Constant-Size Commitments to Polynomials and Their Applications*

Aniket Kate[1], Gregory M. Zaverucha[2], and Ian Goldberg[3]

[1] Max Planck Institute for Software Systems (MPI-SWS)
[2] Certicom Research
[3] University of Waterloo

Abstract. We introduce and formally define polynomial commitment schemes, and provide two efficient constructions. A polynomial commitment scheme allows a committer to commit to a polynomial with a short string that can be used by a verifier to confirm claimed evaluations of the committed polynomial. Although the homomorphic commitment schemes in the literature can be used to achieve this goal, the sizes of their commitments are linear in the degree of the committed polynomial. On the other hand, polynomial commitments in our schemes are of constant size (single elements). The overhead of opening a commitment is also constant; even opening multiple evaluations requires only a constant amount of communication overhead. Therefore, our schemes are useful tools to reduce the communication cost in cryptographic protocols. On that front, we apply our polynomial commitment schemes to four problems in cryptography: verifiable secret sharing, zero-knowledge sets, credentials and content extraction signatures.

Keywords: Polynomial Commitments, Verifiable Secret Sharing, Zero-Knowledge Sets, Credentials

1 Introduction

Commitment schemes are fundamental components of many cryptographic protocols. A commitment scheme allows a committer to publish a value, called the *commitment*, which binds her to a message (*binding*) without revealing it (*hiding*). Later, she may *open* the commitment and reveal the committed message to a verifier, who can check that the message is consistent with the commitment.

We review three well-known ways a committer can commit to a message. Let g and h be two random generators of a group $\mathbb{G}$ of prime order p. The committer can commit to a message $m \in_R \mathbb{Z}_p$ simply as $\mathcal{C}_{\langle g \rangle}(m) = g^m$. This scheme is unconditionally binding, and computationally hiding under the assumption that the discrete logarithm (DL) problem is hard in $\mathbb{G}$. The second scheme, known as a Pedersen commitment [31], is of the form $\mathcal{C}_{\langle g,h \rangle}(m, r) = g^m h^r$, where $r \in_R \mathbb{Z}_p$. Pedersen commitments are unconditionally hiding, and computationally binding

* An extended version of this paper is available [24]. This research was completed at the University of Waterloo.

under the DL assumption. Third, the committer may publish $H(m)$ or $H(m||r)$ for any one-way function H. In practice a collision-resistant hash function is often used. A survey by Damgård [16] covers the commitment schemes in detail.

Now consider committing to a polynomial $\phi(x) \in_R \mathbb{Z}_p[x]$, a problem motivated by verifiable secret sharing. Suppose $\phi(x)$ has degree t and coefficients $\phi_0, \ldots, \phi_t$. We could commit to the string $(\phi_0|\phi_1|\ldots|\phi_t)$, or to some other unambiguous string representation of $\phi(x)$. Based on the commitment function used, this option may have a constant size commitment which uniquely determines $\phi(x)$. However, it limits the options for opening the commitment; opening must reveal the entire polynomial. This is not always suitable for cryptographic applications, most notably secret sharing, that require evaluations of the polynomial (*i.e.*, $\phi(i)$ for $i \in \mathbb{Z}_p$) be revealed to different parties or at different points in the protocol *without* revealing the entire polynomial. One solution is to commit to the coefficients, *e.g.*, $C = (g^{\phi_0}, \ldots, g^{\phi_t})$, which allows one to easily confirm that an opening $\phi(i)$ for index i is consistent with C. However, this has the drawback that the size of the commitment is now $t + 1$ elements of $\mathbb{G}$.

Our Contributions. The main contribution of this paper is an efficient scheme to commit to polynomials $\phi(x) \in \mathbb{Z}_p[x]$ over a bilinear pairing group, called $\mathsf{PolyCommit}_{\mathrm{DL}}$, with the following features. The size of the commitment is constant, a single group element. The committer can efficiently open the commitment to any correct evaluation $\phi(i)$ along with an element called the *witness*, which allows a verifier to confirm that $\phi(i)$ is indeed the evaluation at i of the polynomial $\phi(x)$. The construction is based on an algebraic property of polynomials $\phi(x) \in \mathbb{Z}_p[x]$ that $(x-i)$ *perfectly* divides the polynomial $\phi(x)-\phi(i)$ for any $i \in \mathbb{Z}_p$. The hiding property of the scheme is based on the DL assumption. The binding property of the main scheme is proven under the SDH assumption [6]. Using a technique similar to Pedersen commitments, we also define a stronger commitment scheme $\mathsf{PolyCommit}_{\mathsf{Ped}}$, which achieves unconditional hiding and computational binding under the SDH assumption.

When a set of evaluations $\{\phi(i) : i \in S\}$ is opened at the same time, what we term *batch opening*, the overhead still remains a *single* witness element. Security of batch opening assumes that the bilinear version of the SDH (BSDH) problem [21] is hard. Further, our schemes are homomorphic and easily randomizable. As in other work on reducing communication costs (*e.g.*, [8]) the global system parameters are somewhat large ($O(t)$ in our case). Reducing communication complexity (*i.e.*, the number of bits transferred) is our goal, and to this end we apply the $\mathsf{PolyCommit}$ schemes to the following four applications.

First we apply the $\mathsf{PolyCommit}$ schemes to the Feldman verifiable secret sharing (VSS) protocol [18]. The new VSS protocol requires a broadcast with size $O(1)$ as compared to $O(n)$ required in the best known protocols in the literature (where n is the number of participants) [18, 31].

Second, we define and use the $\mathsf{PolyCommit}$ schemes to construct a relaxed type of zero-knowledge set (ZKS) [27]. A ZKS is a commitment to a set S, such that the committer may prove that $i \in S$, or $i \notin S$ without revealing additional information about S, not even $|S|$. We define *nearly zero-knowledge sets* as ZKS that do not attempt to hide the size of the committed set. This is sufficient for most applications of zero-knowledge sets, and our construction has constant size proofs of (non)membership as compared to the best known constructions of ZKS that require non-constant communication [12, 25]. We apply the same relaxation to elementary zero-knowledge databases (ZK-EDB), and achieve constant communication there as well.

In the next application we leverage the efficiency of batch opening, by using the PolyCommit schemes in an efficient general construction of a *content extraction signature* (CES) scheme [35]. A CES scheme allows a signature holder to extract signatures for subsections of the signed message. The general construction, when instantiated with our commitment scheme and a general secure signature scheme, is as efficient as the best known CES scheme, which relies on specific properties of the RSA signature scheme.

In the special case when the CES scheme is used to authenticate a list of attributes, the result is a digital credential with an efficient *selective show* operation. A selective show allows the credential holder to reveal only a subset of the attributes, with proof that the revealed attributes are signed. More precisely, the communication cost of revealing k attributes in a credential with t attributes is $O(k)$, while known credential systems must communicate $O(t)$ bits. We also show how to efficiently prove knowledge of committed values, allowing predicates on attributes to be proven in zero-knowledge (also with complexity $O(k)$).

Outline. In the rest of this section, we compare our contributions with related work (work related to each application is included in the respective subsection). In §2, we cover some preliminary material and describe our cryptographic assumptions. §3 defines polynomial commitments and presents our constructions. §4 is devoted to applications while §5 presents some open problems. Due to space constraints, all security proofs are included in the extended version [24].

