Modern Zero Knowledge Cryptography - MIT IAP, 2023.1

Lecture 5: Commitment Schemes

18 January 2023

Lecturer: Ying Tong

Modern SNARKs consist of two components: an information-theoretic **interactive oracle proof** (IOP) [1]; and a compatible **cryptographic commitment scheme**, which "compiles" the IOP into an argument system [5]. This note introduces several commonly used commitment schemes.

An IOP is "information-theoretic" in that it provides soundness and zero-knowledge guarantees even when the prover and verifier are *computationally unbounded*. To make this possible, the proof system makes the idealised assumption of "oracle access": in other words, the verifier can only access the prover's messages through random queries.

The commitment scheme instantiates this oracle access using cryptographic primitives (e.g. a one-way function): as a consequence, the resulting argument system is only secure with respect to a *computationally bounded* prover and/or verifier. To realise a *succinct* argument system, the chosen commitment scheme must provide low communication complexity relative to the computation being proven.

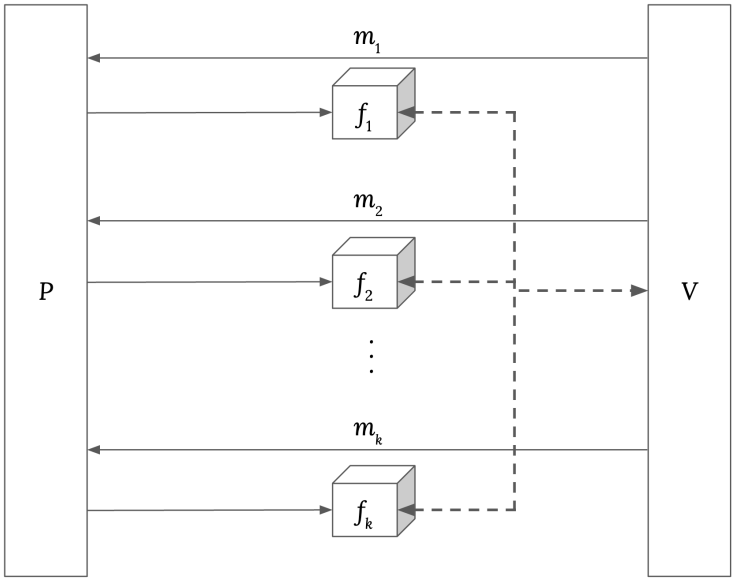

Figure 1: A k-round interactive oracle proof system. In the i-th round: the verifier $\mathcal{V}$ sends a message m_i to the prover $\mathcal{P}$; then, $\mathcal{P}$ replies with a message f_i, which $\mathcal{V}$ can query (via random access) in this and all later rounds. After the k rounds of interaction, $\mathcal{V}$ either accepts or rejects.

1 Formal definition

Definition 1.1 (Commitment scheme)**.** *A commitment scheme is a tuple* $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *of* PPT *algorithms where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *takes security parameter* λ *(in unary) and generates public parameters* pp*;*
- $\mathsf{Commit}(\mathsf{pp}\ ;\ m) \to (C; r)$ *takes a secret message* m *and outputs a public commitment* C *and (optionally) a secret opening hint* r *(which might or might not be the randomness used in the computation).*
- $\mathsf{Open}(\mathsf{pp}, C; m, r) \to b \in \{0,1\}$ *verifies the opening of the commitment* C *to the message* m *provided with the opening hint* r*.*

A commitment scheme Γ *is* ***binding*** *if for all* PPT *adversaries* $\mathcal{A}$*:*

$$\mathbf{Pr}\left[b_0 = b_1 \neq 0 \wedge m_0 \neq m_1 \ : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, m_0, m_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, C, m_0, r_0) \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, m_1, r_1) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Informally, this states that a valid commitment C to a message m_0 is binding if no adversary can produce a valid opening to some different message m_1.

A commitment scheme Γ *is* ***hiding*** *if for any polynomial-time adversary* $\mathcal{A}$*:*

$$\left| \mathbf{Pr}\left[b_0 = b' \ : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (m_0, m_1, st) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b \xleftarrow{\$} \{0,1\} \\ (C_b; r_b) \leftarrow \mathsf{Commit}(\mathsf{pp}; m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{pp}, st, C_b) \end{array} \right] - 1/2 \right| = \mathsf{negl}(\lambda)$$

Informally, this states that if a commitment is hiding if an adversary cannot "reverse-engineer" which of their messages was committed to.

