, e_{\mathcal{P}}), (S_1, W, U), (S_2, Z, V))$, and proceeds as follows.

1. Set $\mathbb{G} := G^3$, $\mathbb{H} := H^3$ and $\hat{\mathbb{G}} := \hat{G}^3$, and define $e : \mathbb{G} \times \mathbb{H} \to \hat{\mathbb{G}}$ as in Equation C.1.
2. Choose generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$.
3. For every $j \in [\sqrt{n}] \backslash \{k\}$ and $b \in \{0,1\}$, sample $w_{j,b}, z_{j,b} \xleftarrow{\$} \mathbb{Z}_p^3$, and set

$$\tilde{g}_{j,b} = g^{w_{j,b}} \qquad\qquad \tilde{h}_{j,b} = h^{z_{j,b}}$$

4. Sample $w_{k,1}, z_{k,1} \xleftarrow{\$} \mathbb{Z}_p^3$ and set

$$\tilde{g}_{k,0} = W \qquad \tilde{h}_{k,0} = Z$$
$$\tilde{g}_{k,1} = g^{w_{k,1}} \qquad \tilde{h}_{k,1} = g^{z_{k,1}}$$

5. Set

$$\widetilde{\mathsf{hk}} = \left((\mathbb{G}, \mathbb{H}, \hat{\mathbb{G}}, e), \begin{pmatrix} \tilde{g}_{0,1}, \ldots, \tilde{g}_{0,\sqrt{n}} \\ \tilde{g}_{1,1}, \ldots, \tilde{g}_{1,\sqrt{n}} \end{pmatrix}, \begin{pmatrix} \tilde{h}_{0,1}, \ldots, \tilde{h}_{0,\sqrt{n}} \\ \tilde{h}_{1,1}, \ldots, \tilde{h}_{1,\sqrt{n}} \end{pmatrix} \right)$$

6. Generate subgroups $((\mathbb{G}_j)_{1 \le j \le 3}, (\mathbb{H}_j)_{1 \le j \le 3}) \leftarrow \mathcal{G}_3^{\mathsf{sub}}(\mathbb{G}, \mathbb{H})$.

7. For every $j \in [k-1]$ and $b \in \{0,1\}$, sample

$$\tilde{u}_{k,b} \xleftarrow{\$} \mathbb{G} \qquad \tilde{v}_{k,b} \xleftarrow{\$} \mathbb{H}$$

8. For $b \in \{0,1\}$, sample $\tilde{\gamma}_{k,b} \xleftarrow{\$} \mathbb{G}_1$ and $\tilde{\delta}_{k,b} \xleftarrow{\$} \mathbb{H}_2$, and set

$$\tilde{u}_{k,0} = U\gamma_{k,0} \qquad \tilde{v}_{k,0} = V\delta_{k,0}$$
$$\tilde{u}_{k,1} = S_1^{w_{k,1}}\gamma_{k,1} \qquad \tilde{v}_{k,1} = S_2^{z_{k,1}}\delta_{k,1}$$

9. For $k < j \le \sqrt{n}$ and $b \in \{0,1\}$, sample $\tilde{\gamma}_{j,b} \xleftarrow{\$} \mathbb{G}_1$ and $\tilde{\delta}_{j,b} \xleftarrow{\$} \mathbb{H}_2$, and set

$$\tilde{u}_{j,b} := S_1^{w_{j,b}}\gamma_{j,b} \qquad \tilde{v}_{j,b} := S_2^{z_{j,b}}\delta_{j,b}$$

10. Set

$$\widetilde{\mathsf{ek}} := \left(S_1, \begin{pmatrix} \tilde{u}_{0,1}, \ldots, \tilde{u}_{0,\sqrt{n}} \\ \tilde{u}_{1,1}, \ldots, \tilde{u}_{1,\sqrt{n}} \end{pmatrix}, S_2, \begin{pmatrix} \tilde{v}_{0,1}, \ldots, \tilde{v}_{0,\sqrt{n}} \\ \tilde{v}_{1,1}, \ldots, \tilde{v}_{1,\sqrt{n}} \end{pmatrix} \right)$$