Related Work. Similar to our scheme, a Merkle hash tree [26] allows many values to be committed to with a single element. Here, the leaves of a binary tree are the messages. Each non-leaf node has the value $H(L||R)$ where L and R are its children, and H is a collision-resistant hash function. One can open the commitment to an individual message by revealing the message, and a path up the tree to the root. The opening has size $O(\log n)$ as compared to $O(1)$ in our scheme, where n is the total number of (leaf) elements.

Chase *et al.* [13] introduce mercurial commitments to construct ZKS, which eventually led to the commitment schemes for committing to a vector of messages [12, 25]. Catalano *et al.* [12], and Libert and Yung [25] construct vector commitment schemes under the name *trapdoor t-mercurial commitments*. The security of both of these commitment schemes is based on SDH-like assumptions and their system parameters have size $O(t)$, as in our scheme. In [12], all messages must be revealed when opening, while in [25], the committer may open a commitment to a single message. However, in [25], it is not possible to have arbitrary indices for committed messages since each of the t committed messages is associated with a value in the system parameters g^{α^j} for $j \in [1, t]$. Our scheme have no such restriction on the domain for the indices, offering greater flexibility.

Another related primitive is an *accumulator* [3], which aggregates a large set of input elements into a single element and can provide a witness as evidence that an element is included in the accumulator. Further, it is possible to use a witness to prove (in zero-knowledge) that the element is included in the accumulator. Camenisch and Lysyanskaya [10] extend the concept to *dynamic accumulators*, which support efficient updates. Au *et al.* [1] observe that a paring-based accumulator by Nguyen [29] is a *bounded* accumulator, *i.e.*, only a fixed number of elements can be accumulated. They then go on to use bounded accumulators to construct a compact e-cash scheme [2]. However, the accumulated elements in this scheme are not ordered, which makes it inappropriate for accumulating polynomials. While our PolyCommit schemes provide the same features as non-dynamic accumulators, they have additional features (hiding and batch opening) and are more general since we can commit to a polynomial instead of a set.

2 Preliminaries

In what follows, all adversaries are probabilistic polynomial time (PPT) algorithms with respect to a security parameter κ expect if stated otherwise. Further, they are *static* and they have to choose their nodes before protocol instances start. A function $\epsilon(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there exists a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the rest of the paper, $\epsilon(\cdot)$ will always denote a negligible function. We use the notation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ to denote a symmetric (type 1) bilinear pairing in groups of prime order $p \geq 2^{2\kappa}$. The choice of type 1 pairings was made to simplify presentation, however, our constructions can easily be modified to work with pairings of types 2 and 3 as well. For details of bilinear pairings, see the extended version of the paper.

We use the *discrete logarithm* (DL) assumption [26, Chap. 3], and the *t-strong Diffie-Hellman* (t-SDH) assumption [6] to prove the security of the $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ schemes. Security of two additional properties of the schemes require a generalization of the *t-Diffie-Hellman inversion* (t-DHI) assumption [28, 5], and the bilinear version of t-SDH, the t-BSDH assumption [21].

Definition 1. Discrete logarithm (DL) Assumption. *Given a generator g of $\mathbb{G}^*$, where $\mathbb{G}^* = \mathbb{G}$ or $\mathbb{G}_T$, and $a \in_R \mathbb{Z}_p^*$, for every adversary $\mathcal{A}_{DL}$,* $\Pr[\mathcal{A}_{DL}(g, g^a) = a] = \epsilon(\kappa)$.

Mitsunari, Sakai and Kasahara [28] introduced the *weak Diffie-Hellman assumption*, which was renamed the t-DHI assumption by Boneh and Boyen [5] as this assumption is stronger than the Diffie-Hellman assumption, especially for large values of t. See Cheon [14] for a security analysis.

The t-DHI problem is, on input $\langle g, g^{\alpha}, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ to output $g^{1/\alpha}$, or equivalently (see [7]), $g^{\alpha^{t+1}}$. In this paper, we use a generalization of the t-DHI assumption, where $\mathcal{A}$ has to output a pair $\langle \phi(x), g^{\phi(\alpha)} \rangle \in \mathbb{Z}_p[x] \times \mathbb{G}$ such that

$2^\kappa > \deg(\phi) > t$. We call this assumption the *t-polynomial Diffie-Hellman* (*t*-polyDH) assumption. This assumption was implicitly made by [1, 2] to support their claim that the accumulator of [29] is bounded.[4]

Definition 2. t-polynomial Diffie-Hellman (t-polyDH) Assumption. *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-polyDH}}$, the probability $\Pr[\mathcal{A}_{t\text{-polyDH}}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle \phi(x), g^{\phi(\alpha)} \rangle] = \epsilon(\kappa)$ for any freely chosen $\phi(x) \in \mathbb{Z}_p[x]$ such that $2^\kappa > \deg(\phi) > t$.*

Boneh and Boyen [6] defined the t-SDH assumption that is related to but stronger than the t-DHI assumption and has exponentially many non-trivially different solutions, all of which are acceptable.

Definition 3. t-Strong Diffie-Hellman (t-SDH) Assumption. *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-SDH}}$, the probability $\Pr[\mathcal{A}_{t\text{-SDH}}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle c, g^{\frac{1}{\alpha+c}} \rangle] = \epsilon(\kappa)$ for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.*

Security of the batch opening extension of our commitment schemes requires the bilinear version of the t-SDH assumption, the t-BSDH assumption [21].

Definition 4. t-Bilinear Strong Diffie-Hellman (t-BSDH) Assumption. *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-BSDH}}$, the probability $\Pr[\mathcal{A}_{t\text{-BSDH}}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle c, e(g, g)^{\frac{1}{\alpha+c}} \rangle] = \epsilon(\kappa)$ for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.*

A similar assumption was also made in [22], but with a different solution: $\langle c, e(g, h)^{1/(\alpha+c)} \rangle$, where $h \in_R G$ is an additional system parameter.

3 Polynomial Commitments

In this section we provide a formal definition of a polynomial commitment scheme, followed by two constructions. In the first construction the commitments are computationally hiding, while in the second they are unconditionally hiding. We also discuss some useful features of our constructions.

3.1 Definition

A polynomial commitment scheme consists of six algorithms: Setup, Commit, Open, VerifyPoly, CreateWitness, and VerifyEval.

[4] Note that we bound $\deg(\phi)$ by 2^κ as evaluations can be found for polynomials with higher degrees in PPT using number theoretic techniques (*e.g.*, for $\phi(x) = x^{p-1}$, $g^{\phi(\alpha)} = g$ for any $\alpha \in \mathbb{Z}_p^*$). In practice, $\deg(\phi) \ll 2^\kappa$.

Setup$(1^\kappa, t)$ generates an appropriate algebraic structure $\mathcal{G}$ and a commitment public-private key pair $\langle \mathsf{PK}, \mathsf{SK}\rangle$ to commit to a polynomial of degree $\leq t$. For simplicity, we add $\mathcal{G}$ to the public key PK. Setup is run by a trusted or distributed authority. Note that SK is not required in the rest of the scheme.

Commit$(\mathbf{PK}, \phi(x))$ outputs a commitment $\mathcal{C}$ to a polynomial $\phi(x)$ for the public key PK, and some associated decommitment information d. (In some constructions, d is null.)

Open$(\mathbf{PK}, \mathcal{C}, \phi(x), d)$ outputs the polynomial $\phi(x)$ used while creating the commitment, with decommitment information d.

