Extractable Witness Encryption for KZG Commitments and Efficient Laconic OT

Nils Fleischhacker[1]*, Mathias Hall-Andersen[2], and Mark Simkin[3]

[1] Ruhr University Bochum
mail@nilsfleischhacker.de
[2] Aarhus University
ma@cs.au.dk
[3] Ethereum Foundation
mark.simkin@ethereum.org

Abstract. We present a concretely efficient and simple extractable witness encryption scheme for KZG polynomial commitments. It allows to encrypt a message towards a triple $(\mathsf{com}, \alpha, \beta)$, where com is a KZG commitment for some polynomial f. Anyone with an opening for the commitment attesting $f(\alpha) = \beta$ can decrypt, but without knowledge of a valid opening the message is computationally hidden. Our construction is simple and highly efficient. The ciphertext is only a single group element. Encryption and decryption both require a single pairing evaluation and a constant number of group operations.
Using our witness encryption scheme, we construct a simple and highly efficient laconic OT protocol, which significantly outperforms the state of the art in most important metrics.

1 Introduction

The polynomial commitment scheme of Kate, Zaverucha, and Goldberg (KZG) [KZG10] is a powerful tool that has allowed for constructing a variety of advanced cryptographic primitives. Many concretely efficient vector commitments [CF13] with all kinds of additional functionalities or security properties are extensions of the KZG polynomial commitment scheme [TAB$^+$20, GRWZ20, LPR22, SCP$^+$22, WUP23]. Many of the currently most efficient proof systems [MBKM19, GWC19, CHM$^+$20] make crucial use of KZG commitments as part of their constructions.

In general, a polynomial commitment scheme allows for committing to a polynomial $f(X)$, such that one can later provide openings, attesting that $f(\alpha) = \beta$ for some chosen evaluation point α. The polynomial commitment scheme should ensure that the committed polynomial is at most of some degree d, that it is position binding in the sense that one cannot open the commitment to two different evaluations at the same point α, and in some cases it may also be desirable that the commitment itself hides the committed polynomial.

The KZG polynomial commitment scheme is highly efficient as both commitments and openings consists of a single group element, no matter how large the degree of the polynomial is. From a security perspective, the construction requires a trusted setup and can be proven secure in the algebraic group model (AGM) [FKL18] under a q-type variant of the standard discrete logarithm assumption.

Given the widespread usefulness of KZG commitments, it seems natural to ask the following somewhat abstract questions: Can we expand the toolkit surrounding KZG commitments in a fundamental way that would allow us to solve an even larger set of problems via these polynomial commitments?

* Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

1.1 Our Contribution

In this work, we present a simple, concretely efficient extractable witness encryption [GGSW13, GKP+13] scheme for KZG commitments and show how it can be used to construct concretely efficient laconic OT [CDG+17]. Concretely, we make the following contributions:

Witness Encryption for KZG Commitments. We present an encryption scheme that takes a KZG commitment $\texttt{com}$, evaluation point α, evaluation β, as well as a message m as input and produces ciphertext $\texttt{ct}$. Correctness of the encryption scheme allows anybody, who knows an opening for $\texttt{com}$ that attests that the committed polynomial evaluates to β at point α, to decrypt the ciphertext $\texttt{ct}$. Informally, security of the encryption scheme ensures that anybody, who can decrypt $\texttt{ct}$, can also compute a valid opening with respect to $(\texttt{com}, \alpha, \beta)$. Now, if $\texttt{com}$ is a commitment to polynomial $f(X)$ and $\texttt{ct}$ is an encryption for $(\texttt{com}, \alpha, \beta)$, where $\beta \neq f(\alpha)$, then $\texttt{ct}$ computationally hides the message, due to the evaluation binding property of the polynomial commitment scheme.

We note that for *any* tuple $(\texttt{com}, \alpha, \beta)$, there *exists* an opening and for this reason the vanilla notion of witness encryption as defined by Garg et al. [GGSW13] would not suffice to guarantee that the message is hidden. By focusing on extractable witness encryption as defined by Goldwasser et al. [GKP+13], we can ensure that the message is hidden, whenever computing the opening is assumed to be hard.

Our encryption scheme is highly efficient in terms of both ciphertext size and computational costs. One ciphertext for the scheme as described above is comprised of a single group element, encryption requires two group operations and one pairing evaluation and decryption requires a single pairing evaluation. The construction is generalized for multiple opening constraints. In that case the costs grow linearly in the number of constraints.

Efficient Laconic Oblivious Transfer. Using our witness encryption scheme for KZG commitments, we present a simple and highly-efficient protocol for laconic oblivious transfer [CDG+17]. Oblivious transfer [Rab81] is an interactive protocol between a sender holding messages m_0 and m_1 and a receiver holding a choice bit b. At the end of the protocol, the receiver should only learn m_b and the sender should learn nothing at all.

In laconic oblivious transfer, the receiver holds a database $D \in \{0,1\}^n$ of n choice bits and publishes a digest $\texttt{digest} \leftarrow \mathsf{H}(D)$, whose size is independent of the size of D. The sender can then repeatedly choose a message pair (m_0, m_1), an index $i \in [n]$, and use the digest to compute a short message for the receiver, which allows them to obtain $m_{D[i]}$. As in regular OT, the receiver does not learn anything about the other message and the sender does not learn anything about the choice bit of the receiver.

Cho et al. [CDG+17] show that laconic OT is a useful building block for constructing secure computation protocols that operate on large inputs and for constructing multi-hop homomorphic encryption for RAM programs. Improving the efficiency of laconic OT, directly translates to faster protocols for these applications.

Implementation & Benchmarks. The full laconic OT construction was implemented[4] in Rust. We provide benchmarks for various parameter regimes and show that our laconic OT construction outperforms the state of the art in most important metrics quite significantly.

[4] https://github.com/rot256/research-we-kzg

1.2 Related Work

In the following, let us discuss related works from several research areas that intersect with our work here.

Witness Encryption. Witness encryption was originally introduced by Garg et al. [GGSW13], who presented a construction based on multilinear maps. Such encryption schemes take a statement $\mathtt{x}$ and a message m as input and produce a ciphertext $\mathtt{ct}$. Correctness guarantees that given a witness $\mathtt{w}$ with $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$, one can decrypt $\mathtt{ct}$ and security ensures that the message is computationally hidden, if $\mathtt{x} \notin \mathcal{L}$. It is important to note though that $\mathtt{ct}$ provides no explicit security guarantees, when $\mathtt{x} \in \mathcal{L}$. In a different work, Garg et al. [GGH+13] showed that one can construct general-purpose witness encryption from indistinguishability obfuscation [BGI+01]. Two recent works by Tsabary [Tsa22] and Vaikuntanathan [VWW22] present construction based on new computational hardness assumptions. Extractable witness encryption is a strengthening of the original notion that was introduced by Goldwasser et al. [GKP+13]. It requires the existence of an extractor, which can recover a witness $\mathtt{w}$ from any adversary that can break the semantic security of the encryption scheme. How to construct general-purpose extractable witness encryption is currently unclear. Garg et al. [GGHW14] show that the existence of a type of special-purpose obfuscation would imply the non-existence of general-purpose extractable witness encryption. Benhamouda and Lin [BL18] introduce the notion of witness selectors, which lie somewhere between extractable and non-extractable witness encryption. They require semantic security of the encryption to hold if finding a witness for the relation is computationally hard.

While remotely practical general-purpose witness encryption from standard assumptions currently seems out of reach, there are several works [GS17, BL18, BL20, BJKL21, CFK22] that construct different types of witness encryption for specific languages. In the work of Garg and Srinivasan [GS17], they construct a primitive they call homomorphic proof commitments with encryption. Given a commitment $\mathtt{com}$ and a message m they allow for encrypting the message, such that a proof π that $\mathtt{com}$ is a commitment to 0 or 1 can be used to decrypt the message. Benhamouda and Lin [BL20] introduce witness encryption for NIZKs of commitments. Their scheme allows for encrypting towards a tuple $(\mathtt{com}, G, y)$, where $\mathtt{com}$ is a commitment, G is a function, and y is an output. The witness for decryption is a non-interactive zero-knowledge proof for the statement "the message committed in $\mathtt{com}$ is m and $G(m) = y$". In the presented construction, the commitment and proof size are at least linear in the statement size. Campanelli, Fiore, and Khoshakhlagh [CFK22] present succinct functional commitments with an associated witness encryption scheme with short commitments and short openings. They introduce a new computational hardness assumption and use it to prove their construction secure in the AGM.

In contrast to these works, we construct an extractable witness encryption scheme for a polynomial, rather than functional, commitment scheme from standard assumptions in the AGM. We rely on KZG commitments which have succinct commitments and openings and the ciphertexts of our construction are equally succinct. Our construction is surprisingly simple and in particular much simpler than the constructions discussed above. We believe this to be a significant advantage, when it comes to engineering and deploying a cryptographic primitive in the real world.

Laconic OT. The concept of laconic OT was originally introduced in a work by Cho et al. [CDG+17]. The authors presented a construction based on a special type of encryption scheme in combination with garbled circuits. While asymptotically interesting, their construction makes non-blackbox use of cryptographic objects and is completely impractical when it comes to concrete performance.

Green, Jain, Van Laer [GJL23] focus on concretely efficient laconic OT protocols, but the security of their proposed construction relies on a new hardness assumption they introduce. In addition, they require a generic and somewhat expensive transformation via garbled circuits to achieve sender privacy. Döttling et al. [DKL+23] present another laconic OT protocol based on the learning with errors (LWE) with error leakage problem. They show that this problem reduces to standard LWE for certain parameter ranges.

From a concrete efficiency perspective, our construction is highly efficient in terms of bandwidth and computational overheads that are induced by the individual OT executions. The main drawback of our construction is a large crs, since we require the same setup as KZG commitments. In comparison the work of Green, Jain, Van Laer, our public parameters are smaller by a factor of 3-4x, our sender's message size is smaller by a factor $\approx$ 5x, and our receiver's computation time is smaller by 1-6 *orders of magnitude* depending on the precise setting. In comparison to the work of Döttling et al., our sender's OT message is smaller by 5 *orders of magnitude.* We provide a detailed discussion of the concrete efficiency of our construction in Section 6.

2 Preliminaries

In this section, we will recall some notation and standard definitions that we will use throughout the paper.

Notation. We denote by $\lambda \in \mathbb{N}$ the security parameter, by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ and by $\mathsf{negl}(\lambda)$ any negligible function in λ. An algorithm is PPT if it is modeled by a probabilistic Turing machine with a running time bounded by $\mathsf{poly}(\lambda)$. Let S be a set. We write $x \leftarrow S$ for the process of sampling an element of S uniformly at random. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. For a vector $\boldsymbol{v} \in S^n$ and $i \in [n]$, we write v_i to denote its i-th component.

