16

Wrap-Up of Polynomial Commitments

16.1 Batch Evaluation of Homomorphically Committed Polynomials

In some applications of polynomial commitment schemes to SNARKs, the verifier will wish to open many committed polynomials at the same point—see for example Section 10.3.2. We now explain that, if the polynomial commitment scheme is homomorphic (as all polynomial commitment schemes in this section are), all openings can be verified with essentially the same prover and verifier costs as a single opening.

Suppose for concreteness that the prover claims that $p(r) = y_1$ and $q(r) = y_2$ where p and q are committed polynomials over field $\mathbb{F}$, with commitments c_1 and c_2 respectively. Rather than verifying both claims independently, the verifier can instead verify a random linear combination of the claims, i.e., check that

$$a \cdot p(r) + q(r) = a \cdot y_1 + \cdot y_2 \tag{16.1}$$

for randomly chosen $a \in \mathbb{F}$. Clearly Equation (16.1) is true if both original claims are true. Meanwhile, because the left hand side and right hand side of Equation (16.1) are both linear functions of a, so if either

one of the original claims is false, Equation (16.1) will be false except with probability at most $1/|\mathbb{F}|$ over the random choice of a.

So up to soundness error $1/|\mathbb{F}|$, verifying both claims is equivalent to verifying Equation (16.1), which is an evaluation claim about a single polynomial $ap + q$, of the same degree as p and q are individually. Via homomorphism, the verifier can compute a commitment c_3 to the polynomial $a \cdot p + q$ unaided. Hence, the prover and verifier can apply the polynomial commitment's evaluation procedure directly to this polynomial to check Equation (16.1).

This means the evaluation procedure has to be applied only *once* to check both original claims. The only extra work the prover does to reduce the two original claims to the one derived claim is to compute the polynomial $a \cdot p + q$, and the only extra work the verifier does is to compute the derived commitment c_3 from c_1 and c_2, and $a \cdot y_1 + y_2$ from y_1 and y_2. Both of these extra computations are typically low-order costs, i.e., far cheaper than computing and verifying an evaluation proof respectively.

More general batching techniques are also known, which can handle evaluating multiple different homomorphically-committed polynomials at distinct points [64].

16.2 Commitment Scheme for Sparse Polynomials

Let us call a degree-D univariate polynomial *dense* if the number of nonzero coefficients is $\Omega(D)$, i.e., at least a constant fraction of the coefficients are nonzero. Similarly, call an ℓ-variate multilinear polynomial dense if the number of coefficients over the Lagrange basis is $\Omega(2^\ell)$. If a polynomial is not dense, we call it *sparse*.

An example of sparse polynomials that we have seen in this survey are $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$, the multilinear extensions of the functions add_i and mult_i that arose in our coverage of the GKR protocol (Section 4.6) and the related functions $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ appearing in the MIP of Section 8.2. Indeed, $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ are defined over $\ell = 3 \log |\mathcal{C}|$ variables, and the number of Lagrange basis polynomials over this many variables is $2^\ell = |\mathcal{C}|^3$. However, the number of nonzero coefficients of $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ in the Lagrange basis is just $|\mathcal{C}|$.

As discussed in Section 4.6.6, when $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ cannot be evaluated in time sublinear in $|\mathcal{C}|$, one technique to save the verifier time is for a trusted party to commit to these polynomials in a pre-processing phase with a polynomial commitment scheme (this can be done transparently, so that any party willing to put in the effort can confirm that the commitment was computer correctly). Then whenever the GKR protocol or MIP of Sections 4.6 and 8.2 (or SNARKs derived thereof) is applied to $\mathcal{C}$ on a new input, the verifier need not evaluate $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ on its own to perform the necessary checks of the prover's messages. Rather, the verifier can ask the prover to reveal the evaluations, using the evaluation phase of the polynomial commitment scheme. This reduces the verifier's runtime after the pre-processing phase from $\Theta(|\mathcal{C}|)$ to whatever is the verification time of the evaluation phase of the polynomial commitment scheme. Note that in this application of the polynomial commitment scheme, there is no need for the protocol to be zero-knowledge or extractable; it only needs to be binding to save the verifier work in the resulting zero-knowledge arguments for circuit satisfiability.

