# O-SNARKs for the Win: A Note on Security and A Concrete Examples

Oana Ciobotaru [*1] and Alistair Stewart[2]

[1]OpenZeppelin
[2]Web3 Foundation Technologies

April 27, 2024

**Abstract**

In this work we revisit the notion of O-SNARKs in relation to a wide class of oracles which we call AGM respecting. These oracles support operations that are independent from the operations performed on the elliptic curve on which a SNARK prover runs. Our main result shows that many modern SNARKs (e.g., PLONK [1], Marlin [2], Groth16 [3]) are in fact O-SNARKs when combined with AGM respecting oracles. On one hand, from a theoretical perspective, AGM respecting oracles is a wide class which includes oracles often used when modelling practical applications. On the other hand, via a concrete example, we shed more light on the subtleties regarding where and how O-SNARK security is useful for complex real-world applications.

## 1 Introduction

TO DO

## 2 Preliminaries

We assume all algorithms receive an implicit security parameter $\lambda$. An efficient algorithm is one that runs in uniform probabilistic polynomial time (PPT) in the length of its input and $\lambda$. We assume the correct parameters for the curves, groups, pairings, the group generators, etc., have been generated and shared with all parties before running any algorithm or protocol. A function $f(\lambda)$ is negligible in $\lambda$, written as $\mathsf{negl}(\lambda)$, if $1/f(\lambda)$ grows faster than any polynomial in $\lambda$ and is overwhelming in $\lambda$ if $1 - f(\lambda)$ is negligible in $\lambda$. By $\mathsf{poly}(\lambda)$ we mean some polynomial in $\lambda$ and e.w.n.p. means except with probability $\mathsf{negl}(\lambda)$. We write $y = A(x; r)$ when algorithm $A$ on input $x$ and randomness $r$, outputs $y$. We write $y \leftarrow A(x)$ for picking randomness $r$ uniformly at random and setting $y = A(x; r)$. We denote by $|S|$ the cardinality of set $S$. $\mathbb{F}_{<d}[X]$ is the set of all polynomials of degree less than $d$ over the field $\mathbb{F}$. For any integer $n \geq 1$, we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote by $\mathbb{B} = \{0, 1\} \subset \mathbb{F}$.

### 2.1 Conditional NP Relations

By $\mathcal{R} = \{(x; w) : p(x, w) = 1\}$ we denote the binary relation such that $(x, w)$ fulfil predicate $p(x, w) = 1$. We say $\mathcal{R}$ is an NP relation if predicate $p$ can be checked in polynomial time in the length of both inputs $x$ and $w$ and $\mathcal{L}(\mathcal{R}) = \{x \mid \exists w \ s.t. \ (x, w) \in \mathcal{R}\}$ is an NP language w.r.t. predicate $p$. In such a case we call $x$ an *instance* and $w$ a *witness*.

[*] A significant part of the research was conducted while the first author was affiliated to Web3 Foundation Technologies.

To model a specific property of our NP relations, we introduce further notation which we call *conditional NP relation*, we denote it by

$$\mathcal{R}^c = \{(x;w) : (p_1(x,w) = 1 \mid c(x,w) = 1) \ \wedge \ p_2(x,w) = 1\}$$

and we interpret it as the NP relation containing the pairs of inputs and witnesses $(x, w)$ such that $c(x, w) = 1$, $p_1(x, w) = 1$ and $p_2(x, w) = 1$ hold. However, to prove that $(x, w) \in \mathcal{R}^c$ we assume/take it as a given that $c(x, w) = 1$ and we are left to prove only that $p_1(x, w) = 1$ and $p_2(x, w) = 1$ hold. The reason we separate predicate $c(x, w)$ from predicate $p_1(x, w)$ in the definition of $\mathcal{R}^c$ is that predicate $c(x, w)$ may be inefficient to prove inside a proof system (e.g., in our case, inside a SNARK); using the above separation, one can delegate the verification of $c(x, w)$ to a trusted party.
We explicitly include in the definition of any NP relation $\mathcal{R}$ or $\mathcal{R}^c$ the corresponding domain for each type of public input and that input is parsed by the honest parties without additional checks. When we make a statement about an NP relation we can interpret that as one about a conditional relation $\mathcal{R}^c$, where $c$ is the predicate that always outputs 1.

# 3 O-SNARKs

In the following, we remind the reader the definition of an O-SNARK from [4] and we prove that SNARKs such as PLONK [1], Marlin [2] or Groth16 [3] on which a flurry of both under development or in production practical applications are based, fulfil this definition for a commonly occurying and well-defined class of oracles. In order to do that, we start with a couple of building block definitions: we give a reminder of algebraic algorithms (see, for example [5]) and introduce our new notion of AGM respecting oracles.

Let $\mathbb{G}$ be a cyclic group of prime order $p$. Informally, an algorithm $A$ is algebraic with respect to $\mathbb{G}$ if it fulfils the following requirement: whenever $A$ outputs a group element $h \in \mathbb{G}$, it also outputs a representation $(s_1, \ldots, s_t) \in \mathbb{Z}_p^t$ such that $h = \sum_{i=1}^{t} s_i \cdot g_i$, where $(g_1, \ldots, g_t)$ is the list of all group elements that were given to $A$ during its execution so far.

It is easy to see that the property above is equivalent to $A$ outputting representation $(v_1, \ldots, v_l) \in \mathbb{Z}_p^l$ such that $h = \sum_{i=1}^{l} v_i \cdot g_i$, where $(g_1, \ldots, g_l)$ is the list of all group elements that were given to $A$ as input.

TO DO: Revise the entire next paragraph.
Let $\mathcal{R}$ be an NP relation, i.e., there exists a random access machine $M$ and a polynomial $t$ such that $(x, w) \in \mathcal{R}$ if and only if $M(x, w)$ outputs 1 after at most $t(\lambda)$ steps and $|x| \leq t(\lambda)$, $|w| \leq t(\lambda)$, where by $|x|$, $|w|$ we denote the length of the public input $x$ and of the witness $w$, respectively. We denote $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists\ w, (x, w) \in \mathcal{R}\}$. Finally, by $|M|$ we denote the upper bound (in the security parameter $\lambda$) of the time taken by $M$ to output 1 on input $(x, w) \in \mathcal{R}$.

Let $\mathbb{O} = \{\mathcal{O}\}_{i \in \mathcal{I}}$ be a family of oracles. We denote by $\mathcal{O} \leftarrow \mathbb{O}$ the process of sampling an oracle $\mathcal{O}$ from the family $\mathbb{O}$ according to some (possibly probabilistic) process. $\mathbb{O}$ can be a random oracle family or a signing oracle corresponding to a signature scheme. For any oracle family $\mathbb{O}$, we define an O-SNARK $\Pi$ for $\mathbb{O}$ as follows [4]. TO DO: Clarify if the next definition is inspired by or identical to the source mentioned!