VerifyPoly$(\mathbf{PK}, \mathcal{C}, \phi(x), d)$ verifies that $\mathcal{C}$ is a commitment to $\phi(x)$, created with decommitment information d. If so it outputs 1, otherwise it outputs 0.

CreateWitness$(\mathbf{PK}, \phi(x), i, d)$ outputs $\langle i, \phi(i), w_i\rangle$, where w_i is a witness for the evaluation $\phi(i)$ of $\phi(x)$ at the index i and d is the decommitment information.

VerifyEval$(\mathbf{PK}, \mathcal{C}, i, \phi(i), w_i)$ verifies that $\phi(i)$ is indeed the evaluation at the index i of the polynomial committed in $\mathcal{C}$. If so it outputs 1, otherwise it outputs 0.

Note that it is possible to commit to a list of messages $(m_1, \ldots, m_{t+1})$ by associating each to a unique key (index) $k_1, \ldots, k_{t+1}$ in $\mathbb{Z}_p$, and interpolating to find $\phi(x) \in \mathbb{Z}_p[x]$, such that $\deg(\phi) \leq t$ and $\phi(k_j) = m_j$.

In terms of security, a malicious committer should not be able to convincingly present two different values as $\phi(i)$ with respect to $\mathcal{C}$. Further, until more than $\deg(\phi)$ points are revealed, the polynomial should remain hidden. Next, we formally define the security and correctness of a polynomial commitment.

Definition 5. *(Setup, Commit, Open, VerifyPoly, CreateWitness, and VerifyEval) is a* secure polynomial commitment scheme *if it satisfies the following properties.*

Correctness. *Let $PK \leftarrow Setup(1^\kappa)$ and $\mathcal{C} \leftarrow Commit(PK, \phi(x))$. For a commitment $\mathcal{C}$ output by $Commit(PK,\ \phi(x))$, and all $\phi(x) \in \mathbb{Z}_p[x]$,*

- *the output of $Open(PK, \mathcal{C}, \phi(x))$ is successfully verified by $VerifyPoly(PK, \mathcal{C}, \phi(x))$, and,*
- *any $\langle i, \phi(i), w_i\rangle$ output by $CreateWitness(PK, \phi(x), i)$ is successfully verified by $VerifyEval(PK,\ \mathcal{C}, i, \phi(i), w_i)$.*

Polynomial Binding. *For all adversaries $\mathcal{A}$:*

$$\Pr\begin{pmatrix} PK \leftarrow Setup(1^\kappa),\ (\mathcal{C}, \langle \phi(x), \phi'(x)\rangle) \leftarrow \mathcal{A}(PK) : \\ VerifyPoly(PK, \mathcal{C}, \phi(x)) = 1\ \wedge \\ VerifyPoly(PK, \mathcal{C}, \phi'(x)) = 1\ \wedge \phi(x) \neq \phi'(x) \end{pmatrix} = \epsilon(\kappa).$$

Evaluation Binding. *For all adversaries $\mathcal{A}$:*

$$\Pr\begin{pmatrix} PK \leftarrow Setup(1^\kappa),\ (\mathcal{C}, \langle i, \phi(i), w_i\rangle, \langle i, \phi(i)', w_i'\rangle) \leftarrow \mathcal{A}(PK) : \\ VerifyEval(PK, \mathcal{C}, i, \phi(i), w_i) = 1\ \wedge \\ VerifyEval(PK, \mathcal{C}, i, \phi(i)', w_i') = 1\ \wedge \phi(i) \neq \phi(i)' \end{pmatrix} = \epsilon(\kappa).$$

Hiding. *Given $\langle PK, \mathcal{C}\rangle$ and $\{\langle i_j, \phi(i_j), w_{\phi_{i_j}}\rangle : j \in [1, \deg(\phi)]\}$ for a polynomial $\phi(x) \in_R \mathbb{Z}_p[x]$ such that $VerifyEval(PK, \mathcal{C}, i_j, \phi(i_j), w_{\phi_{i_j}}) = 1$ for each j,*

– *no adversary $\mathcal{A}$ can determine $\phi(\hat{i})$ with non-negligible probability for any unqueried index $\hat{i}$ (computational hiding) or*
– *no computationally unbounded adversary $\hat{\mathcal{A}}$ has any information about $\phi(\hat{i})$ for any unqueried index $\hat{i}$ (unconditional hiding).*

3.2 Construction: $\mathsf{PolyCommit}_{\mathrm{DL}}$

We now provide an efficient construction of a polynomial commitment scheme. $\mathsf{PolyCommit}_{\mathrm{DL}}$ is based on an algebraic property of polynomials $\phi(x) \in \mathbb{Z}_p[x]$: $(x - i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{Z}_p$. In the literature, Herzberg *et al.* [23] have used this technique in their share recovery scheme.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$, and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t \rangle$. Choose a generator $g \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{Z}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ and outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t} \rangle$. SK is not required in the rest of the construction.

Commit$(\mathbf{PK}, \phi(x))$ computes the commitment $\mathcal{C} = g^{\phi(\alpha)} \in \mathbb{G}$ for polynomial $\phi(x) \in \mathbb{Z}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j}$ as the commitment to $\phi(x)$.

Open$(\mathbf{PK}, \mathcal{C}, \phi(x))$ outputs the committed polynomial $\phi(x)$.

VerifyPoly$(\mathbf{PK}, \mathcal{C}, \phi(x))$ verifies that $\mathcal{C} \overset{?}{=} g^{\phi(\alpha)}$. If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ the algorithm outputs 1, else it outputs 0. Note that this only works when $\deg(\phi) \leq t$.

CreateWitness$(\mathbf{PK}, \phi(x), i)$ computes $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x-i)}$ and outputs $\langle i, \phi(i), w_i \rangle$, where the witness $w_i = g^{\psi_i(\alpha)}$ is computed in a manner similar to $\mathcal{C}$, above.

VerifyEval$(\mathbf{PK}, \mathcal{C}, i, \phi(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \overset{?}{=} e(w_i, g^\alpha / g^i) e(g, g)^{\phi(i)}$, the algorithm outputs 1, else it outputs 0.

$\mathsf{VerifyEval}$ is correct because

$$\begin{aligned} e(w_i, g^\alpha / g^i) e(g, g)^{\phi(i)} &= e(g^{\psi_i(\alpha)}, g^{(\alpha - i)}) e(g, g)^{\phi(i)} = e(g, g)^{\psi_i(\alpha)(\alpha - i) + \phi(i)} \\ &= e(g, g)^{\phi(\alpha)} = e(\mathcal{C}, g) \text{ as } \phi(x) = \psi_i(x)(x - i) + \phi(i) \end{aligned}$$

Theorem 1. *$\mathsf{PolyCommit}_{DL}$ is a secure polynomial commitment scheme (as defined in Definition 5) provided the DL and t-SDH assumptions hold in $\mathcal{G}$.*

A proof is provided in the extended version. The proof of the binding property uses the t-SDH assumption, while the proof of the hiding property uses the DL assumption.

3.3 Construction: $\mathsf{PolyCommit_{Ped}}$

$\mathsf{PolyCommit_{Ped}}$ is also based on the same algebraic property of $\phi(x) \in \mathbb{Z}_p[x]$: $(x-i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{Z}_p$; however, it uses an additional random polynomial $\hat{\phi}(x)$ to achieve unconditional hiding.