2.1 Algebraic Group Model

Some of our proofs will rely on the algebraic group model, which was introduced by Fuchsbauer, Kiltz, and Loss [FKL18]. In this model, one considers algebraic algorithms and, in particular, an algebraic adversary. Let $\mathbb{G}$ of prime order q. Whenever an algorithm returns a value $Y \in \mathbb{G}$, it must also provide an algebraic representation of that element $(e_1, \ldots, e_n)$, such that $Y = \prod_{n}^{i=1} X_i^{e_i}$, where $(X_1, \ldots, X_n)$ are group elements the algorithm has previously seen.

2.2 Pairings and Assumptions

Let GGen be the parameter generation algorithm that takes the security parameter λ as input and returns $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$ as its output, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $p = p(\lambda)$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. For $i \in \{1, 2, T\}$, we write $[\alpha]_i$ as a shorthand notation for g_i^{α} and we write $[\alpha]_i + [\beta]_i = [\alpha + \beta]_i$ as a shorthand notation for the multiplication of two group elements. The type of pairing is irrelevant for our applications.

Definition 1 (ℓ-DLOG Assumption). *The ℓ-DLOG assumption holds with respect to* GGen*, if for any $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\tau = \tau' : \begin{array}{r} \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda) \\ \tau \leftarrow \mathbb{F}_p \\ \tau' \leftarrow \mathcal{A}(\mathsf{par}, ([\tau^0]_1, \ldots, [\tau^\ell]_1, [1]_2, [\tau]_2)) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of the group generation algorithm and the adversary and over the uniform choice of τ.

Remark 1. Throughout the paper we will omit explicitly spelling out the group parameter generation and we will assume all involved parties are always implicitly provided with the parameters.

2.3 Polynomial Commitments

A polynomial commitment allows for computing a short value $\mathtt{com}$ for a polynomial f of potentially high degree over a finite field $\mathbb{F}$. Later on, one can compute short openings that certify that the polynomial committed to by $\mathtt{com}$ evaluates to $\beta \in \mathbb{F}$ at some position $\alpha \in \mathbb{F}$. Polynomial commitment should be binding in the sense that it should be impossible to open the same point to two different values.

Definition 2 (Polynomial Commitments). *A polynomial commitment over $\mathbb{F}$ is a tuple of PPT algorithms* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$**:** *The setup algorithm takes security parameter λ and degree upper bound d as input and returns commitment key* ck*.*

$\mathtt{com} \leftarrow \mathsf{Commit}(\mathsf{ck}, f)$**:** *The commitment algorithm takes commitment key* ck *and polynomial $f \in \mathbb{F}[X]$ as input and returns commitment* $\mathtt{com}$*.*

$\pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, \alpha, \beta)$**:** *The opening algorithm takes commitment key* ck*, polynomial $f \in \mathbb{F}[X]$, point $\alpha \in \mathbb{F}$, and point $\beta \in \mathbb{F}$ as input and returns openings π.*

$b \leftarrow \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta)$**:** *The verification algorithm takes commitment key* ck*, commitment* $\mathtt{com}$*, opening π, point $\alpha \in \mathbb{F}$, and point $\beta \in \mathbb{F}$ as input and returns a bit b.*

Definition 3 (Correctness). *A polynomial commitment scheme* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *over $\mathbb{F}$ is correct, if for all $\lambda, d \in \mathbb{N}$, all* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*, all polynomials $f \in \mathbb{F}[X]$ of degree at most d, all* $\mathtt{com} \leftarrow \mathsf{Commit}(\mathsf{ck}, f)$*, all $\alpha, \beta \in \mathbb{F}$, and all* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, \alpha, \beta)$ *it holds that* $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta) = 1$*.*

Definition 4 (Binding). *A polynomial commitment scheme* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *over $\mathbb{F}$ is binding, if for all $\lambda, d \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{r} \beta_0 \neq \beta_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_0, \alpha, \beta_0) = 1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_1, \alpha, \beta_1) = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathtt{com}, \alpha, \beta_0, \beta_1, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

```
$\mathsf{Setup}(1^\lambda, 1^d)$
    $\tau \leftarrow \mathbb{F}_p$
    return $([\tau^0]_1, \ldots, [\tau^d]_1, [1]_2, [\tau]_2)$

$\mathsf{Commit}(\mathsf{ck}, f)$
    $\mathtt{com} := \sum_{i=0}^{d} f_i \cdot [\tau^i]_1$
    return $\mathtt{com}$

$\mathsf{Open}(\mathsf{ck}, \mathtt{com}, f, \alpha, \beta)$
    $q(X) := \frac{f(X) - \beta}{X - \alpha}$
    $\pi := \sum_{i=0}^{d} q_i \cdot [\tau^i]_1$
    return $\pi$

$\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta)$
    return $e(\mathtt{com} - [\beta]_1, [1]_2) \stackrel{?}{=} e(\pi, [\tau]_2 - [\alpha]_2)$
```

Fig. 1. The KZG polynomial commitment scheme.

2.4 KZG Commitments

Let us recall the KZG polynomial commitment scheme. The scheme's public parameters are powers of a secret point τ in the exponent of a group generator. Committing to a polynomial is done by evaluating it in the exponent at point τ, which can be done using the public parameters, but without knowledge of τ.

Theorem 1 ([CHM+20]). *If the d-DLOG assumption holds with respect to* GGen*, then KZG commitment scheme described in Figure 1 is a correct and binding polynomial commitment scheme in the AGM.*

It was shown by Feist and Khovratovich [FK23] that it is possible to open a KZG commitment in n positions much more efficiently than naively computing each opening separately Specifically, it is possible to perform a batch opening at n points in time $\mathcal{O}(n \log^2(n))$. This can be further improved to $\mathcal{O}(n \log(n))$, if the points are powers of a root of unity. We refer to this batch opening algorithm as $\mathsf{BatchOpen}$.

2.5 Weakly-Hiding Vector Commitments

A vector commitment allows for computing a short value $\mathtt{com}$ for a potentially long vector of messages $(m_1, \ldots, m_n)$. Later on, one can compute short openings that certify that the vector committed to by $\mathtt{com}$ opens to m_i at some position $i \in [n]$. Vector commitment should be binding in the usual sense, meaning that no position can be opened to two different values. In addition, we will require a weak form of hiding from our vector commitments.

Definition 5. *A vector commitment with batch opening is a tuple of PPT algorithms* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$**:** *The setup algorithm takes security parameter λ and vector length n as input and returns commitment key* ck.

$(\mathtt{com}, \mathtt{aux}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m})$**:** *The commitment algorithm takes commitment key* ck *and vector* $\boldsymbol{m} \in \mathcal{M}^n$ *as input and returns commitment* $\mathtt{com}$ *and auxiliary output* $\mathtt{aux}$.

$(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathsf{ck}, \mathtt{com}, \mathtt{aux})$**:** *The batch opening algorithm takes commitment key* ck*, a commitment* $\mathtt{com}$*, and auxiliary input* $\mathtt{aux}$*, as input and returns openings* $\pi_1, \ldots, \pi_n$.

$b \leftarrow \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, i, m)$**:** *The verification algorithm takes commitment key* ck*, commitment* $\mathtt{com}$*, opening* π*, index* i *and message* m *as input and returns a bit* b*.*

Definition 6 (Correctness). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is correct, if for all* $\lambda, n \in \mathbb{N}$*, all* $\boldsymbol{m} \in \mathcal{M}^n$*, all* $(\mathtt{com}, \mathtt{aux}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m})$*, all* $(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathsf{ck}, \mathtt{com}, \mathtt{aux})$*, and all* $i \in [n]$ *it holds that* $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_i, i, m) = 1$*.*

Definition 7 (Position-Binding). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is position binding if for all* $\lambda, n \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{c} m_0 \neq m_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_0, i, m_0) = 1 : \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_1, i, m_1) = 1 \end{array} \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathtt{com}, i, m_0, m_1, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 8 (Perfect Weak Hiding). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is perfectly weakly hiding if for all* $\lambda, n \in \mathbb{N}$ *and all* $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \mathcal{M}^n$ *and all* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ *it holds that over the random coins of the commitment algorithm,* $\mathtt{com}_0$ *and* $\mathtt{com}_1$ *computed as* $(\mathtt{com}_0, \mathtt{aux}_0) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m}_0)$ *and* $(\mathtt{com}_1, \mathtt{aux}_1) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m}_1)$ *are distributed identically.*

Definition 9 (Efficiency). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is efficient, if commitments are of size independent of* n*, the commitment algorithm runs in time* $n \cdot \mathsf{poly}(\lambda)(\lambda)$ *and the batch opening algorithm runs in time* $n \cdot \mathsf{poly}(\lambda)(\log n, \lambda)$*.*

```
$\mathsf{Setup}(1^\lambda, 1^n)$
    $\mathsf{pp} \leftarrow \mathsf{KZG.Setup}(1^\lambda, 1^n)$
    return $\mathsf{pp}$

$\mathsf{Commit}(\mathsf{ck}, \boldsymbol{m})$
    $r \leftarrow \mathbb{F}_p$
    $\mathbb{F}_p[X] \ni f(X) := \begin{cases} m_i \text{ if } X \in [n] \\ r \text{ if } X = 0 \end{cases}$
    $\mathtt{com} \leftarrow \mathsf{KZG.Commit}(\mathsf{pp}, f)$
    return $(\mathtt{com}, f)$

$\mathsf{BatchOpen}(\mathsf{ck}, \mathtt{com}, f)$
    $\boldsymbol{\pi} \leftarrow \mathsf{KZG.BatchOpen}(\mathsf{ck}, \mathtt{com}, f, (i, f(i))_{i \in [n]})$
    return $\boldsymbol{\pi}$

$\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, i, m)$
    return $\mathsf{KZG.Verify}(\mathsf{ck}, \mathtt{com}, \pi, i, m)$
```

Fig. 2. Weakly-hiding vector commitments from KZG polynomial commitments.

Given a polynomial commitment, it is easy to construct a vector commitment by encoding the message vector's entries into distinct polynomial evaluations. In the theorem statement below, we recall this transformation using KZG commitments but add a small twist that makes the construction weakly hiding.

