A Proof of Security for the Sapling Generation of zk-SNARK Parameters in the Generic Group Model

Mary Maller

1 Overview

Recently Bowe, Gabizon and Miers designed a multiparty computation [BGM17] to output the public parameters for a recent zk-SNARK by Groth [Gro16]. Their aim is that this computation will be used to generate the new parameters for Zcash. The public parameters output by the multiparty computation differ from the public parameters of Groth's original scheme. As a consequence, Bowe, Gabizon and Miers provided a proof that the scheme is still secure in the generic group model under the new parameters. The purpose of this document is to provide an alternative proof of security for knowledge soundness in the generic group model. Additionally, we provide a proof of subversion zero-knowledge which is not dissimilar from the proof of subversion zero-knowledge presented in [Fuc18].

2 Preliminaries

2.1 Notation

We use $\lambda \in \mathbb{N}$ to denote the security parameter and 1^λ to denote its unary representation. Algorithms are randomised unless explicitly noted otherwise. We use $y \leftarrow A(x;r)$ to denote running algorithm A on inputs x and random coins r and assigning its output to y. Typically we do not specify the randomness and write write $y \leftarrow A(x)$.

2.2 Bilinear Groups

Definition 1. *A* bilinear group generator $\mathsf{BilinearGen}$ *takes as input a security parameter in unary and returns a bilinear group* $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, G, H)$ *consisting of cyclic groups* $\mathbb{G}$, $\mathbb{H}$, $\mathbb{G}_T$ *of prime order* p *and a* bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ *and generators* G, H *for* $\mathbb{G}, \mathbb{H}$ *respectively such that*

- *there are efficient algorithms for computing group operations, evaluating the bilinear map, deciding membership of the groups, and sampling generators of the groups;*
- *the map is bilinear, i.e., for all* $a, b \in \mathbb{Z}$ *we have* $e(G^a, H^b) = e(G, H)^{ab}$*;*
- *and the map is non-degenerate, i.e., if* $e(G_1, H_1) = 1$ *then* $G_1 = 1$ *or* $H_1 = 1$.

Bilinear groups can be set up as symmetric bilinear groups where $\mathbb{G} = \mathbb{H}$ or as asymmetric bilinear groups where $\mathbb{G} \neq \mathbb{H}$. For simplicity, our proof of knowledge soundness will be in the symmetric setting. For our generic adversary this is equivalent to working in the asymmtric setting where it has access to all elements in both groups. Hence our adversary is given strictly greater or equal information to the information it would have in practice.

We will be working with generic algorithms. The only meaningful operations that the algorithms can do are generic group operations, namely: calculating exponentiations of known group elements by known field elements; multiplying group elements in the same group; and using the bilinear pairing function on known group elements in $\mathbb{G}$.

2.3 Quadratic Arithmetic Programs

Groth 2016 uses quadratic arithmetic circuits QAPs to describe the relation. A QAP is described by

$$\{\mathbb{F}, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^m, t(X)\}$$

a field $\mathbb{F}$, three sets of degree $n-1$ polynomials with coefficients in $\mathbb{F}$ $u_i(X), v_i(X), w_i(X)$, and a degree n polynomial with coefficients in $\mathbb{F}$ $t(X)$. The relation is given by

$$R = \left\{ (\phi, w) \;\middle|\; \begin{array}{l} \phi = (a_0 = 1, a_1, \ldots, a_\ell) \in \mathbb{F}^{1+\ell} \\ w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{F}^{m-\ell} \\ \exists h(X) \in \mathbb{F}[X] \text{ of degree} \leq n-2 \text{ such that} \\ (\sum_{i=0}^m a_i u_i(X)) \, (\sum_{i=0}^m a_i v_i(X)) = \sum_{i=0}^m a_i w_i(X) + h(X)t(X) \end{array} \right\}.$$

3 Proof of Knowledge

Let Hash be a collision resistant hash function $\mathsf{Hash} : \{0,1\}^k \mapsto \mathbb{H}$. By Claim 3.5 in [BGM17] the following scheme is a proof of knowledge in the random oracle model under the KEA assumption. That is, for any PPT algorithm $\mathcal{A}$, there exists a PPT extractor $\mathcal{X}$ such that if $(A, \mathsf{string}, P) \leftarrow \mathcal{A}$ such that $\mathsf{PoKVerify}(bp, A, \mathsf{string}, P) = 1$, then $\alpha \leftarrow \mathcal{X}$ such that $A = G^\alpha$ with overwhelming probability. Here we provide an alternative proof of security in the random oracle model under the B-KEA assumption [ABLZ17], which assumes that it is infeasible to compute $C, \hat{C}$ such that $e(C, H) = e(G, \hat{C})$ without knowing s such that $(C, \hat{C}) = (G^s, H^s)$. Since the Fiat-Shamir heuristic is non-malleable [FKMV12] we do not need to consider show that a strengthened adversary that can see additional proofs can gain additional advantage. Groth et. al. [GKM+18] went for an alternative proof style where their proof was malleable however the adversary could not compute the required updated string if they cheated in this manner. Their technique is not applicable here because we need to argue that the proof of knowledge with respect to the hashed value still holds.