The $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme is homomorphic in nature. Given $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitments $\mathcal{C}_{\phi_1}$ and $\mathcal{C}_{\phi_2}$ associated with polynomials $\phi_1(x)$ and $\phi_2(x)$ respectively, one can compute the commitment $\mathcal{C}_\phi$ for $\phi(x) = \phi_1(x) + \phi_2(x)$ as $\mathcal{C}_\phi = \mathcal{C}_{\phi_1}\mathcal{C}_{\phi_2}$. Further, given two witness-tuples $\langle i, \phi_1(i), w_{\phi_1 i}\rangle$ and $\langle i, \phi_2(i), w_{\phi_2 i}\rangle$ at index i associated with polynomials $\phi_1(x)$ and $\phi_2(x)$ respectively, the corresponding tuple for polynomial $\phi(x)$ can be given as $\langle i, \phi_1(i) + \phi_2(i), w_{\phi_1 i} w_{\phi_2 i}\rangle$. The $\mathsf{PolyCommit_{Ped}}$ construction uses the homomorphic property to combine two commitments (one to $\phi(x)$, one to $\hat{\phi}(x)$), although each commitment uses a different generator. Next, we define our $\mathsf{PolyCommit_{Ped}}$ construction.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t\rangle$. Choose two generators $g, h \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{Z}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(2t+2)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t}\rangle \in \mathbb{G}^{2t+2}$ and outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t}\rangle$. Similar to $\mathsf{PolyCommit}_{\mathrm{DL}}$, SK is not required by the other algorithms of the commitment scheme.

Commit$(\mathsf{PK}, \phi(x))$ chooses $\hat{\phi}(x) \in_R \mathbb{Z}_p[x]$ of degree t and computes the commitment $\mathcal{C} = g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)} \in \mathbb{G}$ for the polynomial $\phi(x) \in \mathbb{Z}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ as the commitment to $\phi(x)$.

Open$(\mathsf{PK}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ outputs the committed polynomials $\phi(x)$ and $\hat{\phi}(x)$.

VerifyPoly$(\mathsf{PK}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}$. If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs 1, else it outputs 0. This only works when both $\deg(\phi)$ and $\deg(\hat{\phi}) \leq t$.

CreateWitness$(\mathsf{PK}, \phi(x), \hat{\phi}(x), i)$ calculates $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x-i)}$ and $\hat{\psi}_i(x) = \frac{\hat{\phi}(x) - \hat{\phi}(i)}{(x-i)}$, and outputs $\langle i, \phi(i), \hat{\phi}(i), w_i\rangle$. Here, the witness $w_i = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$.

VerifyEval$(\mathsf{PK}, \mathcal{C}, i, \phi(i), \hat{\phi}(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g)$, the algorithm outputs 1, else it outputs 0.

In the extended version we show $\mathsf{PolyCommit_{Ped}}$ is correct and prove the following security theorem.

Theorem 2. *$\mathsf{PolyCommit_{Ped}}$ is a secure polynomial commitment scheme (as defined in Definition 5) provided the t-SDH assumption holds in $\mathcal{G}$.*

The proof of the binding property is based on the t-SDH assumption, while the hiding property is unconditional.

3.4 Features

We next discuss some important features of $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

Homomorphism. In §3.3, we describe that the $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme is (additive) homomorphic in nature. In the full version we show that $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is also homomorphic.

Unconditional Hiding for $\mathsf{PolyCommit}_{DL}$. When $t' < \deg(\phi)$ evaluations have been revealed, $\mathsf{PolyCommit}_{\mathrm{DL}}$ unconditionally hides any unrevealed evaluation, since the $t'+1$ evaluations $\langle \alpha, \phi(\alpha)\rangle, \langle i_1, \phi(i_1)\rangle, \ldots, \langle i_{t'}, \phi(i_{t'})\rangle$ are insufficient to interpolate a polynomial of degree $> t'$. Note that the evaluation pair $\langle \alpha, \phi(\alpha)\rangle$ is available in an exponentiated form $\langle g^{\alpha}, g^{\phi(\alpha)}\rangle$.

Indistinguishability of Commitments. When a polynomial commitment scheme is randomized, an unbounded adversary cannot distinguish commitments to chosen sets of key-value pairs. When committing to a set of key-value pairs $(\langle k_1, m_1\rangle, \ldots, \langle k_{t+1}, m_{t+1}\rangle)$, if indistinguishable $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitments are required, it is sufficient to set one m_i to a random value. On the other hand, the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme is inherently randomized and can be used directly.

Trapdoor Commitment. The constructions are also trapdoor commitment schemes, where $\mathsf{SK} = \alpha$ is the trapdoor. Refer to the extended version for details.

Batch Opening. In the case when multiple evaluations in a $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitment must be opened, the opening may be batched to reduce the computation and the communication of both the committer and the verifier; *i.e.*, opening multiple evaluations at the same time is cheaper than opening each of those evaluations individually using $\mathsf{CreateWitness}$ and $\mathsf{VerifyEval}$. Let $B \subset \mathbb{Z}_p$, $|B| < t$ be a set of indices to be opened, for a commitment $\mathcal{C} = g^{\phi(\alpha)}$ created using $\mathsf{PolyCommit}_{\mathrm{DL}}$. The witness for the values $\phi(i)$, for all $i \in B$, is computed as $w_B = g^{\psi_B(\alpha)}$ for the polynomial $\psi_B(x) = \frac{\phi(x)-r(x)}{\prod_{i\in B}(x-i)}$ where $r(x)$ is the remainder of the polynomial division $\phi(x)/(\prod_{i\in B}(x-i))$; *i.e.*, $\phi(x) = \psi_B(x)\left(\prod_{i\in B}(x-i)\right) + r(x)$ and for $i \in B$, $\phi(i) = r(i)$. We define two algorithms for batch opening. The algorithm $\mathsf{CreateWitnessBatch}(\mathsf{PK}, \phi(x), B)$ outputs $\langle B, r(x), w_B\rangle$ and the algorithm $\mathsf{VerifyEvalBatch}(\mathsf{PK}, \mathcal{C}, B, r(x), w_B)$ outputs 1 if $e(\mathcal{C}, g) \stackrel{?}{=} e(g^{\prod_{i\in B}(\alpha-i)}, w_B)e(g, g^{r(\alpha)})$ holds, $\deg r(x) = |B|$, and $r(i) = \phi(i)$ for all $i \in B$.

In terms of security, since commitments are formed in the same way as the Commit algorithm of $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{CreateWitnessBatch}$ reveals no more information than running the $\mathsf{CreateWitness}$ algorithm of $\mathsf{PolyCommit}_{\mathrm{DL}}$ for all batch elements individually, the hiding property (Theorem 1) still holds. For binding, an adversary should not be able to open a batch B containing an index i, in a manner that conflicts with the value $\phi(i)$ output in an individual opening of index i. Formally, we say that batch opening is binding if the following holds:

$$\Pr\left(\begin{array}{c}\mathsf{PK} \leftarrow \mathsf{Setup}(1^{\kappa}, t), (\mathcal{C}, \langle B, w_B, r(x)\rangle, \langle i \in B, w_i, \phi(i)\rangle) \leftarrow \mathcal{A}(\mathsf{PK}) : \\ \mathsf{VerifyEvalBatch}(\mathsf{PK}, \mathcal{C}, B, w_B, r(x)) = 1 \ \wedge \\ \mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, w_i, \phi(i)) = 1 \ \wedge \phi(i) \neq r(i)\end{array}\right) = \epsilon(\kappa).$$

Theorem 3. *The construction of* $\mathsf{CreateWitnessBatch}$, $\mathsf{VerifyEvalBatch}$ *in §3.4 is binding provided the* t*-BSDH assumption holds in* $\mathbb{G}$.