Theorem 2. *If* KZG *is a binding polynomial commitment scheme, then the construction specified in Figure 2 is a correct, efficient, computationally position binding, and perfectly weakly hiding commitment scheme.*

Proof. Correctness is immediate from the correctness of KZG. Efficiency similarly follows from the definition of KZG and the batch opening algorithm of Feist and Khovratovich [FK23].

It is trivial to see, that any adversary breaking position binding of the vector commitment immediately also breaks binding of the KZG commitment scheme with the same probability. Position binding therefore follows immediately from the binding property of the KZG commitment. To see that the scheme is perfectly weakly hiding, consider any $\lambda, n \in \mathbb{N}$ and any $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \mathcal{M}^n$ and any $\mathtt{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$. Any commitment to either vector has the form $\mathtt{com}_b = [\mu_b]_1 + [r]_1$, where μ_b is defined by m_b and independent of r. Since r is chosen uniformly from $\mathbb{F}_p$, $\mathtt{com}_b$ is always a uniformly distributed element of $\mathbb{G}_1$. In particular, this means that $\mathtt{com}_0$ and $\mathtt{com}_1$ are distributed identically.

2.6 Symmetric Encryption

We quickly recall the definition of a symmetric encryption scheme.

Definition 10. *A symmetric encryption scheme with keyspace $\mathcal{K}$ and message space $\mathcal{M}$ is a pair of PPT algorithms $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ defined as follows:*

$\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathcal{K}, m)$**:** *The setup algorithm takes a key $\mathtt{k} \in \mathcal{K}$ and a message $m \in \mathcal{M}$ as input and returns ciphertext $\mathtt{ct}$.*

$\mathtt{ct} \leftarrow \mathsf{Dec}^{\mathsf{sym}}(\mathcal{K}, \mathtt{ct})$**:** *The setup algorithm takes a key $\mathtt{k} \in \mathcal{K}$ and a ciphertext $\mathtt{ct}$ as input and returns message m.*

Definition 11 (Correctness). *A symmetric encryption scheme $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ is correct, if for all $\mathtt{k} \in \mathcal{K}$, all $m \in \mathcal{M}$, and all $\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m)$ it holds that $\mathsf{Dec}^{\mathsf{sym}}(\mathtt{k}, \mathtt{ct}) = m$.*

We will only require a very weak notion of security, namely indistinguishability against eavesdroppers, also called EAV-security. This security notion is e.g. satisfied by the one-time pad or a (nonce-less) stream cipher.

Definition 12 (EAV-Security). *A symmetric encryption scheme $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ has indistinguishable encryptions in the presence of an eavesdropper, or is EAV-secure, if for any PPT adversary $\mathcal{A}$ it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{A}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the experiment is defined as follows.

```
$\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{A}}(1^\lambda)$
$\mathtt{k} \leftarrow \mathcal{K}$
$(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$
$b \leftarrow \{0,1\}$
$\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_b)$
$b' \leftarrow \mathcal{A}(\mathtt{ct})$
**return** $\begin{cases} 1 & \textit{if } b' = b \\ 0 & \textit{otherwise} \end{cases}$
```

3 Extractable Witness KEMs

As a building block for our main constructions, we first define and instantiate the notion of an extractable witness KEM. This notion, with minor differences, has previously been defined by Choi and Vaudenay [CV21]. In their work, the authors present an instantiation of this notion for some class of problems using a new non-falsifiable hardness assumption in a new restricted non-standard computational model. In our work, we will focus on instantiating the notion of an extractable witness KEM for specific relations using a standard assumption in the AGM.

We first define the notions of an indexed family of NP relations. We will later define extractable witness KEMs relative to such a family. This is more general than defining them for an individual relation and allows some part of the relation to be fixed by system parameters.

Definition 13. *Let $\mathcal{I} \subseteq \{0,1\}^*$ be a set. A set $\mathcal{F} = \{\mathcal{R}_I\}_{I\in\mathcal{I}}$ a family of NP relations with index set $\mathcal{I}$ if for all $I \in \mathcal{I}$, $\mathcal{R}_I$ is an NP relation. We call I the index of $\mathcal{R}_I$ and $\mathcal{R}_I$ the relation identified by I. We use $\mathcal{L}_I$ to refer to the corresponding NP language.*

Relative to these families we can now define a witness KEM. The general idea of a witness KEM is that it works like a regular key encapsulation mechanism, but uses the pairs of statement and witness in an NP relation as a key-pair.

Definition 14. *A witness key encapsulation mechanism for a family of NP relations $\mathcal{F}$ and a keyspace $\mathcal{K}$ is a pair of PPT algorithms $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$, defined as follows:*

$(\mathtt{ct}, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$**:** *The encapsulation algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$ and a statement $\mathtt{x}$ and returns as output a ciphertext $\mathtt{ct}$ and a key $\mathtt{k} \in \mathcal{K}$.*

$\mathtt{k} \leftarrow \mathsf{Decap}(I, \mathtt{w}, \mathtt{ct})$**:** *The deterministic decapsulation algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$, a witness $\mathtt{w}$, and a ciphertext $\mathtt{ct}$ and returns a key $\mathtt{k} \in \mathcal{K}$.*

Definition 15 (Correctness). *A witness KEM $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ for a family of NP relations $\mathcal{F}$ is correct, if for any $\mathcal{R}_I \in \mathcal{F}$, any $\lambda \in \mathbb{N}$, any $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_I$, and any $(\mathtt{ct}, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$ it holds that $\mathsf{Decap}(I, \mathtt{w}, \mathtt{ct}) = \mathtt{k}$.*

The above definitions by themselves do not guarantee any kind of security. Security will be derived from the extractability of the scheme. Extractability essentially says that if any efficient adversary can distinguish between a key encapsulated under some statement $\mathtt{x}$ and a random key, then this adversary can also be used to extract a witness $\mathtt{w}$ for $\mathtt{x}$.

Definition 16 (Extractability). *A witness KEM $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ for a family of NP-relations $\mathcal{F}$ is extractable, if there exists a PPT algorithm Ext such that for any stateful PPT adversary $\mathcal{A}$ and any relation $\mathcal{R}_I \in \mathcal{F}$ such that*

$$\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda, I) = 1] \geq \frac{1}{2} + \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$, it holds that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} \mathtt{x} \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot,\cdot)}(I, \mathtt{x}) \end{array}\right] \geq \delta(\lambda),$$

for some non-negligible function $\delta(\lambda)$. The latter probability is taken over the random coins of the adversary and the extractor and the experiment $\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda)$ is defined as follows.

```
$\underline{\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda, I)}$
$\mathtt{x} \leftarrow \mathcal{A}(1^\lambda, I)$
$b \leftarrow \{0,1\}$
$(\mathtt{ct}, \mathtt{k}_0) \leftarrow \mathsf{Encap}(I, \mathtt{x})$
$\mathtt{k}_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}(\mathtt{ct}, \mathtt{k}_b)$
return $\begin{cases} 1 & \textit{if } b' = b \\ 0 & \textit{otherwise} \end{cases}$
```

3.1 An Extractable Witness KEM for KZG Openings

We now proceed with constructing an extractable witness KEM as just defined for a very specific family of relations, specifically those describing valid opening of KZG commitments. Let (Setup, $\mathsf{Commit}, \mathsf{Open}, \mathsf{Verify}$) be the KZG commitment relative to the bilinear group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p as specified in Figure 1. Let $\mathsf{CK} = \{\mathsf{ck} \in \mathbb{G}_1^{d+1} \times \mathbb{G}_2^2 \mid d \in \mathbb{N} \wedge \mathsf{ck} \in \mathsf{Setup}(1^\lambda, 1^d)\}$ be the set of valid KZG commitment keys. We can then define the family of NP relations of valid KZG openings as

$$\mathcal{F}_{KZG} := \{\mathcal{R}_{\mathsf{ck}}\}_{\mathsf{ck} \in \mathsf{CK}}$$

where

$$\mathcal{R}_{\mathsf{ck}} = \left\{ \left((\mathtt{com}_j, \alpha_j, \beta_j)_{j \in [\ell]}, (\pi_j)_{j \in [\ell]} \right) \mid \ell \in \mathbb{N} \wedge \forall j \in [\ell].\ \mathsf{Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, \alpha_j, \beta_j) = 1 \right\}$$

for any $\mathsf{ck} \in \mathsf{CK}$.

```
$\underline{\mathsf{Encap}^{\mathsf{H}}(\mathsf{ck}, (\mathtt{com}, \alpha, \beta))}$
for $1 \le j \le \ell$
    $r_j \leftarrow \mathbb{F}_p$
    $s_j := e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$
    $\mathtt{ct}_j \leftarrow r_j \cdot ([\tau]_2 - [\alpha_j]_2)$
$\mathtt{ct} := (\mathtt{ct}_1, \ldots, \mathtt{ct}_\ell)$
$\mathtt{k} := \mathsf{H}(s_1, \ldots, s_\ell)$
return $(\mathtt{ct}, \mathtt{k})$

$\underline{\mathsf{Decap}^{\mathsf{H}}(\mathsf{ck}, (\pi_1, \ldots, \pi_\ell), \mathtt{ct})}$
parse $\mathtt{ct}$ as $(\mathtt{ct}_1, \ldots, \mathtt{ct}_\ell)$
for $1 \le j \le \ell$
    $s_j := e(\pi_j, \mathtt{ct}_j)$
$\mathtt{k} := \mathsf{H}(s_1, \ldots, s_\ell)$
return $\mathtt{k}$
```

Fig. 3. Construction of an Extractable Witness KEM for $\mathcal{F}_{KZG}$ with keyspace $\{0,1\}^\lambda$ in the combined Algebraic Group and Random Oracle Model.

Theorem 3. *Let* $\mathsf{H} : \mathbb{G}_T^* \to \{0,1\}^\lambda$ *be a hash functioned modeled as a random oracle. If the d-DLOG assumption holds with respect to* GGen*, then the construction described in Figure 3 is an extractable witness KEM for* $\mathcal{F}_{KZG}$ *in the algebraic group model.*

Proof. Let $\mathcal{A}$ be an arbitrary algebraic PPT adversary with non-negligible advantage $\epsilon(\lambda)$ for the extractability of the construction described in Figure 3.