**Definition 1** ($\mathcal{Z}$-auxiliary input (O-)SNARK for oracle family $\mathbb{O}$)**.** *A $\mathcal{Z}$-auxiliary input succinct non-interactive argument of knowledge for a relation $\mathcal{R}$ and with oracle access to oracle family $\mathbb{O}$ is a triple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ working as follows:*

- $\mathsf{Gen}(1^\lambda, \mathcal{R}) \rightarrow \mathsf{crs}$ : *on input a security parameter $\lambda \in \mathbb{N}$, the generation algorithm outputs a common reference string $\mathsf{crs} = (prs, vrs)$ consisting of a public pair of prover reference string $prs$ and a verification state $vrs$ TO DO Verification state or verification reference string?;*
- $\mathsf{Prove}(\mathsf{prs}, x, w, \mathcal{R}) \rightarrow \pi$ : *the proving algorithm takes as input a prover reference string $\mathsf{crs}$ and a pair $(x, w) \in \mathcal{R}$ and returns a proof $\pi$.*
- $\mathsf{Verify}(\mathsf{vrs}, x, \pi, \mathcal{R}) \rightarrow 0/1$ : *the verification algorithm takes as input a verification reference string $\mathsf{vrs}$, some public input $x$ and an alleged proof $\pi$ and if it accepts, it outputs 1 and if it rejects, it outputs 0.*

*that fulfils perfect completeness, succinctness and oracle access adaptive knowledge soundness for oracle family $\mathbb{O}$ as defined below. TO DO decide on use of* crs *or* prs *as parameter for* Prove*; same for* Verify*.*

**Perfect completeness:** *For every $(x, w) \in \mathcal{R}$ we have*

$$Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi, \mathcal{R}) = 1 \mid \mathsf{Gen}(1^\lambda, \mathcal{R}) \rightarrow \mathsf{crs} \ \wedge \ \mathsf{Prove}(\mathsf{crs}, x, w, \mathcal{R}) \rightarrow \pi] = 1.$$

**Succinctness:** *For every large enough security parameter $\lambda \in \mathbb{N}$ and every $x \in \mathcal{L}_\mathcal{R}$, we have that* Gen *runs in* $\mathsf{poly}(\lambda)$ *time,* Prove *runs in* $\mathsf{poly}(\lambda + |M| + |x|)$ *time,* Verify *runs in* $\mathsf{poly}(\lambda + |x|)$ *time and an honestly generated proof has size* $\mathsf{poly}(\lambda)$*. TO DO is this the definition of succinctness that we want? Is this not too relaxed? To replace* poly *by* polylog*?*

**Adaptive knowledge soundness:** *Consider the following experiment for security parameter $\lambda \in \mathbb{N}$, adversary $\mathcal{A}$, extractor $\mathcal{E}_\mathcal{A}$, auxiliary input generator $\mathcal{Z}$ :*

$$\begin{aligned} &AddaptPoK(\lambda, \mathcal{A}, \mathcal{E}_\mathcal{A}, \mathcal{Z}) : \\ &\qquad \mathsf{aux} \leftarrow \mathcal{Z}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda); \\ &\qquad (y, \pi) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{aux}); w \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{crs}, \mathsf{aux}); \\ &\qquad if \ \mathsf{Verify}(\mathsf{crs}, y, \pi) = 1 \wedge (y, w) \in R \ return \ 1 \\ &\qquad else \ return \ 0 \end{aligned}$$

$\Pi$ *satisfies adaptive proof of knowledge soundness with respect to auxiliary input from $\mathcal{Z}$ if for every non-uniform oracle prover $\mathcal{A}$ making at most $Q(\lambda) = \mathsf{poly}(\lambda)$ queries there exists a non-uniform extractor $\mathcal{E}_\mathcal{A}$ and a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}[AddaptPoK(\lambda, \mathcal{A}, \mathcal{E}_\mathcal{A}, \mathcal{Z}) \implies 1] \leq \mathsf{negl}(\lambda).$$

*TO DO 1 Do we want to add at all this variant? Yes! Is this variant fully correct? Is it as per definition introduced by Dario? In fact, eventually, we want the Dario inspired definition to be combined with our light client paper new definition of SNARKs. But, maybe, we can show everything with Dario's definition and motivate at the last step the combined definition.*

**Oracle-access adaptive knowledge soundness for oracle family $\mathbb{O}$:** *Consider the following experiment for security parameter $\lambda \in \mathbb{N}$, adversary $\mathcal{A}$, extractor $\mathcal{E}_\mathcal{A}$, auxiliary input generator $\mathcal{Z}$ and oracle family $\mathbb{O}$:*

$$\begin{aligned} &OAddaptPoK(\lambda, \mathcal{A}, \mathcal{E}_\mathcal{A}, \mathcal{Z}, \mathbb{O}) : \\ &\qquad \mathsf{aux} \leftarrow \mathcal{Z}(1^\lambda); \mathcal{O} \leftarrow \mathbb{O}; \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda); \\ &\qquad (y, \pi) \leftarrow \mathcal{A}^\mathcal{O}(\mathsf{crs}, \mathsf{aux}); w \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{crs}, \mathsf{aux}, \mathsf{qt}); \\ &\qquad if \ \mathsf{Verify}(\mathsf{crs}, y, \pi) = 1 \wedge (y, w) \in R \ return \ 1 \\ &\qquad else \ return \ 0 \end{aligned}$$

*where $\mathsf{qt} = \{q_i, \mathcal{O}(q_i)\}$ is the transcript of all the $\mathcal{O}$ oracle queries and answers made and received by $\mathcal{A}$ during its execution. $\Pi$ satisfies oracle-access adaptive proof of knowledge with respect to oracle family $\mathbb{O}$ and auxiliary input from $\mathcal{Z}$ if for every non-uniform oracle prover $\mathcal{A}^\mathcal{O}$ making at most $Q(\lambda) = \mathsf{poly}(\lambda)$ queries to oracle $\mathcal{O}$ there exists a non-uniform extractor $\mathcal{E}_\mathcal{A}$ and a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}[OAddaptPoK(\lambda, \mathcal{A}, \mathcal{E}_\mathcal{A}, \mathcal{Z}, \mathcal{O}) \implies 1] \leq \mathsf{negl}(\lambda).$$

The following definition is inspired by the definition of the AGM model itself introduced in [6], and, in particular the formalism and concrete examples and counterexamples introduced at the end of Section 5.2.1.

