A security analysis of the Zcash Sapling Protocol

Ariel Gabizon Daira Hopwood

Zcash

1 Introduction

The purpose of this note is to show that the Sapling protocol, that will be used for Zcash shielded (private) transactions as of the Sapling network upgrade, satisfies certain security properties. This document is not completely self contained and while reading it we recommend referring often to the Zcash protocol spec[3] for full details of the Sapling protocol.

A noteable property of the protocol is a separation of proving and signing authority. A "delegated spender/prover" creates transactions with the help of a proof authorizing key (or just proving key for short), but the transaction is not valid until it is signed by the signer with the spending key, that roughly corresponds to the secret key when thinking of the proving key as a public key.

We informally describe the four properties we prove.

1. **Non-malleability**: The delegated spender, after receiving a set of signatures on transactions of his choice, should not be able to create a new valid transaction, containing a nullifier appearing in one of the old transactions (overlapping nullifiers intuitively correspond to transactions from the same spending key). The way non-malleability is defined and proved here is inspired from the Zerocash paper[1].

2. **Indistinguishability**: An adversary should not be able to find two tupples (input notes, output notes) that are consistent in public data - meaning mainly that the amount going in or out of the shielded pool is the same, such that it is possible to distinguish from seeing the transaction which tupple it corresponds to.

3. **Balance**: An adversary should not be able to construct a valid ledger (even when having full control of transactions inserted) such that the total amount coming out of the shielded pool is larger than what came in.

4. **Spendability**: An adversary should not be able to send the honest party a note that was successfully received, but cannot be later spent (such an attack on [1] was found by Zooko Wilcox and coined "Faerie Gold" in [3]).

Before getting into the Sapling protocol and these properties, we begin with preliminary definitions and results regarding signature schemes.

2 Signature schemes

When we say an algorithm $\mathcal{A}$ is *efficient*, we mean it runs in time $\mathrm{poly}(\lambda)$ for the "security parameter" λ.

Definition 2.1. *Let $\mathbb{G}$ be a group of prime order r. A signature scheme $\mathscr{S}$ over $\mathbb{G}$ in the random oracle model consists of algorithms $\mathscr{S} = (\mathsf{sign}, \mathsf{verifySig}, \mathcal{S} = (\mathcal{S}_{\mathsf{sign}}, \mathcal{S}_{\mathcal{R}}))$ where $\mathsf{sign}, \mathsf{verifySig}$ are oracle machines with access to an oracle $\mathcal{R}$ taking as input arbitrary strings and returning uniform elements of $\mathbb{F}_r$. Such that the following holds.*

- *The set of public/verification keys $\{\mathsf{pk}\}$ is $\mathbb{G}$, and the set of private keys $\{\mathsf{sk}\}$ is $\mathbb{F}_r$.*
- *For $\mathsf{sk} \in \mathbb{F}_r$, the verification key of sk is $\mathsf{pk} = \mathsf{sk} \cdot g$ for a fixed generator $g \in \mathbb{G}$.*
- *We have the following "zero-knowledge" property: Fix any efficient $\mathcal{A}$. Suppose that $\mathcal{A}$ interacts with $\mathcal{S}$ with two types of queries*

  1. *Queries x, for an arbitrary string x that are answered according to $\mathcal{S}_{\mathcal{R}}$.*
  2. *Queries $(\mathsf{pk}, \mathbf{m})$, answered according to $\mathcal{S}_{\mathsf{sign}}$.*

  Let π_1 be the distribution of the sequence of queries and replies to $\mathcal{A}$. Let π_2 be the distribution of the sequence of queries and replies to $\mathcal{A}$ when

  1. *$\mathcal{R}$ takes the place of $\mathcal{S}_1$*
  2. *$\mathsf{sign}^{\mathcal{R}}(\mathsf{sk}, \mathbf{m})$ is returned instead of $\mathcal{S}_2(\mathsf{pk}, \mathbf{m})$ where sk is the secret key corresponding to pk.*

  Then the distance between π_1 and π_2 is $\mathrm{negl}(\lambda)$.

We say $\mathscr{S}$ is unforgeable w.r.t key randomization *if the following holds. Fix any efficient $\mathcal{A}$. A party $\mathscr{O}$ chooses uniform $\mathsf{sk} \in \mathbb{F}_r$ and sends $\mathsf{pk} = \mathsf{sk} \cdot g$ to $\mathcal{A}$. $\mathscr{O}$ also initializes an empty set T. $\mathcal{A}$ adaptively makes $\mathrm{poly}(\lambda)$ queries of the form $(\alpha, \mathbf{m})$. $\mathscr{O}$ replies with $\sigma := \mathsf{sign}(\mathsf{pk} + \alpha \cdot g, \mathbf{m})$ and adds $(\alpha, \mathbf{m}, \sigma)$ to T.*

Finally $\mathcal{A}$ outputs $(\alpha^*, \mathbf{m}^*, \sigma^*)$. Let $\mathsf{pk}^* := \mathsf{pk} + \alpha^* \cdot g$. Then the probability that

1. *$\mathsf{verifySig}(\mathsf{pk}^*, \mathbf{m}^*, \sigma^*)$, and*
2. *$(\alpha^*, \mathbf{m}^*, \sigma^*) \notin T$*

is $\mathrm{negl}(\lambda)$.

We assume our group $\mathbb{G}$ has a hard DL problem; meaning that for any efficient $\mathcal{A}$, given uniform $g, \mathsf{sk} \cdot g \in \mathbb{G}$ the probability of outputting sk is $\mathrm{negl}(\lambda)$.

We define the non-malleable version of Schnorr's signature scheme:

<u>Schnorr</u>**:**

Parameters: Group $\mathbb{G}$ of prime order r. Non-zero $g \in \mathbb{G}$.

Signing: Given message $\mathbf{m}$ and sk,

- Choose random $a \in \mathbb{F}_r$ and let $R := a \cdot g$
- Compute $c := \mathcal{R}(R, \mathsf{pk}, \mathbf{m})$
- Let $u := a + c \cdot \mathsf{sk}$.
- Define $\mathsf{sign}^{\mathcal{R}}(\mathsf{sk}, \mathbf{m}) := (R, u)$.

Verifying: Given $\mathsf{pk}, \mathbf{m}, \sigma = (R, u)$, $\mathsf{verifySig}^{\mathcal{R}}(\mathsf{pk}, \mathbf{m}, \sigma)$ accepts iff:

- Computing $c := \mathcal{R}(R, \mathsf{pk}, \mathbf{m})$; we have $u \cdot g = R + c \cdot \mathsf{pk}$.

Simulating:

- $\mathcal{S}_{\mathcal{R}}(\mathrm{x})$ checks if x has been queried before; if so answers consistently, otherwise answers uniformly in $\mathbb{F}_r$ and records the answer.
- $\mathcal{S}_{\mathsf{sign}}(\mathsf{pk}, \mathbf{m})$: Choose uniform $c, u \in \mathbb{F}_r$. Define $R := u \cdot g - c \cdot \mathsf{pk}$ and $\mathrm{x} := (R, \mathsf{pk}, \mathbf{m})$. Check if $\mathcal{S}_{\mathcal{R}}(\mathrm{x})$ has been defined. If so, abort. Otherwise define $\mathcal{S}_{\mathcal{R}}(\mathrm{x}) = c$ and return (R, u).

Remark 2.2. *At times when we wish to change the parameter g we work with from default to an element h, we will use it in the subscript, e.g. $sign_h^{\mathcal{R}}(\mathsf{sk}, \mathbf{m})$.*

We refer by $\mathsf{Schnorr}' = (\mathsf{sign}', \mathsf{verifySig}')$ to the Schnorr scheme where pk is omitted from the computation of c.

Theorem 2.3. $\mathsf{Schnorr}$ *is non-forgeable w.r.t randomization.*

Proof. Similarly to [2], we reduce to the non-forgeability of standard Schnorr (where the public key is not part of the signature & without randomization) that was proven in [4].

Suppose we are given $\mathcal{A}$ interacting with $\mathscr{O}$ as described above, and finally outputting $(\alpha^*, \mathbf{m}^*, \sigma^*)$. We construct $\mathcal{A}'$ that interacts with $\mathscr{O}'$ which is a "standard" Schnorr oracle.

That is:

1. $\mathscr{O}'$ begins by choosing a uniform $\mathsf{sk} \in \mathbb{F}_r$
2. $\mathscr{O}'$ computes $\mathsf{pk} = \mathsf{sk} \cdot g$ and sends pk to $\mathcal{A}'$. $\mathscr{O}'$ intializes an empty set T'.
3. $\mathcal{A}'$ sends queries $\mathbf{m}$ to $\mathscr{O}'$ and receives replies $\sigma = \mathsf{sign}'_{\mathsf{sk}}(\mathbf{m})$. $\mathscr{O}'$ adds $(\mathbf{m}, \sigma)$ to T'.
4. After all queries $\mathcal{A}'$ outputs $(\mathbf{m}^*, \sigma^*)$.

$\mathcal{A}'$ wins if

- $\mathsf{verifySig}'(\mathsf{pk}, \mathbf{m}^*, \sigma^*)$, and
- $(\mathbf{m}^*, \sigma^*) \notin T'$

$\mathcal{A}'$ will simulate $(\mathcal{A})$'s interaction with $\mathscr{O}$ using $\mathscr{O}'$: Given a query $(\alpha, \mathbf{m})$ of $\mathcal{A}$, $\mathcal{A}'$ queries $\mathscr{O}'$ with $\mathbf{m}' := (\mathsf{pk} + \alpha \cdot g, \mathbf{m})$, to receive reply $\sigma' = (R, u')$ - *this is a* $\mathsf{Schnorr}'$*-signature of* $\mathbf{m}'$ *with* sk*, and we now convert this to a* $\mathsf{Schnorr}$*-signature of* $\mathbf{m}$ *with* $\mathsf{sk} + \alpha$. Let $c := \mathcal{R}(R, \mathbf{m}') = \mathcal{R}(R, \mathsf{pk} + \alpha \cdot g, \mathbf{m})$. It sends $\sigma := (R, u := u' + c\alpha)$ to $\mathcal{A}$.

We have

$$u \cdot g = u' \cdot g + c\alpha \cdot g = R + c \cdot \mathsf{pk} + c\alpha \cdot g = R + c \cdot (\mathsf{pk} + \alpha \cdot g).$$

So we have $\mathsf{verifySig}(\mathsf{pk} + \alpha \cdot g, \mathbf{m}, \sigma)$. Also R is uniformly distributed, thus $\mathcal{A}'$ is answering $(\mathcal{A})$'s queries with the same distribution $\mathscr{O}$ would have.

Note that the mapping $F(\alpha, \mathbf{m}, \sigma) := (\mathbf{m}', \sigma')$ where $\mathbf{m}' := (\mathsf{pk} + \alpha \cdot g, \mathbf{m}), \sigma' := (R, u - c\alpha)$ is injective.

Let T be the set of tupples $(\alpha, \mathbf{m}, \sigma)$ such that $\mathcal{A}$ queried $(\alpha, \mathbf{m})$ and $\mathcal{A}'$ answered σ. We have $T' = \{F(x)\}_{x \in T}$.