We have now seen several polynomial commitment schemes in which the committer's runtime is dominated by doing a number of group exponentiations that is linear in the number of coefficients for *dense* univariate and multilinear polynomials (e.g., Section 14.3). However, these schemes do not offer any additional runtime savings for the committer if the polynomials are sparse. For example, applying these schemes directly to $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ requires $\Omega(|\mathcal{C}|^3)$ time, which is totally impractical.[1]

In this section, we sketch a commitment scheme proposed by Setty [226] for any polynomial q such that the runtime of the committer is proportional to the *nonzero coefficients* M of q. The commitment scheme uses any polynomial commitment scheme for *dense, multilinear polynomials* as a subroutine. This result is analogous to the holography achieved in Section 10.3.2, which gave a way to commit to any sparse *bivariate* polynomial given a commitment scheme for dense *univariate*

[1] Directly applying the KZG-based scheme for multilinear polynomials of Section 15.3 would allow the commitment to be computed in $O(|C|)$ time, but the SRS would have length $|\mathcal{C}|^3$, and the evaluation phase may take time $\Omega(|\mathcal{C}|^3)$ for the committer.

polynomials. The schemes of this section and Section 10.3.2 share conceptual similarities, but also key differences.

For presentation purposes, we first describe a protocol that achieves the above goals up to a logarithmic factor. That is, the committer will have to apply a dense multilinear polynomial commitment scheme to a multilinear polynomial defined over $\ell' = \log_2 M + \log_2 \ell$ variables. Assuming M is $2^{\Omega(\ell)}$ (as is the case for $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$), $O(M\ell) \leq O(M \log M)$, so the dense polynomial to be committed is an $O(\log M)$ factor larger than can be hoped for.

At the end of this section we sketch a technique to remove even this extra $\ell = \Theta(\log M)$ factor from the committer's runtime by exploiting additional structure in $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$. For brevity, our sketch is very high-level, and deviates in certain details from Setty's scheme–the interested reader is directed to [137, Section 6] for a self-contained exposition of Setty's full scheme.

Simple Scheme with Logarithmic Factor Overhead. The idea is to identify a layered arithmetic circuit $\mathcal{C}'$ of fan-in two that takes as input a description of a sparse ℓ-variate multilinear polynomial q (we specify the input description shortly) and a second input $z \in \mathbb{F}^\ell$, and such that $\mathcal{C}'$ outputs $q(z)$. We will ensure that the input to $\mathcal{C}'$ consists of $O(M\ell)$ field elements, and $\mathcal{C}'$ has size $O(M\ell)$ and depth $O(\log M)$. Also, $\mathcal{C}'$ will have wiring predicates $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ that can be evaluated any point in $O(\log(M\ell))$ time.

If s denotes the input to $\mathcal{C}'$ specifying q, the commitment to q in our sparse polynomial commitment scheme will simply be a commitment to the multilinear extension $\widetilde{s}$ of s using any commitment scheme for dense multilinear polynomials. The reveal phase of our sparse polynomial commitment scheme works as follows. When the verifier requests the committer reveal $q(z)$ for a desired $z \in \mathbb{F}^\ell$, the committer sends the claimed value v of $q(z)$, and then the committer and verifier apply the GKR protocol to the claim that $\mathcal{C}'(s, z) = v$.[2] At the end of the GKR protocol, the verifier needs to evaluate the multilinear extension

[2] They could also apply the MIP of Section 8.2 to verify this claim, replacing the second prover with a polynomial commitment scheme for dense multilinear polynomials.

of the input (s, z) at a random point. Since the verifier knows z but not s, using an observation analogous to Equation (7.1), the multilinear extension of (s, z) can be efficiently evaluated at any desired point so long as the verifier learns an evaluation of $\widetilde{s}$ at a related point. The verifier can obtain this evaluation from the prover using the reveal phase of the dense polynomial commitment scheme.