We construct an extractor Ext as follows. The extractor receives as input an index $\mathsf{ck} = ([\tau^0]_1, \dots, [\tau^d]_1, [1]_2, [\tau]_2) \in \mathsf{CK}$ as well as a statement $((\mathtt{com}_1, \alpha_1, \beta_1), \dots, (\mathtt{com}_\ell, \alpha_\ell, \beta_\ell) \in (\mathbb{G}_1 \times \mathbb{F}_p^2)^\ell$. Further, since $\mathcal{A}$ is an algebraic algorithm, the extractor also receives an algebraic representations of each $\mathtt{com}_j$ in the form of coefficients $f_{j,0}, \dots, f_{j,d}$ such that

$$\mathtt{com}_j := \sum_{i=0}^{d} f_{j,i} \cdot [\tau^i]_1.$$

The extractor chooses $r_1, \dots, r_\ell \leftarrow \mathbb{F}_p$, computes $\mathtt{ct}_j := r_j \cdot ([\tau]_2 - [\alpha_j]_2)$ for each $j \in [\ell]$, sets $\mathtt{ct} := (\mathtt{ct}_1, \dots, \mathtt{ct}_\ell)$, chooses a random key $\mathtt{k} \leftarrow \{0,1\}^\lambda$, initializes an empty list $\mathcal{L} := \emptyset$ and invokes $\mathcal{A}(\mathtt{ct}, \mathtt{k})$. The adversary $\mathcal{A}$ expects access to a random oracle that Ext simulates as follows. For any query $\boldsymbol{s} \in \mathbb{G}_T^*$ such that either $|\boldsymbol{s}| \neq \ell$ or for at least one $j \in [\ell]$, $s_j \neq e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$, Ext continues to simulate the random oracle via lazy sampling. I.e., if there exists an entry $(\boldsymbol{s}, \mathtt{k}_{\boldsymbol{s}}) \in \mathcal{L}$ it returns $\mathtt{k}_{\boldsymbol{s}}$. If such an entry does not yet exist, Ext samples $\mathtt{k}_{\boldsymbol{s}} \leftarrow \{0,1\}^n$ and returns $\mathtt{k}_{\boldsymbol{s}}$.

If, however, $|\boldsymbol{s}| = \ell$ and for all $j \in [\ell]$, $s_j = e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$ then the extractor aborts $\mathcal{A}$ and continues as follows. Since $\mathcal{A}$ is an algebraic algorithm it also provides an algebraic description of each s_j in the form of of coefficients[5]

$$\tilde{w}_{j,0}, \dots, \tilde{w}_{j,2d}, \tilde{q}_{j,1,0}, \dots, \tilde{q}_{j,1,d}, \tilde{q}_{j,2,0}, \dots, \tilde{q}_{j,\ell,d}, \tilde{h}_{j,1}, \dots, \tilde{h}_{j,\ell}$$

such that

$$\begin{aligned} s_j := \quad & \sum_{i=0}^{2d} \tilde{w}_{j,i} \cdot [\tau^i]_T + \sum_{k=1}^{\ell} \sum_{i=0}^{d} \tilde{q}_{j,k,i} \cdot [r_k \cdot (\tau - \alpha_k) \cdot \tau^i]_T \\ & + \sum_{k=1}^{\ell} \tilde{h}_{j,k} \cdot [r_j \cdot (\tau - \alpha_j) \cdot r_k \cdot (\tau - \alpha_k)]_T. \end{aligned}$$

Since Ext computed each $\mathtt{ct}_k$, $k \neq j$ algebraically, however, we can simplify this representation. Specifically Ext can compute an algebraic description of each s_j in the form of coefficients $w_{j,0}, \dots, w_{j,2d}, q_{j,0}, \dots, q_{j,d}, h_j$ defined as

$$w_{j,i} := \begin{cases} \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i} r_k \alpha_k & \text{if } i = 0 \\ \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i} r_k \alpha_k + \tilde{q}_{j,k,i-1} r_k & \text{if } 0 < i \leq d \\ \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i-1} r_k & \text{if } i = d+1 \\ \tilde{w}_{j,i} & \text{if } i > d+1 \end{cases}$$

$$q_{j,i} := \begin{cases} \tilde{q}_{j,j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{h}_{j,k} r_k & \text{if } i = 0 \\ \tilde{q}_{j,j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{h}_{j,k} r_k \alpha_k & \text{if } i = 1 \\ \tilde{q}_{j,j,i} & \text{if } i > 1 \end{cases}$$

$$h_j := \tilde{h}_{j,j}$$

[5] Note that we allow the algebraic adversary the maximum amount of freedom here that they would only have in the context of a Type-1 pairing. In the context of Type-2 or Type-3 pairings, many of these coefficients would necessarily be 0.

such that

$$\begin{aligned} s_j :=& \sum_{i=0}^{2d} w_{j,i} \cdot [\tau^i]_T + \sum_{i=0}^{d} q_{j,i} \cdot [r_j \cdot (\tau - \alpha_j) \cdot \tau^i]_T + h_j \cdot [r^2 \cdot (\tau - \alpha_j)^2)]_T \\ =& \sum_{i=0}^{2d} w_{j,i} \cdot [\tau^i]_T + r_j \cdot (\tau - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot [\tau^i]_T + h_j r_j^2 \cdot [(\tau - \alpha_j)^2]_T \end{aligned}$$

For each $j \in [\ell]$ we define the two polynomials

$$F_j(X) := \sum_{i=1}^{d} f_{j,i} \cdot X^i$$

and

$$G_j(X,Y) := \sum_{i=0}^{2d} w_{j,i} \cdot X^i + Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i + h_j Y^2 \cdot (X - \alpha_j)^2.$$

It is easy to see that $\mathtt{com}_j = [F_j(\tau)]_T$ and $s_j = [G_j(\tau, r_j)]_T$.

If for all $j \in [\ell]$, $G_j(X,Y) = Y(F_j(X) - \beta_j)$ the extractor computes $\pi_j := \sum_{i=1}^{d} q_{j,i} \cdot [\tau^i]_1$ and outputs $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_\ell)$. Otherwise the extractor outputs $\perp$. If the adversary terminates without querying an $\boldsymbol{s}$ as described above, then the extractor also outputs $\perp$.

We now prove two claims about the extractor.

Claim 4. *If* Ext *outputs* $\boldsymbol{\pi} \neq \perp$*, then for all* $j \in [\ell]$ *it holds that* $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, \alpha_j, \beta_j) = 1$.

Proof. The extractor only outputs $\boldsymbol{\pi} \neq \perp$, if for all $j \in [\ell]$, $G_j(X,Y) = Y(F_j(X) - \beta_j)$. We observe that, this can only be true if $h_j = 0$ and $w_{j,i} = 0$ for all $j \in [\ell]$ and $0 \le i \le 2d$. Therefore,

$$G_j(X,Y) := Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i.$$

and thus

$$\begin{aligned} & G_j(X,Y) = Y(F_j(X) - \beta_j) \\ \iff & Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i = Y(F_j(X) - \beta_j) \\ \iff & \sum_{i=0}^{d} q_{j,i} \cdot X^i = \frac{F_j(X) - \beta_j}{X - \alpha_j} \end{aligned}$$

It follows that

$$\pi_j = \left[\frac{F_j(\tau) - \beta_j}{\tau - \alpha_j}\right]_1$$

and thus the verification equation

$$e(\pi_j, [\tau]_2 - [\alpha_j]_2) = e\left(\left[\frac{F_j(\tau) - \beta_j}{\tau - \alpha_j}\right]_1, [\tau - \alpha_j]_2\right)$$

$$
\begin{aligned}
&= [F_j(\tau) - \beta_j]_T \\
&= e([F_j(\tau) - \beta_j]_1, [1]_2) = e(\mathtt{com}_j - [\beta_j]_1, [1]_2)
\end{aligned}
$$

holds as required. $\square$

Claim 5. Ext *outputs* $\boldsymbol{\pi} = \bot$ *with probability at most* $1 - 2\epsilon(\lambda) + \mathsf{negl}(\lambda)$.

Proof. Let Hit denote the event that $\boldsymbol{s}$ with $|\boldsymbol{s}| = \ell$ and $s_j = e(r_j \cdot (\mathtt{com}_j - [\beta_j]_1), [1]_2)$ for all $j \in [\ell]$ is queried to the random oracle. The extractor outputs $\bot$, either if Hit does not occur, or if Hit occurs but for at least one $j \in [\ell]$ $G_j(X, Y) \neq Y(F_j(X) - \beta_j)$. Since those two events are mutually exclusive, we thus have

$$
\Pr[\boldsymbol{\pi} = \bot] = \Pr[\overline{\mathsf{Hit}}] + \Pr[\mathsf{Hit} \wedge \exists j \in [\ell].\ G_j(X, Y) \neq Y(F_j(X) - \beta_j)]. \tag{1}
$$

We bound the two probabilities separately.

Let us first bound $\Pr[\overline{\mathsf{Hit}}]$. Recall, that $\mathcal{A}$ is an adversary with advantage $\epsilon(\lambda)$. Whenever Hit does not occur, the view of $\mathcal{A}$ remains independent of the bit b in the KEM-CPA experiment. This is the case, since unless Hit occurs, both $\mathtt{k}_0$ and $\mathtt{k}_1$ are uniformly distributed. Therefore we have that

$$
\begin{aligned}
\frac{1}{2} + \epsilon(\lambda) &= \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM}_{KZG}, \mathcal{A}}(1^\lambda) = 1] \\
&= \Pr[b' = b] \\
&= \overbrace{\Pr[b' = b \mid \mathsf{Hit}]}^{\leq 1} \Pr[\mathsf{Hit}] + \overbrace{\Pr[b' = b \mid \overline{\mathsf{Hit}}]}^{=1/2} \Pr[\overline{\mathsf{Hit}}] \\
&\leq 1 - \Pr[\overline{\mathsf{Hit}}] + \frac{1}{2} \Pr[\overline{\mathsf{Hit}}] \\
&= 1 - \frac{1}{2} \Pr[\overline{\mathsf{Hit}}]
\end{aligned}
$$

and thus, we have that

$$
\Pr[\overline{\mathsf{Hit}}] \leq 1 - 2\epsilon(\lambda). \tag{2}
$$

Next we bound $\Pr[\mathsf{Hit} \wedge \exists j \in [\ell].\ G_j(X, Y) \neq Y(F_j(X) - \beta_j)]$. For each $j \in [\ell]$, define the bivariate polynomial $Q_j(X, Y) = G_j(X, Y) - Y(F_j(X) - \beta_j)$. Since $s_j = e(r_j \cdot (\mathtt{com}_j - [\beta_j]_1), [1]_2) = [r_j \cdot (F_j(\tau) - \beta_j)]_T$, it must hold that $Q_j(\tau, r_j) = 0$, however, since $G_j(X, Y) \neq Y(F_j(X) - \beta_j)$, it also holds that $Q_j(X, Y) \neq 0$. I.e., $Q_j(X, Y)$ is a non-zero polynomial with a root at (τ, r_j).