When $\mathcal{A}$ finally outputs $x^* = (\alpha^*, \mathbf{m}^*, \sigma^*)$; $\mathcal{A}'$ outputs $F(x^*)$. As F is injective $x^* \notin T$ implies $F(x^*) \notin T'$.

Denote $(m', \sigma') := F(x^*)$. From [4]'s results on unforgeability of $\mathsf{Schnorr}'$, the probability that

- $\mathsf{verifySig}'(\mathsf{pk}, \mathbf{m}', \sigma')$, and
- $(\mathbf{m}', \sigma') \notin T'$

is negl(λ). Noting that $\mathsf{verifySig}'(\mathsf{pk}, \mathbf{m}', \sigma') \equiv \mathsf{verifySig}(\mathsf{pk} + \alpha \cdot g, \mathbf{m}^*, \sigma^*)$, this means that the probability that

- $\mathsf{verifySig}(\mathsf{pk} + \alpha \cdot g, \mathbf{m}^*, \sigma^*)$, and
- $x^* \notin T$

is negl(λ). This is exactly what we had to prove. $\square$

Invertible group samplers We assume that for our group $\mathbb{G}$ we have an efficient randomized procedure **sample** that produces a group element in $\mathbb{G}$ that is negl(λ)-close to uniform, such that there is an efficient deterministic algorithm **invert** that given the output of **sample**, produces w.p $1/\text{poly}(\lambda)$ over the randomness of **sample**, the randomness r used in that execution.

Note that when $\mathbb{G}$ is an elliptic curve group over $\mathbb{F}_r$ such a pair (**sample**, **invert**) is having **sample** try λ iterations of: Choose random $x \in \mathbb{F}_r$, check if there exists some $(x, y) \in \mathbb{G}$, if so output one of the two such elements randomly; and otherwise try another random $x \in \mathbb{F}_r$.

invert, given $(x, y) \in \mathbb{G}$, will output $(x, \mathsf{sign}(y))$, which will be correct in the case that the first iteration of **sample** produced a good x, which happens w.p. approximately half.

We also need that $\mathsf{Schnorr}$ is a proof of knowledge of discrete log. For this, we state the following theorem that is almost implicit in [4], but we provide a proof for completeness.

Theorem 2.4 (Extractability of Schnorr)**.** *Fix any integer function $\gamma = \gamma(\lambda)$ with $0 \leq \gamma(\lambda) \leq 1$ for any λ. There is an algorithm ξ with the following property. Fix any efficient $\mathcal{A}$ and group element $\mathsf{g} \in \mathbb{G}$. Suppose that $\mathcal{A}$ produces w.p. γ $(\mathsf{pk}, \mathbf{m}, \sigma)$ such that $\mathsf{verifySig}_{\mathsf{g}}^{\mathcal{R}}(\mathsf{pk}, \mathbf{m}, \sigma)$. Then, given the internal randomness used by $\mathcal{A}$ in the run and the vector r of replies of $\mathcal{R}$, ξ produces w.p $\gamma/2$ over $(\mathcal{A})$'s randomness, the randomness of $\mathcal{R}$ in answering $(\mathcal{A})$'s queries and its own randomness $s \in \mathbb{F}_r$ such that $\mathsf{pk} = s \cdot \mathsf{g}$. Furthermore, ξ's running time will be $F(\lambda, 1/\gamma)$ where F is a polynomial depending only on the running time of $\mathcal{A}$.*

Proof. Assume first that $\mathcal{A}$ is deterministic. Let $Q = \text{poly}(\lambda)$ be a bound on the number of queries to $\mathcal{R}$ made by $\mathcal{A}$. When $\mathcal{A}$ is deterministic its execution is fully determined by the vector $\mathsf{r} \in \mathbb{F}_r^q$ of replies by $\mathcal{R}$ to its queries.

Recall that $\mathsf{r} \in \mathbb{F}_r^Q$ denotes the Q oracle replies to the queries of $\mathcal{A}$ to $\mathcal{R}$. We call r *good* if $\mathcal{A}(\mathsf{r})$ outputs a verifying $(\mathsf{pk}, \mathbf{m}, \sigma)$. We assume for simplicity that whenever r is good $\mathcal{A}$ queries $\mathcal{R}$ at $(R, \mathsf{pk}, \mathbf{m})$ where $\sigma = (R, u)$. (Otherwise ξ can simulate an altered $\mathcal{A}$ that asks this query whenever r is good and the query hasn't been made yet.) For good r we define the index $i(\mathsf{r}) \in [Q]$ where the query $(R, \mathsf{pk}, \mathbf{m})$ was made. For $i \in [Q]$ and $\mathsf{r} \in \mathbb{F}_r^Q$ we define the subset $W|_{\mathsf{r}\setminus i}$ of $\mathbb{F}_r^Q$ to be the set of $\mathsf{r}' \in \mathbb{F}_r^Q$ that are equal to r outside of index i. We denote by $W_{\mathsf{r},i}$ the set of $\mathsf{r}' \in \mathbb{F}_r^Q$ that are contained in $W|_{\mathsf{r}\setminus i}$, are good, and have $i(\mathsf{r}') = i$. Note that there are at most $r^{Q-1} \cdot Q$ distinct sets $W_{\mathsf{r},i}$.

Note also that given two distinct elements $\mathsf{r} \neq \mathsf{r}' \in W_{\mathsf{r},i}$, the executions $\mathcal{A}(\mathsf{r}), \mathcal{A}(\mathsf{r}')$ give us two valid signatures with the same public key pk message $\mathbf{m}$ and first part R; and such two signatures enable computing sk.

Define the two functions

$$P = 4Q/\gamma, T = \lceil \ln 3 \cdot 2P \rceil.$$

Given r and $\mathcal{A}$, ξ does the following.

1. If r is not good, abort.
2. If λ is such that $r(\lambda) < 2P(\lambda)$, conduct a brute for search for sk such that $\mathsf{sk} \cdot \mathsf{g} = \mathsf{pk}$.
3. Otherwise, set $i = i(\mathsf{r})$. Sample T elements $\mathsf{r} \in W|_{\mathsf{r}\setminus i}$.
4. Run $\mathcal{A}$ using each of the samples as $\mathcal{R}$'s reply vector. If one of the sampled elements is good and different from r, use it to compute and output sk.

We claim ξ retrieves sk with probability at least $\gamma/2$. This claim will follow from two subclaims described below. Fix good r and let $i = i(\mathsf{r})$. Call r *dense* if (it is good and)

$$|W_{\mathsf{r},i}| \geq |W|_{\mathsf{r}\setminus i}|/P.$$

We first show that given dense r, ξ succeeds w.p. at least $2/3$ over its inner randomness: The event that ξ fails implies that in T samples of $W|_{\mathsf{r}\setminus i}$, it didn't find a distinct $\mathsf{r}' \in W_{\mathsf{r},i}$. This probability is bounded by

$$(1 - (1/P - 1/r))^T \leq (1 - 1/2P)^T \leq e^{-T/2P} \leq 1/3.$$

Next, we bound the probability of $\mathsf{r} \in \mathbb{F}_r^Q$ not being dense: The number of such r is at most

$$r^{Q-1} Q \cdot \frac{r}{P} \leq \frac{r^Q \cdot Q}{P}.$$

Thus the density of such elements is at most

$$Q/P \leq \gamma/4.$$

Now using these two subclaims, the probability of ξ succeeding to output sk is at least the probability of r being dense, multiplied by the probability of ξ succeeding conditioned on r being dense. This gives success probability at least

$$(3\gamma/4) \cdot (2/3) = \gamma/2.$$

Finally, if $\mathcal{A}$ is randomized, running ξ as above when $\mathcal{A}$ is fixed to whatever inner randomness it used and ξ received as input, gives the same success probability of ξ for randomized $\mathcal{A}$.

$\square$

3 Description of Sapling

3.1 Basic components

Functions, and their requirements:

We do not explicitly state function domains and ranges; see the spec for more details. Whenever discussing a function in the properties below, we always think of an infinite sequence of functions indexed by the security parameter λ.

1. For any fixed values $\mathsf{g}, \mathsf{pk}, \mathsf{v}$, and for any $\epsilon \geq 0$, $\mathbf{NC}(\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$ is ϵ-close to uniform when rcm is ϵ-close to uniform.

2. $\mathbf{NC}$ is collision resistant - i.e. the probability of finding $\mathsf{note}, \mathsf{note}'$ such that $\mathbf{NC}(\mathsf{note}) = \mathbf{NC}(\mathsf{note}')$ is $\mathrm{negl}(\lambda)$. [1]

3. For any fixed v and any $\epsilon \geq 0$, $\mathbf{VC}(\mathsf{v}, \mathsf{rcv})$ is ϵ-close to uniform whenever rcv is ϵ-close to uniform.

4. $\mathbf{VC}$ is collision-resistant.

5. **sighash** is collision-resistant.

6. $\mathbf{IVK}$ is collision-resistant.

7. $\mathbf{NF}$ is collision resistant (see another requirement for the indistinguishability property in Section 5).

Generators of $\mathbb{G}$ We assume we are given generators $\mathsf{g}_{\mathbf{sig}}, \mathsf{g}_{\mathbf{n}}, \mathsf{g}_{\mathbf{r}}, \mathsf{g}_{\mathbf{v}}$ that were sampled in a way that except w.p $\mathrm{negl}(\lambda)$ an efficient $\mathcal{A}$ cannot discover the discrete log relation between any two of them.

Statements:

$\underline{\mathsf{OUT}(\mathsf{cv}, \mathsf{cm}, \mathsf{epk})}$**:** I know $\mathsf{note} = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm}), \mathsf{rcv}, \mathsf{esk}$ such that

1. $\mathsf{cm} = \mathbf{NC}(\mathsf{note})$.

2. $\mathsf{cv} = \mathbf{VC}(\mathsf{v}, \mathsf{rcv})$.

3. $\mathsf{epk} = \mathsf{esk} \cdot \mathsf{g}$.

4. g has order greater than eight.

[1] A caveat here is that this is true when the rcm parameter is thought of as a field element; in the actual circuit it is received as a string of bits where some elements of $\mathbb{F}_r$ have multiple representations; inspection of the proof shows that it suffices that CR w.r.t rcm as a field element; same story with rcv in $\mathbf{VC}$.

$\underline{\mathsf{SPEND}(\mathsf{rt},\mathsf{cv},\mathsf{nf},\mathsf{rk})}$**:** I know $\mathsf{path},\mathsf{pos},\mathsf{note}=(\mathsf{g},\mathsf{pk},\mathsf{v},\mathsf{rcm}),\mathsf{cm},\mathsf{rcv},\alpha,\mathsf{ak},\mathsf{nsk}$ such that

1. $\mathsf{cm}=\mathbf{NC}(\mathsf{note})$.
2. Either $\mathsf{v}=0$ ("dummy note"); or path is a merkle path from cm at position pos to rt.
3. $\mathsf{rk}=\mathsf{ak}+\alpha\cdot\mathsf{g}_{\mathbf{sig}}$.
4. Setting $\mathsf{nk}:=\mathsf{nsk}\cdot\mathsf{g}_{\mathbf{n}},\mathsf{ivk}:=\mathbf{IVK}(\mathsf{ak},\mathsf{nk})$; we have $\mathsf{pk}=\mathsf{ivk}\cdot\mathsf{g}$.
5. $\mathsf{nf}=\mathbf{NF}(\mathsf{nk},\mathsf{cm},\mathsf{pos})$

Components

A *note* is a tupple $\mathsf{note}=(\mathsf{g},\mathsf{pk},\mathsf{v},\mathsf{rcm})$ where

1. $\mathsf{g},\mathsf{pk}\in\mathbb{G}$.
2. $\mathsf{v},\mathsf{rcm}\in\mathbb{F}_r$
3. $\mathsf{v}\leq\mathsf{MAX}$.

