Investigating IVC with Halo2

Rasmus Kirk Jakobsen - 201907084

2024-12-16 - 16:43:23 UTC

Contents

Introduction 1

Prerequisites 2

$\mathrm{PC_{DL}}$: The Polynomial Commitment Scheme 2
- Outline . . . 3
  - $\mathrm{PC_{DL}}$.Commit . . . 3
  - $\mathrm{PC_{DL}}$.Open . . . 4
  - $\mathrm{PC_{DL}}$.SuccinctCheck . . . 5
  - $\mathrm{PC_{DL}}$.Check . . . 5
- Completeness . . . 5
- Soundness . . . 6
- Zero-knowledge . . . 6
- Benchmarks . . . 6

$\mathrm{AS_{DL}}$: The Accumulation Scheme 6
- Common subroutine . . . 6
- Prover . . . 6
- Verifier . . . 7
- Decider . . . 7
- Completeness . . . 7
- Soundness . . . 7
- Zero-knowledge . . . 7
- Benchmarks . . . 7

Appendix 8
- Notation . . . 8
- CM: Pedersen Commitment . . . 8

Introduction

Halo2, can be broken down into the following components:

- **Plonk**: A general-purpose zero-knowledge proof scheme.
- **$\mathrm{PC_{DL}}$**: A Polynomial Commitment Scheme in the Discrete Log setting.
- **$\mathrm{AS_{DL}}$**: An Accumulation Scheme in the Discrete Log setting.
- **Pasta**: A Cycle of Elliptic Curves, namely **Pa**llas and Ve**sta**.

This project is focused on the components of $\mathrm{PC_{DL}}$ and $\mathrm{AS_{DL}}$. I used the 2020 paper *"Proof-Carrying Data from Accumulation Schemes"* as a reference. The project covers both the theoretical aspects of the scheme described in this document along with a rust implementation, both of which can be found in the project's repository.

The original paper had additional functions for generating prover and verifier keys, getting the public parameters and trimming input to fit the public parameters. I have chosen to omit these from the discussion below as these are fixed per-implementation.

Prerequisites

Basic knowledge on elliptic curves, groups, interactive arguments are assumed in the following text. There is also a heavy reliance on the Inner Product Proof from the Bulletproofs protocol, see the following resources on bulletproofs if need be:

- Section 3 in the original Bulletproofs paper
- From Zero (Knowledge) to Bulletproofs writeup
- Rust Dalek Bulletproofs implementation notes
- Section 4.1 of my bachelors thesis

PC_{DL}: The Polynomial Commitment Scheme

$$
\begin{aligned}
C_{i+1} &= \langle \boldsymbol{c}_{i+1}, \boldsymbol{G}_{i+1} \rangle + \langle \boldsymbol{c}_{i+1}, \boldsymbol{z}_{i+1} \rangle H' \\
&= \langle l(\boldsymbol{c}_i) + \xi_{i+1}^{-1} r(\boldsymbol{c}_i), l(\boldsymbol{G}_i) + \xi_{i+1} r(\boldsymbol{G}_i) \rangle + \langle l(\boldsymbol{c}_i) + \xi_{i+1}^{-1} r(\boldsymbol{c}_i), l(\boldsymbol{z}_i) + \xi_{i+1} r(\boldsymbol{z}_i) \rangle H' \\
&= \langle l(\boldsymbol{c}_i), l(\boldsymbol{G}_i) \rangle + \xi_{i+1} \langle l(\boldsymbol{c}_i)), r(\boldsymbol{G}_i \rangle + \xi_{i+1}^{-1} \langle r(\boldsymbol{c}_i), l(\boldsymbol{G}_i) \rangle + \langle r(\boldsymbol{c}_i), r(\boldsymbol{G}_i) \rangle \\
&+ (\langle l(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle + \xi_{i+1} \langle l(\boldsymbol{c}_i), r(\boldsymbol{z}_i) \rangle + \xi_{i+1}^{-1} \langle r(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle + \langle r(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle) H'
\end{aligned}
$$