Since $\mathcal{C}'$ has size $O(M\ell)$, the GKR protocol prover applied to C' can be implemented in $O(M\ell)$ total time, and committing to $\widetilde{s}$ using an appropriate dense polynomial commitment scheme requires a multi-exponentiation of size $O(M\ell)$. Since $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ for $\mathcal{C}'$ can be evaluated in $O(\log(M\ell))$ time, the verifier's runtime in the protocol is dominated by the cost of the evaluation phase of the dense polynomial commitment scheme.

Here is how the input to $\mathcal{C}'$ will specify the polynomial q. Let $T_1, \ldots, T_M \in \{0,1\}^\ell$ denote the Lagrange basis polynomials $\chi_{T_1}, \ldots, \chi_{T_M}$ that have nonzero coefficients $c_1, \ldots, c_M$ in q. That is, let

$$q(X) = \sum_{i=1}^{M} c_i \cdot \prod_{j=1}^{\ell} \left(T_{i,j}X_j + (1 - T_{i,j})(1 - X_j)\right). \tag{16.2}$$

The description s of q will consists of two lists $L[1], \ldots, L[M]$ and $B[1], \ldots, B[M]$, where $L[i] = c_i \in \mathbb{F}_p$ and $B[i] = T_i \in \{0,1\}^\ell$. The circuit $\mathcal{C}'$ will simply evaluate Equation (16.2) at input z. It is not hard to verify that Equation (16.2) can be evaluated by an arithmetic circuit with $O(M\ell)$ gates, such that the multilinear extensions of the wiring predicates add_i and mult_i for each layer of $\mathcal{C}'$ can be evaluated in $O(\log(M\ell))$ time.

Saving a logarithmic factor (sketch). The idea to shave a factor of ℓ from the size of $\mathcal{C}'$ and the length of its input when committing to $q = \widetilde{\text{add}}$ or $q = \widetilde{\text{mult}}$ is as follows. First, modify the description of q, so as to reduce the description length from $O(M \cdot \ell)$ field elements down to $O(M)$ field elements. Then identify a Random Access Machine $\mathcal{M}$ that takes as input this modified description of q and an input $z \in \mathbb{F}^\ell$ and outputs $q(z)$. We make sure that $\mathcal{M}$ runs in time $O(M)$, and that $\mathcal{M}$ can be transformed into a circuit of size that is just a constant factor larger than its runtime.

Here is how to modify the description of $q = \widetilde{\text{add}}$. Rather than having the identities of the nonzero Lagrange coefficients of $\widetilde{\text{add}}$ be specified via a list of bit-strings $T_1, \ldots, T_M \in \{0,1\}^{3 \log |\mathcal{C}|}$, we instead specify their identities with triples of *integers* $(u_1, u_2, u_3) \in \{1, \ldots, |\mathcal{C}|\}^3$, and interpret a triple as indicating that the u_1'st gate of $\mathcal{C}$ is an addition gate with in-neighbors assigned integer labels u_2 and u_3.

The Random Access Machine $\mathcal{M}$ works as follows. It is given as input the modified description of $q = \widetilde{\text{add}}$ and a point $z = (r_1, r_2, r_3) \in \left(\mathbb{F}_p^{\log |\mathcal{C}|}\right)^3$, runs in $O(M)$ time and outputs $q(z)$. Recall from Section 8.2 that $\text{add}(a,b,c)\colon \{0,1\}^{3 \log S} \to \{0,1\}$ interprets its input as three gate labels a, b, c and outputs 1 if and only if b and c are the in-neighbors of gate a, and a is an addition gate. This means that

$$\widetilde{\text{add}}(X,Y,Z) = \sum_{a \in \{0,1\}^{\log |\mathcal{C}|}\colon a \text{ an add gate}} \chi_a(X) \cdot \chi_{\text{in}_1(a)}(Y) \cdot \chi_{\text{in}_2(a)}(Z), \tag{16.3}$$