We can rewrite $Q_j(X, Y)$ as a polynomial of the form $Q_j(X, Y) = C_{j,0}(X) + C_{j,1}(X) \cdot Y + C_{j,2}(X) \cdot Y^2$, where each $C_{j,i}(X) \in \mathbb{F}_p[X]$ is a univariate polynomial of degree at most $2d$. Since $Q_j(X, Y)$ is non-zero, at least one of $C_{j,0}(X), C_{j,1}(X), C_{j,2}(X)$ must also be non-zero.

Now consider the univariate Polynomial $P_j(Y) := Q_j(\tau, Y)$. We can consider two cases, either $P_j(Y) = 0$, or $P_j(Y) \neq 0$. In the case that $P_j(Y) = 0$, it must hold that $C_{j,0}(\tau) = C_{j,1}(\tau) = C_{j,2}(\tau) = 0$, but at least one of these is a non-zero polynomial. We can therefore find the roots of one of the non-zero $C_{j,0}(Y), C_{j,1}(Y), C_{j,2}(Y)$ to recover τ. On the other hand, if $P_j(Y) \neq 0$, we can find the roots of $P_j(Y)$ to recover r_j.

We use this to specify the following reduction to d-DLOG. On input $[\tau^0]_1, \ldots, [\tau^d]_1, [\tau^0]_2, [\tau^1]_2$, the reduction $\mathcal{B}$ defines $\mathsf{ck} = ([\tau^0]_1, \ldots, [\tau^d]_1, [1]_2, [\tau]_2)$ and invokes $\mathcal{A}(1^\lambda, \mathsf{ck})$. Eventually the adversary will output $((\mathtt{com}_1, \alpha_1, \beta_1), \ldots, (\mathtt{com}_\ell, \alpha_\ell, \beta_\ell))$ together with the algebraic explanation of each

$\mathtt{com}_j$. The reduction now chooses $j^* \leftarrow [\ell]$. For $j \in [\ell] \setminus \{j^*\}]$, the reduction chooses $r_j \leftarrow \mathbb{F}_p$ and computes $\mathtt{ct}_j := r_j \cdot ([\tau]_2 - [\alpha_j]_2)$. For j^* it chooses $z \leftarrow \mathbb{F}_p$ and computes $\mathtt{ct}_{j^*} := (z \cdot [\tau^2]_2 - z\alpha_{j^*} \cdot [\tau]_2)$. Note, that, if we define $r_{j^*} := z\tau$ it holds that $(z \cdot [\tau^2]_2 - z\alpha_{j^*} \cdot [\tau]_2) = r_{j^*}([\tau]_2 - [\alpha_{j^*}]_2)$. Therefore, the ciphertext $\mathtt{ct}_{j^*}$ is computed correctly for an implicitly defined *but uniformly distributed* r_{j^*}. From this point, $\mathcal{B}$ proceeds exactly as Ext until Hit occurs. It is important to verify that $\mathcal{B}$ can actually check whether this query occurs, even on position j^*. However, due to the bilinearity of e, $\mathcal{B}$ can compute the relevant value as

$$\begin{aligned} e(\mathtt{com}_{j^*}, z \cdot [\tau]_2) - e(z\beta_{j^*} \cdot [\tau]_1, [1]_2) &= e(\mathtt{com}_{j^*}, z\tau \cdot [1]_2) - e(z\tau \cdot [\beta_{j^*}]_1, [1]_2) \\ &= e(r_{j^*} \cdot \mathtt{com}_{j^*}, [1]_2) - e(r_{j^*} \cdot [\beta_{j^*}]_1, [1]_2) \\ &= e(r_{j^*} \cdot (\mathtt{com}_{j^*} - [\beta_{j^*}]_1), [1]_2) \end{aligned}$$

and thus can check each query against this value. If Hit does not occur, or if $G_{j^*}(X, Y) = Y(F_{j^*}(X) - \beta_{j^*})$ then $\mathcal{B}$ aborts. Otherwise, we consider two cases.

If $P_{j^*}(Y) = 0$, then, as discussed above, there exists a $C_{j^*}(X) \in \{C_{j^*,0}(X), C_{j^*,1}(X), C_{j^*,2}(X)\}$ such that $C_{j^*}(X)$ is non-zero. $\mathcal{B}$ factors $C_{j^*}(X)$ to finds all of its at most $2d$ roots. For each root η, $\mathcal{B}$ checks whether $\eta \cdot [1]_1 = [\tau]_1$ and returns η if that's the case. Since τ must be one of the roots, $\mathcal{B}$ will always correctly identify τ in this case.

If $P_{j^*}(Y) \neq 0$, then $\mathcal{B}$ factors $P_{j^*}(Y)$ to find the all of its at most 2 roots. For each root η, $\mathcal{B}$ checks whether $\eta \cdot [1]_1 = z \cdot [\tau]_1$ and returns ηz^{-1} if that's the case. Since $r_{j^*} = z\tau$ must be one of the roots, $\mathcal{B}$ will always correctly identify $\tau = r_{j^*} z^{-1}$ in this case. It thus follows under the d-DLog assumption, that

$$\begin{aligned} \mathsf{negl}(\lambda) \geq\ & \Pr\left[\tau = \tau' : \begin{array}{r} \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda) \\ \tau \leftarrow \mathbb{F}_p \\ \tau' \leftarrow \mathcal{B}(\mathsf{par}, ([\tau^0]_1, \ldots, [\tau^d]_1, [1]_2, [\tau]_2)) \end{array}\right] \\ =\ & \Pr[\mathsf{Hit} \wedge G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*})] \\ \geq\ & \frac{1}{\ell} \Pr[\mathsf{Hit} \wedge \exists j \in [\ell].\ G_j(X, Y) \neq Y(F_j(X) - \beta_j)]. \end{aligned} \tag{3}$$

Since $\mathcal{A}$ is a PPT algorithm, ℓ must be polynomial and the claim follows. □

By combining the two claims we finally get

$$\begin{aligned} &\Pr\left[\forall j \in [\ell].\ \mathsf{Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, \alpha_j, \beta_j) = 1 : \begin{array}{r} (\mathtt{com}_j, \alpha_j, \beta_j)_{j \in [\ell]} \leftarrow \mathcal{A}(1^\lambda, \mathsf{ck}) \\ \boldsymbol{\pi} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot,\cdot)}(\mathsf{ck}, (\mathtt{com}_j, \alpha_j, \beta_j)_{j \in [\ell]}) \end{array}\right] \\ &= \Pr[\pi \neq \bot] = 1 - \Pr[\pi = \bot] = 2\epsilon(\lambda) - \mathsf{negl}(\lambda). \end{aligned}$$

which is non-negligible for any non-negligible function $\epsilon(\lambda)$, as required. □

We will later require an extractable witness KEM not just for plain KZG, but in fact for the derived weakly hiding vector commitment described in Figure 3. Let VC be the vector commitment and let $\mathsf{CK}_{\mathsf{VC}} = \{\mathsf{ck} \mid \mathsf{ck} \in \mathsf{VC.Setup}(1^\lambda, 1^n)\}$ be the set of valid commitment keys. We can then define the family of NP relations of valid VC openings as

$$\mathcal{F}_{\mathsf{VC}} := \{\mathcal{R}_{\mathsf{ck}}\}_{\mathsf{ck} \in \mathsf{CK}_{\mathsf{VC}}}$$

where

$$\mathcal{R}_{\mathsf{ck}} = \left\{ \left((\mathtt{com}_j, i_j, m_j)_{j\in[\ell]}, (\pi_j)_{j\in[\ell]} \right) \mid \ell \in \mathbb{N} \wedge \forall j \in [\ell].\ \mathsf{VC.Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, i_j, m_j) = 1 \right\}$$

for any $\mathsf{ck} \in \mathsf{CK}_{\mathsf{VC}}$.

Since VC commitments are simply KZG commitments, openings are KZG openings, and verification is KZG verification, it follows immediately that $\mathcal{F}_{\mathsf{VC}} = \mathcal{F}_{KZG}$ and we thus get the following corollary from Theorem 3.

Corollary 6. *Let* $\mathsf{H} : \mathbb{G}_T^* \to \{0,1\}^\lambda$ *be a hash functioned modeled as a random oracle. If the d-DLOG assumption holds with respect to* GGen*, then the construction described in Figure 3 is an extractable witness KEM for* $\mathcal{F}_{\mathsf{VC}}$ *in the algebraic group model.*

4 Extractable Witness Encryption

In this section, we recall the definition of extractable witness encryption following broadly the definitions of [GKP+13], with the extension to *families* of relations. We then go on to show that one can generically construct it from any extractable witness KEM using the standard KEM/DEM paradigm.

As is the case for regular encryption and KEMs, witness encryption and witness KEMs are very similar, with the only difference being that a witness encryption scheme is capable of encrypting a freely chosen message instead of a random one. The definitions are otherwise very similar to the definitions from the previous section.

Definition 17. *A witness encryption scheme for a family of NP relations* $\mathcal{F}$ *and a messagespace* $\mathcal{M}$ *is a pair of PPT algorithms* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$*, defined as follows:*

$\mathtt{ct} \leftarrow \mathsf{Enc}(I, \mathtt{x}, m)$**:** *The encryption algorithm takes as input an index* I *identifying a relation* $\mathcal{R}_I \in \mathcal{F}$*, a statement* $\mathtt{x}$*, and a message* $m \in \mathcal{M}$ *and returns as output a ciphertext* $\mathtt{ct}$*.*

$m/\bot \leftarrow \mathsf{Dec}(I, \mathtt{w}, \mathtt{ct})$**:** *The deterministic decryption algorithm takes as input an index* I *identifying a relation* $\mathcal{R}_I \in \mathcal{F}$*, a ciphertext* $\mathtt{ct}$*, and a witness* $\mathtt{w}$ *and returns a message* $m \in \mathcal{M}$ *or a error symbol* $\bot$*.*

Definition 18 (Correctness). *A witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a family of NP relations* $\mathcal{F}$ *is correct, if for any* $\mathcal{R}_I \in \mathcal{F}$*, any* $\lambda \in \mathbb{N}$*, any* $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_I$*, any* $m \in \mathcal{M}$*, and any* $\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{x})$ *it holds that* $\mathsf{Dec}(I, \mathtt{w}, \mathtt{ct}) = m$*.*

Definition 19 (Extractability). *A witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a family of NP-relations* $\mathcal{F}$ *is extractable, if there exists a PPT algorithm* Ext *such that for any stateful PPT adversary* $\mathcal{A}$ *and any relation* $\mathcal{R}_I \in \mathcal{F}$ *such that*