**Definition 2** (AGM Respecting Oracles)**.** *Let $\{\mathbb{G}_i\}_{i \in I}$ be a set of cyclic groups of prime orders $\{p_i\}_{i \in I}$, respectively. We say that a family of oracles $\mathbb{O}$ is AGM respecting relative to $\{\mathbb{G}_i\}_{i \in I}$ if the probability of any oracle $\mathcal{O} \in \mathbb{O}$ outputting an element of $\{\mathbb{G}_i\}_{i \in I}$ or outputting an element which can be transformed via an efficient algorithm into an element of $\{\mathbb{G}_i\}_{i \in I}$ is 0.*

**Theorem 3** (O-SNARKS for AGM Respecting Oracles). *Let $\{\mathbb{G}_i\}_{i\in I}$ be a set of cyclic groups of prime orders $\{p_i\}_{i\in I}$, respectively, and let $\mathbb{O}$ be an AGM respecting oracle family relative to $\{\mathbb{G}_i\}_{i\in I}$ as per Definition 2. If $\Pi$ is a $\mathcal{Z}$-auxiliary input SNARK secure in the AGM model for any benign distribution $\mathcal{Z}$, then $\Pi_{\mathbb{O}}$ which follows $\Pi$ with the exception that the prover has oracle access to $\mathbb{O}$, is a $\mathcal{Z}$-auxiliary input O-SNARK for $\mathbb{O}$.*

*Proof.* TO DO 2; First try the proof using Dario's inspired definition only. □

**Corollary 4** (O-SNARKs from Pairing-based SNARKs). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order $p$ and let $\mathbb{O}$ be an AGM respecting oracle family relative to both $\mathbb{G}_1$ and $\mathbb{G}_2$ as per Definition 2. If $\Pi$ is a $\mathcal{Z}$-auxiliary input SNARK secure in the AGM model for any benign distribution $\mathcal{Z}$, then $\Pi_{\mathbb{O}}$ which follows $\Pi$ with the exception that the prover has oracle access to $\mathbb{O}$, is a $\mathcal{Z}$-auxiliary input O-SNARK for $\mathbb{O}$.*

*Proof.* TO DO □

# 4 A Fully Succinct Public Keys Aggregation Argument

Assume a set $S$ of public keys instantiated by points on a pairing-friendly elliptic curve and assume polynomial commitments $C_{S,x}$ and $C_{S,y}$ to the vectors of $x$ and $y$ of affine coordinates, respectively, of the public keys in $S$. In the following, we construct a *counting SNARK* which allows a prover to convince an efficient verifier that an alleged aggregated public key has been computed correctly as an aggregate (i.e., a sum) of a subset $U$ of the public keys in $S$, when $C_{S,x}$ and $C_{S,y}$ have been given as inputs. Additionally, our counting SNARK ensures that the alleged aggregated public key, in turn, is a scalar product between the entire set of public keys in $S$ and a *bitvector* with one bit associated to each public key for signalling the inclusion or omission of the respective public key in the subset $U$. TO DO We conclude this section by mentioning how to transform the counting SNARK into a SNARK for building non-accountable light client systems.

To compile our desired SNARK, we proceed as follows: in Section 4.1, we define vector-based conditional NP relations $\mathcal{R}_{\mathsf{c}}^{incl}$ (i.e., counting) and we design a ranged polynomial protocol $\mathcal{R}_{\mathsf{c}}^{incl}$ for this relation. The ranged polynomial protocol notion originates in [1]; we give a reminder of it in Appendix A including a refinement introduced in [7]. In Section 4.2, we use the two-steps PLONK-based compiler introduced in [7] which is necessary to compile the counting ranged polynomial protocol $\mathcal{R}_{\mathsf{c}}^{incl}$ into an O-SNARK for an NP relation based on mixed vector and pair of polynomial commitments; we denote this relation by $\mathcal{R}_{\mathsf{c},com}^{incl}$.

To start with, we define our vector based counting NP relation over $\mathbb{F}$ which is the base field of a pairing friendly elliptic curve $E_{inn}$. Our SNARK prover's circuit is defined as well over $\mathbb{F}$ as the scalar field of a second pairing friendly elliptic curve $E_{out}$. The vector of public keys, which is part of the public input for $\mathcal{R}_{\mathsf{c}}^{incl}$ and is denoted by $\mathbf{pk} = (pk_0, \ldots, pk_{n-2})$, is a vector of pairs with each component in $\mathbb{F}$. This vector has size $n-1$, where $n$ is the order of a group $H$ defined below. For $\mathcal{R}_{\mathsf{c}}^{incl}$, we denote by $\mathbf{bit}$ a bitvector with $n$ components where $\mathbf{bit} = (bit_0, \ldots, bit_{n-1})$, meaning that each component $bit_i$ belongs to $\mathbb{B} \subset \mathbb{F}$). Each bit $bit_i$ signals whether the index-wise corresponding public key ($pk_i$ is aggregated (i.e., summed in) or not into the aggregated public key $apk$. The $n$-th component $bit_{n-1}$ does not correspond to any public key, but, as will become clear in the following, has been included for easier design of constraints.

We denote by $H$ the multiplicative subgroup of $\mathbb{F}$ generated by an $n$-th root of unity $\omega$. We denote by $incl(a_0, \ldots, a_{n-2})$ the inclusion predicate that checks if $(a_0, \ldots, a_{n-2}) \in \mathbb{G}_{1,inn}^{n-1}$, where $\mathbb{G}_{1,inn}$ is the first source group of pairing function associated with $E_{inn}$. Moreover, let $h = (h_x, h_y)$ be some fixed, publicly known element in $E_{inn} \setminus \mathbb{G}_{1,inn}$. We denote by $(a_x, a_y)$ the affine representation in $x$ and $y$ coordinates, respectively, of $a \in E_{inn}$ and by $\oplus$ the point addition in affine coordinates on the elliptic curve $E_{inn}$.

## 4.1 Counting Ranged Polynomial Protocol

Next, we introduce the following Lagrange interpolation polynomials of degree at most $n-1 = |H|$ over cyclic group $H$:

- $b(X)$ - interpolates the bits of bitvector $bit$;
- $pkx(X)$, $pky(X)$ - interpolate all public keys' $x$ and $y$ coordinates, respectively;
- $kaccx(X)$, $kaccy(X)$ - interpolate $x$ and $y$ coordinates, respectively, of the iterative partial aggregate sum of the actual signing validators' public keys.