This theorem is proven in the full version. The batch construction can be modified for $\mathsf{PolyCommit}_{\mathsf{Ped}}$ due to homomorphic nature of $\mathsf{PolyCommit}_{\mathrm{DL}}$. In the full version we also compare the overhead of various commitment schemes, when Alice commits to t values, and then must reveal k of them. *Overhead* excludes the communication cost of sending the committed values. Notably, the communication overhead of $\mathsf{PolyCommit}_{\mathrm{DL}}$ is constant when batch opening is used.

Strong Correctness. VSS schemes will require an additional property of the $\mathsf{PolyCommit}$ scheme: it should not be possible to commit to a polynomial of degree greater than t. This is called the *strong correctness* property.

To define strong correctness for the $\mathsf{PolyCommit}$ schemes is not easy, *e.g.*, there are many polynomials ϕ' of degree greater than t such that $\phi(\alpha) = z \in_R \mathbb{Z}_p$ and so g^z is trivially a $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitment to some polynomial of degree t' such that $2^\kappa > t' > t$. To avoid this triviality, we require that an adversary $\mathcal{A}$ must convince a challenger $\mathcal{B}$ that he knows ϕ with the following game. $\mathcal{A}$ creates a commitment to a claimed polynomial ϕ' of degree t'. $\mathcal{B}$ challenges $\mathcal{A}$ with $t'+1$ indices $I \subset \mathbb{Z}_p$. $\mathcal{A}$ wins if he is able to produce $\{\langle i, \phi(i), w_i \rangle\}_{i\in I}$ accepted by $\mathsf{VerifyEval}$ and that the interpolation (in exponents) of any $t'+1$ witnesses generates a degree $t-1$ polynomial. Similarly for $\mathsf{PolyCommit}_{\mathsf{Ped}}$. Refer to the extended version of the paper for proof of the following theorem.

Theorem 4. $\mathsf{PolyCommit}_{DL}$ *and* $\mathsf{PolyCommit}_{\mathsf{Ped}}$ *have the strong correctness property if the* t*-polyDH assumption holds in* $\mathcal{G}$.

Practicality and Efficiency Improvements. In absence of a single trusted party, computing Setup can be distributed. Here, $\mathsf{SK} = \alpha$ is computed in a distributed form (*i.e.*, shared by multiple parties forming a distributed authority) using the concept of distributed key generation [31]. PK is computed using a distributed multiplication protocol [20]. As we do not require SK during our protocols and as anybody can verify the correctness of PK using pairings, it is possible to consider PK as a global reusable set, shared in many systems.

Further, the exponentiations required when committing and creating witnesses can be trivially parallelized. Also, since $\mathcal{C} = g^{\phi(\alpha)}$ is computed as a product of powers (sometimes called a *multi-exponentiation*), we suggest using fast exponentiation techniques [32] instead of a naïve implementation. It is also possible to precompute $e(\mathcal{C}, g)$ and $e(g, g)$ for use during verification.

4 Applications

In this section, we describe applications of our commitment schemes to verifiable secret sharing (§4.1), zero-knowledge sets and elementary databases (§4.2), and selective disclosure of signed data and credentials (§4.3).

4.1 Verifiable Secret Sharing (VSS)

For integers n and t such that $n > t \geq 0$, an (n, t)-*secret sharing* scheme [34, 4] is a method used by a *dealer* P_d to share a secret s among a set of n participants (the *sharing* Sh phase) in such a way that in the *reconstruction* Rec phase any subset of $t+1$ or more honest participants can compute the secret s, but subsets of size t or fewer cannot. Furthermore, in secret sharing, nodes may need a procedure to verify the correctness of the dealt values in order to prevent malicious behaviour by the dealer. To solve this problem, Chor *et al.* [15] introduced verifiability in secret sharing, which led to the concept of *verifiable secret sharing* (VSS).

VSS schemes have two security requirements: *secrecy* and *correctness.*

Secrecy (VSS-S). A t-*limited* adversary who compromises t nodes cannot compute s during the Sh phase.

Correctness (VSS-C). The reconstructed value should be equal to the shared secret s or every honest node concludes that the dealer is malicious by outputting $\perp$.

In the computational complexity setting, any malicious behaviour by P_d is caught by the honest nodes in the Sh phase itself and the VSS-C property simplifies to the following: the reconstructed value should be equal to the shared secret s.

Many VSS applications requires that broadcasts from any $t+1$ honest nodes or any $2t+1$ nodes is sufficient to reconstruct s. Therefore, along with VSS-S and VSS-C, we mandate the correctness property that we refer as the *strong correctness* property. Further, some VSS schemes achieve a stronger secrecy guarantee.

Strong Correctness (VSS-SC). The same unique value s is reconstructed regardless of the subset of nodes (of size greater than $2t$) chosen by the adversary in the Rec algorithm.

Strong Secrecy (VSS-SS). The adversary who compromises t nodes have no more information about s except what is implied by the public parameters.

Feldman [18] developed the first efficient VSS protocol, which forms the basis of all VSS schemes defined in the literature. In the literature, the discrete logarithm commitment or Pedersen commitment is used in the Feldman VSS achieve the binding (correctness) and the hiding (secrecy) properties. Both of these commitment schemes trivially satisfy the strong correctness (VSS-SC) property of VSS by the fact that the size of a commitment to a polynomial $\phi(x) \in \mathbb{Z}_p[x]$ is equal to $\deg(\phi) + 1$, which is $O(n)$ (since for optimal resiliency, $\deg(\phi) = t = \lfloor \frac{n-1}{2} \rfloor$). However, the commitment to a polynomial has to be broadcast to all nodes, which results in a linear-size broadcast for Feldman VSS and their variants and a linear complexity gap between the message and the bit complexities. Our goal is to close this gap using any of the $\mathsf{PolyCommit}$ schemes. Next, we apply $\mathsf{PolyCommit}_{\mathrm{DL}}$ to existing polynomial-based VSS schemes and reduce the broadcast message size of VSS by a linear factor, making it equal to the message complexity. Although $\mathsf{PolyCommit}_{\mathrm{DL}}$ can be used in any univariate polynomial-based scheme, we choose the Feldman VSS for ease of exposition.

Sh Phase

1. To share a secret $s \in \mathbb{Z}_p^*$, the dealer P_d chooses a random degree t polynomial $\phi(x) = \sum_{j=0}^{t} \phi_j x^j \in \mathbb{Z}_p[x]$ such that $\phi(0) = \phi_0 = s$. It then broadcasts $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$.
2. For $\ell \in [1, n]$, P_d computes a share $s_\ell = \phi(\ell)$, a witness $w_\ell = \mathsf{CreateWitness}(\mathsf{PK}, \phi(x), \ell)$ and sends $\langle \ell, \phi(\ell), w_\ell \rangle$ to node P_ℓ over a secure and authenticated channel.
3. After receiving $\langle i, \phi(i), w_i \rangle$ from P_d, node P_i runs $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i)$. If the verification fails, P_i broadcasts an accusation message against P_d.
4. If more than t nodes accuse P_d, then it is clearly faulty and is disqualified. If not, for each accusing party P_ℓ, P_d broadcasts the corresponding share and witness $\langle \ell, \phi(\ell), w_\ell \rangle$ such that $\mathsf{VerifyEval}$ holds.
5. If any of the revealed shares fails $\mathsf{VerifyEval}$, P_d is disqualified and the protocol stops. If there is no disqualification, each node P_ℓ accepts $s_\ell = \phi(\ell)$.