An *output base* $\mathsf{output}=(\mathsf{g},\mathsf{pk},\mathsf{v})$ is the same as a note excluding the rcm component.

Remark 3.1. *It is convenient for us to define a note with* g *rather than its* GH*-preimage* d *as in the spec, as this is what's given as input to the circuits; there are minor non-exploitable issues with this, see e.g. https://github.com/zcash/zcash/issues/3490.*

For $\mathsf{ivk}\in\mathbb{F}_r$ we say note *belongs to* ivk if $\mathsf{pk}=\mathsf{ivk}\cdot\mathsf{g}$.

An *input base*, usually denoted input, will consist of the values required to make an input in a Sapling transaction, except the spending key; namely $\mathsf{input}=(\mathsf{note},\mathsf{path},\mathsf{pos},\mathsf{pak})$ where

- note is a note
- path is a path in a merkle tree beginning from a leaf of value $\mathsf{cm}=\mathbf{NC}(\mathsf{note})$.
- pos is the position of cm amongst the leaves of the Merkle tree (pos is redundant here as it can be derived from path, but convenient).
- pak is a proving key to make SNARK spend proofs about the note.

We say input is *consistent with* rt if path ends at rt.

A *transaction input*, usually denoted inp, is the final form in which an input appears in a transaction; inp consists of

1. A value commitment cv.
2. A nullifier nf.
3. A Merkle root rt of the tree containing the used note.
4. A public key rk that is (allegedly) a randomized version of the spent note's proving key ak.
5. A SNARK proof π for the statement $\mathsf{SPEND}(\mathsf{rt},\mathsf{cv},\mathsf{nf},\mathsf{rk})$.

3.2 Methods

We use the convention that ℓ denotes the number of inputs in a transaction, and s the number of outputs.

<u>**makeinp**$(\mathsf{rt}, \mathsf{input} = (\mathsf{note}, \mathsf{path}, \mathsf{pos}, \mathsf{pak}), \mathsf{rcv}, \alpha)$</u>

where input is an input base consistent with rt.

1. $\mathsf{cm} = \mathbf{NC}\ (\mathsf{note})$
2. $\mathsf{nf} = \mathbf{NF}\ (\mathsf{nk},\mathsf{note},\mathsf{pos})$
3. Define $\mathsf{rk} := \mathsf{ak} + \alpha \cdot \mathsf{g}_{\mathbf{sig}}, \mathsf{cv} := \mathsf{v} \cdot \mathsf{g}_{\mathbf{v}} + \mathsf{rcv} \cdot \mathsf{g}_{\mathbf{r}}$.
4. Let $\pi = \pi_{\mathsf{spend}}(\mathsf{cv}, \mathsf{rt}, \mathsf{nf}, \mathsf{rk}; \mathsf{note}, \mathsf{pak}, \alpha, \mathsf{path}, \mathsf{pos})$.
5. Output $\mathsf{inp} = (\mathsf{cv}, \mathsf{rt}, \mathsf{nf}, \mathsf{rk}, \pi)$.

<u>**makeout** $(\mathsf{note} = (\mathsf{g},\mathsf{pk},\mathsf{v},\mathsf{rcm}),\mathsf{rcv})$,</u>

1. Choose random $\mathsf{esk} \in \mathbb{F}_r$.
2. Let $\mathsf{cv} := \mathbf{VC}(\mathsf{v}, \mathsf{rcv}) = \mathsf{v} \cdot \mathsf{g}_{\mathbf{v}} + \mathsf{rcv} \cdot \mathsf{g}_{\mathbf{r}}$.
3. Let $\mathsf{note} = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$ and $\mathsf{cm} := \mathbf{NC}(\mathsf{note})$.
4. Let $\mathsf{epk} = \mathsf{esk} \cdot \mathsf{g}$.
5. Let $\mathsf{enc} = \mathbf{ENC}_{\mathbf{KDF}(\mathsf{esk}\cdot\mathsf{pk},\mathsf{epk})}(\mathsf{note})$
6. Let $\pi = \pi_{\mathsf{output}}(\mathsf{epk}, \mathsf{cm}, \mathsf{cv}; \mathsf{note}, \mathsf{rcv}, \mathsf{esk})$.
7. Output $(\mathsf{cv}, \mathsf{cm}, \mathsf{epk}, \pi, \mathsf{enc})$

<u>**makerandomizedout** $(\mathsf{note} = (\mathsf{g},\mathsf{pk},\mathsf{v}),\mathsf{rcv})$,</u>

1. Choose random $\mathsf{esk}, \mathsf{rcm} \in \mathbb{F}_r$.
2. Let $\mathsf{cv} := \mathbf{VC}(\mathsf{v}, \mathsf{rcv}) = \mathsf{v} \cdot \mathsf{g}_{\mathbf{v}} + \mathsf{rcv} \cdot \mathsf{g}_{\mathbf{r}}$.
3. Let $\mathsf{note} = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$ and $\mathsf{cm} := \mathbf{NC}(\mathsf{note})$.
4. Let $\mathsf{epk} = \mathsf{esk} \cdot \mathsf{g}$.
5. Let $\mathsf{enc} = \mathbf{ENC}_{\mathbf{KDF}(\mathsf{esk}\cdot\mathsf{pk},\mathsf{epk})}(\mathsf{note})$
6. Let $\pi = \pi_{\mathsf{output}}(\mathsf{epk}, \mathsf{cm}, \mathsf{cv}; \mathsf{note}, \mathsf{rcv}, \mathsf{esk})$.
7. Output $(\mathsf{cv}, \mathsf{cm}, \mathsf{epk}, \pi, \mathsf{enc})$

<u>**bindval** $(\mathsf{raw}_{\mathsf{tx}} = (\overrightarrow{\mathsf{inp}},\overrightarrow{\mathsf{out}},\mathrm{v}^{\mathrm{bal}}),\overrightarrow{\mathsf{rcv}})$</u>

1. Let $r := \sum_{i=1}^{\ell} \mathsf{rcv}_i - \sum_{i=\ell+1}^{\ell+s} \mathsf{rcv}_i$
2. Let $S := \sum_{i=1}^{\ell} \mathsf{cv}_i - \sum_{i=\ell+1}^{\ell+s} \mathsf{cv}_i - \mathrm{v}^{\mathrm{bal}} \cdot \mathsf{g}_{\mathbf{v}}$

3. Let $\sigma_{\mathsf{bind}} := \mathsf{sign}_{\mathbf{g_r}}(r, \mathbf{sighash}(\mathsf{raw_{tx}}))$.

4. Output $\mathsf{pre\text{-}tx} := (\mathsf{raw_{tx}}, \sigma_{\mathsf{bind}})$.

<u>$\mathbf{signtx}(\mathsf{pre\text{-}tx} = (\mathsf{raw_{tx}}, \sigma_{\mathsf{bind}}), \overrightarrow{\mathsf{ask}}, \overrightarrow{\alpha})$</u>

1. For each $i \in [\ell]$, let $\sigma_i := \mathsf{sign}_{\mathbf{g_{sig}}}(\mathsf{ask}_i + \alpha_i, \mathbf{sighash}(\mathsf{raw_{tx}}))$

2. Let $\overrightarrow{\sigma} := (\sigma_1, \ldots, \sigma_\ell)$.

3. Output $(\mathsf{raw_{tx}}, \overrightarrow{\sigma})$.

Given $(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}})$ we say $(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}})$ is *consistent* with $\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}$, if

- for each $j \in [\ell]$ input_j is consistient with rt, i.e. pak_j is from $\mathbf{NC}(\mathsf{note}_j)$ to rt,
- $\sum_{j=1}^{\ell} \mathsf{v}_j - \sum_{j=\ell+1}^{\ell+s} \mathsf{v}_j = \mathrm{v}^{\mathrm{bal}}$.
- the positions $\left\{\mathsf{pos}_j\right\}_{j\in[\ell]}$ are all distinct.

and
<u>$\mathbf{makerandomizedtx}$ $(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}})$</u>
where $\mathsf{input}_j = (\mathsf{note}_j, \mathsf{pak}_j, \mathsf{path}_j, \mathsf{pos}_j), \mathsf{output}_j = (\mathsf{g}_j, \mathsf{pk}_j, \mathsf{v}_j)$

1. Choose random $\overrightarrow{\mathsf{rcv}} \in \mathbb{F}_r^s$.

2. For $j \in [\ell]$, $\mathsf{inp}_j = \mathbf{makeinp}(\mathsf{rt}, \mathsf{input}_j, \mathsf{rcv}_j)$

3. For $j \in [s]$, $\mathsf{out}_j = \mathbf{makeout}(\mathsf{output}_j, \mathsf{rcv}_j)$

4. $\mathsf{pre\text{-}tx} = \mathbf{bindval}(\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \mathrm{v}^{\mathrm{bal}})$.

5. Choose random $\overrightarrow{\alpha} \in \mathbb{F}_r^\ell$.

6. Output $\mathsf{tx} = \mathbf{signtx}(\mathsf{pre\text{-}tx}, \mathsf{ask}, \overrightarrow{\alpha})$

<u>$\mathbf{maketx}$ $(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \overrightarrow{\mathsf{rcv}}, \mathsf{ask}, \mathsf{pak})$</u> where $\mathsf{input}_j = (\mathsf{v}_j, \mathsf{note}_j, \mathsf{pak}_j, \mathsf{path}_j, \mathsf{pos}_j), \mathsf{output}_j = (\mathsf{g}_j, \mathsf{pk}_j, \mathsf{v}_j, \mathsf{rcm}_j)$

1. Choose random $\overrightarrow{\alpha} \in \mathbb{F}_r^\ell$.

2. For $j \in [\ell]$, $\mathsf{inp}_j = \mathbf{makeinp}(\mathsf{input}_j, \mathsf{rcv}_j, \alpha_j, \mathsf{pak})$

3. For $j \in [s]$, $\mathsf{out}_j = \mathbf{makeout}(\mathsf{output}_j, \mathsf{rcv}_j)$

4. Let $\mathrm{v}^{\mathrm{bal}} := \sum_{i=1}^{\ell} \mathsf{v}_i - \sum_{j=\ell+1}^{\ell+s} \mathsf{v}_j$.

5. $\mathsf{pre\text{-}tx} = \mathbf{bindval}(\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{rcv}})$.