We also define six polynomial identities $id_1(X) = 0, \ldots, id_6(X) = 0$ supporting the following intuition:

- If $id_1(X) = 0$ and $id_2(X) = 0$ hold over $H$, this ensures the $x$ and, respectively, the $y$ coordinates of the iterative partial aggregate sums of the public keys (up to each index $i \leq n-2$) follow formulas $(*)$, $(**)$ from Observation 1 (see Appendix B) which gives all possible cases of complete curve point addition when the second curve point is multiplied by a bit;
- If $id_3(X) = 0$ and $id_4(X) = 0$ hold over $H$, this ensure the first partial aggregate sum is $h$ and the total aggregate sum is $h + apk$; this is necessary in order to ensure the addition of the public keys (i.e., elliptic curve points) never falls into condition (3) defined in Observation 1 (see Appendix B). This recursively implies the partial aggregate sum at every step is a well defined curve point, hence, it is a suitable input for the next step;
- If $id_5(X) = 0$ holds over $H$, this ensures $b(X)$ evaluates to bits over $H$.
- If $id_6(X) = 0$ holds over $H$, this ensures that the sum of bits in the bitvector $bit$ is $s+1$.

Intuitively and overall, if the identities $\{id_i(X) = 0\}_{i\in[6]}$ hold over $H$, this ensures $apk$ is the aggregated public key of at least $s$ and at most $s+1$ public keys. Hence, we interpret $s$ as a threshold on the number of public keys included in the aggregated public key. Note that since $bit_{n-1}$ as the last component of the bitmask witness $bit$ does not correspond to any public key and we have to account for the fact that $bit_{n-1}$ may be $1 \in \mathbb{F}$, relation $\mathcal{R}_{\mathsf{c}}^{incl}$ includes the off-by-one constraint $\sum_{i=0}^{n-1} bit_i = s+1$.

Conditional Counting Relation $\mathcal{R}_{\mathsf{c}}^{incl}$

$$\mathcal{R}_{\mathsf{c}}^{incl} = \{(\mathbf{pk} \in (\mathbb{F}^2)^{n-1}, s \in \mathbb{F}, apk \in \mathbb{F}^2; \mathbf{bit}) : apk = \sum_{i=0}^{n-2} [bit_i] \cdot pk_i \mid \mathbf{pk} \in \mathbb{G}_{1,inn}^{n-1} \wedge$$
$$\wedge \ \mathbf{bit} \in \mathbb{B}^n \ \wedge \ \sum_{i=0}^{n-1} bit_i = s+1\}$$

TO DO: EXPLAIN THE CONDITIONAL PART IN THIS RELATION.
The polynomials and polynomial identities computed and used are:

Polynomials as Computed by Honest Parties

$$\mathsf{b(X)} = \sum_{i=0}^{n-1} bit_i \cdot \mathsf{L_i(X)}; \mathsf{pkx(X)} = \sum_{i=0}^{n-2} pkx_i \cdot \mathsf{L_i(X)}; \mathsf{pky(X)} = \sum_{i=0}^{n-2} pky_i \cdot \mathsf{L_i(X)}$$
$$\mathsf{kaccx(X)} = \sum_{i=0}^{n-1} kaccx_i \cdot \mathsf{L_i(X)}; \mathsf{kaccy(X)} = \sum_{i=0}^{n-1} kaccy_i \cdot \mathsf{L_i(X)}; acc_{vt}(X) = \sum_{i=0}^{n-1} acc_{vt,i} \cdot L_i(X),$$

where $(pkx_0, \ldots, pkx_{n-2})$ and $(pky_0, \ldots, pky_{n-2})$ are computed such that $\forall i \in \{0, \ldots, n-2\}$, $pk_i$ is interpreted as a pair $(pkx_i, pky_i)$ with its components in $\mathbb{F}$; we also have $(kaccx_0, kaccy_0) = (h_x, h_y)$ and $(kaccx_{i+1}, kaccy_{i+1}) = (kaccx_i, kaccy_i) \oplus bit_i(pkx_i, pky_i)$, $\forall i < n-1$. Finally, $acc_{vt,i}$ are the components of the vector $(0, bit_0, bit_0 + bit_1, \ldots, \sum_{i=0}^{n-2} bit_i)$, $\forall i < n$.

Polynomial Identities

$$id_1(X) = (X - \omega^{n-1}) \cdot [b(X) \cdot ((kaccx(X) - pkx(X))^2 \cdot (kaccx(X) + pkx(X) + kaccx(\omega \cdot X)) -$$
$$- (pky(X) - kaccy(X))^2) + (1 - b(X)) \cdot (kaccy(\omega \cdot X) - kaccy(X))].$$
$$id_2(X) = (X - \omega^{n-1}) \cdot [b(X) \cdot ((kaccx(X) - pkx(X)) \cdot (kaccy(\omega \cdot X) + kaccy(X)) -$$
$$- (pky(X) - kaccy(X)) \cdot (kaccx(\omega \cdot X) - kaccx(X))) + (1 - b(X)) \cdot (kaccx(\omega \cdot X) - kaccx(X))].$$
$$id_3(X) = (kaccx(X) - h_x) \cdot L_0(X) + (kaccx(X) - (h \oplus apk)_x) \cdot L_{n-1}(X).$$
$$id_4(X) = (kaccy(X) - h_y) \cdot L_0(X) + (kaccy(X) - (h \oplus apk)_y) \cdot L_{n-1}(X).$$
$$id_5(X) = b(X)(1 - b(X)).$$
$$id_6(X) = acc_{vt}(\omega \cdot X) - acc_{vt}(X) - b(X) + (s+1) \cdot L_{n-1}(X).$$

### $H$-ranged Polynomial Protocol for Conditional Counting Relation $\mathcal{R}_{\mathsf{c}}^{incl}$

#### Protocol $\mathscr{P}_{\mathsf{c}}$

$\mathcal{P}_{poly}$ and $\mathcal{V}_{poly}$ know public input $s \in \mathbb{F}^2$, $\mathbf{pk} \in (\mathbb{F}^2)^{n-1}$ and $apk \in \mathbb{F}^2$ which are interpreted as per their respective domains.

1. $\mathcal{V}_{poly}$ computes $pkx(X)$, $pky(X)$.
2. $\mathcal{P}_{poly}$ sends polynomials $b(X)$, $kaccx(X)$, $kaccy(X)$, $acc_{vt}(X)$ to $\mathcal{I}$.
3. $\mathcal{V}_{poly}$ asks $\mathcal{I}$ to check whether the following polynomial relations hold over range $H$:
$$id_i(X) = 0, \forall i \in [6].$$
4. $\mathcal{V}_{poly}$ accepts if all of $\mathcal{I}$'s checks verify.

