# On the privacy of Zcash Joinsplits with a Witness Indistinguishable SNARK

Ariel Gabizon

Zcash

## 1 Introduction

The purpose of this note is to show that witness indistinguishability (WI), a weaker property than zero-knowledge, is a sufficient guarantee from a SNARK to achieve privacy in Zcash Joinsplits. The advantage of using WI, is that the WI property can be verified just by checking that 3 elements of the [1] proving key are non-zero; with no need to augment the SNARK parameters, as done e.g. in [2]; or download and verify the MPC transcript that created the parameters.

**In a nutshell, why is WI enough for Zcash?** The Zcash Sprout protocol is designed in a way that any shielded input and output notes can correspond to any SNARK public input, as long as they are consistent in the value moving from the transparent to shielded pool. Thus it is enough to hide the witness (i.e. input and output notes) that was used, and there is no need to hide the *set* of possible witnesses. Contrast this with a Sudoku puzzle with only one solution (i.e. only one satisfying witness for the given public input), in which case a WI SNARK may leak information regarding this unique solution, where a zk-SNARK wouldn't

## 2 Definitions

The proof requires the random-oracle model, and practical security is based on the (standard) assumption that the SHA-256 and BLAKE-2 functions, used to instantiate the various independent oracles used in the protocol description below, "behave like a random function".

We formally define the WI property;

**Definition 2.1.** *Suppose that* $\mathscr{S} = (\mathsf{Gen}, \mathsf{P}, \mathsf{ver})$ *is a SNARK for a relation* $\mathsf{R}$. $\mathscr{S}$ *is* Witness Indistinguishable *(WI) for an output* $\sigma$ *of the CRS-generator* $\mathsf{Gen}$*, if for any public input* $\mathsf{inp}$*, there exists a distribtion* $\pi$ *such that for any witness* $\mathsf{w}$ *such that* $(\mathsf{inp}, \mathsf{w}) \in \mathsf{R}$ *the distribtion of* $\mathsf{P}(\sigma, \mathsf{inp}, \mathsf{w})$ *is* $\pi$*.*

It is easy to verify that

**Lemma 2.2.** *Let* $\mathscr{S} = (\mathsf{Gen}, \mathsf{P}, \mathsf{ver})$ *be the SNARK of [1] for a QAP relation. Then* $\mathscr{S}$ *is WI for any output* $\sigma$ *of* $\mathsf{Gen}$ *such that, in the notation of [1],* $A_{m+1}, B_{m+2}, C_{m+3} \neq 0$ *in* $\sigma$*.*

*Proof.* (sketch) When the mentioned elements of $\sigma$ are non-zero, then the $\pi_A, \pi_B, \pi_C$ elements of $\mathsf{P}(\sigma, \mathsf{inp}, \mathsf{w})$ are uniform, and the five other proof elements are deterministic functions of the first three and the public input. □

# 3 The Zcash Sprout Joinsplit

We describe the joinsplits in notation that will be convenient for us. We assume familiarity with the Zcash spec [3], which may also be refered to for more details.

A *sprout note* $\mathsf{note}$ is a tupple $\mathsf{note} = (\mathsf{a_{pk}}, \mathsf{v}, \rho, \mathsf{rcm})$ where $\mathsf{a_{pk}}$ is the address, $\rho$ is the serial number, $\mathsf{v}$ is the value of the note and $\mathsf{rcm}$ is the commitment randomness.

We describe the Sprout $\mathsf{makeJSD}$ algorithm whose inputs consist of two input and two output notes, a root of a Merkle tree $\mathsf{rt}$ containing the commitments of the two input notes as leaves, and a witness $\mathsf{w}$ containing Merkle paths from the commitments of $(\mathsf{input}_1, \mathsf{input}_2)$ to $\mathsf{rt}$, together with the secret keys $\mathsf{a_{sk}}_i^{\mathsf{old}}$ of $\{\mathsf{input}\}$, and two bits indicating if the input notes are dummy notes of value 0, in which case the Merkle path need not be valid.

We will use the functions **NC**, **NF**, **PK**, **RHO**, **ADDR**,**KDF** from the spec (with different names), and assume here they are independent random oracles.