Rec Phase

Any $t+1$ or more nodes P_i publish their accepted shares and witnesses $\langle i, s_i, w_i \rangle$. All $t+1$ (or more) nodes verify each of the broadcast shares $\langle i, \phi(i), w_i \rangle$ using $\mathsf{VerifyEval}$ and then interpolate the pairs $\langle i, \phi(i) \rangle$ to determine the secret $s = \phi(0)$.

Fig. 1. eVSS: An efficient Feldman VSS using $\mathsf{PolyCommit}_{\mathrm{DL}}$

Our *efficient Feldman VSS* (eVSS) scheme runs $\mathsf{Setup}(1^\kappa, t)$ of $\mathsf{PolyCommit}_{\mathrm{DL}}$ once, which outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t} \rangle$. Further, as we are working in the synchronous communication model, a *resiliency bound* of $n \geq 2t+1$ is required for VSS to provide correctness against a t-limited Byzantine adversary as the $n-t$ honest nodes available during the Sh and Rec phases should at least be equal to $t+1$ (the required threshold). In Figure 1, we present eVSS that uses the $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme in the Feldman VSS. In the Sh and the Rec phases of the eVSS scheme, the VSS methodology remains exactly the same as that of Feldman VSS except here $t+1$ commitments of the form g^{ϕ_j} for $\phi(x) = \sum_{j=0}^{t} \phi_j x^j$ are replaced by a single polynomial commitment $\mathcal{C} = g^{\phi(\alpha)}$. In addition, along with a share s_i, P_d now sends a witness w_i to node P_i. Overall, the eVSS protocol needs $O(1)$ broadcast instead of $O(n)$ required by the Feldman VSS. In case of multiple accusations, dealer P_d can use the batch opening feature described in §3.4 to provide a single witness for the complete batch. Furthermore, due to the homomorphic nature of $\mathsf{PolyCommit}$, the eVSS scheme can easily converted to a distributed key generation protocol [31].

Theorem 5. *The eVSS protocol implements a synchronous VSS scheme with the VSS-S and VSS-SC properties for $n \geq 2t+1$ provided the DL, t-SDH and t-polyDH assumptions hold in $\mathcal{G}$.*

We need to prove the secrecy, correctness and strong correctness properties of a synchronous VSS scheme. Secrecy and correctness result directly from Theorem 1, while Theorem 4 provides the strong correctness property. The secrecy provided by eVSS is computational against a t-bounded adversary, and unconditional against a $t-1$ bounded adversary. Share correctness is computational.

$\mathsf{PolyCommit}_{\mathrm{DL}}$ can easily be replaced by $\mathsf{PolyCommit}_{\mathsf{Ped}}$ in the above eVSS scheme. In that case, we achieve the strong secrecy (VSS-SS) property due to the unconditional hiding property (Theorem 2) of $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

4.2 Nearly ZKSs and Nearly ZK-EDBs

Micali *et al.* [27] define zero-knowledge sets (ZKSs). Basically a ZKS allows a committer to create a short commitment to a set of values S, such that he may later efficiently prove statements of the form $k_j \in S$ or $k_j \notin S$ in zero-knowledge. No additional information about S is revealed. Perhaps the most challenging aspect in ZKS construction is that not even an upper bound on $|S|$ may be revealed. The closely related notion of *zero-knowledge elementary databases* (ZK-EDB) is also defined in [27]. Loosely speaking, an EDB is a list of key-value pairs, and a ZK-EDB allows a committer to prove that a given value is associated with a given key with respect to a short commitment.

We argue that relaxing the requirements of a ZKS is sufficient for known applications, and show this leads to a significantly more efficient primitive. In particular, by not hiding $|S|$, the size of the proof that an element is (or is not) in a committed set is reduced by a factor of sixteen or more, when compared to the best known ZKS construction.

Motivation. Much of the literature on ZKSs does not consider applications [12, 13, 19, 25, 33]. In the applications of ZKSs (and ZK-EDBs) suggested in [27] the size of the set (or DB) is not crucial to the intended security or privacy of the application. The applications given are to improve privacy and access control when the records of an EDB contain sensitive information about people, *e.g.*, medical records. In such cases, revealing a bound on the number of records in the database clearly does not affect the privacy of an individual whose information is kept in the database.

Another use of ZKSs and ZK-EDBs is for committed databases [30]. In this application, a database owner commits to the database and then proves for every query that the response is consistent with the commitment. For many applications the *contents* of the committed database must be hidden, but the size may be revealed. An example is given in Buldas *et al.* [9]. Here ZK-EDBs are used to increase the accountability of a certification authority by preventing it from providing inconsistent responses to queries about the validity of a certificate. Clearly, keeping the number of certificates hidden is not required. Therefore, a weaker type of ZKS primitive that does not hide the size of the set will suffice for most practical applications of ZKSs. We call a ZKS that may leak information about the size of the set a *nearly ZKS*. Similarly, a *nearly ZK-EDB* is a ZK-EDB that may leak information about the number of records it contains.

Note that an accumulator also represents a set of values with proofs of membership, and some even allow proofs of non-membership (*e.g.*, see [17]). They do

SetupZKS$(1^\kappa, t)$ outputs $\mathsf{PK} = \mathsf{Setup}(1^\kappa, t)$. t is an upper bound on the size of the set which may be committed.

CommitZKS(PK, S) requires $|S| \le t$. Define $\phi(x) = \prod_{k_j \in S}(x - k_j) \in \mathbb{Z}_p[x]$. Output $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$. Let $\hat{\phi}(x) \in \mathbb{Z}_p[x]$ be the random degree t polynomial chosen in $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

QueryZKS$(\mathsf{PK}, \mathcal{C}, k_j)$ allows the committer to create a proof that either $k_j \in S$ or $k_j \notin S$. Compute $\langle k_j, \phi(k_j), \hat{\phi}(k_j), w_j \rangle = \mathsf{CreateWitness}(\mathsf{PK}, \phi(x), \hat{\phi}(x), k_j)$.

(i) If $k_j \in S$, output $\pi_{Sj} = (k_j, w_j, \hat{\phi}(k_j), \perp)$.

(ii) If $k_j \notin S$, create $z_j = g^{\phi(k_j)} h^{\hat{\phi}(k_j)}$ and a ZK proof of knowledge of $\phi(k_j)$ and $\hat{\phi}(k_j)$ in $z_j = g^{\phi(k_j)} h^{\hat{\phi}(k_j)}$. Let $\gamma_j = \langle z_j, \text{ZK proof} \rangle$. Output $\pi_{Sj} = (k_j, w_j, \perp, \gamma_j)$.

VerifyZKS$(\mathsf{PK}, \mathcal{C}, \pi_{Sj})$ parses π_{Sj} as $(k_j, w_j, \hat{\phi}(k_j), \gamma_j)$.