Recall: "locked boxes". *[Lecture 3: "Mathematical Building Blocks"]*

In the Hamilton cycle example, we use a collision-resistant hash function to construct our commitment scheme:

- $r \leftarrow \{0,1\}^{256}$ is a randomly sampled secret key that is input to the hash function.
- $\mathsf{Commit} : \mathsf{hash}(m; r) = h$.
- Open : check $h \stackrel{?}{=} \mathsf{hash}(m; r)$.

2 Constructions

2.1 Vector commitment scheme

A vector commitment scheme [2] for the message space M is a **commitment scheme** for a vector $\vec{m} = (m_1, \ldots, m_k) \in \mathsf{M}^k$. The main security property for a vector commitment is *position binding*:

Definition 2.1 (Position binding)**.** *A vector commitment scheme* Γ *is* ***position binding*** *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathbf{Pr}\left[\begin{array}{l|l} \mathsf{Open}(\mathsf{pp}, C, \vec{m}, i) \to 1 & \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ \mathsf{Open}(\mathsf{pp}, C, \vec{m'}, i) & \mathcal{A}(\mathsf{pp}) \to (c, \vec{m}, \vec{m'}, i) \\ m \neq m' & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Informally, this states that no adversary can open C to two different values at the same position.

Vector Pedersen commitment. The Pedersen commitment [9] is a binding and hiding commitment scheme for the message space $\mathbb{F}_q$. For a secret message $m \in \mathbb{Z}_q$:

- $\mathsf{Pedersen.Setup}(1^\lambda, q) \to \mathsf{pp}$: $\mathsf{pp} = G, H \in \mathbb{G}$, where $\mathbb{G}$ is a **cryptographic group** of order q.
- $\mathsf{Pedersen.Commit}(\mathsf{pp}; m) \to (C; r)$: $C = [m]G + [r]H$, where $r \in \mathbb{Z}_q$ is a random secret.
- $\mathsf{Pedersen.Open}(\mathsf{pp}, C; m, r) \to \{0, 1\}$: the prover $\mathcal{P}$ reveals m and r, and the verifier $\mathcal{V}$ checks $C \stackrel{?}{=} [m]G + [r]H$.

Note that the Pedersen commitment is additively homomorphic:

$$\begin{aligned} \mathsf{Commit}(m, r) + \mathsf{Commit}(m', r') &= [m]G + [r]H + [m']G + [r']H \\ &= [m + m']G + [r + r']H \\ &= \mathsf{Commit}(m + m', r + r'). \end{aligned}$$

Exercise. *Convince yourself that the* Pedersen *commitment scheme is hiding and binding.*

Note that a ***cryptographic group*** $\mathbb{G}$ *is a group where the problem of finding a discrete logarithm of a group element* P *to a given base* G *is hard in general. In other words, it is hard to find* x *such that* $P = [x]G$*.*

- **hiding**: given a commitment C, is every value m equally likely to be the value committed in C?
- **binding**: can a prover find $m, m', m \neq m'$ such that $C = \mathsf{Pedersen.Commit}(\mathsf{pp}; m) \wedge \mathbb{C} = \mathsf{Pedersen.Commit}(\mathsf{pp}; m')$?

We can extend the Pedersen commitment scheme to get VectorPedersen over vectors in the message space $\mathbb{F}_q^k$. For a message $\vec{m} = (m_0, \ldots, m_{k-1})$:

- $\mathsf{VectorPedersen.Setup}(1^\lambda, q, k) \to \mathsf{pp}$: $\mathsf{pp} = (G_0, \ldots, G_{k-1}), H \in \mathbb{G}$, where $\mathbb{G}$ is a cryptographic group of order q.
- $\mathsf{VectorPedersen.Commit}(\mathsf{pp}; \vec{m}) \to (C; r)$: $C = [r]H + \sum_{i=0}^{k-1}[m_i]G_i$, where $r \in \mathbb{Z}_q$ is a random secret.
- $\mathsf{VectorPedersen.Open}(\mathsf{pp}, C; \vec{m}, r) \to \{0, 1\}$: the prover $\mathcal{P}$ reveals m and r, and the verifier $\mathcal{V}$ checks $C \stackrel{?}{=} [r]H + \sum_{i=0}^{k-1}[m_i]G_i$.