6. Let $\mathsf{tx} = \mathbf{signtx}(\mathsf{pre\text{-}tx}, \overrightarrow{\alpha}, \mathsf{ask})$

<u>$\mathsf{verify\text{-}tx}(\mathrm{L}, \mathsf{tx})$</u>

1. Suppose that $\mathsf{tx} = (\mathsf{raw_{tx}}, \overrightarrow{\sigma})$.

2. For each $\mathsf{inp}_i = (\mathsf{rt}, \mathsf{cv}, \mathsf{nf}, \mathsf{rk}, \pi) \in \overrightarrow{\mathsf{inp}}(\mathsf{tx})$,

   - Check that $\mathsf{nf} \notin \mathsf{nf}(\mathrm{L}) \cup \left\{\mathsf{nf}(\mathsf{inp}_1), \ldots, \mathsf{nf}(\mathsf{inp}_{i-1})\right\}$.
   - Check that $\mathsf{spendverify}(\mathsf{rt}, \mathsf{cv}, \mathsf{nf}, \mathsf{rk}; \pi)$.
   - Check that $\mathsf{verifySig}^{\mathcal{R}}_{\mathsf{g_{sig}}}(\mathsf{rk}, \mathbf{sighash}(\mathsf{raw_{tx}}), \sigma_i)$

3. For each $\mathsf{out} = (\mathsf{cv}, \mathsf{cm}, \mathsf{epk}, \pi, \mathsf{enc}) \in \overrightarrow{\mathsf{out}}(\mathsf{tx})$, check that $\mathbf{outverify}(\mathsf{cv}, \mathsf{cm}, \mathsf{epk}; \pi)$

4. Let $S := \sum_{i=1}^{\ell} \mathsf{cv}_i - \sum_{i=\ell+1}^{\ell+s} \mathsf{cv}_i - \mathrm{v}^{\mathrm{bal}} \cdot \mathsf{g}_{\mathbf{v}}$.

5. Check that $\mathsf{verifySig}^{\mathcal{R}}_{\mathsf{g_r}}(S, \mathbf{sighash}(\mathsf{raw_{tx}}), \sigma_{\mathsf{bind}})$.

4 Non-Malleability of Sapling w.r.t. delegated spenders

We make the simplifying assumption when modelling non-malleability in this writeup; that *there is only one spending key* $(\mathsf{ask}, \mathsf{nsk})$ *of the honest signer involved, and all addresses are diversifed addresses derived from this spending key.*

Modelling the adversary:

We wish to show that the delegated spender cannot create any new transactions of her own. We model this claim with the following non-malleability game: We model the honest signer as an oracle $\mathscr{O}$ that $\mathcal{A}$ interacts with as follows.

$\mathscr{O}$ begins by choosing a new spending key $(\mathsf{ask}, \mathsf{nsk}) \leftarrow \mathcal{K}$ and sending the corresponding proof authorizing key $\mathsf{pak} = (\mathsf{ak}, \mathsf{nsk})$ to $\mathcal{A}$. Where $\mathsf{ak} = \mathsf{ask} \cdot \mathsf{g_{sig}}$.

Afterwords, $\mathcal{A}$ can make $\mathsf{sign\text{-}all\text{-}inputs}$ queries to $\mathscr{O}$, which intuitively correspond to asking for signatures on transactions whose inputs have spending key $(\mathsf{ask}, \mathsf{nsk})$ (though see remark).

Sign-all-inputs queries

1. $\mathcal{A}$ sends $(\mathsf{pre\text{-}tx} = (\mathsf{raw_{tx}}, \sigma_{\mathsf{bind}}), \overrightarrow{\alpha})$ to $\mathscr{O}$. Where $\mathsf{raw_{tx}} = (\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \mathrm{v}^{\mathrm{bal}})$

2. $\mathscr{O}$ checks if $\mathsf{spendverify}(\mathrm{pub}_i, \pi_i)$ holds for each $i \in [\ell]$ and otherwise aborts.

3. $\mathscr{O}$ computes for $i \in [\ell]$, $\sigma_i = \mathsf{sign}_{\mathsf{g_{sig}}}(\mathsf{ask} + \alpha_i, \mathbf{sighash}(\mathsf{raw_{tx}}))$.

4. Let $\overrightarrow{\sigma} := (\sigma_1, \ldots, \sigma_\ell)$. $\mathscr{O}$ return $\mathsf{tx} := (\mathsf{raw_{tx}}, \sigma_{\mathsf{bind}}, \overrightarrow{\sigma})$.

Remark 4.1. *The second item is another way of saying we assume $\mathcal{A}$ can only ask $\mathscr{O}$ for signatures of transactions with legitimate spend proofs. Otherwise the proof currently fails as we need to be able to extract the witness from each input.*

Terminology: We refer below to a transaction tx as $\mathsf{tx} = (\mathsf{raw}_{\mathsf{tx}}, \sigma_{\mathsf{bind}}, \overrightarrow{\sigma})$, where $\overrightarrow{\sigma}$ contains the ℓ input signatures and σ_{bind} is as described above in **maketx** that are added during $\mathsf{sign\text{-}all\text{-}inputs}$ *and* the signature σ_{bind} added in the last step of **maketx**.

Non-malleability says, $\mathcal{A}$ should not be able to create a new valid transaction with inputs belonging to $\mathscr{O}$, even after seeing transactions of its choice with inputs of $\mathscr{O}$. New will mean that the $\mathsf{raw}_{\mathsf{tx}}$ part will be new. (If we had changed the signature scheme to sign in order and have each signature sign the previous ones we could have required that tx including the signature part must be different from all previous transactions).

The way we formalize "transaction with inputs of $\mathscr{O}$" is that the transaction created by $\mathcal{A}$ contains overlapping nullifiers with the transactions signed previously by $\mathscr{O}$; precisely transactions that are outputs of $\mathsf{sign\text{-}all\text{-}inputs}$ queries.

Remark 4.2. *A somewhat odd thing about the construction with the delegated spender, is that valid transactions signed by $\mathscr{O}$, do not exactly correspond to transactions whose inputs $\mathscr{O}$ knows the spending key of. We can only say $\mathscr{O}$ and $\mathcal{A}$* together *know the spending key. For example, given $(\mathsf{ak}, \mathsf{nsk})$, $\mathcal{A}$ can choose random $s \in \mathbb{F}_r$, set $\mathsf{ak}' := \mathsf{ak} + s \cdot \mathsf{g}_{\mathsf{sig}}$. Now when $\mathcal{A}$ wants to sign an input in address ak', i.e. with some randomized key $\mathsf{rk} = \mathsf{ak}' + \alpha \mathsf{g}_{\mathsf{sig}} = \mathsf{ak} + (s + \alpha) \cdot \mathsf{g}_{\mathsf{sig}}$, it can give $\mathscr{O}$ the randomization $\alpha' = s + \alpha$.*

A way to avoid these oddities is to have $\mathscr{O}$ only sign transactions where he recognizes the nullifiers as belonging to a note of his. For our purposes here, we get a stronger result without this restriction by showing non-malleability holds when $\mathscr{O}$ signs any *transaction.*

Some more terminology Given a validly formatted transaction $\mathsf{tx} = ((\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \mathrm{v}^{\mathrm{bal}}), \sigma_{\mathsf{bind}}, \overrightarrow{\sigma})$, we define

- $\mathsf{nf}(\mathsf{tx})$ to be the set of nullifiers appearing in one of its inputs; so $\mathsf{nf}(\mathsf{tx}) := \{\mathsf{nf}(\mathsf{inp})\}_{\mathsf{inp} \in \overrightarrow{\mathsf{inp}}}$.
- $\mathsf{rk}(\mathsf{tx})$ the set of randomized public keys appearing in inputs of tx, so $\mathsf{rk}(\mathsf{tx}) := \{\mathsf{rk}(\mathsf{inp})\}_{\mathsf{inp} \in \overrightarrow{\mathsf{inp}}}$.
- $\mathsf{raw}(\mathsf{tx}) := (\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \mathrm{v}^{\mathrm{bal}})$. For a set T of validly formed transactions we define $\mathsf{raw}(\mathsf{T}) := \{\mathsf{raw}(\mathsf{tx})\}_{\mathsf{tx} \in T}$

Claim 4.3 (Non-malleability w.r.t delegated spenders)**.** *Fix any efficient $\mathcal{A}$ interacting with $\mathscr{O}$ as described above. Let $T = \{\mathsf{tx}'\}$ be the set of transactions that are replies of $\mathscr{O}$ to $\mathcal{A}$'s* $\mathsf{sign\text{-}all\text{-}inputs}$ *queries. The probability that $\mathcal{A}$ manages to output a ledger L and transaction tx such that*

1. $\mathsf{verify\text{-}tx}(\mathrm{L}, \mathsf{tx}) = \mathsf{acc}$,
2. $\mathsf{raw}(\mathsf{tx})$ *is not a prefix of an element of T.*
3. $\mathsf{nf}(\mathsf{tx}) \cap \mathsf{nf}(\mathsf{tx}') \neq \emptyset$ *for some* $\mathsf{tx}' \in T$.

is $\mathrm{negl}(\lambda)$.

Proof. Let $\mathcal{A}$ be an algorithm that after interacting with $\mathscr{O}$ as described above outputs L,tx. Let ϵ be the probability that L, tx satisfy the above, and assume for contradiction $\epsilon = 1/\mathrm{poly}(\lambda)$.

We construct $\mathcal{A}'$ that receives a randomized forgery challenge for $\mathsf{Schnorr}$ as described in Definition 2.1, and with probability $\epsilon - \mathrm{negl}(\lambda)$ either

- outputs a collision of **sighash**
- outputs a collision of **NF**,
- outputs a collision of **IVK**,
- Constructs a signature forgery for $\mathsf{Schnorr}$ w.r.t randomization.

Then, from CR of **sighash**, **NF**,**NC**,**IVK** and Theorem 2.3 the claim follows. $\mathcal{A}'$ works as follows:

1. $\mathcal{A}'$ will receive a challenge ak^* for the signature scheme $\mathsf{Schnorr}$ sampled by the procedure **sample** described in Section 2.
2. $\mathcal{A}'$ chooses random $\mathsf{nsk} \in \mathbb{G}$ and sends to $\mathcal{A}$ the proof authorizing key $\mathsf{pak} = (\mathsf{nsk}, \mathsf{ak})$
3. When $\mathcal{A}$ makes a $\mathsf{sign\text{-}all\text{-}inputs}$ query $(\mathsf{raw_{tx}}, \overrightarrow{\alpha})$ $\mathcal{A}'$ first checks that the proofs in $\mathsf{raw_{tx}}$ are valid (as $\mathscr{O}$ does in the description of $\mathsf{sign\text{-}all\text{-}inputs}$ queries) and then answers with $\overrightarrow{\sigma}$ where $\sigma_i := \mathcal{S}_{\mathsf{sign}}(\mathsf{ak} + \alpha_i \cdot \mathsf{g_{sig}}, \mathbf{m})$. If during invocations to $\mathcal{S}_{\mathsf{sign}}$, $\mathcal{S}_{\mathcal{R}}$ is queried on a point on which $\mathcal{A}$ queried $\mathcal{R}$, $\mathcal{A}'$ aborts. (Note that the point queried by $\mathcal{S}_{\mathcal{R}}$ is $(R, \mathsf{rk}, \mathbf{m})$ for a uniform R chosen only during the execution of $\mathcal{S}_{\mathsf{sign}}$, so the probability such a point was already queried is $\mathrm{negl}(\lambda)$.)
4. When $\mathcal{A}'$ makes a query to $\mathcal{R}$, $\mathcal{A}$ answers according to $\mathcal{R}$ unless the query has already been answered according to $\mathcal{S}_{\mathcal{R}}$ during invocations of $\mathcal{S}_{\mathsf{sign}}$ in $\mathsf{sign\text{-}all\text{-}inputs}$ queries; in which case $\mathcal{A}'$ answers according to $\mathcal{S}_{\mathcal{R}}$. (This doesn't change the distribution of $\mathcal{R}$ from the perspective of $\mathcal{A}$.)
5. When $\mathcal{A}$ outputs L, tx: $\mathcal{A}'$ checks that it indeed satisfies the challenge - that is $\mathsf{verify\text{-}tx}(\mathrm{L}, \mathsf{tx})$; tx contains an input inp with $\mathsf{nf} = \mathsf{nf}(\mathsf{inp})$ being equal to $\mathsf{nf}(\mathsf{inp}')$ for some $\mathsf{inp}' \in \mathsf{tx}'$ for some $\mathsf{tx}' \in T$; appearing in one of the $\mathsf{sign\text{-}all\text{-}inputs}$ queries of $\mathcal{A}$; and $\mathsf{raw_{tx}} \notin \mathsf{raw(T)}$. If not, $\mathcal{A}'$ aborts.
6. $\mathcal{A}'$ checks if $\mathbf{sighash}(\mathsf{raw_{tx}}) = \mathbf{sighash}(\mathsf{raw_{tx}}'')$ for some $\mathsf{tx}'' \in T$ with $\mathsf{raw_{tx}} \neq \mathsf{raw_{tx}}''$. If so it outputs $(\mathsf{raw_{tx}}, \mathsf{raw_{tx}}'')$ as a collision of **sighash**.

Explanation of where we are so far: Denote by rk and σ the public key and signature in inp. Let $\mathbf{m} := \mathbf{sighash}(\mathsf{raw_{tx}})$. σ is a valid signature for message $\mathbf{m}$ and public key rk, and $\mathbf{m}$ was never signed in reply to the $\mathsf{sign\text{-}all\text{-}inputs}$ queries by $\mathscr{O}$. To obtain a forgery w.r.t randomizatoin for the challenge ak^*, what is left is to find the α^* such that $\mathsf{rk} = \mathsf{ak}^* + \alpha^* \cdot \mathsf{g_{sig}}$. The purpose of the next steps is to obtain such α^* or a collision of one of our CRH functions.

7. Otherwise, denote by B the algorithm consisting of execution of all parties up to this point outputting tx and tx'. Note that B's randomness consists[2] of that of $\mathcal{A}, \mathcal{A}', \mathcal{R}$ used up to this point and the randomness of **sample**. Let ξ be the extractor guaranteed to exist for B for the input inp in tx. Recall that ξ requires B's internal randomness to produce a SNARK witness. $\mathcal{A}'$ can give ξ the randomness of $\mathcal{A}'$,$\mathcal{A}$,$\mathcal{R}$ used up to this point, but instead

[2] We have not defined collision-resistant functions too formally. To be more accurate we assume all CRH functions are "public" in the sense that their seed is just a random string, and this randomness is also one of the inputs to B.

of using the actual randomness of **sample** as input to ξ, $\mathcal{A}'$ uses the **invert** method to obtain this randomness correctly from ak with $1/\mathrm{poly}(\lambda)$ probability. Given this input, with probability $1/\mathrm{poly}(\lambda) - \mathrm{negl}(\lambda) = 1/\mathrm{poly}(\lambda)$, ξ outputs for the input inp in tx a witness $\mathsf{w} = (\mathsf{note}, \mathsf{pak} = (\mathsf{ak}, \mathsf{nsk}), \alpha, \mathsf{path}, \mathsf{pos})$. Similarly there is an extractor ξ' for the input inp' in tx' giving us a witness $\mathsf{w}' = (\mathsf{note}', \mathsf{pak}' = (\mathsf{ak}', \mathsf{nsk}'), \alpha', \mathsf{path}', \mathsf{pos}')$. If ξ or ξ' fails $\mathcal{A}'$ aborts (note that the probability of both succeeding is $1/\mathrm{poly}(\lambda)$).

8. Let $\mathsf{nk} := \mathsf{nsk} \cdot \mathsf{g}_{\mathbf{n}}, \mathsf{nk}' := \mathsf{nsk}' \cdot \mathsf{g}_{\mathbf{n}}$. We have

$$\mathbf{NF}(\mathsf{nk}, \mathsf{note}, \mathsf{pos}) = \mathbf{NF}(\mathsf{nk}', \mathsf{note}', \mathsf{pos}') = \mathsf{nf}.$$

   If $\mathsf{nk} \neq \mathsf{nk}', \mathsf{note} \neq \mathsf{note}'$ or $\mathsf{pos} \neq \mathsf{pos}'$, $\mathcal{A}'$ outputs $(\mathsf{nk}, \mathsf{note}, \mathsf{pos}), (\mathsf{nk}', \mathsf{note}', \mathsf{pos}')$ as a collision of $\mathbf{NF}$.

9. Otherwise we have $\mathsf{note} = \mathsf{note}' = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$. Defining $\mathsf{ivk} := \mathbf{IVK}(\mathsf{ak}, \mathsf{nk}), \mathsf{ivk}' := \mathbf{IVK}(\mathsf{ak}', \mathsf{nk})$, we have $\mathsf{pk} = \mathsf{ivk} \cdot \mathsf{g} = \mathsf{ivk}' \cdot \mathsf{g}$. Thus, $\mathsf{ivk} = \mathsf{ivk}'$. (Important here that ivk representation is unique and it is cause dfn of $\mathbf{IVK}$ has *mod* $2^{\ell_{\mathsf{ivk}}=251}$.) If $\mathsf{ak} \neq \mathsf{ak}'$, $\mathcal{A}'$ outputs $(\mathsf{ak}, \mathsf{nk}), (\mathsf{ak}, \mathsf{nk}')$ as a collision of $\mathbf{IVK}$.

10. Otherwise, we have $\mathsf{ak} = \mathsf{ak}'$. Now, $\mathcal{A}'$ knows α^* such that $\mathsf{rk}' = \mathsf{ak}^* + \alpha^* \cdot \mathsf{g}_{\mathbf{sig}}$, where ak^* was the forgery challenge from $\mathscr{O}$ (as $\mathcal{A}$ used $(\alpha^*, \mathbf{sighash}(\mathsf{raw}_{\mathsf{tx}}'))$ in the $\mathsf{sign\text{-}all\text{-}inputs}$ query for tx' for input inp'). And also $\mathsf{rk}' = \mathsf{ak}' + \alpha' \cdot \mathsf{g}_{\mathbf{sig}}$. So $\mathsf{ak} = \mathsf{ak}' = \mathsf{ak}^* + (\alpha^* - \alpha') \cdot \mathsf{g}_{\mathbf{sig}}$. And $\mathsf{rk} = \mathsf{ak}^* + (\alpha^* - \alpha' + \alpha) \cdot \mathsf{g}_{\mathbf{sig}}$. Thus, in this case $\mathcal{A}'$ outputs $(\alpha^* - \alpha' + \alpha, \mathbf{sighash}(\mathsf{raw}_{\mathsf{tx}}), \sigma)$ as a signature forgery w.r.t randomization of ak^*.

□

5 Indistinguishability w.r.t outside adversaries

For a sequence of random variables $X_1, \ldots, X_n$ it will be convenient in this section to denote $X_{<i} := (X_1, \ldots, X_{i-1})$. Let us say that random variables X, Y are γ-independent if for any events A, B

$$|\Pr(X \in A \wedge Y \in B) - \Pr(X \in A) \cdot \Pr(Y \in B)| \leq \gamma.$$

We recall that the *statistical distance* between X and Y is the maximum over all events T of

$$|\Pr(X \in T) - \Pr(Y \in T)|.$$

We say X, Y are *γ-close* if they have statistical distance at most γ.

A calculation proves

Claim 5.1. *Suppose $X = (X_1, X_2), Y = (Y_1, Y_2)$ are such that*

- *X_1 and Y_1 are on the same range, are γ_1-independent and γ_1-close.*
- *e.w.p γ_2 over the value (x_1, y_1) of (X_1, Y_1), $(X|X_1 = x_1), (Y|Y_1 = y_1)$ are γ_3-independent.*
- *e.w.p γ_2 over the value x_1 of X_1, we have that $(X|X_1 = x_1), (Y|Y_1 = x_1)$ are γ_3-close.*

Then X, Y are $\gamma_1 + \gamma_2 + \gamma_3$-independent and $\gamma_1 + \gamma_2 + \gamma_3$-close.

Induction then shows that

Claim 5.2. *Suppose $t = \text{poly}(\lambda)$. Suppose random variables $X = (X_1, \ldots, X_t), Y = (Y_1, \ldots, Y_t)$ are such that for any $i \in [n]$,*

- *e.w.p $\text{negl}(\lambda)$ over the value (x, y) of $(X_{<i}, Y_{<i})$,*
  $(X_i | X_{<i} = x)$ and $(Y_i | Y_{<i} = y)$ are $\text{negl}(\lambda)$-independent; and
- *e.w.p $\text{negl}(\lambda)$ over the value x of $X_{<i}$, $(X_i | X_{<i} = x)$ and $(Y_i | Y_{<i} = x)$ are $\text{negl}(\lambda)$-close.*

Then X, Y are $\text{negl}(\lambda)$-independent and $\text{negl}(\lambda)$-close.

Below we use $\mathcal{R}_{sig}$ to denote the random oracle used by the signature algorithm.

Theorem 5.3. *Assume that*

1. $\mathbf{NF}(\mathsf{nk}, \mathbf{NC}(\mathsf{note}), \mathsf{pos}) = \mathcal{R}(\mathsf{nk}, \mathbf{MPH}(\mathsf{note}, \mathsf{pos}))$ *where $\mathcal{R}$ is a random oracle and* $\mathbf{MPH}$ *is a collision-resistant function*[3]
2. $\mathbf{KDF}$ *and $\mathcal{R}_{sig}$ are also random oracles.*
3. $\mathbf{ENC}_K(m)$ *produces a uniform output when K is uniform and m is fixed.*
4. *The SNARK we are using is witness indistinguishable - i.e. the proof distribution depends* only *on the public input and not on the witness.*

Then, the probability of an efficient $\mathcal{A}$ finding $\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \overrightarrow{\mathsf{input}'}, \overrightarrow{\mathsf{output}'}$ such that

- $|\overrightarrow{\mathsf{input}}| = |\overrightarrow{\mathsf{input}'}| = \ell$, $|\overrightarrow{\mathsf{output}}| = |\overrightarrow{\mathsf{output}'}| = s$.
- *The positioned notes in $\overrightarrow{\mathsf{input}}$ and $\overrightarrow{\mathsf{input}'}$ are all distinct.*
- $(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}})$ *and* $(\overrightarrow{\mathsf{input}'}, \overrightarrow{\mathsf{output}'})$ *are both consistent with* $\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}$.
- *The distributions of the random variables $D :=$ **makerandomizedtx**$(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}})$ and*
  *$D' :=$ **makerandomizedtx**$(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}'}, \overrightarrow{\mathsf{output}'})$, over the randomness of the oracles $\mathcal{R}$,**KDF** and $\mathcal{R}_{sig}$, and the inner randomness of the signer, SNARK prover and the* **makerandomizedtx** *method, are not $\text{negl}(\lambda)$-close and $\text{negl}(\lambda)$-independent*

is $\text{negl}(\lambda)$.