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] \geq \frac{1}{2} + \epsilon(\lambda),$$

for some non-negligible function $\epsilon(\lambda)$ *it holds that*

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{c} (\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot)}(I, \mathtt{x}, m_0, m_1) \end{array} \right] \geq \delta(\lambda),$$

for some non-negligible function $\delta(\lambda)$*. The latter probability is taken over the random coins of the adversary and the extractor and the experiment* $\mathsf{Expt}^{\mathsf{CPA}}_{\mathcal{A}}$ *is defined as follows.*

```
$\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I)$
$(\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I)$
$b \leftarrow \{0,1\}$
$\mathtt{ct} \leftarrow \mathsf{Enc}(I, \mathtt{x}, m_b)$
$b' \leftarrow \mathcal{A}(\mathtt{ct})$
return $\begin{cases} 1 & \textit{if } b' = b \\ 0 & \textit{otherwise} \end{cases}$
```

4.1 Extractable Witness Encryption from Extractable Witness KEMs

We can construct extractable witness encryption for a family of NP-relations $\mathcal{F}$ and a message space $\mathcal{M}$ from any extractable witness KEM for $\mathcal{F}$ and any EAV secure symmetric encryptions scheme with message space $\mathcal{M}$, as long as the two schemes share a compatible key space $\mathcal{K}$. The construction, shown in Figure 4 essentially follows the standard KEM/DEM paradigm instantiated with an extractable witness KEM. Even though the security of the KEM/DEM paradigm has been

```
$\mathsf{Enc}(I, \mathtt{x}, m)$                              $\mathsf{Dec}(I, \mathtt{w}, (\mathtt{ct}_1, \mathtt{ct}_2))$
$(\mathtt{ct}_1, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$     $\mathtt{k} := \mathsf{Decap}(I, \mathtt{w}, \mathtt{ct}_1)$
$\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m)$       $m := \mathsf{Dec}^{\mathsf{sym}}(\mathtt{k}, \mathtt{ct}_2)$
return $(\mathtt{ct}_1, \mathtt{ct}_2)$                return $m$
```

Fig. 4. Construction of an extractable witness encryption Scheme for $\mathcal{F}$ based on an extractable witness KEM and an EAV secure symmetric encryption scheme.

proven ad nauseam, we will not skip the proof here. Since we are not proving indistinguishability, but *extractability* we need to be a bit more careful. Note that encapsulated keys are in general *not* indistinguishable from random keys. In fact, since the *adversary* chooses the statement, they may very well *know the witness* and thus be capable of distiguishing with overwhelming probability. It is merely the case in this case *we are capable of extracting the witness.* We need to be careful in our proof that this capability is preserved in the KEM/DEM paradigm. This may not always be the case, depending on how one executes the standard hybrid argument.

Theorem 7. *Let* $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ *be an extractable witness KEM for* $\mathcal{F}$ *and key space* $\mathcal{K}$*. Let* $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ *be an EAV secure symmetric encryption scheme with key space* $\mathcal{K}$*, message space* $\mathcal{M}$*. Then* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *as specified in Figure 4 is an extractable witness encryption scheme for* $\mathcal{F}$ *and message space* $\mathcal{M}$*.*

Proof. We first define a *modified* witness encryption scheme $\widetilde{\mathsf{WE}} = (\widetilde{\mathsf{Enc}}, \cdot)$ that, instead of using the encapsulated key $\mathtt{k}$ to perform the symmetric encryption, chooses a fresh key $\mathtt{k}' \leftarrow \mathcal{K}$ to do so. This scheme has no well-defined decryption algorithm.Nevertheless, for any PPT adversary $\mathcal{A}$, the probability $\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]$ is still well defined. Let $\mathcal{A}$ be an arbitrary PPT adversary such that

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] = \frac{1}{2} + \epsilon(\lambda). \tag{4}$$

for some non-negligible function $\epsilon(\lambda)$. We first prove the following claim.

Claim 8. *For any $I \in \mathcal{I}$, it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1] \leq \tfrac{1}{2} + \mathsf{negl}(\lambda).$$

Proof. We construct a PPT adversary $\mathcal{B}$ against the EAV security of SE as follows. Upon input 1^λ, $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, I)$, receiving $\mathtt{x}, m_0, m_1$ in response, and outputs m_0, m_1. After receiving as input the challenge ciphertext $\mathtt{ct}_2$, it compues $(\mathtt{ct}_1, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$ and invokes $\mathcal{A}((\mathtt{ct}_1, \mathtt{ct}_2))$. In response, $\mathcal{A}$ will output a bit b', which $\mathcal{B}$ also outputs.

It is easy to see that $\mathcal{B}$ perfectly simulates the $\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}}\mathcal{A}}(1^\lambda, I)$ experiment for $\mathcal{A}$. Further, whenever $\mathcal{A}$ would be successful, so is $\mathcal{B}$. It thus follows from the EAV security of SE that

$$\frac{1}{2} + \mathsf{negl}(\lambda) \geq \Pr[\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{B}}(1^\lambda) = 1] = \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]$$

as claimed. □

Claim 9. *There exists a PPT adversary $\mathcal{B}$ such that for any $I \in \mathcal{I}$, it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1] = \tfrac{1}{2} + \tfrac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda).$$

Proof. Upon input 1^λ and I, $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, I)$, receiving $\mathtt{x}, m_0, m_1$ in response and outputs $\mathtt{x}$. After receiving $\mathtt{ct}_1, \mathtt{k}$, $\mathcal{B}$ samples $b' \leftarrow \{0,1\}$, computes $\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_{b'})$, and invokes $\mathcal{A}((\mathtt{ct}_1, \mathtt{ct}_2))$. Eventually $\mathcal{A}$ will output a bit b'' and $\mathcal{B}$ will output 0, if $b'' = b'$ and 1 otherwise.

Let b denote the random bit of the experiment $\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I)$. It is then easy to see that, if $b = 0$, then $\mathcal{B}$ perfectly simulates the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I)$ and outputs 0 iff the experiment outputs 1. Similarly, if $b = 1$, then $\mathcal{B}$ perfectly simulates the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I)$ and outputs 0 iff the experiment outputs 1. It then follows from Claim 8 and Equation 4 that

$$\begin{aligned}
&\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1] \\
=\ &\Pr[b = 0] \cdot \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1 \mid b = 0] + \Pr[b = 1] \cdot \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1 \mid b = 1] \\
=\ &\frac{1}{2} \cdot \left(\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] + 1 - \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]\right) \\
\geq\ &\frac{1}{2} \cdot \left(\frac{1}{2} + \epsilon(\lambda) + 1 - \frac{1}{2} - \mathsf{negl}(\lambda)\right) \\
=\ &\frac{1}{2} + \frac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda)
\end{aligned}$$

□

Finally, since WKEM is known to be extractable and $\frac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda)$ is a non-negligible function whenever $\epsilon(\lambda)$ is non-negligible, there exists a PPT extractor $\widetilde{\mathsf{Ext}}$, such that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} \mathtt{x} \leftarrow \mathcal{B}(1^\lambda, I) \\ \mathtt{w} \leftarrow \widetilde{\mathsf{Ext}}^{\mathcal{B}(\cdot)}(I, \mathtt{x}) \end{array}\right] \geq \delta(\lambda)$$

for some non-negligible function $\delta(\lambda)$. We can thus construct an extractor Ext for WE as follows. The extractor receives as input $(I, \mathtt{x}, m_0, m_1)$ and is given oracle access to $\mathcal{A}$. It then invokes $\widetilde{\mathsf{Ext}}^{\mathcal{B}}(\cdot)(I, \mathtt{x})$. Whenever $\widetilde{\mathsf{Ext}}$ queries $\mathtt{ct}_1$ to its oracle, Ext samples $b' \leftarrow \{0,1\}$, computes $\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_{b'})$, and queries $(\mathtt{ct}_1, \mathtt{ct}_2)$ to its own oracle, forwarding the reply to $\widetilde{\mathsf{Ext}}$. It is easy to see that this perfectly simulates oracle access to $\mathcal{B}$. Therefore, it holds that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{c}(\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot)}(I, \mathtt{x}, m_0, m_1)\end{array}\right] \geq \delta(\lambda)$$

as required. □

5 Laconic OT

As discussed in the introduction, we will now show how to construct a concretely efficient laconic OT protocol from our extractable witness encryption scheme for KZG commitments and openings.

Definition 20 ([CDG+17]). *A laconic oblivious transfer scheme is a tuple of PPT algorithms* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$**:** *The setup algorithm takes security parameter λ and database length n as input and returns public parameters* pp.

$(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$**:** *The hashing algorithm takes public parameters* pp *and database D as input and returns a public hash* $\mathtt{digest}$ *and some secret auxiliary information* $\mathtt{aux}$.

$c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$**:** *The sender algorithm takes public parameters* pp, *database hash* $\mathtt{digest}$, *index i, and message m_0 and m_1 as input and returns message c.*

$m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathtt{aux}, c, i)$**:** *The receiver algorithm takes public parameters* pp, *auxiliary information* $\mathtt{aux}$, *sender message m, and index i as input and returns message m.*

Definition 21 (Correctness). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is correct, if for all $\lambda, n \in \mathbb{N}$ with $n = \mathsf{poly}(\lambda)$, all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, all databases $D \in \{0,1\}^n$, all $(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$, all $i \in [n]$, all $m_0, m_1 \in \mathcal{M}$, and all $c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$ it holds that $\mathsf{Receive}(\mathsf{pp}, \mathtt{aux}, c, i) = m_{D[i]}$.*

Sender privacy requires the sender's message to the receiver to hide the message that was not selected by the receiver's choice bit.

Definition 22 (Sender Privacy). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is sender private against semi-honest adversaries, if for all $\lambda, n \in \mathbb{N}$ with $n = \mathsf{poly}(\lambda)$, any PPT adversary $\mathcal{A}$, any database $D \in \{0,1\}^n$, any $i \in [n]$, and any pair of messages $m_0, m_1 \in \mathcal{M}$, it holds that*

$$\left|\begin{array}{l}\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] \\ - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]\end{array}\right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}$ *and* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}$ *are defined as follows*

```
$\mathsf{Expt}_{\mathsf{LOT},\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}(1^\lambda, 1^n, D, m_0, m_1, i)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$
$c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$
$b \leftarrow \mathcal{A}(\mathsf{pp}, c, \mathtt{aux})$
return $b$
```

```
$\mathsf{Expt}_{\mathsf{LOT},\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}(1^\lambda, 1^n, D, m_0, m_1, i)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$c \leftarrow \mathsf{Sim}(\mathsf{pp}, D, i, m_{D[i]})$
$b \leftarrow \mathcal{A}(\mathsf{pp}, c, \mathtt{aux})$
return $b$
```

In the original work of Cho et al. [CDG+17], no explicit definition for receiver privacy was stated. The authors informally argued that any protocol that is not receiver privacy can be transformed into one that is, via the use of generic secure computation. In our work, we do define receiver privacy and we focus on the arguably strongest possible notion, namely that of perfect receiver privacy, where the sender learns no information about the receivers choice bits in the information-theoretic sense. We do not make use of generic secure computation and instead our construction will directly satisfy this security notion.