Exercise. *Convince yourself that* VectorPedersen *is additively homomorphic, i.e.*

$$\begin{aligned}&\mathsf{VectorPedersen.Commit}(\mathsf{pp}; \vec{m}) + \mathsf{VectorPedersen.Commit}(\mathsf{pp}; \vec{m'})\\ &= \mathsf{VectorPedersen.Commit}(\mathsf{pp}; m + m').\end{aligned}$$

Merkle tree. A well-known solution for building vector commitments is a Merkle tree [8]. It is used in distributed systems like Git, Cassandra, and Bitcoin for summarizing sets of data. Merkle trees also have hiding and extractability properties, which make them an ideal candidate for compiling an IOP into a non-interactive proof secure in the random oracle model [1].

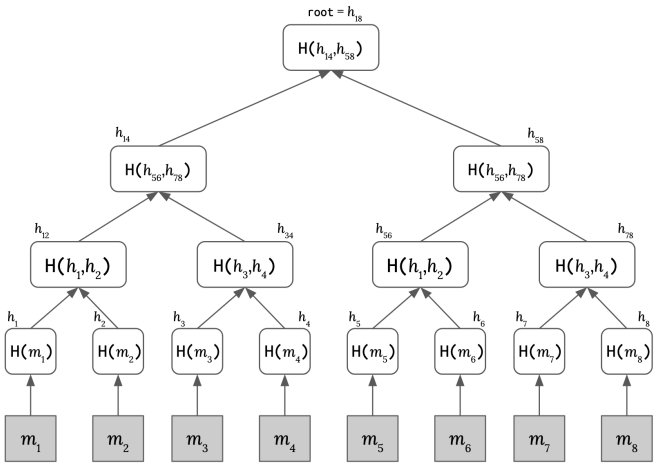

Figure 2: Each inner node of a Merkle tree is the hashed value of its two children.

- $\mathsf{Merkle.Commit}(\mathsf{pp}; \vec{m}) \to C$: for each $m_i \in \vec{m}$, compute a hash $h_i = \mathsf{Hash}(m_i)$. Compute the inner nodes of the Merkle tree $h_{ij} = \mathsf{Hash}(h_i, h_j)$. Output $C = \mathsf{root} = h_{1q}$.
- $\mathsf{Merkle.Open}(\mathsf{pp}, C, \vec{m}, i) \to b \in \{0, 1\}$:
  a) the prover $\mathcal{P}$ computes the path of inner nodes from h_i to root, and $\pi = (m_i, \mathsf{path})$;
  b) the verifier $\mathcal{V}$ checks that root can be recovered by hashing m_i with path.

2.2 Polynomial commitment scheme

A **univariate** polynomial commitment scheme is a **commitment scheme** for the message space $\mathbb{F}^{\leq d}[X]$, the ring of univariate polynomials with maximum degree $d \in \mathbb{N}$ and coefficients in the field $\mathbb{F} = \mathbb{Z}_p$.

It supports an argument of knowledge for proving the correct evaluation of a committed polynomial at a given point. A lot of information can be encoded in a polynomial: we will see in *Lecture 7 ("Arithmetizations")* and *Lecture 8 ("PlonK and polynomial identities")* how an arbitrary relation can be represented as a polynomial.

KZG commitment scheme [6]. This is used in protocols like Sonic [7], Marlin [3], and PlonK [4]. (Marlin and PlonK improve on Sonic by constructing a different polynomial IOP.)

Recall: Pairing-based cryptography. *[Lecture 3: "Mathematical Building Blocks"]*

Given cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, all of the same prime order p, a **pairing** is a nondegenerate bilinear map

$$e \ : \ \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,$$

a) **bilinear**: $e([a]P, [b]Q) = e(P, Q)^{a \cdot b}$

b) **nondegenerate**: with generators $G_1 \in \mathbb{G}_1$ and $G_2 \in \mathbb{G}_2, G_T := e(G_1, G_2) \in \mathbb{G}_T$ is a generator.

In this lecture, we will use the shorthand notation $[x]_1 := [x]G_1, [x]_2 := [x]G_2$, for any $x \in \mathbb{F}_p$.