where $\text{in}_1(a)$ and $\text{in}_2(a)$ respectively denote the labels in $\{0,1\}^{\log |\mathcal{C}|}$ of the first and second in-neighbors of gate a in $\mathcal{C}$. Recall in addition from Lemma 3.8 that evaluating all $(\log |\mathcal{C}|)$-variate Lagrange basis polynomials at a specified input $r \in \mathbb{F}^{\log |\mathcal{C}|}$ can be done in $O(|\mathcal{C}|)$ time. So to evaluate $\widetilde{\text{add}}$ at an input $(r_1, r_2, r_3) \in \left(\mathbb{F}^{\log |\mathcal{C}|}\right)^3$ in $O(|\mathcal{C}|)$ time, it suffices for $\mathcal{M}$ to operate in two phases. In the first phase, $\mathcal{M}$ evaluates all $(\log |\mathcal{C}|)$-variate Lagrange basis polynomials at the three inputs $r_1, r_2, r_3 \in \mathbb{F}^{\log |\mathcal{C}|}$ in $O(|\mathcal{C}|)$ time (this can be done without even examining the list of triples (u_1, u_2, u_3)), and stores the $3 \cdot |\mathcal{C}|$ results in memory. In the second phase, $\mathcal{M}$ evaluates $\widetilde{\text{add}}$ at (r_1, r_2, r_3) via Equation (16.3) in $O(|\mathcal{C}|)$ additional time, given random access to the memory contents. Note that the the memory accesses made by $\mathcal{M}$ depend only on the committed polynomial q and are independent of the evaluation point z.

Using the computer-programs-to-circuit-satisfiability transformation of Section 6, specifically using the fingerprinting-based memory-checking procedure described in Section 6.6.2, $\mathcal{M}$ can be transformed into a circuit-satisfiability instance for a circuit $\mathcal{C}'$. As described in Sections 6.6.2 and 6.6.3, the transformation procedure from $\mathcal{M}$ to the circuit-satisfiability instance is itself interactive, but the transformation can

be rendered non-interactive in the random oracle model using the Fiat-Shamir transformation.

16.3 Polynomial Commitment Schemes: Pros and Cons

We have seen three approaches to the construction of practical polynomial commitment schemes. The first is based on IOPs, specifically FRI (Section 10.4.2), and Ligero-and Brakedown-commitments (Section 10.5). The second (Section 14) builds in sophisticated ways on homomorphic commitment schemes such as Pedersen commitments [206], and Schnorr-style [223] techniques for proving inner product relations between a committed vector and a public vector; this approach based binding on the assumed hardness of the discrete logarithm problem. The third (Section 15) is derived from work of KZG [164] and is based on bilinear maps and requires a trusted setup. Roughly, the practical pros and cons of the three approaches to polynomial commitment schemes are the following.

Pros and Cons of the IOP-Based Polynomial Commitments. The IOP approach is the only one of the three approaches that is plausibly quantum-secure (it can lead to security in the quantum random oracle model [99]). The other two approaches assume hardness of the discrete logarithm problem, which is a problem that quantum computers can solve in polynomial time. Another advantage of the IOP approach is that it uses very small public parameters (these simply specify one or more cryptographic hash functions) that moreover can be generated at random, i.e., they simply specify a randomly chosen hash function from a collision-resistant hash family. That is, unlike the third approach, the first (and also the second) approach is transparent, meaning it does not require a structured reference string (SRS) generated by a trusted party who will have the power to forge proofs of evaluations if they fail to discard toxic waste.

The downsides of the IOP-based schemes include the following. FRI's evaluation proofs, while polylogarithmic in size and the smallest amongst known IOP-based schemes, are concretely rather large, especially relative to commitment schemes from other approaches with

logarithmic or constant size proofs, e.g., Bulletproofs (Section 14.4), Dory (Section 15.4), and KZG commitments (Section 15).