We also make the simplfying assumption that we have a symmetric encryption scheme with the property that a fixed message under a uniformly chosen key is itself uniform.

We describe in our notation the function $\mathsf{makeJSD}$, that given input and output notes creates the Joinsplit Description of the transaction from these inputs to outputs.

We use below the notation $\overrightarrow{\mathsf{input}} = (\mathsf{input}_1, \mathsf{input}_2)$ where $\mathsf{input}_i = (\mathsf{a_{pk}}_i^{\mathsf{old}}, \mathsf{v}_i^{\mathsf{old}}, \rho_i^{\mathsf{old}}, \mathsf{rcm}_i^{\mathsf{old}})$; and $\overrightarrow{\mathsf{output}} = (\mathsf{output}_1, \mathsf{output}_2)$ where $\mathsf{output}_i = (\mathsf{a_{pk}}_i^{\mathsf{new}}, \mathsf{v}_i^{\mathsf{new}})$.

$\mathsf{makeJSD}\ (\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w})$:

1. For $i \in \{1, 2\}$, let
   (a) $\mathsf{nf}_i = \mathbf{NF}(\rho_i^{\mathsf{old}}, \mathsf{a_{sk}}_i^{\mathsf{old}})$
   (b) Choose random $\mathsf{rcm}_i^{\mathsf{new}}$ and define $\mathsf{note}_i^{\mathsf{new}} := (\mathsf{a_{pk}}_i^{\mathsf{new}}, \mathsf{v}_i^{\mathsf{new}}, \rho_i^{\mathsf{new}}, \mathsf{rcm}_i^{\mathsf{new}})$.
   (c) $\mathsf{cm}_i = \mathbf{NC}(\mathsf{note}_i^{\mathsf{new}})$
2. Choose uniform $\mathsf{randomseed}$ and $\varphi$. Compute $\mathsf{h_{sig}}$ as described in the spec using the joinsplit pubkey together with $\mathsf{randomseed}, \mathsf{nf}_1, \mathsf{nf}_2$.
3. For $i \in \{1, 2\}$, let
   (a) $h_i = \mathbf{PK}(\mathsf{a_{sk}}_i^{\mathsf{old}}, i, \mathsf{h_{sig}})$.
   (b) $\rho_i^{\mathsf{new}} = \mathbf{RHO}(\varphi, i, \mathsf{h_{sig}})$.
   (c) Choose a random key pair $(\mathsf{esk}_i, \mathsf{epk}_i)$, and let $\mathbf{C}^{\mathsf{enc}}{}_i$ be the encryption under $K_i := \mathbf{KDF}(\mathsf{pk_{enc}}_i^{\mathsf{new}} \cdot \mathsf{esk})$ of $\mathsf{note}_i^{\mathsf{new}}$.
4. Compute a SNARK proof $\pi$ for public input $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{nf}_1, \mathsf{nf}_2, \mathsf{cm}_1, \mathsf{cm}_2, h_1, h_2, \mathsf{h_{sig}})$ using the witness $\left(\mathsf{w}, \overrightarrow{\mathsf{input}}, \{\mathsf{note}_i^{\mathsf{new}}\}_{i \in \{1,2\}}, \varphi\right)$.
5. Output[1] $(\mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{rt}, \mathsf{randomseed}, \mathsf{nf}_1, \mathsf{nf}_2, \mathsf{cm}_1, \mathsf{cm}_2, h_1, h_2, \mathsf{h_{sig}}, \pi, \mathsf{epk}_1, \mathsf{epk}_2, \mathbf{C}^{\mathsf{enc}}{}_1, \mathbf{C}^{\mathsf{enc}}{}_2)$.

[1] It is convenient for us to write $\mathsf{h_{sig}}$ as an explicit part of the Joinsplit description; in practice it is derived online deterministically from other values in the transaction, and not actually written down in the JSD.