We show that protocol $\mathscr{P}_{\mathsf{c}}$ is an $H$-ranged polynomial protocol for conditional relation $\mathcal{R}_{\mathsf{c}}^{incl}$.

**Lemma 5.** *$\mathscr{P}_{\mathsf{c}}$ as described above is an $H$-ranged polynomial protocol for conditional relation $\mathcal{R}_{\mathsf{c}}^{incl}$.*

*Proof.* It is easy to see that perfect completeness holds. Indeed, if $(\mathbf{bit}, \mathbf{pk}, apk) \in \mathcal{R}_{\mathsf{c}}^{incl}$ holds, meaning that $\mathbf{bit} \in \mathbb{B}^n$ and $\mathbf{pk} \in \mathbb{G}_{1,inn}^{n-1}$ and $apk = \sum_{i=0}^{n-2}[bit_i] \cdot pk_i$ and $\sum_{i=0}^{n-1} bit_i = s + 1$ hold, then it is easy to see that the honest prover $\mathcal{P}_{poly}$ in $\mathscr{P}_{\mathsf{c}}$ will convince the honest verifier $\mathcal{V}_{poly}$ in $\mathscr{P}_{\mathsf{c}}$ to accept with probability 1.
Regarding knowledge-soundness, if the verifier $\mathcal{V}_{poly}$ in $\mathscr{P}_{\mathsf{c}}$ accepts, then we construct the extractor $\mathcal{E}$ in the following way. Using the polynomial $b(X)$ which was part of the messages from $\mathcal{P}_{poly}$ to $\mathcal{I}$ and evaluating it at the elements of the set $H$, $\mathcal{E}$ obtains evaluation vector $\mathbf{bit} = (b(1), \ldots, b(\omega^{n-1}))$ which, in the following, we denote as $(bit_0, \ldots, bit_{n-1}) \in \mathbb{F}^n$.
Next, we show that if $(pk_0, \ldots, pk_{n-2}) \in \mathbb{G}_{1,inn}^{n-1}$ holds and the verifier in $\mathscr{P}_{\mathsf{c}}$ accepts, then
$$((pk_0, \ldots, pk_{n-2}), s, apk, (bit_0, \ldots, bit_{n-1})) \in \mathcal{R}_{\mathsf{c}}^{incl},$$
which is equivalent to proving that $apk = \sum_{i=0}^{n-2}[bit_i] \cdot pk_i$ and $\mathbf{bit} \in \mathbb{B}^n$ and $\sum_{i=0}^{n-1} bit_i = s + 1$. First, since $id_6(X) = 0$ holds over $H$, we can expand that as follows:
$$\begin{aligned} acc_{vt,1} &= acc_{vt,0} + bit_0 \\ acc_{vt,2} &= acc_{vt,1} + bit_1 \\ &\ldots \\ acc_{vt,n-1} &= acc_{vt,n-2} + bit_{n-2} \\ acc_{vt,0} &= acc_{vt,n-1} + bit_{n-1} - (s+1). \end{aligned}$$
By summing together the LHS and, respectively, the RHS of the equalities above and reducing the equal terms, we obtain $s + 1 = \sum_{i=0}^{n-1} bit_i$ (1).
Second, since it holds over $H$ that $id_i(X) = 0$, $\forall i \in [5]$ and $b(\omega^i) = bit_i, \forall i < n$ (by the definition of $\mathcal{E}$), the properties concluded in Claim 8 from Appendix B hold. We call this property (2). TO DO: Is the next sentence correct/enough? By combining the properties in (1) and (2), we obtain the desired conclusion. □

## 4.2 From a Polynomial Protocol to an O-SNARK

One can use the standard PLONK compiler (see Lemma 4.7 [1]) to compile the counting ranged polynomial protocol for relation $\mathcal{R}_{\mathsf{c}}^{incl}$ introduced in Section 4.1 into a SNARK $\Pi_{\mathcal{R}_{\mathsf{c}}^{incl}}$ for the same relation. More precisely, let $\mathbb{O}$ be an AGM respecting oracle as per Definition 2 and let $\mathcal{Z}_{\mathbb{O}}$ be defined as in the statement of Theorem 3. Then according to the PLONK compiler, $\mathcal{Z}_{\mathbb{O}}$ auxiliary input $\Pi_{\mathcal{R}_{\mathsf{c}}^{incl}}$ is a SNARK.
In fact, what we have described so far corresponds to the first step of the compiler for the two-step compiler introduced in [7].[1] TO DO 3: Prove this compilation step holds or give a short argument. Next, analogously to the step two in the compilation introduced in [7], we prove that $\Pi_{\mathcal{R}_{\mathsf{c},com}^{incl}}$ is an O-SNARK for relation $\mathcal{R}_{\mathsf{c},com}^{incl}$ TO DO 4:, where

$$\mathcal{R}_{\mathsf{c},com}^{incl} = \{(\mathbf{C} \in \mathcal{C}, apk \in \mathbb{F}^2, s \in \mathbb{F}; \mathbf{pk}, \mathbf{bit}) : apk = \sum_{i=0}^{n-2} [bit_i] \cdot pk_i \mid \mathbf{pk} \in \mathbb{G}_{1,inn}^{n-1} \wedge$$

$$\wedge \ \mathbf{bit} \in \mathbb{B}^n \ \wedge \ \sum_{i=0}^{n-1} bit_i = s + 1 \ \wedge \ \mathbf{C} = \mathbf{Com}(\mathbf{pk})\}$$

where *Com* denotes the KZG polynomial commitment, $\mathcal{C}$ is the set of all *KZG* poly commitments or vectors of such poly commitments and $\mathbf{C}$ is an alleged pair of such commitments. Finally, either invoking Theorem 3 or via a direct proof, we show that $\Pi_{\mathcal{R}_{\mathsf{c},com}^{incl}}$ is an ($\mathcal{Z}_{\mathbb{O}}$ auxiliary input ??) O-SNARK. TO DO 5.
Having proven the above result, we can now combine the ($\mathcal{Z}_{\mathbb{O}}$ auxiliary input ??) O-SNARK $\Pi_{\mathcal{R}_{\mathsf{c},com}^{incl}}$ and an aggregatable signature scheme as defined in [7], and we can construct a non-accountable light client scheme which is a very close analog to the accountable light client schemes introduced in [7]. While the perfect completeness for the non-accountable light client scheme requires perfect completeness of the O-SNARK (which, in turn, is identical to the perfect completeness for a SNARK scheme), the soundness of the non-accountable scheme makes direct use of the knowledge-soundness property of the ($\mathcal{Z}_{\mathbb{O}}$ auxiliary input ??) O-SNARK for $\Pi_{\mathcal{R}_{\mathsf{c},com}^{incl}}$ which is specific for O-SNARKs. For more details on why is crucial we use an O-SNARK for the soundness security proof, please read Section 4.3.