We can further group these terms:

$$
\begin{aligned}
C_{i+1} &= \langle l(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle \quad + \langle r(\boldsymbol{c}_i), r(\boldsymbol{G}_i) \rangle \quad + \xi_{i+1} \langle l(\boldsymbol{c}_i), r(\boldsymbol{G}_i) \rangle \quad + \xi_{i+1}^{-1} \langle r(\boldsymbol{c}_i), l(\boldsymbol{G}_i) \rangle \\
&+ (\langle l(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle \quad + \langle r(\boldsymbol{c}_i), r(\boldsymbol{z}_i) \rangle) H' + \xi_{i+1} \langle l(\boldsymbol{c}_i), r(\boldsymbol{z}_i) \rangle H' + \xi_{i+1}^{-1} \langle r(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle H' \\
&= C_i \qquad\qquad\qquad\qquad\qquad\qquad\quad + \xi_{i+1} R_i \qquad\qquad\qquad\quad + \xi_{i+1}^{-1} L_i \\
&\textbf{Where:} \\
L_i &= \langle r(\boldsymbol{c}_i), l(\boldsymbol{G}_i) \rangle + \langle r(\boldsymbol{c}_i), l(\boldsymbol{z}_i) \rangle H' \\
R_i &= \langle l(\boldsymbol{c}_i), r(\boldsymbol{G}_i) \rangle + \langle l(\boldsymbol{c}_i), r(\boldsymbol{z}_i) \rangle H'
\end{aligned}
$$

And then simplify further:

$$
\begin{aligned}
\boldsymbol{L} &= (L_0, \dots, L_{\lg(n)-1}) \\
\boldsymbol{R} &= (R_0, \dots, R_{\lg(n)-1}) \\
\boldsymbol{C} &= (C_0, \dots, C_{\lg(n)}) \\
\boldsymbol{\xi} &= (\xi_0, \dots, \xi_{\lg(n)})
\end{aligned}
$$

Now we are ready to look at the check that the verifier makes:

$$
\begin{aligned}
C_0 &= \bar{C} + vH' = C + vH' \\
C_{\lg(n)} &= C_0 + \sum_{i=0}^{\lg(n)-1} \xi_{i+1}^{-1} L_i + \xi_{i+1} R_i
\end{aligned}
$$

The original definition of C_i:

$$= \langle \boldsymbol{c}_{\lg(n)}, \boldsymbol{G}_{\lg(n)} \rangle + \langle \boldsymbol{c}_{\lg(n)}, \boldsymbol{z}_{\lg(n)} \rangle H'$$

Vectors have length one, use the single elements $c^{(0)}, G^{(0)}, c^{(0)}, z^{(0)}$:

$$= c^{(0)} G^{(0)} + c^{(0)} z^{(0)} H'$$

The verifier has $c^{(0)} = c, G^{(0)} = U$ from $\pi \in \textbf{EvalProof}$:

$$= cU + cz^{(0)} H'$$

And finally, by construction of $h(X) \in \mathbb{F}_q^d[X]$

$$= cU + ch(z)H'$$

Which corresponds exactly to the check that the verifier makes.

Outline

We have four main functions:

- $\text{PC}_{\text{DL}}.\textsc{Commit}(p : \mathbb{F}_q^d[X], \omega : \textbf{Option}(\mathbb{F}_q)) \to \mathbb{E}(\mathbb{F}_q)$:

  Creates a commitment to the coefficients of the polynomial q of degree d with optional hiding ω, using pedersen commitments.