(i) If $\hat{\phi}(k_j) \ne \perp$, then $k_j \in S$. Output 1 if $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, k_j, 0, \hat{\phi}(k_j), w_j) = 1$.

(ii) If $\gamma_j \ne \perp$, then $k_j \notin S$. Parse γ_j as $\langle z_j, \text{ZK proof} \rangle$. If $e(\mathcal{C}, g) \overset{?}{=} e(w_j, g^{\alpha - k_j}) e(z_j, g)$, and the ZK proof of knowledge of z_j is valid, output 1. Output 0 if either check fails.

Fig. 2. A nearly ZKS scheme based on $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

not however, guarantee hiding (the ZK property), in [17] after seeing responses to t non-membership queries we may recover the entire accumulated set.

Construction of a Nearly ZKS. This construction (Figure 2) will use $\mathsf{PolyCommit}_{\mathsf{Ped}}$, and allows us to commit to $S \subset \mathbb{Z}_p$ such that $|S| \le t$. The basic idea is to commit to a polynomial ϕ, such that $\phi(k_j) = 0$ for $k_j \in S$, and $\phi(k_j) \ne 0$ for $k_j \notin S$. Our construction relies on a ZK proof that proves $\phi(k_j) \ne 0$ without revealing $\phi(k_j)$ to maintain privacy for queries when $k_j \notin S$. Although a construction based on $\mathsf{PolyCommit}_{\mathrm{DL}}$ is also possible, we choose $\mathsf{PolyCommit}_{\mathsf{Ped}}$ as the required ZK proof is simpler in the latter case. For convenience we describe our protocols assuming the ZK proof is non-interactive, however, an interactive ZK proof may be used as well.

A security definition and proof are provided in the full version. The ZK proof of knowledge may be implemented using any secure ZK proof system allowing proof of knowledge of a discrete logarithm (see [11] for examples).

Construction of a Nearly ZK-EDB. This construction (Figure 3) makes use of the above nearly ZKS construction and $\mathsf{PolyCommit}_{\mathrm{DL}}$. Let $D = (K, V) \subset \mathbb{Z}_p^t \times \mathbb{Z}_p^t$ be a list of key-value pairs that will define the database (K and V are ordered lists of equal length such that the value $m_j \in V$ corresponds to the key $k_j \in K$). The values may repeat, but the keys must be distinct. We write $D(k_j)$ to denote the value associated to key k_j (if $k_j \notin K$, then $D(k_j) = \perp$). The underlying idea of our construction is to commit to the keys using our nearly ZKS, and also commit to ϕ, such that $\phi(k_j) = m_j$, using $\mathsf{PolyCommit}_{\mathrm{DL}}$, since it

$\mathsf{SetupEDB}(1^\kappa, t)$ runs $\mathsf{SetupZKS}(1^\kappa, t)$, and outputs PK.

$\mathsf{CommitEDB}(\mathsf{PK}, D = (K, V))$ sets $\mathcal{C}_1 = \mathsf{CommitZKS}(\mathsf{PK}, K)$. It then interpolates the t (or fewer) points $(k_j, m_j) \in D$, and one or more random points $(k_r, m_r) \in_R \mathbb{Z}_p \times \mathbb{Z}_p$ to get a polynomial $\phi_2(x) \in \mathbb{Z}_p[x]$, assured to be of degree t. Set $\mathcal{C}_2 = \mathsf{Commit}(\mathsf{PK}, \phi_2(x))$ and output $\mathcal{E} = (\mathcal{C}_1, \mathcal{C}_2)$.

$\mathsf{QueryEDB}(\mathsf{PK}, \mathcal{E}, k_j)$ parses $\mathcal{E}$ as $(\mathcal{C}_1, \mathcal{C}_2)$.

(i) If $k_j \in K$, compute $\pi_{Sj} = \mathsf{QueryZKS}(\mathsf{PK}, \mathcal{C}_1, k_j)$ to show that $k_j \in K$ and $\langle k_j, m_j, w_{m_j}\rangle = \mathsf{CreateWitness}(\mathsf{PK}, \phi_2(x), k_j)$ to show that $D(k_j) = m_j$. Output $\pi_{Dj} = (\pi_{Si}, m_j, w_{m_j})$.

(ii) If $k_j \notin K$, we show that $k_j \notin K$, set $\pi_{Sj} = \mathsf{QueryZKS}(\mathsf{PK}, \mathcal{C}_1, k_j)$. Output $\pi_{Dj} = (\pi_{Sj}, \perp, \perp)$.

$\mathsf{VerifyEDB}(\mathsf{PK}, \mathcal{E}, \pi_{Dj})$ parses π_{Dj} as (π_{Sj}, m_j, w_{m_j}) and $\mathcal{E}$ as $(\mathcal{C}_1, \mathcal{C}_2)$.

(i) If $m_j \neq \perp$, then $k_j \in K$, output (k_j, m_j) if $\mathsf{VerifyZKS}(\mathsf{PK}, \mathcal{C}_1, \pi_{Sj}) = 1$ and $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}_2, k_j, m_j, w_{m_j}) = 1$.

(ii) If $m_j = \perp$, then $k_j \notin K$, output 1 if $\mathsf{VerifyZKS}(\mathsf{PK}, \mathcal{C}_1, \pi_{Sj}) = 1$.

Fig. 3. A nearly ZK-EDB scheme constructed using our nearly ZKS construction (Figure 2) and $\mathsf{PolyCommit}_{\mathrm{DL}}$.

is sufficient for security and more efficient. The reason for using the nearly ZKS is to respond to queries when $k \notin D$ without revealing any additional information.

Efficiency of our nearly ZKS and ZK-EDBs. The size of the commitment is a single group element for a nearly ZKS, or two elements for a nearly ZK-EDB. Proof that $k_j \in S$, consists of two group elements, while proof that $k_j \notin S$ consists of about five group elements (when ZK proofs are implemented using a standard three-move ZK protocol, made non-interactive with the Fiat-Shamir heuristic). The proof sizes for our nearly ZK-EDB construction are three and about five group elements (respectively).

The ZK-EDB in the literature with the shortest proofs is that of Libert and Yung [25] (based on their ZKS construction). Asymptotically, (non)membership proofs are $O(\kappa/\log(t))$ bits, where κ is a security parameter, and t is the size of the system parameters. For the parameter choices given [25], proof sizes range from 80–220 group elements. The computation of their scheme and ours is nearly equal. Therefore, using nearly ZK-EDBs in the place of ZK-EDBs reduces communication costs by at least a factor of sixteen.

4.3 Credentials and Selective Disclosure of Signed Data

In this section we briefly describe two applications of the $\mathsf{PolyCommit}$ schemes, and we will show how polynomial commitments can reduce communication costs. Both applications are based on the following idea. Suppose Alice has a list of values $(m_1, \ldots, m_t)$, which will be signed by Trent. If Trent signs the concatenation, then Alice must reveal all m_i to allow Bob to verify the signature. However, if Trent signs $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$ where $\phi(i) = m_i$, then Alice may allow Bob to verify that Trent has signed m_i without revealing the other m_j. Bob verifies the signature on $\mathcal{C}$, and Alice produces a witness to prove that $\mathcal{C}$ opens to m_i at position i, allowing Alice to convince Bob that Trent signed m_i.