## 4.3 Discussion

TO DO: COMPARISON BETWEEN THE SOUNDNESS PROOFS OF LIGHT CLIENT SYSTEMS WITH/WITHOUT O-SNARKs.

# 5 Implementation

We implemented and benchmarked our counting custom SNARK. The implementation allows us to evaluate the performance of our protocol and serve as prototype for future deployment. The implementation is open source and publicly available at `https://github.com/w3f/apk-proofs`. It is written in Rust and uses the Arkworks library.

Table 1 gives the prover and verifier time for three very related custom SNARK schemes: basic accountable and packed accountable as described in full in [7] and the counting custom SNARK introduced in this work with $v = n - 1 = 2^{10} - 1$, $v = n - 1 = 2^{16} - 1$ and $v = n - 1 = 2^{20} - 1$ signers. The benchmarks were run on commodity hardware, with an 2.2 GHz i7 and 16GB RAM. We remind the reader that by $v$ we denote the maximum number of validators in our system and that $n$ was defined in Section 4.

These signer numbers are approximately the range of the number of validators that we aimed our implementation at e.g. the Kusama blockchain (`https://kusama.network/`) has 1000 validators which is also the number that Polkadot is aiming for, and Ethereum 2 has about 348,000 validators and it

[1] As a reminder, the first compilation step in [7] uses a standard PLONK compiler.

| Custom SNARK | $v = 2^{10} - 1$ | | $v = 2^{16} - 1$ | | $v = 2^{20} - 1$ | |
|---|---|---|---|---|---|---|
| | prover | verifier | prover | verifier | prover | verifier |
| Basic Accountable | 564ms | 26ms | 22s | 46ms | 332s | 231ms |
| Packed Accountable | 830ms | 29ms | 31s | 33ms | 447s | 57ms |
| Counting | 761ms | 27ms | 31s | 30ms | 692s | 107ms |

Table 1: Proof and verifier times for the different schemes and numbers of signers

has been suggested that there will be no more than $2^{19}$ [8].

At $v = n - 1 = 1023$, the prover can generate a proof in any scheme in well under a second, which is short enough to generate a proof for every block in most prominent blockchains. Even for $v = n - 1 = 2^{20} - 1$, the prover time is under 6.4 minutes, the time for an Ethereum 2 epoch, the time that validators finalise the chain. For verification time, the basic accountable scheme is slower, considerably so for larger signer numbers.

Table 2 gives the number of operations the prover and verifier use. Table 3 gives the proof constituents and also the total proof and input sizes in bits. The basic accountable scheme's verifier performance at large numbers is so slow because it includes $O(n)$ field operations, which dominate the running time, however at 1023 signers it gives the smallest size. The packed accountable scheme, which includes $O(n/\lambda)$ field operations, fairs better on the benchmarks for large signer sets. The prover is considerably slower for the latter scheme because it needs to do additional operations. At larger signer sizes, the proof size is dominated by the bitfield.

| Scheme | Prover operations | Verifier operations |
|---|---|---|
| Basic Accountable | $12 \times FFT(N) + FFT(4N) + 9ME(N)$ | $2P + 11E + O(n)F$ |
| Packed Accountable | $18 \times FFT(N) + FFT(4N) + 12ME(N)$ | $2P + 16E + O(n/\lambda + log(n))F$ |
| Counting | $13 \times FFT(N) + FFT(4N) + 11ME(N)$ | $2P + 14E + O(log(n))F$ |

Table 2: Expensive prover and verifier operations. $FFT(M)$ is an FFT of size M. $ME(M)$ is a multi-scalar multiplication of size $M$. $P$ is a pairing, $E$ is a single scalar multiplication and $F$ is a field operation.

| Scheme | Proof | Input | Actual proof + input size in bits | | |
|---|---|---|---|---|---|
| | | | $v = 2^{10} - 1$ | $v = 2^{16} - 1$ | $v = 2^{20} - 1$ |
| Basic Accountable | $5\mathbb{G}_{1,out} + 5\mathbb{F}$ | $2\mathbb{G}_{1,out} + 1\mathbb{G}_{1,inn} + n$ bits | 9088 | 73600 | 1056640 |
| Packed Accountable | $8\mathbb{G}_{1,out} + 8\mathbb{F}$ | $2\mathbb{G}_{1,out} + 1\mathbb{G}_{1,inn} + n$ bits | 12544 | 77056 | 1060096 |
| Counting | $7\mathbb{G}_{1,out} + 7\mathbb{F}$ | $2\mathbb{G}_{1,out} + 1\mathbb{G}_{1,inn}$ | 10368 | 10368 | 10368 |

Table 3: Proof/input constituents and total proof/input size for implementation.

# 6 Conclusions

In this work we have revisited the notion of O-SNARKs in relation with a wide class of oracles which we call AGM respecting. Our result shows that many modern SNARKs (i.e., PLONK [1], Marlin [2], Groth16 [3], etc.) are in fact O-SNARKs with respect to AGM respecting oracles. Additionally, we shed more light via a concrete example where and how O-SNARK security is useful for practical and non-trivial real-world applications.

# References

[1] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge." Cryptology ePrint Archive, Paper 2019/953,

2019. https://eprint.iacr.org/2019/953.

[2] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," in *EUROCRYPT 2020*, pp. 738–768, 2020.

[3] J. Groth, "On the size of pairing-based non-interactive arguments," in *EUROCRYPT 2016*, pp. 305–326, 2016.

[4] D. Fiore and A. Nitulescu, "On the (in)security of snarks in the presence of oracles." Cryptology ePrint Archive, Paper 2016/112, 2016. https://eprint.iacr.org/2016/112.

[5] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *CRYPTO 2018*, pp. 33–62, 2018.

[6] J. Loss, *New techniques for the modular analysis of digital signature schemes*. PhD thesis, Ruhr University Bochum, Germany, 2019.

[7] O. Ciobotaru, F. Shirazi, A. Stewart, and S. Vasilyev, "Accountable light client systems for pos blockchains." Cryptology ePrint Archive, Paper 2022/1205, 2022. https://eprint.iacr.org/2022/1205.

[8] "Simplified active validator cap and rotation proposal," 2022. https://ethresear.ch/t/simplified-active-validator-cap-and-rotation-proposal/9022.