# 4 The privacy of the Joinsplit description

Let us say that $(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w})$ is *consistent with* $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$ if the values add up, i.e.

$$\mathrm{v}_{\mathrm{pub}}^{\mathrm{old}} + \sum_{i \in \{1,2\}} \mathsf{v}_i^{\mathsf{old}} = \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}} + \sum_{i \in \{1,2\}} \mathsf{v}_i^{\mathsf{new}},$$

and for $i \in \{1, 2\}$ there is a Merkle path in $\mathsf{w}$ from $\mathbf{NC}(\mathsf{input}_i)$ to $\mathsf{rt}$ (or the dummy note bit is one, and the value of the note is zero).

We say a pair of Sprout notes $\overrightarrow{\mathsf{note}} = (\mathsf{note}_1, \mathsf{note}_2)$ is *an extension* of $\overrightarrow{\mathsf{output}}$ if $\mathsf{note}_i$ coincides with $\mathsf{output}_i$ on the values $\mathsf{a_{pk}}_i^{\mathsf{new}}, \mathsf{v}_i^{\mathsf{new}}$.

Fix consistent $(\mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w})$. Let $\overrightarrow{\mathsf{note}}$ be an extension of $\overrightarrow{\mathsf{output}}$.

We denote by

$$\mathsf{distJSD}(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w}, \overrightarrow{\mathsf{note}})$$

the output distribution of $\mathsf{makeJSD}(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w})$ conditioned on $\mathsf{note}_i^{\mathsf{new}} = \mathsf{note}_i$ for $i \in \{1, 2\}$.

The following theorem implies that seeing a Joinsplit description (including the SNARK proof) in a transaction tells us nothing about the inputs and outputs of the transaction, except that they are consistent with $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$.

**Theorem 4.1.** *Fix values* $\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}$. *Fix* $(\overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w})$ *consistent with* $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$. *Fix* $\overrightarrow{\mathsf{note}}$ *that is an extension of* $\overrightarrow{\mathsf{output}}$. *Then the distribution* $\mathsf{distJSD}(\mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{rt}, \overrightarrow{\mathsf{input}}, \overrightarrow{\mathsf{output}}, \mathsf{w}, \overrightarrow{\mathsf{note}})$ *depends only on* $\mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{rt}$ *and the joinsplit pubkey in the transaction.*

*Proof.* The output of $\mathsf{makeJSD}$ is of the form

$$(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{randomseed}, \mathsf{nf}_1, \mathsf{nf}_2, \mathsf{cm}_1, \mathsf{cm}_2, h_1, h_2, \mathsf{h_{sig}}, \pi, \mathsf{epk}_1, \mathsf{epk}_2, \mathbf{C}^{\mathsf{enc}}{}_1, \mathbf{C}^{\mathsf{enc}}{}_2)$$

We go over the elements sequentially and show that their distribution, even conditioned on any fixing of the previous elements in the sequence, depends only on $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$ and the value of previous elements in the sequence, and thus, inductively, depend only on $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$.

- The first three output values $\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}$ clearly only depend on themselves :)
- $\mathsf{randomseed}$ is uniform and independent of $(\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}})$.
- The values $\mathsf{nf}_1, \mathsf{nf}_2, \mathsf{cm}_1, \mathsf{cm}_2, h_1, h_2$ are random oracle outputs on distinct inputs, and thus uniform and independent of previous elements and each other.
- $\mathsf{h_{sig}}$ is a function of previous elements together with the Joinsplit pubkey.
- Since $\mathscr{S}$ is WI, the distribution of $\pi$ depends only on the public input

  $$\mathsf{rt}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{old}}, \mathrm{v}_{\mathrm{pub}}^{\mathrm{new}}, \mathsf{randomseed}, \mathsf{nf}_1, \mathsf{nf}_2, \mathsf{cm}_1, \mathsf{cm}_2, h_1, h_2, \mathsf{h_{sig}}$$

- $\mathsf{epk}_i$ is a deterministic function of the randomly and independently chosen $\mathsf{esk}_i$

- $\mathbf{C}^{\mathsf{enc}}{}_1$ and $\mathbf{C}^{\mathsf{enc}}{}_2$ are encryptions under the uniformly chosen, and independent of all previous elements so far, keys $K_i$ and thus (using our simplyfing assumption on uniformity of the encryption scheme when the key is uniform), are simply uniform.

□

## References


[1] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 781–796, 2014.

[2] G. Fuchsbauer. Subversion zero-knowledge snarks. 2017.

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