Assumption 1 (The B-KEA Assumption) *Let $\mathcal{A}$ be a PPT adversary. Then there exists a PPT extractor $\mathcal{X}$ such that*

$$\Pr[bp \leftarrow \mathsf{BilinearGen}(1^\lambda); (C, \hat{C}) \leftarrow \mathcal{A}(bp); s \leftarrow \mathcal{X}(\mathsf{trans}_\mathcal{A}) : \ e(C, H) = e(G, \hat{C}) \ \wedge \ (C, \hat{C}) \neq (G^s, H^s)]$$

is negligible in λ.

$\mathsf{PoKProve}(bp, \mathsf{string}, G^\alpha, \alpha)$	$\mathsf{PoKVerify}(bp, A, \mathsf{string}, P)$
$R \leftarrow \mathsf{Hash}(G^\alpha, \mathsf{string})$	check $A \in \mathbb{G}, P \in \mathbb{H}$
$P \leftarrow R^\alpha$	$R \leftarrow \mathsf{Hash}(G^\alpha, \mathsf{string})$
return P	check $e(A, R) = e(G, P)$
	return 1 if both checks pass, else return 0

Lemma 1. *The algorithms ($\mathsf{PoKProve}, \mathsf{PoKVerify}$) are a proof of knowledge under the B-KEA assumption.*

Proof. We prove soundness in the interactive setting, i.e. when the random element R is provided by an honest verifier. Soundness in the non-interactive setting then follows from the Fiat-Shamir paradigm. Let $\mathcal{A}$ be a PPT adversary. Suppose that $bp \leftarrow \mathsf{BilinearGen}(1^\lambda)$ and that $(A, \mathsf{string}) \leftarrow \mathcal{A}(bp)$ is sent to the verifier. The verifier responds with $R \xleftarrow{\$} \mathbb{H}$. The adversary $\mathcal{A}$ then returns P such that $e(A, R) = e(G, P)$. Since R is a random generator of $\mathbb{H}$, by the B-KEA assumption, there exists an extractor $\mathcal{X}$ that outputs a such that $(A, P) = (G^a, R^a)$.

An additional algorithm to check the ratio of 4 elements was given in [BGM17]. It works as follows.

$\mathsf{CheckRatio}(bp, (A_1, B_1), (A_2, B_2))$
check $A_1, A_2 \in \mathbb{G}, B_1, B_2 \in \mathbb{H}$
check $e(A_1, B_1) = e(A_2, B_2)$
return 1 if both checks pass, else return 0

4 Security properties

We take our definitions from Groth et al [GKM+18]. Their definitions are simpler because they only have one round in the setup. Instead of update security, we define 2-round security 2RS to capture the notion of a scheme that is secure under a 2 round setup. In terms of zero-knowledge, we aim to prove simulation zero-knowledge which is strictly stronger than 2RS zero-knowledge. For simplicity, we shall avoid discussing 2RS zero-knowledge.

For each security property, the game in the left column of Figure 1 resembles the usual security game for zero-knowledge, soundness, and knowledge soundness. The difference is in the creation of the CRS crs, which is initially set to $\perp$. We then model the process of generating the CRS as an interaction between the adversary and a setup oracle $\mathcal{O}_\mathsf{s}$, at the end of which the oracle sets this value crs and returns it to the adversary.

In the definition of zero-knowledge, we require the existence of a PPT simulator consisting of algorithms $(\mathsf{SimSetup}, \mathsf{Simulate})$ that share state with each other. The idea is that it can be used to simulate the generation of common reference strings and simulate proofs without knowing the corresponding witnesses.

Definition 2. *Let $\mathsf{P} = (\mathsf{Setup}, \mathsf{2RSSetup}, \mathsf{2RSUpdate}, \mathsf{SetupVerify}, \mathsf{Prove}, \mathsf{Verify})$ be a non-interactive argument for the relation R. Then the argument is*

- *complete if for all PPT algorithms $\mathcal{A}$ the advantage $|1 - \Pr[\mathsf{COMP}_\mathcal{A}(\lambda)]|$ is negligible in λ.*
- *subversion zero-knowledge if for all PPT algorithms $\mathcal{A}$ there exists a simulator $\mathsf{Sim}_\mathcal{A} = (\mathsf{SimSetup}, \mathsf{Simulate}_\mathcal{A})$ where the advantage $|2\Pr[\mathsf{S\text{-}ZK}_{\mathcal{A}, \mathsf{Sim}_\mathcal{A}}(1^\lambda) = 1] - 1|$ is negligible in λ.*
- *2RS-sound if for all PPT algorithms $\mathcal{A}$ the probability $\Pr[\mathsf{X\text{-}SND}_\mathcal{A}(1^\lambda) = 1]$ is negligible in λ.*
- *2RS-knowledge-sound if for all PPT algorithms $\mathcal{A}$ there exists a PPT extractor $\mathcal{X}_\mathcal{A}$ such that $|\Pr[\mathsf{X\text{-}KSND}_{\mathcal{A}, \mathcal{X}_\mathcal{A}}(1^\lambda)]|$ is negligible in λ.*