Proof. Let us denote by $(\overrightarrow{\mathsf{inp}}, \overrightarrow{\mathsf{out}}, \sigma_{\mathsf{bind}}, \overrightarrow{\sigma})$ the output of **makerandomizedtx**$(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}})$ and by $(\overrightarrow{\mathsf{inp}'}, \overrightarrow{\mathsf{out}'}, \sigma'_{\mathsf{bind}}, \overrightarrow{\sigma}')$ the output of **makerandomizedtx**$(\mathsf{rt}, \mathrm{v}^{\mathrm{bal}}, \overrightarrow{\mathsf{input}'}, \overrightarrow{\mathsf{output}'})$ when using independent inner randomness, but joint randomness for the oracles $\mathcal{R}, \mathcal{R}_{sig}, \mathbf{KDF}$.

We will consider D and D' as sequences of random variables $D = (X_1, \ldots, X_m), D' = (Y_1, \ldots, Y_m)$, and show that for every $i \in [m]$ they satisfy the conditions of Claim 5.2.

We begin with the inputs. Letting, for $i \in [\ell]$, $X_i = \mathsf{inp}_i, Y_i = \mathsf{inp}'_i$, the following claim shows those conditions hold for the first $i \in [\ell]$.

[3]The requirement here may seem a bit odd; it models the fact that $\mathbf{NC}(\mathsf{note})$ is a pedersen hash which is combined in $\mathbf{NF}$ with a pos-multiple of an independent group generator, followed by an application of BLAKE-2 on the result prefixed with nk. In particular, BLAKE-2 takes the place of $\mathcal{R}$ in the implementation.

Claim 5.4. *E.w.p* $\text{negl}(\lambda)$ *over the randomness of* $\mathcal{A}$*, for each* $i \in [\ell]$ $\mathsf{inp}_i, \mathsf{inp}'_i$ *are identically distributed and independent given any fixing of* $\mathsf{inp}_{<i}, \mathsf{inp}'_{<i}$.

Proof. We show first that e.w.p. $\text{negl}(\lambda)$ over the randomness of $\mathcal{A}$, $\mathsf{inp}_i, \mathsf{inp}'_i$ are independent conditioned on any fixing of $\mathsf{inp}_{<i}, \mathsf{inp}'_{<i}$. $\mathsf{inp}_1, \ldots, \mathsf{inp}_\ell, \mathsf{inp}'_1, \ldots, \mathsf{inp}'_\ell$ are results of invocations of **makeinp** with independent randomness rcv, α and independent randomness of the SNARK prover. Inspection shows the only opportunity for dependence amongst any two of them, even after conditioning on the value of the others, is having the random oracle $\mathcal{R}$ queried at the same point during the invocations. $\mathcal{R}$ is queried for the computation of **NF**; so this only happens if

$$(\mathsf{nk}_i, \mathbf{MPH}(\mathsf{note}_i, \mathsf{pos}_i)) = (\mathsf{nk}'_i, \mathbf{MPH}(\mathsf{note}'_i, \mathsf{pos}'_i)).$$

This implies $\mathbf{MPH}(\mathsf{note}_i, \mathsf{pos}_i) = \mathbf{MPH}(\mathsf{note}'_i, \mathsf{pos}'_i)$, but $\mathcal{A}$ will only find such a collision w.p $\text{negl}(\lambda)$. When this doesn't happen inp_i and inp'_i are independent also given any fixing of the previous inputs.

Now to show they are identically distributed given a fixing of $\mathsf{inp}_{<i}, \mathsf{inp}'_{<i}$.

Suppose $\mathsf{inp}_i = (\mathsf{nf}, \mathsf{rt}, \mathsf{rk}, \mathsf{cv}, \pi)$, and $\mathsf{inp}'_i = (\mathsf{nf}', \mathsf{rt}', \mathsf{rk}', \mathsf{cv}', \pi')$. We show each element is identically distributed conditioned on any fixing of the previous ones.

- $\mathsf{nf} = \mathcal{R}(q)$ and $\mathsf{nf}' = \mathcal{R}(q')$ where $q = (\mathsf{nk}, \mathbf{MPH}(\mathsf{note}, \mathsf{pos})), q' = (\mathsf{nk}', \mathbf{MPH}(\mathsf{note}', \mathsf{pos}'))$. These are both uniform *unless* one of the queries q, q' was already made to $\mathcal{R}$ in a previous invocation; which would mean $\{(\mathsf{note}^*, \mathsf{pos}^*)\}_{(\mathsf{note}^*, \mathsf{pos}^*) \in \mathsf{inp}_{<i+1} \cup \mathsf{inp}_{<i+1}}$ contains a collision of **MPH** which $\mathcal{A}$ can find only w.p $\text{negl}(\lambda)$.

- $\mathsf{rt} = \mathsf{rt}'$.

- $\mathsf{rk} = \mathsf{ak} + \alpha \cdot \mathsf{g}$, $\mathsf{rk}' = \mathsf{ak}' + \alpha' \cdot \mathsf{g}$. Are both uniform in $\mathbb{G}$ because of the uniform choice of α, α' in **makerandomizedtx**.

- $\mathsf{cv} = \mathsf{v} \cdot \mathsf{g}_{\mathbf{v}} + \mathsf{rcv} \cdot \mathsf{g}_{\mathbf{r}}$,$\mathsf{cv}' = \mathsf{v}' \cdot \mathsf{g}'_{\mathbf{v}} + \mathsf{rcv}' \cdot \mathsf{g}'_{\mathbf{r}}$. Are both uniform in $\mathbb{G}$ because of the uniform choices of $\mathsf{rcv}, \mathsf{rcv}' \in \mathbb{F}_r$ in the executions of **makerandomizedtx**.

- π, π' - When $(\mathsf{nf}, \mathsf{rt}, \mathsf{rk}, \mathsf{cv}) = (\mathsf{nf}', \mathsf{rt}', \mathsf{rk}', \mathsf{cv}')$, it follows from the witness indistinguishability of the SNARK that π and π' are identically distributed. They are independent for any fixing of the previous values, as given this fixing the value of π, π' depends only on the inner randomness of the SNARK prover.

□

We proceed with the elements of the the ouputs. It will be convenient now to view each element in $\mathsf{out}_j, \mathsf{out}'_j$ as separate random variables X_i, Y_i, and show that

1. E.w.p $\text{negl}(\lambda)$ over the fixing of $X_{<i}$, they are identically distributed given this fixing of both $X_{<i}$ and $Y_{<i}$.

2. E.w.p $\text{negl}(\lambda)$ over the fixing of $X_{<i}, Y_{<i}$ they are independent given the fixing.

We show this for the different types of elements in $\mathsf{out}_j, \mathsf{out}'_j$:

- $\mathsf{cv} = \mathsf{v}\cdot\mathsf{g}_{\mathbf{v}} + \mathsf{rcv}\cdot\mathsf{g}_{\mathbf{r}}, \mathsf{cv}' = \mathsf{v}'\cdot\mathsf{g}_{\mathbf{v}} + \mathsf{rcv}'\cdot\mathsf{g}_{\mathbf{r}}$: are independent and uniform in $\mathbb{G}$ because of the independent uniform choices of $\mathsf{rcv}, \mathsf{rcv}' \in \mathbb{F}_r$ in **makerandomizedtx**.

- $\mathsf{cm} = \mathbf{NC}(\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$,$\mathsf{cm}' = \mathbf{NC}(\mathsf{g}', \mathsf{pk}', \mathsf{v}', \mathsf{rcm}')$: are uniform and independent in $\mathbb{G}$ because of the independent uniform choices of $\mathsf{rcm}, \mathsf{rcm}' \in \mathbb{F}_r$ in **makerandomizedout**.

- $\mathsf{epk} = \mathsf{esk}\cdot\mathsf{g}, \mathsf{epk}' = \mathsf{esk}'\cdot\mathsf{g}$ are uniform and independent in $\mathbb{G}$ because of the independent and uniform choices of $\mathsf{esk}, \mathsf{esk}' \in \mathbb{F}_r$ in **makerandomizedout**.

- π, π' - Assuming the pubic inputs $(\mathsf{epk}, \mathsf{cm}, \mathsf{cv}) = (\mathsf{epk}', \mathsf{cm}', \mathsf{cv}')$, it follows from the witness indistinguishability of the SNARK that π and π' are identically distributed. They are independent for any fixing of the previous values, as given this fixing the value of π, π' depends only on the inner randomness of the SNARK prover.

- $\mathsf{enc} = \mathbf{ENC}_{\mathbf{KDF}(k)}((\mathsf{g}, \mathsf{pk}, \mathsf{v}))$,$\mathsf{enc}' = \mathbf{ENC}_{\mathbf{KDF}(k')}((\mathsf{g}', \mathsf{pk}', \mathsf{v}'))$ where $k := (\mathsf{esk}\cdot\mathsf{pk}, \mathsf{epk})$ and $k' := (\mathsf{esk}'\cdot\mathsf{pk}', \mathsf{epk}')$: Assuming $k \neq k'$, and moreover $\{k, k'\}$ are different from all the "key seeds" $\left\{k_j, k'_j\right\}$ used in previous outputs; we have that the encryption keys $\mathbf{KDF}(k), \mathbf{KDF}(k')$ are uniform and independent of all previous variables. And thus by the theorem's assumption that **KDF** is a random oracle $\mathsf{enc}, \mathsf{enc}'$ are uniform and independent in this case. Thus there are at most ℓ values of the preceding esk and at most ℓ values of the preceding esk' that can prevent enc and enc' from being uniform and independent; which is a $\mathrm{negl}(\lambda)$-fraction of the possible values of the preceding values.

It is now left to deal with the signature elements. $\sigma_{\mathsf{bind}}, \sigma'_{\mathsf{bind}}, \{\sigma_i, \sigma'_i\}$.

The distribution of these elements is determined by the public key $\mathsf{pk} = \mathsf{rk}_i$ (or $\mathsf{pk} = S$ the sum of value commitments in the case of σ_{bind}), the message $\mathbf{m} = \mathbf{sighash}(\mathsf{raw}_{\mathsf{tx}})$ they are signing, the internal randomness of the signing algorithm and the reply of the random oracle $\mathcal{R}_{sig}$ on the query point $(R, \mathsf{pk}, \mathbf{m})$. Thus, given a fixing of previous variables, the only case where a dependence between σ_i or σ'_i could be created is if there as a collision between the signatures in the choice of R which happens w.p. $\mathrm{negl}(\lambda)$. □

5.1 Balance

The following claim states an adversary should not be able to create "money out of thin air"; or more specifically, extract more money from the shielded pool than was put in it. In Sapling, the value $\mathrm{v}^{\mathrm{bal}} = \mathrm{v}^{\mathrm{bal}}(\mathsf{tx})$ in a transaction tx corresponds to the alleged difference of spend and output values (see Section 4.12 in the spec) and tx is thought of as having ; thus over-extracting from the pool corresponds to a constructing a ledger where the sum of all $\mathrm{v}^{\mathrm{bal}}$ values is strictly positive.