- $\mathsf{KZG.Setup}(1^\lambda, d) \to \mathsf{srs}$: set $\mathsf{srs} = (\mathsf{ck}, \mathsf{vk}) = (\{[\alpha^i]_1\}_{i=0}^{d-1}, [\alpha]_2)$. α here is a secret element and must be discarded after the Setup.
- $\mathsf{KZG.Commit}(\mathsf{ck}; f(X)) \to C$: for $f(X) = \sum_{i=0}^{n-1} f_i X^i, C = \sum_{i=0}^{n-1} [f_i][\alpha^i]_1 = [f(\alpha)]_1$.
- $\mathsf{KZG.Open}(\mathsf{srs}, C, x, y; f(X)) \to \{0, 1\}$: To "open" the commitment at evaluation point x to a claimed value y,
  a) the prover $\mathcal{P}$ computes the quotient polynomial $q(X) = \frac{f(X)-y}{X-x}$ and sends the verifier $\pi = \mathsf{KZG.Commit}(\mathsf{ck}; q(X)) = [q(\alpha)]_1$;
  b) the verifier $\mathcal{V}$ checks $e(C - [y]_1, H) \stackrel{?}{=} e(\pi, [\alpha]_2 - [x]_2)$.

The a) and b) steps in KZG.Open are often written as two separate algorithms:

- $\mathsf{Open}(\mathsf{ck}, C, x, y; f(X)) \to \pi$ returns an opening proof for the relation

$$\mathcal{R} := \left\{ (\mathsf{ck}, C, x, y; f(X)) \ : \ \begin{array}{l} C = \mathsf{KZG.Commit}(\mathsf{ck}; f(X)) \\ \wedge \deg(f(X)) \leq d \\ \wedge \ y = f(x) \end{array} \right\};$$

- $\mathsf{Verify}(\mathsf{vk}, C, x, y, \pi) \to \{0, 1\}$ verifies the opening proof's correctness.

2.3 Additional resources

This note hopes to serve as a high-level introduction to commitment schemes, and where they fit in the construction of modern SNARKs. Below are a few excellent resources for further understanding and comparing commitment schemes:

2.3.1 More polynomial commitments

- **polynomial commitments: building block for universal SNARKS** (Justin Drake): includes a taxonomy of polynomial commitment schemes based on the cryptographic primitives used (hash functions, pairing group, unknown order group, and discrete log group). (Parts [1], [2], and [3].)
- **KZG polynomial commitments** (Dankrad Feist): an introduction to the KZG polynomial commitment scheme, and how to extend it to multiproofs and vector commitments.
- **Inner Product Arguments** (Dankrad Feist): an introduction to the inner product argument (IPA) protocol, a primitive that can be used to build a polynomial commitment scheme. The IPA is often instantiated with the vector Pedersen commitment scheme.
- **`bulletproofs::notes::inner_product_proof`**: excellent write-up on IPA.
- **Anatomy of a STARK, Part 3: `FRI`** (Alan Szepieniec): an introduction to the FRI (Fast Reed-Solomon IOP of Proximity) protocol, an oracle proof of proximity. The STARK polynomial IOP instantiates FRI with Merkle trees.
- **Linear commitments** work over linear functions. (Note that a polynomial commitment is a special form of a linear commitment, since $p(X) = \sum p_i x^i$ can be written as the dot product of two vectors $(p_0, \ldots, p_{d-1})$ and $(1, x, \ldots, x^{d-1})$.) These are used in constructions like Vortex (lattice-based), Brakedown, and Orion.
- **Multilinear commitments** work over *multivariate linear polynomials*. They can be used to instantiate the **sumcheck protocol**, an interactive proof (IP) which is by itself neither zero-knowledge nor succinct for NP statements. zk-SNARKs which use the sumcheck protocol with multilinear commitment schemes include: Hyrax, Libra, Virgo, and Spartan.

2.3.2 Implementations and benchmarks

- **`arkworks-rs/poly-commit`**: a Rust library supporting four polynomial commitment schemes.
- **Polynomial Commitment Benchmark** (Remco Bloemen): benchmarks for the Commit algorithm in implementations of KZG, IPA-based, and FRI-based polynomial commitment schemes.

References

[1] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference*, pages 31–60. Springer, 2016.

[2] D. Catalano and D. Fiore. Vector commitments and their applications. In *International Workshop on Public Key Cryptography*, pages 55–72. Springer, 2013.

[3] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. Marlin: preprocessing zksnarks with universal and updatable srs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–768. Springer, 2020.

[4] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, 2019.

[5] Y. Ishai. Zero-knowledge proofs from information-theoretic proof systems. *Zkproofs Blog*, 2020.

[6] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.

[7] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

[8] R. C. Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1988.

[9] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1992.