# A Ranged Polynomial Protocols for NP Relations

In the following, we keep the convention that all algorithms receive an implicit security parameter $\lambda$. The definition below is a natural extension of the notions of polynomial protocols and polynomial protocols for relations from Section 4 of PLONK [1] to polynomial protocols over ranges for conditional NP relations with additional refinements required by our specific use case; these refinements are incorporated into steps 4, 5 and 6 as follows:

**Definition 6.** *(Polynomial Protocols over Ranges for Conditional NP Relations) Assume three parties, a prover $\mathcal{P}_{poly}$, a verifier $\mathcal{V}_{poly}$ and a trusted party $\mathcal{I}$. Let $\mathcal{R}^c$ be a conditional NP relation (with $c$ being a predicate) and let $x$ be a public input both of which have been given to $\mathcal{P}_{poly}$ and $\mathcal{V}_{poly}$ by an InitGen efficient algorithm. For positive integers $d$, $D$, $t$, $l$, $u$, $e$ and for set $S \subset \mathbb{F}$, an $S$-ranged $(d, D, t, l, u, e)$-polynomial protocol $\mathscr{P}_{\mathcal{R}^c}$ for relation $\mathcal{R}^c$ is a multi-round protocol between $\mathcal{P}_{poly}$, $\mathcal{V}_{poly}$ and $\mathcal{I}$ such that:*

1. *The protocol $\mathscr{P}_{\mathcal{R}^c}$ definition includes a set of pre-processed polynomials $g_1(X), \ldots, g_l(X) \in \mathbb{F}_{<d}[X]$.*
2. *The messages of $\mathcal{P}_{poly}$ are sent to $\mathcal{I}$ and are of the form $f(X)$ for $f(X) \in \mathbb{F}_{<d}[X]$.*
   *If $\mathcal{P}_{poly}$ sends a message not of this form, the protocol is aborted.*
3. *The messages from $\mathcal{V}_{poly}$ to $\mathcal{P}_{poly}$ are random coins.*
4. *$\mathcal{V}_{poly}$ may perform arithmetic computations using input $x$ and the random coins used in the communication with $\mathcal{P}_{poly}$. Let $(res_1, \ldots, res_u)$ be the results of those computations which $\mathcal{V}_{poly}$ sends to $\mathcal{I}$.*
5. *Using vectors which are part of input $x$ and/or other ad-hoc vectors which $\mathcal{V}_{poly}$ deems useful, $\mathcal{V}_{poly}$ may compute interpolation polynomials $s_1(X), \ldots, s_e(X)$ over domain $S$ such that $s_1(X), \ldots, s_e(X) \in \mathbb{F}_{<d}[X]$. $\mathcal{V}_{poly}$ sends $s_1(X), \ldots, s_e(X)$ to $\mathcal{I}$.*
6. *At the end of the protocol, suppose $f_1(X), \ldots, f_t(X)$ are the polynomials that were sent from $\mathcal{P}_{poly}$ to $\mathcal{I}$. $\mathcal{V}_{poly}$ may ask $\mathcal{I}$ if certain polynomial identities hold between*

$$\{f_1(X), \ldots, f_t(X), g_1(X), \ldots, g_l(X), s_1(X), \ldots, s_e(X)\}$$

*over set $S$ (i.e., if by evaluating all the polynomials that define the identity at each of the field elements from $S$ we obtain a true statement). Each such identity is of the form*

$$F(X) := G(X, h_1(v_1(X)), \ldots, h_M(v_M(X))) \equiv 0,$$

*for some $h_i(X) \in \{f_1(X), \ldots, f_t(X), g_1(X), \ldots, g_l(X), s_1(X), \ldots, s_e(X)\}$,*
*$G(X, X_1, \ldots, X_M) \in \mathbb{F}[X, X_1, \ldots, X_M]$, $v_1(X), \ldots, v_M(X) \in \mathbb{F}_{<d}[X]$ such that $F(X) \in \mathbb{F}_{<D}[X]$ for every choice of $f_1(X), \ldots, f_t(X)$ made by $P_{poly}$ when following the protocol correctly. Note that some of the coefficients in the identities above may be from the set $\{res_1, \ldots, res_u\}$.*

7. *After receiving the answers from $I$ regarding the polynomial identities, $\mathcal{V}_{poly}$ outputs* acc *if all identities hold over set $S$, and outputs* rej *otherwise.*

*Additionally, the following properties hold:*

***Perfect Completeness:*** *If $\mathcal{P}_{poly}$ follows the protocol correctly and uses a witness $\omega$ with $(x, \omega) \in \mathcal{R}^c$, $\mathcal{V}_{poly}$ accepts with probability one.*
***Knowledge Soundness:*** *There exists an efficient algorithm $E$, that given access to the messages of $\mathcal{P}_{poly}$ to $\mathcal{I}$ it outputs $\omega$ such that, for any strategy of $\mathcal{P}_{poly}$, the probability of $\mathcal{V}_{poly}$ outputting* acc *at the end of the protocol and, simultaneously, $(x, \omega) \in \mathcal{R}^c$ is overwhelming in $\lambda$.*

Our definition for polynomial protocols over ranges does not include a zero-knowledge property as it is not required in our current work.

TO DO: Make sure that the definition of SNARK we use in the rest of the paper (unification of Dario's and of hybrid model SNARKs) is what we also mean below. Also, make sure we call the unified SNARK notion the same way throughout the paper. Add a proof for lemma below and refer to it in the rest of the paper.

Given the definition for polynomial protocols over ranges for conditional relations as detailed above, we are now ready to state the following result. The proof follows with only minor changes from that of Lemmas 4.5. and 4.7. from [1].