Claim 5.5. *The probability that an efficient $\mathcal{A}$ generates ledger $\mathrm{L} = (\mathsf{tx}_1, \ldots, \mathsf{tx}_n)$ such that*

$$\sum_{\mathsf{tx}\in\mathrm{L}} \mathrm{v}^{\mathrm{bal}}(\mathsf{tx}) > 0$$

is $\mathrm{negl}(\lambda)$.

Proof. Given $\mathcal{A}$ that produces a ledger as in the claim statement w.p. γ, we construct an efficient $\mathcal{A}$' that w.p $\gamma/2 - \mathrm{negl}(\lambda)$ produces a collision of **IVK**,**NC**,$\mathsf{treehash}$ or **VC**. It follows that $\gamma = \mathrm{negl}(\lambda)$.

1. $\mathcal{A}$' begins by running $\mathcal{A}$ and aborting if $\mathcal{A}$ hasn't output a ledger as in the claim.

2. Otherwise, given such a ledger L, $\mathcal{A}$' can apply an extractor for each SNARK proof in all inputs and ouputs in all transactions. For each transaction input $\mathsf{inp} \in \mathsf{tx} \in \mathrm{L}$, $\mathsf{inp} = (\mathsf{cv}, \mathsf{nf}, \mathsf{rt}, \mathsf{rk}, \pi)$, the extractor except w.p. $\mathrm{negl}(\lambda)$ outputs an input witness $\mathsf{inpwit} = (\mathsf{input} = (\mathsf{note}, \mathsf{path}, \mathsf{pos}), \mathsf{pak}, \mathsf{rcv}, \alpha))$. We denote by $\mathsf{posnote}$ the *positioned note* corresponding to inp, $\mathsf{posnote} := (\mathsf{note}, \mathsf{pos})$. Similarly for every transaction output in some tx in L, $\mathsf{out} = (\mathsf{cv}, \mathsf{cm}, \mathsf{epk}, \pi, \mathsf{enc})$, the extractor outputs $\mathsf{outwit} = (\mathsf{note}, \mathsf{esk}, \mathsf{rcv})$. The value pos for the output note can be deduced from when it was added to L, i.e., the location of cm in the commitment tree. So again we can define for each out, the corresponding positioned note $\mathsf{posnote} = (\mathsf{note}, \mathsf{pos})$. For $i \in [n]$ let us denote respectively by $\mathcal{I}_i, \mathcal{O}_i$ the positioned input and output notes in tx_i *with non-zero value*[4].

   We also use the extractor from Theorem 2.4 to find s such that $S = s \cdot \mathsf{g}_{\mathbf{r}}$ where

$$S := \sum_{i=1}^{\ell} \mathsf{cv}_i - \sum_{i=\ell+1}^{\ell+s} \mathsf{cv}_i - \mathrm{v}^{\mathrm{bal}} \cdot \mathsf{g}_{\mathbf{v}}$$

   is the public key in the value binding signature σ_{bind}.

   If one of the extractor runs fails $\mathcal{A}$' aborts. Note that w.p. at least $\gamma/2 - \mathrm{negl}(\lambda)$ $\mathcal{A}$' doesn't abort.

3. $\mathcal{A}$' checks if for some $i \in [n]$ and $\mathsf{inp} \in \mathsf{tx}_i, \mathsf{posnote}(\mathsf{inp}) \notin \mathcal{O}_{<i}$.

   If so, let $\mathsf{tx} = \mathsf{tx}_i$. Let rt be the root of the tree used in the public input of inp; this is the tree T_j formed from $\{\mathsf{tx}_1, \ldots, \mathsf{tx}_j\}$ for some $j < i$. Let $\mathsf{posnote} = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm}, \mathsf{pos})$ and $\mathsf{cm} = \mathbf{NC}(\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$. inpwit contains a path path from cm to rt. If pos is an index of a leaf in T_j, there exists an extended note $\mathsf{posnote}'$ that was inserted to this position when constructing the ledger and from $\mathsf{posnote}$' we can derive a path path' from $\mathsf{cm}' = \mathbf{NC}(\mathsf{g}', \mathsf{pk}', \mathsf{v}', \mathsf{rcm}')$ in position pos to rt. If $\mathsf{path} \neq \mathsf{path}'$, then going down from rt to the first difference between path and path' (ask Sean/Daira : is T always a full tree with zeroes on other leaves? No you have filler values for the empty subtrees, need to check this are values that are hard to find route to - their impossible to find rout to - have no preimage) this difference gives a collision of $\mathsf{treehash}$ that $\mathcal{A}$' can output.

   Otherwise, we have $\mathsf{cm} = \mathsf{cm}'$. note must be different from note' because $\mathsf{posnote}' = (\mathsf{note}', \mathsf{pos}) \in \mathcal{O}_{<i}$ but $(\mathsf{note}, \mathsf{pos}) \notin \mathcal{O}_{<i}$.

   Thus $\mathsf{note}, \mathsf{note}'$ is a collision of $\mathbf{NC}$. In this case, $\mathcal{A}$' outputs this collision and terminates.

   Now suppose pos is not a position of a leaf in T_j. This means there is only a partial path path' in T_j from rt to a filler value with no preimage (see spec for details). So, similarly we follow path and path' to their first difference - a difference that must exist becaues of the filler value; and this gives us a collision of $\mathsf{treehash}$ that $\mathcal{A}$' outputs.

4. Now $\mathcal{A}$' checks if as a multiset $\mathcal{I} := \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_n$ contains a repetition. That is, there exists $\mathsf{posnote} = (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm}, \mathsf{pos})$ such that for two distinct transaction inputs $\mathsf{inp} = (\mathsf{cv}, \mathsf{nf}, \mathsf{rt}, \mathsf{rk}, \pi), \mathsf{inp}' =$

[4]Sapling enables the creation of dummy notes with zero value, for which the spend statement doesn't check Merkle path validity, cf. Section 4.7.2 in the spec).

$(\mathsf{cv}', \mathsf{nf}', \mathsf{rt}', \mathsf{rk}', \pi')$ in L; if the corresponding extracted witnesses are $\mathsf{inpwit} = (\mathsf{input} = (\mathsf{note}, \mathsf{path}, \mathsf{pos}), \mathsf{pak}, \mathsf{rcv}, \alpha)$, $\mathsf{inpwit}' = (\mathsf{input}' = (\mathsf{note}', \mathsf{path}', \mathsf{pos}'), \mathsf{pak}', \mathsf{rcv}', \alpha')$; then $(\mathsf{note}, \mathsf{pos}) = (\mathsf{note}', \mathsf{pos}') = \mathsf{posnote}$.

We show in this case that $\mathcal{A}$' can output a collision of **IVK**:

Let $\mathsf{cm} = \mathbf{NC}(\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$. Since $\mathsf{nf} \neq \mathsf{nf}'$, and $\mathsf{nf} = \mathbf{NF}(\mathsf{nk}, \mathsf{note}, \mathsf{pos})$, $\mathsf{nf}' = \mathbf{NF}(\mathsf{nk}', \mathsf{note}, \mathsf{pos})$; we have $\mathsf{nk} \neq \mathsf{nk}'$.

Also $\mathsf{ivk} = \mathbf{IVK}(\mathsf{ak}, \mathsf{nk}), \mathsf{ivk}' = \mathbf{IVK}(\mathsf{ak}', \mathsf{nk}')$, and $\mathsf{pk} = \mathsf{ivk} \cdot \mathsf{g} = \mathsf{ivk}' \cdot \mathsf{g}$. So $\mathsf{ivk} = \mathsf{ivk}'$ And thus, $\mathcal{A}$' can output $(\mathsf{ak}, \mathsf{nk}), (\mathsf{ak}', \mathsf{nk}')$ as a collision of **IVK**.

5. Let us denote by $\mathrm{bal}(\mathsf{tx})$ the (integer) sum of values in inputs of tx minus the sum of values in output of tx (notes meaning those output by the extractors); and by $\mathsf{rcv}(\mathsf{tx})$ the sum of values rcv in input witnesses of tx minus the sum of values rcv in output witnesses of tx. When reaching this point with no output we know that:

   For each $i \in [n], \mathcal{I}_i \subset \mathcal{O}_1 \cup \ldots \cup \mathcal{O}_{i-1} \setminus (\mathcal{I}_1 \cup \ldots \cup \mathcal{I}_{i-1})$.

   This implies
   $$\sum_{\mathsf{tx} \in \mathrm{L}} \mathrm{bal}(\mathsf{tx}) \leq 0.$$

   We claim that we must have for some $\mathsf{tx} \in \mathrm{L}$, $\mathrm{bal}(\mathsf{tx}) \neq \mathrm{v}^{\mathrm{bal}}(\mathsf{tx})$: Otherwise, we would have
   $$\sum_{\mathsf{tx} \in \mathrm{L}} \mathrm{v}^{\mathrm{bal}}(\mathsf{tx}) = \sum_{\mathsf{tx} \in \mathrm{L}} \mathrm{bal}(\mathsf{tx}) \leq 0,$$

   contradicting the fact that $\mathcal{A}$ has managed to output L with a positive sum of $\mathrm{v}^{\mathrm{bal}}$ values.

   Thus, let $\mathsf{tx} = \mathsf{tx}_i$ be such that $\mathrm{bal}(\mathsf{tx}) \neq \mathrm{v}^{\mathrm{bal}}(\mathsf{tx})$. We show in the next step how $\mathcal{A}'$ uses this to output a collision of **VC**.

6. At this point, we know that $\mathrm{bal}(\mathsf{tx}) \neq \mathrm{v}^{\mathrm{bal}}(\mathsf{tx})$. As both these values are in the open interval [5] $(-r/2, r/2)$, we have also $\mathrm{bal}(\mathsf{tx}) \neq \mathrm{v}^{\mathrm{bal}}(\mathsf{tx}) \pmod r$. We show how to find a collision of **VC** with probability $\gamma/\mathrm{poly}(\lambda)$. Since tx verifies, we know that $\mathsf{verifySig}^{\mathcal{R}}_{\mathbf{g_r}}(S, \mathbf{sighash}(\mathsf{raw}_{\mathsf{tx}}), \sigma_{\mathsf{bind}})$ for
   $$S = \sum_{i=1}^{\ell} \mathsf{cv}_i - \sum_{i=\ell+1}^{\ell+s} \mathsf{cv}_i - \mathrm{v}^{\mathrm{bal}} \cdot \mathsf{g}_{\mathbf{v}} = \left(\sum_{i=1}^{\ell} \mathsf{v}_i - \sum_{i=\ell+1}^{s} \mathsf{v}_i\right) \cdot \mathsf{g}_{\mathbf{v}} + \left(\sum_{i=1}^{\ell} \mathsf{rcv}_i - \sum_{i=\ell+1}^{s} \mathsf{rcv}_i\right) \cdot \mathsf{g}_{\mathbf{r}} - \mathrm{v}^{\mathrm{bal}} \cdot \mathsf{g}_{\mathbf{v}}.$$

   Let $R := \sum_{i=1}^{\ell} \mathsf{rcv}_i - \sum_{i=\ell+1}^{s} \mathsf{rcv}_i$ and $v := \mathrm{bal}(\mathsf{tx}) - \mathrm{v}^{\mathrm{bal}}(\mathsf{tx}) (\mathrm{mod}\ r)$. We have $\mathbf{VC}(v, R) = S$.