Content Extraction Signatures. If $(m_1, \ldots, m_t)$ are parts of a document, then signing a polynomial commitment is a *content extraction signature* (CES) scheme. Steinfeld *et al.* [35] introduce CES and give a generic construction of CES signatures. The scheme requires a standard commitment scheme, which is then used to commit to each of the t sub-messages $(m_1, \ldots, m_t)$ individually, forming a vector of commitments, which is signed. The scheme is secure, provided the signature scheme is secure, and the commitment scheme is hiding and binding.

Since both $\mathsf{PolyCommit}$ schemes are hiding and binding, and allow a list of messages to be committed, they can be used in the general construction of Steinfeld *et al.* Along with the commitment, Trent should also sign t so that Bob knows that only indices $\{1, \ldots, t\}$ correspond to valid sub-messages. The new scheme is nearly as communication efficient as a specific scheme in [35] which has the lowest communication cost. The latter, however, depends on specific properties of the RSA signature scheme and is secure in the random oracle model. Using a polynomial commitment scheme gives an efficient generic construction. Therefore, efficient standard model CES schemes are possible by combining any of the $\mathsf{PolyCommit}$ schemes with a signature scheme secure in the standard model.

Pseudonymous Credentials. If $(m_1, \ldots, m_t)$ are attributes about Alice, and Trent is an identity provider, then the signature Alice holds on $\mathcal{C}$ is a digital credential that allows Alice to reveal only as much information as is necessary to complete an online transaction. Here, we create $\mathcal{C}$ using $\mathsf{PolyCommit}_{\mathrm{DL}}$, as batched openings are efficient for $\mathsf{PolyCommit}_{\mathrm{DL}}$. Disclosing a single m_i requires Alice to transmit $(\mathcal{C}, \mathsf{Sign}_{\mathrm{Trent}}(\mathcal{C}), \langle i, m_i, w_i \rangle)$, the size of which is independent of t. If Alice reveals a subset of the attributes, a single witness may be used to reduce communication even further using batch opening (described in §3.4). Further, if Trent signs multiple commitments to the same attributes (but includes an extra randomized attribute), Alice may present a different commitment to the same verifier unlinkably.

For many interesting applications of credentials, selective show is insufficient because Alice would like to prove something about m_i (*e.g.*, $m_i < 1990$) without revealing m_i. Alice may prove knowledge of a nonzero committed value $\phi(i)$ without revealing it, and compose this proof with other proofs about m_i using standard ZK proof techniques for proving knowledge of, relations between or the length of discrete logarithms [11]. Since the communication costs per attribute of proving knowledge of a committed value are constant, if k attributes are involved in showing a credential, the complexity of the show will be $O(k)$. In existing schemes the communication is $O(t)$ where t is the total number of attributes in the credential. Further details of this application are given in the full version of this paper.

5 Open Problems

Finally, we list a few open problems related to polynomial commitment schemes. 1. Is it possible to construct efficient polynomial commitment schemes under weaker assumptions? 2. What other protocols does PolyCommit improve? (For example, can PolyCommit reduce the communication of asynchronous VSS protocols or verifiable shuffles?) 3. We have mainly focused on the communication costs, but our construction asks for nontrivial computation. Is it possible to reduce computation cost as well?

Acknowledgements. We thank the anonymous reviewers for providing many helpful comments, and thank Benoit Libert for sending us a preprint of [25]. We gratefully acknowledge the financial support of NSERC, MITACS, and the David R. Cheriton Graduate Scholarship program.

References

1. M. H. Au, Q. Wu, W. Susilo, and Y. Mu. Compact E-Cash from Bounded Accumulator. In *Proceedings of CT-RSA'07*, pages 178–195, 2007.
2. M.H. Au, W. Susilo, and Y. Mu. Practical anonymous divisible e-cash from bounded accumulators. In *Proceedings of FC'08*, pages 287–301, 2008.
3. J.C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In *Proceedings of EUROCRYPT'93*, pages 274–285. Springer, 1993.
4. G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, pages 313–317, 1979.
5. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Proceedings of EUROCRYPT'04*, pages 223–238, 2004.
6. D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *Proceedings of EUROCRYPT'04*, pages 56–73. Springer, 2004.
7. D. Boneh, X. Boyen, and E.J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Eurocrypt*, volume 3494, pages 440–456. Springer, 2005.
8. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proceedings of CRYPTO'05*, pages 258–275. Springer, 2005.
9. A. Buldas, P. Laud, and H. Lipmaa. Eliminating Counterevidence with Applications to Accountable Certificate Management. *Journal of Computer Security*, 10(3):273–296, 2002.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of CRYPTO'02*, pages 61–76. Springer, 2002.
11. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical report, 1997. 260, Dept. of Computer Science, ETH Zurich.
12. D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. In *Proceedings of EUROCRYPT'08*, pages 433–450. Springer, 2008.
13. M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *Proceedings of EUROCRYPT'05*, pages 422–439. Springer, 2005.

14. J.H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Proceedings of EUROCRYPT'06*, pages 1–13. Springer, 2006.
15. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *Proceedings of FOCS'85*, pages 383–395, 1985.
16. I. Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security*, pages 63–86. Springer, 1999.
17. I. Damgård and N. Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. 2008. Cryptology ePrint Archive: Report 2008/538.
18. P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Proceedings of FOCS'87*, pages 427–437, 1987.
19. R. Gennaro and S. Micali. Independent zero-knowledge sets. In *Proceedings of ICALP'06*, pages 34–45. Springer, 2006.
20. R. Gennaro, M.O. Rabin, and T. Rabin. Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In *Proceedings of PODC'98*, pages 101–111. ACM Press, 1998.
21. V. Goyal. Reducing Trust in the PKG in Identity Based Cryptosystems. In *Advances in Cryptology—CRYPTO'07*, pages 430–447, 2007.
22. F. Guo, Y. Mu, and Z. Chen. Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key. In *Proceedings of Pairing'07*, pages 392–406. Springer, 2007.
23. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Proceedings of CRYPTO'95*, pages 339–352, 1995.
24. A. Kate, G. Zaverucha, and I. Goldberg. Polynomial commitments. Technical report, 2010. CACR 2010-10, Centre for Applied Cryptographic Research, University of Waterloo.
25. B. Libert and M. Yung. Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In *TCC'10*, pages 499–517, 2010.
26. A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1st edition, 1997.
27. S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *Proceedings of FOCS'03*, pages 80–91. IEEE, 2003.
28. S. Mitsunari, R. Sakai, and M. Kasahara. A New Traitor Tracing. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E85-A(2):481–484, 2002.
29. L. Nguyen. Accumulators from bilinear pairings and applications. In *Proceedings of CT-RSA*, pages 275–292, 2005.
30. R. Ostrovsky, C. Rackoff, and A. Smith. Efficient Consistency Proofs for Generalized Queries on a Committed Database. In *Proceedings of ICALP'04*, pages 1041–1053, 2004.
31. T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Proceedings of CRYPTO'91*, pages 129–140. Springer, 1991.
32. N. Pippenger. On the evaluation of powers and related problems. In *IEEE SFCS(FOCS)'76*, pages 258–263, 1976.
33. M. Prabhakaran and R. Xue. Statistically Hiding Sets. In *Proceedings of CT-RSA*, pages 100–116. Springer, 2009.
34. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.
35. R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *Proceedings of ICISC'01*, pages 285–304. Springer, 2002.