**Lemma 7.** *(Compilation of Ranged Polynomial Protocols for Conditional NP Relations into SNARKs using PLONK) Let $\mathscr{P}_{\mathcal{R}^c}$ be a public coin $S$-ranged $(d, D, t, l, u, e)$-polynomial protocol for relation $\mathcal{R}^c$ where only one identity is checked by $\mathcal{V}_{poly}$ and predicate $c$ from the definition of $\mathcal{R}^c$ needs to be fulfilled only by a part $x_1$ of the public input of the relation $\mathcal{R}^c$. Then one can construct a SNARK protocol $\mathscr{P}^*_{\mathcal{R}^c}$ for relation $\mathscr{P}_{\mathcal{R}^c}$ with $SNARK.PartInput$ as defined below and with $\mathscr{P}^*_{\mathcal{R}^c}$ secure in the AGM under the $2d$-DLOG assumption[2] such that:*

1. *The prover $\mathbf{P}$ in $\mathscr{P}^*_{\mathcal{R}^c}$ requires $\mathsf{e}(\mathscr{P}_{\mathcal{R}^c})$ $\mathbb{G}_{1,out}$-exponentiations where $\mathsf{e}(\mathscr{P}_{\mathcal{R}^c})$ is define analogously as in PLONK (see preamble of Section 4.2.), however it additionally takes into account polynomials $s_1(X), \ldots, s_e(X)$.*
2. *The total prover communication consists of $t + t^*(\mathscr{P}_{\mathcal{R}^c}) + 1$ $\mathbb{G}_{1,out}$-elements and $M$ $\mathbb{F}$-elements, where $t^*(\mathscr{P}_{\mathcal{R}^c})$ is defined identically as in PLONK (see preamble of Section 4.2.).*
3. *The verifier $\mathbf{V}$ in $\mathscr{P}^*_{\mathcal{R}^c}$ requires $t + t^*(\mathscr{P}_{\mathcal{R}^c}) + 1$ $\mathbb{G}_{1,out}$-exponentiations, two pairings and one evaluation of the polynomial $G$, and, additionally, the verifier in $\mathscr{P}^*_{\mathcal{R}^c}$ computes $e$ polynomial commitments to polynomials in the set $\{s_1(X), \ldots, s_e(X)\}$.*
4. *For $x_1$ part of $state_1$, the algorithm for computing partial inputs is defined as*

$$
\begin{aligned}
&SNARK.PartInput(srs, state_1, \mathcal{R}^c)\\
&If\ \ c(x_1) = 0\\
&\quad Return\\
&Else\\
&\quad Return(state_1, x_1)
\end{aligned}
$$

# B Delayed Proofs for Section 4

The following statements and proofs are taken over from [7] and included below for convenience. They are delayed from Section 4.

First, we remind the reader the incomplete addition formulae for curve points in affine coordinates, over elliptic curve in short Weierstrasse form and state:

[2]Definition 2.1. in PLONK [1] formally describes the $2d$-DLOG assumption.

*Observation 1:* Suppose that $bit \in \{0,1\}$, $(x_1, y_1)$ is a point on an elliptic curve in short Weierstrasse form, and, if $bit = 1$, so is $(x_2, y_2)$. We claim that the following equations:

$$bit((x_1 - x_2)^2(x_1 + x_2 + x_3) - (y_2 - y_1)^2) + (1 - bit)(y_3 - y_1) = 0 \ (*)$$
$$bit((x_1 - x_2)(y_3 + y_1) - (y_2 - y_1)(x_3 - x_1)) + (1 - bit)(x_3 - x_1) = 0 \ (**)$$

hold if and only if one of the following three conditions hold

1. $bit = 1$ and $(x_1, y_1) \oplus (x_2, y_2) = (x_3, y_3)$ and $x_1 \neq x_2$
2. $bit = 0$ and $(x_3, y_3) = (x_1, y_1)$
3. $bit = 1$ and $(x_1, y_1) = (x_2, y_2)$[3].

It is easy to see that each of the conditions 1,2,3 above implies equations $(*)$ and $(**)$. For the implication in the opposite direction, if we assume that $(*)$ and $(**)$ hold, then

*Case a:* For $bit = 0$, the first term of each equation $(*)$ and $(**)$ vanishes, leaving us with $y_3 - y_1 = 0$ and $x_3 - x_1 = 0$ which are equivalent to condition 2.

*Case b:* For $bit = 1$ and $x_1 = x_2$, by simple substitution in $(*)$ and $(**)$, we obtain $y_1 = y_2$, i.e., condition 3.

*Case c:* For $bit = 1$ and $x_1 \neq x_2$, then we can substitute $\beta = \frac{y_2 - y_1}{x_2 - x_1}$ into equations $(*)$ and $(**)$, leaving us with

$$x_1 + x_2 + x_3 = \beta^2 \text{ and } y_3 + y_1 = \beta(x_3 - x_1).$$

which are the usual formulae for short Weierstrass form addition of affine coordinate points when $x_1 \neq x_2$ so this is equivalent to Condition 1.

**Claim 8.** *Assume that $\forall i < n-1$ such that $bit_i = 1$, $pk_i = (pkx_i, pky_i) \in \mathbb{G}_{1,inn}$. If polynomial identities $id_i(X) = 0, \forall i \in [5]$, hold over range $H$ and the polynomial $b(X)$ has been constructed via interpolation on $H$ such that $b(\omega^i) = bit_i, \forall i < n$ then the following four properties hold:*
$bit_i \in \mathbb{B} = \{0,1\} \subset \mathbb{F}, \forall i < n \qquad (1)$,
$(kaccx_0, kaccy_0) = (h_x, h_y) \qquad (2)$,
$(kaccx_{n-1}, kaccy_{n-1}) = (h_x, h_y) \oplus (apk_x, apk_y) \qquad (3)$,
$(kaccx_{i+1}, kaccy_{i+1}) = (kaccx_i, kaccy_i) \oplus bit_i(pkx_i, pky_i), \ \forall i < n-1 \qquad (4)$.

*Proof.* The properties (1), (2) and (3) in the claim are easy to derive from the polynomial identities $id_3(X) = 0, id_4(X) = 0, id_5(X) = 0$ holding over $H$. To prove property (4), we apply the above *Observation* 1 by noticing that if $id_1(X)$ and $id_2(X)$ hold over $H$, then $(*)$ and $(**)$ hold with $(x_1, y_1)$ substituted by $(kaccx_i, kaccy_i)$, $(x_2, y_2)$ substituted by $(pkx_i, pky_i)$, $(x_3, y_3)$ substituted by $(kaccx_{i+1}, kaccy_{i+1})$ and $bit$ substituted by $bit_i$ for $0 \leq i \leq n-2$. Moreover, since

$$(kaccx_0, kaccy_0) = (h_x, h_y) \in E_{inn} \setminus \mathbb{G}_{1,inn}$$

and if $(pkx_i, pky_i) \in \mathbb{G}_{1,inn}$ whenever $bit_i = 1$, then $\forall i < n-1$ equations $(*)$ and $(**)$ obtained after the substitution defined above are equivalent to either condition 1 or condition 2, but never condition 3, so the result of the sum (i.e., $(kaccx_{i+1}, kaccy_{i+1})$, $0 \leq i \leq n-2$) is, by induction, at each step a well-defined point on the curve. □

[3]Note that under condition 3, $(x_3, y_3)$ can be any point whatsoever, maybe not even on the curve. The same holds true for $(x_2, y_2)$ under the condition 2.