- $\text{PC}_{\text{DL}}.\textsc{Open}(p : \mathbb{F}_q^d[X], C : \mathbb{E}(\mathbb{F}_q), z : \mathbb{F}_q, \omega : \textbf{Option}(\mathbb{F}_q)) \to \pi_{\text{EVAL}}$:

  Creates a proof π that states: "I know $p \in \mathbb{F}_q^d[X]$ with commitment $C \in \mathbb{E}(\mathbb{F}_q)$ s.t. $p(z) = v$" where p is private and d, z, v are public.

- $\text{PC}_{\text{DL}}.\textsc{SuccinctCheck}(C : \mathbb{E}(\mathbb{F}_q), d : \mathbb{N}, z : \mathbb{F}_q, v : \mathbb{F}_q, \pi : \pi_{\text{EVAL}}) \to \textbf{Result}(\mathbb{F}_q^d[X], \mathbb{G})$:

  Cheaply checks that a proof π is correct. It is not a full check however, since an expensive part of the check is deferred until a later point.

- $\text{PC}_{\text{DL}}.\textsc{Check}(C : \mathbb{E}(\mathbb{F}_q), d : \mathbb{N}, z : \mathbb{F}_q, v : \mathbb{F}_q, \pi : \pi_{\text{EVAL}}) \to \textbf{Result}(\top, \bot)$:

  The full check on π.

PC_{DL}.Commit

$\text{PC}_{\text{DL}}.\textsc{Commit}$ is rather simple, we just take the coefficients of the polynomial and commit to them using a pedersen commitment:

Algorithm 1 $\text{PC}_{\text{DL}}.\textsc{Commit}$

Inputs

$p : \mathbb{F}_q^d[X]$	The univariate polynomial that we wish to commit to.
$\omega : \textbf{Option}(\mathbb{F}_q)$	Optional hiding factor for the commitment.

Output

$C : \mathbb{E}(\mathbb{F}_q)$	The pedersen commitment to the coefficients of polynomial p.

Require: $d \leq D$
Require: $(d + 1) = 2^k$, where $k \in \mathbb{N}$
1: Let $\boldsymbol{p}$ be the coefficient vector for p
2: Output $C := \text{CM}.\textsc{Commit}(\boldsymbol{G}, \boldsymbol{p}, \omega)$.

$\mathbf{PC_{DL}.Open}$

Algorithm 2 $\text{PC}_{\text{DL}}.\text{OPEN}$

Inputs

$p : \mathbb{F}_q^d[X]$	The univariate polynomial that we wish to open for.
$C : \mathbb{E}(\mathbb{F}_q)$	A commitment to the coefficients of p.
$z : \mathbb{F}_q$	The element that z will be evaluated on $v = p(z)$.
$\omega : \textbf{Option}(\mathbb{F}_q)$	Optional hiding factor for C. *Must* be included if C was created with hiding!

Output

EvalProof	Proof that states: "I know $p \in \mathbb{F}_q^d[X]$ with commitment $C \in \mathbb{E}(\mathbb{F}_q)$ s.t. $p(z) = v$"

Require: $d \leq D$
Require: $(d+1) = 2^k$, where $k \in \mathbb{N}$

Compute $v = p(z)$ and let $n = d + 1$.
Sample a random polynomial $\bar{p} \in \mathbb{F}_q^{\leq d}[X]$ such that $\bar{p}(z) = 0$.
Sample corresponding commitment randomness $\bar{\omega} \in \mathbb{F}_q$.
Compute a hiding commitment to $\bar{p}$: $\bar{C} \leftarrow \text{CM.COMMIT}(\boldsymbol{G}, \bar{p}, \bar{\omega}) \in \mathbb{G}$.
Compute the challenge $\alpha := \rho_0(C, z, v, \bar{C}) \in \mathbb{F}_q^*$.
Compute the polynomial $p' := p + \alpha\bar{p} = \sum_{i=0} c_i X_i \in \mathbb{F}_q[X]$.
Compute commitment randomness $\omega' := \omega + \alpha\bar{\omega} \in \mathbb{F}_q$.
Compute a non-hiding commitment to p': $C' := C + \alpha\bar{C} - \omega' S \in \mathbb{G}$.
Compute the 0-th challenge field element $\xi_0 := \rho_0(C', z, v) \in \mathbb{F}_q$, then $H' := \xi_0 H \in \mathbb{G}$.
Initialize the vectors ($\boldsymbol{c_0}$ is defined to be coefficient vector of p'):