FRI also has relatively high concrete costs for the prover. To be more precise, FRI exhibits a strong tension between prover costs and verification costs, with the tradeoff between the two determined by the *rate* of the Reed-Solomon code used in the protocol (see Section 10.4.4 for details). Under current security analyses, if FRI is configured to have a slower prover than other polynomial commitment schemes (e.g., using Reed-Solomon code rate 1/16), the size of evaluation proofs at 100 bits of security is substantially over 100 KBs. See, for example, [145] for details. When configured to have a faster prover, the proof sizes are considerably larger. For example, when using rate rate 1/2 rather than rate 1/16, the proofs are roughly 4× larger though the prover is roughly 8× faster.

Ligero- and Brakedown-commitments are faster for the prover than FRI, but have significantly bigger proofs. Some approaches toward mitigating their proof sizes via SNARK composition [95], [253] are discussed in Section 19.

Another major disadvantage of the IOP approach is that the other two approaches yield homomorphic commitments (i.e., given two commitments c_p, c_q to two polynomials p and q, a commitment c to the sum of $p + q$ can be derived). This homomorphism property leads to excellent batch-opening properties (Section 16.1) and is essential in some applications of polynomial commitment schemes (see for example Section 18.5 later in this survey). FRI and Ligero also require the prover to perform FFTs, which can place some restrictions on the field used and for large instances can be a bottleneck in prover time and space.

Pros and Cons of Discrete-Logarithm Based Polynomial Commitments. The second approach is also transparent because the public parameters are simply random group elements— though depending on the commitment scheme, there can be a lot of them, and generating many random elements of elliptic-curve groups can be expensive in practice. This approach does not require FFTs, and the committer performs $O(n)$ many group exponentiations in any group for which the discrete logarithm problem is hard (with a Pippenger-style multi-exponentiation

speedup possible, with the exception of Bulletproof's evaluation proofs). Hence, this approach currently has very good concrete efficiency for the committer, roughly comparable to Ligero- and Brakedown-commitments for large enough polynomials. Until the advent of Dory (Section 15.4), this approach did require larger verifier runtime than the other two approaches. For example, in Hyrax's commitment scheme (Section 14.3), the verifier computes a multi-exponentiation of size $O(\sqrt{n})$, while in Bulletproofs (Section 14.4) the multi-exponentiation done by the verifier has *linear size*, $\Theta(n)$. Dory reduces the verifier time to $O(\log n)$ group exponentiations, at the cost of constant-factor increase in the time to compute commitments, mostly due to Dory's need to operate in a pairing-friendly group, which can lead concretely slower group operations. The costs of the transparent polynomial commitment schemes covered in this survey are summarized in Table 16.1.

Pros and Cons of KZG-Based Polynomial Commitments. The primary benefit of the third approach is its superior verification costs when applied to univariate polynomials. Specifically, both the commitment and proofs of evaluation consist of a constant number of group elements, and can be verified with a constant number of group operations and two bilinear map evaluations. For multilinear polynomials (Section 15.3) these costs are logarithmic in the number of coefficients instead of constant.

A significant downside of the third approach is that it requires an SRS, with toxic waste that must be discarded to avoid forgeability of evaluation proofs. There has been significant work to mitigate the trust assumptions required, and it is now known how to make the SRS "updatable". This means that the SRS can be updated at any point by any party, such that if even a single party is honest—meaning a single party discards their toxic waste—then no one can forge proofs of evaluation [187]. The rough idea for why this is possible is that the SRS consists of powers of a random nonzero field element $\tau \in \{1, \ldots, p-1\}$ in the exponent of a group generator g, so any party can "rerandomize the choice of τ" by picking a random $s \in \{1, \ldots, p-1\}$ and updating the ith element of the SRS from g^{τ^i} to $(g^{\tau^i})^{s^i} = g^{(\tau s)^i}$. That is, by