11. Sample $\tilde{\gamma}_{i_1,1-\mathsf{x}[i]} \xleftarrow{\$} \mathbb{G}_3$ and $\tilde{\delta}_{i_2,1-\mathsf{x}[i]} \xleftarrow{\$} \mathbb{H}_3$ (when $k \le i$, these appear in $\widetilde{\mathsf{ek}}$), and output

$$\mathcal{D}(\mathsf{x}, \widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}, \mathsf{e}(\mathsf{x}, \widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}, \tilde{\gamma}_{i_1,1-\mathsf{x}[i]}, \tilde{\delta}_{i_2,1-\mathsf{x}[i]}; \rho))$$

Now, observe that if $((S_1, W, U), (S_2, Z, V)) \equiv ((g^{s_1}, g^w, g^{s_1 w}), (h^{s_2}, h^z, h^{s_2 z}))$ for $s_1, s_2, w, z \xleftarrow{\$} \mathbb{Z}_p^3$, then $(\widetilde{\mathsf{pp}}, \widetilde{\mathsf{ct}}) \equiv \mathsf{Hybrid}_{k-1,1}$. Otherwise, if $((S_1, W, U), (S_2, Z, V)) \equiv ((g^{s_1}, g^w, g^u), (h^{s_2}, h^z, h^v))$ for s_1, s_2, w, z, u, $v \xleftarrow{\$} \mathbb{Z}_p^3$, then $(\widetilde{\mathsf{hk}}, \widetilde{\mathsf{ek}}) \equiv \mathsf{Hybrid}_{k,0}$. This completes the proof of the lemma. □

We now proceed to prove function privacy and secret encoding hold. In fact, function privacy is immediately implied by Lemma C.1, which is even a stronger claim. Thus, it remains to show secret encoding, for which we prove the following lemma.

Theorem C.1. *The* TDH *scheme from Construction C.1 has secret encoding under SXDH.*

Proof. It appears that also for secret encoding, we have already done the hard work in Lemma C.1.

Let $\mathsf{hk} \xleftarrow{\$} \mathsf{S}(1^\lambda, 1^n)$, $(\mathsf{ek}, \mathsf{td}) \xleftarrow{\$} \mathsf{G}(\mathsf{hk}, f_{[i]})$ and $(\mathsf{e}_0, \mathsf{e}_1) = \mathsf{D}(\mathsf{td}, \mathsf{H}(\mathsf{hk}, \mathsf{x}; \rho))$ for $\rho \xleftarrow{\$} \{0,1\}^*$. By combining the indistinguihsability from Lemma C.1 and observation (i) from above, we obtain

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}_{1-\mathsf{x}[i]}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}(\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \gamma, \delta; \rho))$$

where ek' is uniform, and $\gamma \xleftarrow{\$} \mathbb{G}_3$ and $\delta \xleftarrow{\$} \mathbb{H}_3$ for subgroups $\mathbb{G}_3$ and $\mathbb{H}_3$ sampled by $\mathcal{G}_3^{\mathsf{sub}}(\mathbb{G}, \mathbb{H})$. Since such subgroups are sampled independently of hk and the uniform ek', then γ and δ are essentially uniform over $\mathbb{G}$ and $\mathbb{H}$, independently of hk and ek'. This fact allows us to apply observation (ii) from above, and obtain that

$$(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}_{1-\mathsf{x}[i]}) \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}')$$

for a uniform $\mathsf{k}' \xleftarrow{\$} \hat{\mathbb{G}}$.

It now suffices to show that $(\mathsf{x}, \mathsf{hk}, \mathsf{ek}, \mathsf{e}') \stackrel{c}{\equiv} (\mathsf{x}, \mathsf{hk}, \mathsf{ek}', \mathsf{e}')$. For this, we can use a reduction similar to the one in Lemma C.1, only now, $\mathcal{A}$ just generates a random $\tilde{\mathsf{e}} \xleftarrow{\$} \hat{\mathbb{G}}$ (notice that e' is independent of ek or ek'). This concludes the proof. □

Achieving Reusable Secret Encoding. Similarly to the basic DDH-based scheme, although the construction above satisfies the secret encoding property, it fails to provide the stronger security notion for the reusable case. We observe that a correlation with a similar nature is obtained here as well, and therefore, the same approach can be taken to solve the encoding correlation problem, namely using CIH. Typical known hash function constructions can be assumed to be a CIH for the required type of correlation in bilinear groups. This yields a TDH with reusable secret encoding under the SXDH assumption.