$$\boldsymbol{c_0} := (c_0, c_1, \dots, c_d) \in F_q^n$$
$$\boldsymbol{z_0} := (1, z^1, \dots, z^d) \in F_q^n$$
$$\boldsymbol{G_0} := (G_0, G_1, \dots, G_d) \in \mathbb{G}_n$$

for $i \in [\lg(n)]$ **do**
  Compute $L_i := \text{CM.COMMIT}(l(\boldsymbol{G_{i-1}}) \mathbin{+\!\!+} H',\ r(\boldsymbol{c_{i-1}}) \mathbin{+\!\!+} \langle r(\boldsymbol{c_{i-1}}), l(\boldsymbol{z_{i-1}})\rangle,\ \perp)$
  Compute $R_i := \text{CM.COMMIT}(r(\boldsymbol{G_{i-1}}) \mathbin{+\!\!+} H',\ l(\boldsymbol{c_{i-1}}) \mathbin{+\!\!+} \langle l(\boldsymbol{c_{i-1}}), r(\boldsymbol{z_{i-1}})\rangle,\ \perp)$
  Generate the i-th challenge $\xi_i := \rho_0(\xi_{i-1}, L_i, R_i) \in \mathbb{F}_q$.
  Construct commitment inputs for the next round:

$$\boldsymbol{G_i} := l(\boldsymbol{G_{i-1}}) + \xi_i \cdot r(\boldsymbol{G_{i-1}})$$
$$\boldsymbol{c_i} := l(\boldsymbol{c_{i-1}}) + \xi_i^{-1} \cdot r(\boldsymbol{c_{i-1}})$$
$$\boldsymbol{z_i} := l(\boldsymbol{z_{i-1}}) + \xi_i \cdot r(\boldsymbol{z_{i-1}})$$

end for
Finally output the evaluation proof $\pi := (\boldsymbol{L}, \boldsymbol{R}, U := \boldsymbol{G}_{lg(n)}, c := \boldsymbol{c}_{lg(n)}, \bar{C}, \omega')$

The $\text{PC}_{\text{DL}}.\text{OPEN}$ algorithm simply follows the proving algorithm from bulletproofs. Except,in this case we are trying to prove we know polynomial p s.t. $v = \langle \boldsymbol{c_0}, \boldsymbol{z_0} \rangle$. So because z is public, we can get away with omitting the generators for $\boldsymbol{b}$ in the original protocol ($\boldsymbol{H}$).

$\mathbf{PC_{DL}}$.SuccinctCheck

Algorithm 3 $\text{PC}_{\text{DL}}.\textsc{SuccinctCheck}$

Inputs

$C : \mathbb{E}(\mathbb{F}_q)$	A commitment to the coefficients of p.
$d : \mathbb{N}$	The degree of p
$z : \mathbb{F}_q$	The element that p is evaluated on.
$v : \mathbb{F}_q$	The claimed element $v = p(z)$.
π : **EvalProof**	The evaluation proof produced by $\text{PC}_{\text{DL}}.\textsc{Open}$

Output

Result$((\mathbb{F}_q^d[X], \mathbb{G}), \bot)$	The algorithm will either succeed and output $(h : \mathbb{F}_q^d[X], U : \mathbb{G})$ if π is a valid proof and otherwise fail $(\bot)$.