   Recall that if $\mathcal{A}$' has reached this stage without aborting, it has obtained s such that $s \cdot \mathsf{g}_{\mathbf{r}} = S$. Thus, we also have $\mathbf{VC}(0, s) = S$. Hence, noticing that $v \neq 0$, $\mathcal{A}'$ can output $(0, s), (v, R)$ as a collision of **VC**.

□

[5] See the spec for details: $\mathrm{v}^{\mathrm{bal}}$ and v in each transaction input/output are at most 2^{64} in absolute value, so assuming less than, e.g., 2^{r-66} transaction inputs and outputs in any transaction, this is true.

5.2 Spendability

Valid transaction bases: A sequence $\mathrm{x} = (\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathrm{v}^{\mathrm{bal}})$ is a *valid transaction base* if $\mathrm{v}^{\mathrm{bal}} = \sum \mathsf{v}(\mathsf{input}_i) - \sum \mathsf{v}(\mathsf{output}_j)$.

We review note encryption and decryption from the spec in our notation.

Decrypting notes:
$\underline{\mathbf{dec}(\mathsf{ivk}, \mathsf{out} = (\mathsf{cv}, \mathsf{cm}, \mathsf{epk}, \pi, \mathsf{enc}))}$

1. Let $K := \mathbf{KDF}(\mathsf{epk} \cdot \mathsf{ivk})$
2. Let $\mathsf{np} = \mathbf{DEC}_K(\mathsf{enc})$. If $\mathbf{DEC}()$ fails output rej.
3. Suppose $\mathsf{np} = (\mathsf{d}, \mathsf{v}, \mathsf{rcm}, \mathsf{memo})$. If $\mathsf{rcm} \geq r$ output rej.
4. Let $\mathsf{g} := \mathsf{GH}(\mathsf{d})$.
5. Let $\mathsf{pk} := \mathsf{g} \cdot \mathsf{ivk}$. Let $\mathsf{note} := (\mathsf{g}, \mathsf{pk}, \mathsf{v}, \mathsf{rcm})$.
6. Check that $\mathsf{cm} = \mathbf{NC}(\mathsf{note})$. Output rej if not.
7. Output note.

We define

$$\mathbf{dec}(\mathsf{ivk}, \mathsf{tx}) := \cup_{\mathsf{out} \in \mathsf{tx}} \mathbf{dec}(\mathsf{ivk}, \mathsf{out}),$$

$$\mathbf{dec}(\mathsf{ivk}, \mathrm{L}) := \cup_{\mathsf{tx} \in \mathrm{L}} \mathbf{dec}(\mathsf{ivk}, \mathsf{tx})$$

And also

$$\mathsf{nf}(\mathsf{tx}) := \cup_{\mathsf{inp} \in \overrightarrow{\mathsf{inp}}(\mathsf{tx})} \mathsf{nf}(\mathsf{inp}), \mathsf{nf}(\mathrm{L}) := \cup_{\mathsf{tx} \in \mathrm{L}} \mathsf{nf}(\mathsf{tx})$$

In the spendability game $\mathcal{A}$ tries to create a ledger where a note successfully decrypted with ivk cannot be spent. Formally, the game proceeds as follows.

1. We choose uniform $\mathsf{sk} = (\mathsf{ask}, \mathsf{nsk})$; and give $\mathsf{pak} = (\mathsf{ask} \cdot \mathsf{g}_{\mathbf{sig}}, \mathsf{nsk})$ to $\mathcal{A}$.
2. $\mathcal{A}$ outputs a ledger L, a positioned note $(\mathsf{note}, \mathsf{pos})$, a set of output notes $\overrightarrow{\mathsf{output}}$, and a set of incoming viewing keys $\overrightarrow{\mathsf{ivk}}$.
3. We choose random $\overrightarrow{\mathsf{rcv}} \in \mathbb{F}_r^{\ell+s}$ and compute $\mathsf{tx} = \mathbf{maketx}(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathrm{v}^{\mathrm{bal}}, \mathsf{ask})$.
4. Let $\mathsf{ivk} := \mathbf{IVK}(\mathsf{ak}, \mathsf{nk})$. $\mathcal{A}$ wins iff
   (a) $\mathsf{note} \in \mathbf{dec}(\mathsf{ivk}, \mathrm{L})$.
   (b) $((\mathsf{note}), \overrightarrow{\mathsf{output}}, \mathrm{v}^{\mathrm{bal}})$ is a valid transaction base.
   (c) For each $i \in [s]$, output_i belongs to ivk_i.
   (d) $\mathsf{verify\text{-}tx}(\mathrm{L}, \mathsf{tx})$.
   (e) For some $i \in [s]$, $\mathbf{dec}(\mathsf{ivk}_i, \mathsf{out}_i)$ does not return output_i.

We wish to show that the success of any efficient $\mathcal{A}$ in this game is $\mathrm{negl}(\lambda)$.

Let $\mathsf{nk} = \mathsf{nsk} \cdot \mathsf{g}_{\mathbf{n}}$. Inspection of the protocol shows this exactly corresponds to the nullifier of note with nullifier key nk already appearing in the ledger. Thus, it suffices to prove the following.

Claim 5.6. *Fix any efficient $\mathcal{A}$. Suppose that $\mathcal{A}$ is given uniformly chosen* pak*, and let* $\mathsf{ivk} := \mathbf{IVK}(\mathsf{pak})$. *The probability that $\mathcal{A}$ generates a ledger* L *and positioned note (*$\mathsf{note,pos}$*) such that*

1. $(\mathsf{note}, \mathsf{pos}) \in \mathbf{dec}(\mathsf{ivk}, \mathrm{L})$
2. $\mathbf{NF}(\mathsf{nk}, \mathbf{NC}(\mathsf{note}), \mathsf{pos}) \in \mathsf{nf}(\mathrm{L})$

is $\mathrm{negl}(\lambda)$.

Proof. Let γ be the probability that $\mathcal{A}$ outputs $\mathrm{L}, \mathsf{note}$ satisfying the two properties in the claim. We construct an efficient $\mathcal{A}'$ that receives a forgery challenge ak of $\mathsf{Schnorr}$ and w.p. $\gamma - \mathrm{negl}(\lambda)$ does one of the following.

- Output a collision of either **NF**, **NC** or **IVK**.
- Output a forgery w.r.t to randomization of $\mathsf{Schnorr}$ for the challenge ak.

$\mathcal{A}'$ works as follows.

1. $\mathcal{A}'$ receives a challenge ak; chooses random $\mathsf{nsk} \in \mathbb{F}_r$ and sends $\mathsf{pak} = (\mathsf{ak}, \mathsf{nsk})$ to $\mathcal{A}$.
2. $\mathcal{A}'$ receives the output $(\mathrm{L}, \mathsf{note}, \mathsf{pos})$ of $\mathcal{A}$.
3. $\mathcal{A}'$ checks that $\mathrm{L}, (\mathsf{note}, \mathsf{pos})$ satisfy the two properties in the claim; if not it aborts.
4. Let $\mathsf{nf} := \mathbf{NF}(\mathsf{nk}, \mathbf{NC}(\mathsf{note}), \mathsf{pos})$. Fix the $\mathsf{out}, \mathsf{tx}$ with $\mathsf{out} \in \mathsf{tx} \in \mathrm{L}$ such that $\mathbf{dec}(\mathsf{ivk}, \mathsf{out}) = (\mathsf{note}, \mathsf{pos})$. out contains a valid SNARK proof for $\mathsf{SPEND}(\mathsf{rt}, \mathsf{cv}, \mathsf{nf}, \mathsf{rk})$ for some cv, rt. Apply the relevant extractor ξ relating to the snark proof to obtain e.w.p $\mathrm{negl}(\lambda)$ a witness $\mathsf{path}, \mathsf{pos}', \mathsf{g}', \mathsf{pk}', \mathsf{v}', \mathsf{rcm}', \mathsf{cm}', \mathsf{rcv}', \alpha, \mathsf{ak}', \mathsf{nsk}'$ for the statement.
5. Let $\mathsf{nk}' := \mathsf{nsk}' \cdot \mathsf{g}_{\mathbf{n}}$. If $(\mathsf{nk}, \mathsf{cm}, \mathsf{pos}) \neq (\mathsf{nk}', \mathsf{cm}', \mathsf{pos}')$, $\mathcal{A}'$ outputs $(\mathsf{nk}, \mathsf{cm}, \mathsf{pos}), (\mathsf{nk}', \mathsf{cm}', \mathsf{pos}')$ as a collision of **NF**.
6. Otherwise, let $\mathsf{note}' = (\mathsf{g}', \mathsf{pk}', \mathsf{v}', \mathsf{rcm}')$. We have $\mathsf{cm} = \mathbf{NC}(\mathsf{note}) = \mathbf{NC}(\mathsf{note}')$. If $(\mathsf{g}', \mathsf{pk}', \mathsf{v}') \neq (\mathsf{g}, \mathsf{pk}, \mathsf{v})$, $\mathcal{A}'$ outputs $(\mathsf{note}, \mathsf{note}')$ as a collision of **NC**.
7. Otherwise, we must have $\mathsf{ivk}' = \mathsf{ivk}$ (cause $\mathsf{g} \cdot \mathsf{ivk} = \mathsf{g} \cdot \mathsf{ivk}' = \mathsf{pk}$). Then $\mathsf{ivk} = \mathbf{IVK}(\mathsf{ak}', \mathsf{nk})$ (by this stage we know $\mathsf{nk} = \mathsf{nk}'$). If $\mathsf{ak} \neq \mathsf{ak}'$, $\mathcal{A}'$ outputs $(\mathsf{ak}, \mathsf{nk}), (\mathsf{ak}', \mathsf{nk})$ as a collision of **IVK**.
8. Otherwise $\mathsf{ak} = \mathsf{ak}'$, and $\mathsf{rk} = \mathsf{ak} + \alpha \cdot \mathsf{g}$. Let σ be the signature of $\mathsf{raw}_{\mathsf{tx}}$ with public key rk in inp. and $\mathcal{A}'$ outputs $(\alpha, \mathsf{raw}_{\mathsf{tx}}, \sigma)$ as a forgery of $\mathsf{Schnorr}$ with challenge ak.

□

Remark 5.7. *Note that in the spendability and non-malleability property $\mathcal{A}$ can choose what value* nf *to work with. It seems likely that in a weaker model where the values* nf *are generated randomly via honest users' notes, a second preimage resistance property of* **NF** *would suffice (Thanks to Sean Bowe and Zooko Wilcox for mentioning this).*

Acknowledgements

We thank Matthew D. Green for conversations on simulation extractability that in particular inspired the definition and use of invertible group samplers.

References

[1] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014. DBLP:conf/sp/2014.

[2] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. *IET Information Security*, 12(3):166–183, 2018.

[3] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox. Zcash protocol spec - https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.

[4] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.