Table 16.1: Costs of transparent polynomial commitment schemes covered in this survey. For simplicity, this table restricts its focus to univariate polynomials of degree N with coefficients over the standard monomial basis. λ denotes the security parameter. Commit time and $\mathcal{P}$ time columns respectively list a dominant operation to produce the commitment and an evaluation proof. $\mathcal{V}$ time lists a dominant operation to verify any evaluation proof. FRI and Ligero-commit are plausibly post-quantum, but not homomorphic; the others are homomorphic, but not post-quantum. Dory requires operations over a pairing-friendly group and a pre-processing phase costing $O(\sqrt{N})$ group exponentiations. The public parameters for FRI and Ligero simply specify a cryptographic hash function. For Bulletproofs, they are $O(N)$ random group elements, while for all other rows they are $O(\sqrt{N})$ random group elements. The term exp is short for group exponentiation, and the term field ops is short for field operations.

Scheme	**Commit Size**	**Evaluation Proof Size**	**$\mathcal{V}$ Time**	**Commit Time**	**$\mathcal{P}$ Time**
FRI (Sections 10.4.4 and 10.4.2)	one hash value	$O(\log^2(N) \cdot \lambda)$ hash values	$O(\log^2(N) \cdot \lambda)$ hash evaluations	$O(N \log N)$ field ops	$O(N \log N)$ field ops
Ligero-commit (Section 10.5)	one hash value	$O(\sqrt{N \cdot \lambda})$ field elements	$O(\sqrt{N \cdot \lambda})$ field ops	$O(N \log N)$ field ops	$O(N)$ field ops
Hyrax-commit (Section 14.3)	$O(\sqrt{N})$ group elements	$O(\sqrt{N})$ group elements	multi-exp of size $\sqrt{N}$	$\sqrt{N}$ multi-exps of size $\sqrt{N}$	$O(N)$ field ops
Bulletproofs (Section 14.4)	one group element	$O(\log N)$ group elements	multi-exp of size $O(N)$	multi-exp of size $O(N)$	$O(N)$ group exps
Hyrax-commit+ Bulletproofs (Section 14.4.2)	$O(\sqrt{N})$ group elements	$O(\log N)$ groups elements	multi-exp of size $\sqrt{N}$	$\sqrt{N}$ multi-exps of size $\sqrt{N}$	$O(N)$ field ops
Dory (Section 15.4)	one group element	$O(\log N)$ group elements	multi-exp of size $O(\log N)$	$\sqrt{N}$ multi-exps of size $\sqrt{N}$	$O(N)$ field ops

raising the ith entry of the SRS to the power s^i, τ is effectively updated to $\tau \cdot s \mod p$, which is a random element of $\{1, \ldots, p-1\}$.

KZG commitments can also be more computation-intensive for the committer than alternatives. For example, it requires a similar number of public-key cryptographic operations (i.e., group exponentiations) as the second approach, but as with Dory these operations must be in pairing-friendly groups, for which group operations may be concretely more expensive.

16.4 Additional Approaches

Recent polynomial commitment schemes that we do not discuss in this survey include [13], [57], [83], which are based on a cryptographic notion called groups of unknown order. Specifically, so-called DARKs [83] claimed a polynomial commitment scheme with similar asymptotic verifier complexity to Dory (logarithmic size evaluation proofs and verifier time), but contained a flaw in the security analysis. This was rectified in [57] and a later version of [83], and [13] improved the size of the evaluation proof from a logarithmic number of group elements to constant (verification time remains logarithmic, and prover costs increase substantially). Current constructions of groups of unknown order require a trusted setup or appear to be impractical. One construction is based on so-called class groups, which are currently impractical in this context [108] (see [179, Table 2] for microbenchmarks), and the other is based on so-called RSA groups modulo $\{-1, 1\}$, which require a trusted setup.

Another example is recent work [22], [71] that operates in a manner similar to Bulletproofs (Section 14.4), but modifies it in order to base security on lattice assumptions that are believed to be post-quantum secure. This approach currently appears to yield significantly larger proofs than the discrete-logarithm based protocols that it is inspired by.