Require: $d \leq D$
Require: $(d+1) = 2^k$, where $k \in \mathbb{N}$
Parse π as $(\boldsymbol{L}, \boldsymbol{R}, U := \boldsymbol{G}_{lg(n)}, c := \boldsymbol{c}_{lg(n)}, \bar{C}, \omega')$ and let $n = d + 1$.
Compute the challenge $\alpha := \rho_0(C, z, v, \bar{C}) \in F_q^*$.
Compute the non-hiding commitment $C' := C + \alpha\bar{C} - \omega' S \in \mathbb{G}$.
Compute the 0-th challenge: $\xi_0 := \rho_0(C', z, v)$, and set $H' := \xi_0 H \in \mathbb{G}$.
Compute the group element $C_0 := C' + vH' \in \mathbb{G}$.
for $i \in [\lg(n)]$ **do**
  Generate the i-th challenge: $\xi_i := \rho_0(\xi_{i-1}, L_i, R_i) \in \mathbb{F}_q$.
  Compute the i-th commitment: $C_i := \xi_i^{-1} L_i + C_{i-1} + \xi_i R_i \in \mathbb{G}$.
end for
Define the univariate polynomial $h(X) := \prod_{i=0}^{\lg(n)-1}(1 + \xi_{\lg(n)-i} X^{2^i}) \in \mathbb{F}_q[X]$.
Compute the evaluation $v' := c \cdot h(z) \in \mathbb{F}_q$.
Check that $C_{lg(n)} \stackrel{?}{=} \text{CM}.\textsc{Commit}(U \mathbin{+\!\!+} H', c \mathbin{+\!\!+} v', \bot)$
Output (h(X), U).

$\mathbf{PC_{DL}}$.Check

Algorithm 4 $\text{PC}_{\text{DL}}.\textsc{Check}$

Inputs

$C : \mathbb{E}(\mathbb{F}_q)$	A commitment to the coefficients of p.
$d : \mathbb{N}$	The degree of p
$z : \mathbb{F}_q$	The element that p is evaluated on.
$v : \mathbb{F}_q$	The claimed element $v = p(z)$.
π : **EvalProof**	The evaluation proof produced by $\text{PC}_{\text{DL}}.\textsc{Open}$

Output

Result$(\top, \bot)$	The algorithm will either succeed $(\top)$ if π is a valid proof and otherwise fail $(\bot)$.

Require: $d \leq D$
Require: $(d+1) = 2^k$, where $k \in \mathbb{N}$
Check that $\text{PC}_{\text{DL}}.\textsc{SuccinctCheck}(C, d, z, v, \pi)$ accepts and outputs (h, U).
Check that $U \stackrel{?}{=} \text{CM}.\textsc{Commit}(\boldsymbol{G}, \boldsymbol{h}, \bot)$, where $\boldsymbol{h}$ is the coefficient vector of the polynomial h.

Completeness

This section will both function as an explainer of what is going on in this algorithm, along with a more formal proof of completeness.

Soundness

Zero-knowledge

Benchmarks

AS_{DL}: The Accumulation Scheme

Common subroutine

Algorithm 5 $\text{AS}_{\text{DL}}.\textsc{CommonSubroutine}$

Inputs

$d : \mathbb{N}$	The degree of p.
$\boldsymbol{q} : \mathbb{F}_q^m$	New instances *and accumulators* to be accumulated.
π_V : **Option**(**AccHiding**)	Necessary parameters if hiding is desired.

Output

Result$((\mathbb{E}(\mathbb{F}_q), \mathbb{N}, \mathbb{F}_q, \mathbb{F}_q^d[X]), \bot)$	The algorithm will either succeed $(\mathbb{E}(\mathbb{F}_q), \mathbb{N}, \mathbb{F}_q, \mathbb{F}_q^d[X])$ if the instances has consistent degree and hiding parameters and otherwise fail ($\bot$).