Definition 23 (Receiver Privacy). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is receiver private against semi-honest adversaries, if for all* $\lambda, n \in \mathbb{N}$ *with* $n = \mathsf{poly}(\lambda)$*, any PPT adversary* $\mathcal{A}$*, all databases* $D \in \{0,1\}^n$*, it holds that*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D) = 1] - \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathcal{A}}(1^\lambda, 1^n, D) = 1]\right|,$$

where $\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}$ *and* $\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}$ *are defined as follows.*

```
$\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$
$b \leftarrow \mathcal{A}(\mathsf{pp}, \mathtt{digest})$
return $b$
```

```
$\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$\mathtt{digest} \leftarrow \mathsf{Sim}(\mathsf{pp})$
$b \leftarrow \mathcal{A}(\mathsf{pp}, \mathtt{digest})$
return $b$
```

The main property that makes laconic OT non-trivial and interesting is its efficiency. The digest is required to be independent of the original database size, the hashing should run in quasi-linear time in the database size, and both the computational complexity of sending and receiving should have at most a polylogarithmic dependency on the size of the database.

Definition 24 (Efficiency). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is efficient, if* $|\mathtt{digest}| \in \mathsf{poly}(\lambda)$ *and in particular independent of* $|D|$*, the hashing algorithm* H *runs in time* $|D| \cdot \mathsf{poly}(\log|D|, \lambda)$*, and both* Send *and* $\mathsf{Receive}$ *run in time* $\mathsf{poly}(\log|D|, \lambda)$*.*

5.1 Constructing Laconic OT

Our construction is conceptually very simple. The receiver will use a hiding vector commitment to compute a commitment $\mathtt{com}$ to their database D. Upon the i-th invocation of the OT, the sender will use our extractable witness encryption scheme to separately encrypt m_0 and m_1, such that they can be decrypted, if $\mathtt{com}$ can be opened to 0 and 1, respectively. The sender, who will receive the two ciphertexts, can then use their opening π_i for commitment $\mathtt{com}$ to decrypt the ciphertext containing $m_{D[i]}$. Sender privacy effectively follows from the security guarantees of the extractable witness encryption scheme, whereas receiver privacy follows from the hiding properties of the vector commitment.

Theorem 10. *Let* $\lambda, n \in \mathbb{N}$ *with* $n = \mathsf{poly}(\lambda)$*. Let* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *be a correct, efficient, position binding, and perfectly weakly-hiding vector commitment. Let* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *be an extractable witness encryption scheme for* $\mathcal{F}_{\mathsf{VC}}$*. Then the construction in Figure 5 is an efficient, sender-private, and perfectly receiver-private laconic OT.*

```
$\mathsf{Setup}(1^\lambda, 1^n)$
    $\mathtt{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$

$\mathsf{H}(\mathtt{pp}, D)$
    $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathtt{pp}, D)$
    $(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathtt{pp}, \overline{\mathtt{aux}})$
    return $(\mathtt{com}, (D, \pi_1, \ldots, \pi_n))$

$\mathsf{Send}(\mathtt{pp}, \mathtt{digest}, i, m_0, m_1)$
    $\mathtt{ct}_0 \leftarrow \mathsf{WE.Enc}(\mathtt{pp}, (\mathtt{digest}, i, 0), m_0)$
    $\mathtt{ct}_1 \leftarrow \mathsf{WE.Enc}(\mathtt{pp}, (\mathtt{digest}, i, 1), m_1)$
    return $(\mathtt{ct}_0, \mathtt{ct}_1)$

$\mathsf{Receive}(\mathtt{pp}, \mathtt{aux}, (\mathtt{ct}_0, \mathtt{ct}_1), i)$
    parse $\mathtt{aux}$ as $(D, \pi_1, \ldots, \pi_n))$
    $b := D_i$
    $m_b \leftarrow \mathsf{WE.Dec}(\mathtt{pp}, \pi_i, \mathtt{ct}_b)$
    return $m_b$
```

Fig. 5. Laconic OT construction.

Proof. Efficiency follows immediately from the efficiency of VC. We will prove the other two properties seperately.

Lemma 11. *Let* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *be a correct and position binding vector commitment. Let* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *be an extractable witness encryption scheme for* $\mathcal{F}_{\mathsf{VC}}$*. Then the construction in Figure 5 is sender private.*

Proof. We specify a simulator as follows. On input $\mathtt{pp}, D, i, m_{D[i]}$ the simulator simply computes

$$\begin{aligned} &\mathtt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathtt{pp}, (\mathtt{digest}, i, D[i]), m_{D[i]}) \\ \text{and} \quad &\mathtt{ct}_{1-D[i]} \leftarrow \mathsf{WE.Enc}(\mathtt{pp}, (\mathtt{digest}, i, 1 - D[i]), 0) \end{aligned}$$

and outputs $(\mathtt{ct}_0, \mathtt{ct}_1)$.

Let $D \in \{0,1\}^n$ be an arbitrary database, $i \in [n]$ an arbitrary index, $m_0, m_1 \in \mathbb{F}$ an arbitrary pair of messages, and $\mathcal{A}$ an arbitrary PPT adversary against sender privacy with

$$\begin{aligned} \epsilon(\lambda) = \; &\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] \\ &- \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]. \end{aligned} \tag{5}$$

Assume wlog, that $\epsilon(\lambda) \geq 0$.

We construct a PPT adversary $\mathcal{B}$ against the extractability of WE as follows. On input $1^\lambda, \mathsf{ck}$, the adversary $\mathcal{B}$ computes $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathtt{pp}, D)$ and outputs

$$\big(\underbrace{(\mathtt{com}, i, 1 - D[i])}_{=:\mathrm{x}}, 0, m_{1-D[i]}\big).$$

Upon receiving $\mathtt{ct}_{1-D[i]}$, $\mathcal{B}$ further computes $\mathtt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{ck}, (\mathtt{digest}, i, D[i]), m_{D[i]})$, sets $\mathtt{aux} := (D, \mathtt{com}, \overline{\mathtt{aux}})$ and invokes $\mathcal{A}(\mathsf{ck}, (\mathtt{ct}_0, \mathtt{ct}_1), \mathtt{aux})$. Eventually $\mathcal{A}$ will output a bit, which $\mathcal{B}$ will also output.

Let $b \in \{0,1\}$ be the bit in the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{B}}(1^\lambda)$. It is easy to see that, if $b = 1$, then $\mathcal{B}$ perfectly simulates $\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i)$. On the other hand, if $b = 0$, then $\mathcal{B}$ perfectly simulates $\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i)$. It thus follows that

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{B}}(1^\lambda, \mathsf{ck}) = 1]$$

$$
\begin{aligned}
&= \frac{1}{2}\big(\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda) = 1 \mid b = 1] + \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda) = 1 \mid b = 0]\big)\\
&= \frac{1}{2}\big(\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] + \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 0]\big)\\
&= \frac{1}{2}\big(\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] + 1 - \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]\big)\\
&= \frac{1}{2} + \frac{1}{2}\epsilon(\lambda).
\end{aligned}
$$

Since WE is extractable, it follows that there exists a PPT algorithm Ext, such that, if $\epsilon(\lambda)$, and thereby also $\epsilon(\lambda)/2$ were non-negligible, it would hold that

$$
\Pr\left[\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, i, 1 - D[i]) = 1 : \begin{array}{r} ((\mathtt{com}, i, 1 - D[i]), 0, m_{1-D[i]}) \leftarrow \mathcal{B}(1^\lambda, \mathsf{ck}) \\ \pi \leftarrow \mathsf{Ext}^{\mathcal{B}(\cdot)}(\mathsf{ck}, (\mathtt{com}, i, 1 - D[i]), 0, m_{1-D[i]}) \end{array}\right] \geq \delta(\lambda)
$$

for some non-negligible function $\delta(\lambda)$.

To show that this can't be the case, we construct an adversary $\mathcal{C}$ against the position binding of VC as follows. On input ck, the adversary $\mathcal{C}$ computes $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathtt{pp}, D)$, and invokes $\mathsf{Ext}^{\mathcal{B}(\cdot)}((\mathtt{com}, i, 1-D[i]), 0, m_{1-D[i]})$. Whenever Ext queries its oracle with $\mathtt{ct}_{1-D[i]}$, $\mathcal{C}$ further computes $\mathtt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{ck}, (\mathtt{digest}, i, D[i]), m_{D[i]})$, sets $\mathtt{aux} := (D, \mathtt{com}, \overline{\mathtt{aux}})$, invokes $b \leftarrow \mathcal{A}(\mathsf{ck}, (\mathtt{ct}_0, \mathtt{ct}_1), \mathtt{aux})$ and replies with b. Eventually, Ext will output π. $\mathcal{C}$ will then compute $\pi' \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \overline{\mathtt{aux}}, i)$ and output $\mathtt{com}, i, 1 - D[i], D[i], \pi, \pi'$.

It is easy to see, that $\mathcal{C}$ perfectly simulates $\mathcal{B}$ for the extractor. Therefore, it holds with probability at least $\delta(\lambda)$, that $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, i, 1-D[i]) = 1$. The correctness of the vector commitment guarantees that $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi', i, D[i]) = 1$. Since further $D[i] \neq 1 - D[i]$, it thus holds that

$$
\mathsf{negl}(\lambda) \geq \Pr\left[\begin{array}{c} m_0 \neq m_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_0, i, m_0) = 1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_1, i, m_1) = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathtt{com}, i, m_0, m_1, \pi_0, \pi_1) \leftarrow \mathcal{C}(\mathsf{ck}) \end{array}\right] = \delta(\lambda)
$$

Which immediately implies that $\epsilon(\lambda) \leq \mathsf{negl}(\lambda)$ as required. □

Lemma 12. *Let* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *be a perfectly weakly hiding vector commitment. Then the construction in Figure 5 is perfectly receiver private.*

Proof. The simulator takes as input the public parameters $\mathtt{pp} = \mathsf{ck}$, computes $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathsf{ck}, 0^n)$ and outputs $\mathtt{com}$. The only difference between the two experiments from the point of view of an adversary is that in one case they receive a commitment to D, whereas in the other they receive a commitment to 0^n. However, since VC is perfectly weakly hiding, those two commitments are distributed identically. □

Now Theorem 10 follows directly by combining Lemmas 11 and 12. □

6 Benchmarks

The laconic OT protocol from Section 5 was instantiated with the KZG based vector commitment from Section 2.5 and the extractable witness commitment derived by combining the extractable witness KEM from Section 3 with the generic transformation from Section 4. A simple one-time pad as the symmetric encryption scheme in the transformation. The full construction was implemented[6] in Rust using `arkworks` [ac22]. The `BLS12-381` [Bow17] curve was used as the pairing-friendly elliptic curve throughout the implementation. All involved computational costs and bandwidth overheads were measured for various parameters. All benchmarks were run on a personal laptop with an `i7-11800H @ 2.30GHz` CPU and 64 GB of RAM.