Require: $d \leq D$
Require: $(d+1) = 2^k$, where $k \in \mathbb{N}$

for $i \in [m]$ **do**
  Parse q_i as a tuple $((C_i, d_i, z_i, v_i), \pi_i)$.
  Compute $(h_i(X), U_i) := \text{PC}_{\text{DL}}.(C_i, z_i, v_i, \pi_i)$.
  Check that $d_i = D$ (We accumulate only the degree bound D.)
end for
Parse π_V as (h_0, U_0, ω), where $h_0(X) = aX + b \in \mathbb{F}_q^1[X], U_0 \in \mathbb{G}$ and $\omega \in \mathbb{F}_q$
Check that U_0 is a deterministic commitment to h_0: $U_0 = \text{PC}_{\text{DL}}.\textsc{Commit}(h, \bot)$.
Compute the challenge $\alpha := \rho_1(\boldsymbol{h}, \boldsymbol{U}) \in \mathbb{F}_q$
Let the polynomial $h(X) := \sum_{i=0}^{m} \alpha^i h_i \in \mathbb{F}_q[X]$
Compute the accumulated commitment $C := \sum_{i=0}^{m} \alpha^i U_i$
Compute the challenge $z := \rho_1(C, h) \in \mathbb{F}_q$.
Randomize C: $\bar{C} := C + \omega S \in \mathbb{G}$.
Output $(\bar{C}, d, z, h(X))$.

Prover

Algorithm 6 $\text{AS}_{\text{DL}}.\textsc{Prover}$

Inputs

$d : \mathbb{N}$	The degree of p.
$\boldsymbol{q} : \mathbb{F}_q^m$	New instances *and accumulators* to be accumulated.

Output

Result(**Acc**, $\bot$)	The algorithm will either succeed $((\bar{C}, d, z, v, \pi), \pi_V) \in \textbf{Acc}$) if the instances has consistent degree and hiding parameters and otherwise fail ($\bot$).

Require: $d \leq D$
Require: $(d+1) = 2^k$, where $k \in \mathbb{N}$

Let $n = d + 1$
Sample a random linear polynomial $h_0 \in F_q[X]$
Then compute a deterministic commitment to h_0: $U_0 := \text{PC}_{\text{DL}}.\textsc{Commit}(h_0, \bot)$
Sample commitment randomness $\omega \in F_q$, and set $\pi_V := (h_0, U_0, \omega)$.
Then, compute the tuple $(\bar{C}, d, z, h(X)) := \text{AS}_{\text{DL}}.\textsc{CommonSubroutine}(d, \boldsymbol{q}, \pi_V)$.
Compute the evaluation $v := h(z)$
Generate the hiding evaluation proof $\pi := \text{PC}_{\text{DL}}.\textsc{Open}(h(X), \bar{C}, d, z, \omega)$.
Finally, output the accumulator $acc = ((\bar{C}, d, z, v, \pi), \pi_V)$.

Verifier

Algorithm 7 $\text{AS}_{\text{DL}}.\textsc{Verifier}$

Inputs

$\boldsymbol{q} : \mathbb{F}_q^m$	New instances *and accumulators* to be accumulated.
acc : **Acc**	The accumulator.

Output

Result$(\top, \bot)$	The algorithm will either succeed $(\top)$ if TODO and otherwise fail $(\bot)$.

Require: $acc.d \leq D$
Require: $(acc.d + 1) = 2^k$, where $k \in \mathbb{N}$
1: Parse acc as $((\bar{C}, d, z, v, \_), \pi_V)$
2: The accumulation verifier computes $(\bar{C}', d', z', h(X)) := \text{AS}_{\text{DL}}.\textsc{CommonSubroutine}(d, \boldsymbol{q}, \pi_V)$
3: Then checks that $\bar{C}' \stackrel{?}{=} \bar{C}, d' \stackrel{?}{=} d, z' \stackrel{?}{=} z$, and $h(z) \stackrel{?}{=} v$.

Decider

Algorithm 8 $\text{AS}_{\text{DL}}.\textsc{Decider}$

Inputs

acc : **Acc**	The accumulator.

Output

Result$(\top, \bot)$	The algorithm will either succeed $(\top)$ if TODO and otherwise fail $(\bot)$.

Require: $acc.d \leq D$
Require: $(acc.d + 1) = 2^k$, where $k \in \mathbb{N}$
1: Parse acc as $((\bar{C}, d, z, v, \pi), \_)$
2: Check $\top \stackrel{?}{=} \text{PC}_{\text{DL}}.\textsc{Check}(\bar{C}, d, z, v, \pi)$

Completeness

Soundness

Zero-knowledge

Benchmarks

Appendix

Notation

$(\mathbb{F}_q, \ldots, \mathbb{E}(\mathbb{F}_q))$: Same as $\mathbb{F}_q \times \cdots \times \mathbb{E}(\mathbb{F}_q)$

$\langle \boldsymbol{a}, \boldsymbol{G} \rangle$ where $\boldsymbol{a} \in \mathbb{F}_q^n, \boldsymbol{G} \in \mathbb{E}^n(\mathbb{F}_q)$: The dot product of field elements $\boldsymbol{a}$ and curve points $\boldsymbol{G}$ $(\sum_{i=0}^n a_i G_i)$.

$\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ where $\boldsymbol{a} \in \mathbb{F}_q^n, \boldsymbol{b} \in \mathbb{F}_q^n$: The dot product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

$l(\boldsymbol{a})$: Gets the left half of $\boldsymbol{a}$.

$r(\boldsymbol{a})$: Gets the right half of $\boldsymbol{a}$.

$\boldsymbol{a} \mathbin{+\!\!+} \boldsymbol{b}$ where $\boldsymbol{a} \in \mathbb{F}_q^n, \boldsymbol{b} \in \mathbb{F}_q^m$: Concatinate vectors to create $\boldsymbol{c} \in \mathbb{F}_q^{n+m}$.

$a \mathbin{+\!\!+} b$ where $a \in \mathbb{F}_q$: Create vector $\boldsymbol{c} = (a, b)$.

"Type aliases"

$\textbf{EvalProof} = (\mathbb{E}^{lg(n)}(\mathbb{F}_q), \mathbb{E}^{lg(n)}(\mathbb{F}_q), \mathbb{E}(\mathbb{F}_q), \mathbb{F}_q, \mathbb{E}(\mathbb{F}_q), \mathbb{F}_q)$

$\textbf{AccHiding} = (\mathbb{E}(\mathbb{F}_q), \mathbb{N}, \mathbb{F}_q, \mathbb{F}_q^d)$

$\textbf{Acc} = ((\mathbb{E}(\mathbb{F}_q), \mathbb{N}, \mathbb{F}_q, \mathbb{F}_q, \textbf{EvalProof}), \textbf{AccHiding})$

CM: Pedersen Commitment

As a reference, we include the Pedersen Commitment algorithm we use:

Algorithm 9 CM.Commit

Inputs

$\boldsymbol{m} : \mathbb{F}^n$	The vectors we wish to commit to.
$\boldsymbol{G} : \mathbb{E}(\mathbb{F})^n$	The generators we use to create the commitment.
$\omega : \textbf{Option}(\mathbb{F}_q)$	Optional hiding factor for the commitment.

Output

$C : \mathbb{E}(\mathbb{F}_q)$	The pedersen commitment.

1: Output $C := \boldsymbol{m} \cdot \boldsymbol{G} + \omega S)$.

```rust
pub fn commit(w: Option<&PallasScalar>, Gs: &[PallasAffine], ms: &[PallasScalar]) -> PallasPoint {
    assert!(
        Gs.len() == ms.len(),
        "Length did not match for pedersen commitment: {}, {}",
        Gs.len(),
        ms.len()
    );

    let acc = point_dot_affine(ms, Gs);
    if let Some(w) = w {
        S * w + acc
    } else {
        acc
    }
}
```