As already explained in Section 2.5, the batch opening technique of Feist and Khovratovich was used [FK23] to precompute all openings of the used vector commitment during the hashing of the database of choice indices. For concreteness the sender's OT inputs are assumed to be 256-bit messages. A single witness encryption for such a message consists of a 96 byte witness KEM ciphertext and a 32 bytes large one-time pad encryption of the message itself, thus leading to a 128 byte ciphertext. The public parameters for a database of size n consists of $n+1$ many $\mathbb{G}_1$ elements and two $\mathbb{G}_2$ elements with $\mathbb{G}_1$ and $\mathbb{G}_2$ for BLS12-381 clocking in at 48 and 96 bytes respectively. The benchmark results can be found in Table 1.

$\lvert D\rvert$	Sizes			Times		
	`pp`	`digest`	Sender Msg.	Hash	Send	Receive
2^6	3.2 KB	48 B	256 B	173 ms	4 ms	1 ms
2^8	12.2 KB	48 B	256 B	723 ms	4 ms	1 ms
2^{10}	48.2 KB	48 B	256 B	3 s	4 ms	1 ms
2^{12}	192.2 KB	48 B	256 B	10 s	4 ms	1 ms
2^{14}	768.2 KB	48 B	256 B	43 s	4 ms	1 ms
2^{16}	3.0 MB	48 B	256 B	3 min	5 ms	1 ms
2^{18}	12.0 MB	48 B	256 B	8 min	5 ms	1 ms
2^{31}	96.0 GB	48 B	256 B	—	5 ms	1 ms

Table 1. Runtimes and bandwidth overheads of our laconic OT protocol for varying sizes of the receiver's database D. The computation times for $|D| = 2^{31}$ are extrapolated to compare with [GJL23].

Our construction is highly efficient in most parameters, in particular database digest and the sender's message are both not just constant in size but concretely very small. The time to compute the sender's message and to decode on the receiver's side in an OT invocation is below 5 ms. The main drawbacks of our construction the initial one-time cost of computing the hash of the receiver's database and the size of the public parameters.

Weakly Laconic OT. We also examine the efficiency of a variant of our construction, which does not strictly satisfy the formal efficiency requirements for laconic OT protocols from Definition 24 but has significantly smaller public parameters at the cost of somewhat larger digests. The idea for achieving this trade-off is to simply partition the receiver's database D into $\sqrt{|D|}$ smaller databases and to then hash each of them individually. The hash returned by the receiver is simply the concatenation of all $\sqrt{|D|}$ hashes. This construction does not formally satisfy the asymptotic efficiency requirement of a laconic OT because the digest size now (sublinearly) depends on the size of the input database. The main observation here is that all of those hashes can be computed using the same trusted setup for instances of size $\sqrt{|D|}$. This simple trick was already observed by Green,

[6] https://github.com/rot256/research-we-kzg

$\lvert D\rvert$	Sizes			Times		
	`pp`	`digest`	Sender Msg.	Hash	Send	Receive
2^6	624 B	384 B	256 B	194 ms	4 ms	1 ms
2^8	1008 B	768 B	256 B	743 ms	4 ms	1 ms
2^{10}	1.7 KB	1.5 KB	256 B	3 s	4 ms	1 ms
2^{12}	3.2 KB	3.0 KB	256 B	11 s	4 ms	1 ms
2^{14}	6.2 KB	6.0 KB	256 B	44 s	4 ms	1 ms
2^{16}	12.2 KB	12.0 KB	256 B	3 min	4 ms	1 ms
2^{18}	24.2 KB	24.0 KB	256 B	12 min	4 ms	1 ms
2^{31}	3 MB	1.5 MB	256 B	69 days	4 ms	1 ms

Table 2. Runtimes and bandwidth overheads for the variant of our laconic OT protocol, which splits the database D into $\sqrt{|D|}$ many smaller databases. The numbers are extrapolated from our experimental results for laconic OT.

Jain, and Van Laer in their work [GJL23]. We calculate the sizes of objects and we extrapolate the timing benchmarks for this construction from our benchmarks for regular laconic OT. The results can be found in Table 2.

Comparison to Related Works. Equipped with the concrete performance numbers for our constructions, we can compare its efficiency to that of existing protocols. Green, Jain, Van Laer [GJL23] provide benchmarks for a single data point for their laconic OT construction, which does not achieve sender privacy. They argue that one can generically obtain sender privacy by using garbled circuits, which would incur a significant overhead on top of their provided benchmarks. In their favour, we shall ignore this and simply compare it to our construction that achieves both sender and receiver privacy.

For a database size of 2^{31}, their construction's public parameters 412.3 GB, their digest is 48 B, their sender's message size is 1.34 KB and receiving takes 27.7 *minutes*. That means their public parameters are $\approx$ 4x larger, their digests have the same size, their sender's message is $\approx$ 5x larger, and their receiving time is larger by 6 *orders of magnitude*. No further data points were provided in their benchmarks.

The authors also provide benchmark results for a weakly laconic version of their construction, which splits the database in $\sqrt{|D|}$ chunks as described above. Comparing it to our weakly laconic OT construction, they have $\approx$ 3x larger public parameters, comparable digest size, and $\approx$ 5x larger sender's message. Their receiver's computational time is still $\approx$ 38x larger.

Döttling et al. [DKL+23] only sketch how to construct laconic OT from another cryptographic primitive, which they do provide some benchmarks for. The main advantage of their work is that the public parameters do not grow with the receiver's database size. Taking the benchmarks for their lowest security levels, their public parameters are 8 MB, their digest is 2 KB, and their sender's message size is 98 MB. The constant size of their public parameters makes them concretely smaller than ours once the databases size reaches 2^{18}. Their digests are always $\approx$ 8x larger and their sender's message size is 5 *orders of magnitude* larger.

Comparing with our weakly laconic OT protocol, our public parameters would only become concretely larger than theirs, when the database size reaches at least 2^{35}. For a database size of 2^{18}, our digest is $\approx$ 12x larger, but our sender's message is still 5 orders of magnitude smaller. We believe that a slightly larger digest size in exchange for much smaller sender messages is a valuable trade-off. The digest is sent once, but the sender's messages need to be send for each OT invocation.

References

ac22. arkworks contributors. `arkworks` zksnark ecosystem, 2022. URL: https://arkworks.rs. 6

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_1. 1.2

BJKL21. Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 724–753, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77886-6_25. 1.2

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_17. 1.2

BL20. Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 349–378, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64378-2_13. 1.2

Bow17. Sean Bowe. Bls12-381: New zk-snark elliptic curve construction, March 2017. URL: https://electriccoin.co/blog/new-snark-curve/. 6

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. doi:10.1007/978-3-319-63715-0_2. 1.1, 1.1, 1.2, 20, 5

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 55–72, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-36362-7_5. 1

CFK22. Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. Cryptology ePrint Archive, Report 2022/1510, 2022. https://eprint.iacr.org/2022/1510. 1.2

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45721-1_26. 1, 1

CV21. Gwangbae Choi and Serge Vaudenay. Towards witness encryption without multilinear maps - extractable witness encryption for multi-subset sum instances with no small solution to the homogeneous problem. In Jong Hwan Park and Seung-Hyun Seo, editors, *ICISC 21: 24th*, volume 13218, pages 28–47, Seoul, South Korea, December 1–3 2021. doi:10.1007/978-3-031-08896-4_2. 3

DKL+23. Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 417–446, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30620-4_14. 1.2, 6

FK23. Dankrad Feist and Dmitry Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Report 2023/033, 2023. https://eprint.iacr.org/2023/033. 2.4, 2.5, 6

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96881-0_2. 1, 2.1

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. doi:10.1109/FOCS.2013.13. 1.2

GGHW14. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_29. 1.2

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. doi:10.1145/2488608.2488667. 1.1, 1.1, 1.2

GJL23. Matthew Green, Abhishek Jain, and Gijs Van Laer. Efficient set membership encryption and applications. pages 1080–1092, Copenhagen, Denmark, November 26–30, 2023. doi:10.1145/3576915.3623131. 1.2, 1, 6, 6

GKP$^+$13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_30. 1.1, 1.1, 1.2, 4

GRWZ20. Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 2007–2023, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3417244. 1

GS17. Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 588–599, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press. doi:10.1109/FOCS.2017.60. 1.2

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953. 1

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-17373-8_11. 1

LPR22. Benoît Libert, Alain Passelègue, and Mahshid Riahinia. PointProofs, revisited. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 220–246, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-22972-5_8. 1

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2111–2128, London, UK, November 11–15, 2019. ACM Press. doi:10.1145/3319535.3339817. 1

Rab81. Michael O. Rabin. How to exchange secrets with oblivious transfer. *Technical Report TR-81, Aiken Computation Lab, Harvard University,*, 1981. URL: http://eprint.iacr.org/2005/187. 1.1

SCP$^+$22. Shravan Srinivasan, Alexander Chepurnoy, Charalampos Papamanthou, Alin Tomescu, and Yupeng Zhang. Hyperproofs: Aggregating and maintaining proofs in vector commitments. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 3001–3018, Boston, MA, USA, August 10–12, 2022. USENIX Association. 1

TAB$^+$20. Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir

Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 45–64, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-57990-6_3. 1

Tsa22. Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 535–559, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15802-5_19. 1.2

VWW22. Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 195–221, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-22963-3_7. 1.2

WUP23. Weijie Wang, Annie Ulichney, and Charalampos Papamanthou. Balanceproofs: Maintainable vector commitments with fast aggregation. pages 4409–4426, Anaheim, CA, USA, August 9–11, 2023. 1