Moreover, if a definition holds with respect to an adversary with unbounded computation, we say it holds statistically, and if the advantage is exactly 0, we say it holds perfectly.

$\underline{\text{MAIN } \mathsf{COMP}_{\mathcal{A}}(\lambda)}$
$(\mathsf{crs}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M, \phi, w) \leftarrow \mathcal{A}(1^\lambda)$
$b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$
if $b = 0$ or $(\mathsf{crs}, \phi, w) \notin R$ return 1
$\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, \phi, w)$
return $\mathsf{Verify}(\mathsf{crs}, \phi, \pi)$

$\underline{\text{MAIN } \mathsf{X\text{-}ZK}_{\mathcal{A},\mathsf{Sim}_{\mathcal{A}}}(\lambda)}$
$b \xleftarrow{\$} \{0,1\}$
if $b = 0$
  $\mathsf{Setup} \leftarrow \mathsf{SimSetup}$
$\mathsf{crs} \leftarrow \bot$; $Q \leftarrow \emptyset$
$\mathsf{state} \xleftarrow{r} \mathcal{A}^{\mathsf{X}\text{-}\mathcal{O}_\mathsf{s}}(1^\lambda)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{pf}}}(\mathsf{state})$
return 1 if $b' = b$ else return 0

$\underline{\mathcal{O}_{\mathsf{pf}}(\phi, w)}$
if $(\mathsf{crs}, \phi, w) \notin R$ return $\bot$
if $b = 0$ return $\mathsf{Simulate}_{\mathcal{A}}(\mathsf{crs}, r, \phi)$
else return $\mathsf{Prove}(\mathsf{crs}, \phi, w)$

$\underline{\text{MAIN } \mathsf{X\text{-}SND}_{\mathcal{A}}(\lambda)}$
$\mathsf{crs} \leftarrow \bot$; $Q \leftarrow \emptyset$
$(\phi, \pi) \xleftarrow{\$} \mathcal{A}^{\mathsf{X}\text{-}\mathcal{O}_\mathsf{s}}(1^\lambda)$
return $\mathsf{Verify}(\mathsf{crs}, \phi, \pi) \ \wedge \ \phi \notin L_R$

$\underline{\text{MAIN } \mathsf{X\text{-}KSND}_{\mathcal{A},\mathcal{X}_{\mathcal{A}}}(\lambda)}$
$\mathsf{crs} \leftarrow \bot$, $Q \leftarrow \emptyset$
$(\phi, \pi) \xleftarrow{r} \mathcal{A}^{\mathsf{X}\text{-}\mathcal{O}_\mathsf{s}}(1^\lambda)$
$w \xleftarrow{\$} \mathcal{X}_{\mathcal{A}}(\mathsf{crs}, r)$
return $\mathsf{Verify}(\mathsf{crs}, \phi, \pi) \ \wedge \ (\phi, w) \notin R$

$\underline{\mathsf{T}\text{-}\mathcal{O}_\mathsf{s}(x)}$
if $\mathsf{crs} \neq \bot$ return $\bot$
$(\mathsf{crs}, \rho) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
return (crs, ρ)

$\underline{\mathsf{2RS}\text{-}\mathcal{O}_\mathsf{s}(\mathsf{intent}, \mathsf{crs}_N, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)}$
if $\mathsf{crs} \neq \bot$ return $\bot$
if $\mathsf{intent} = \mathsf{setup}$
  if $N = 0$:
    $(\mathsf{crs}_1, \rho_1) \xleftarrow{\$} \mathsf{2RSSetup}(1^\lambda)$
    $Q1 \leftarrow \{\rho_1\}$
    return (crs_1, ρ_1)

  else:
    $(\mathsf{crs}_{N+1}, \psi_1) \xleftarrow{\$} \mathsf{2RSSetup}(\mathsf{crs}_N, (\rho_i)_{i=1}^N)$
    $Q2 \leftarrow \{\psi_1\}$
    return $(\mathsf{crs}_{N+1}, \psi_1)$

if $\mathsf{intent} = \mathsf{1round}$
  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_N, (\rho_i)_{i=1}^N)$
  if $b = 0$ return $\bot$
  $(\mathsf{crs}', \rho') \xleftarrow{\$} \mathsf{2RSUpdate}(1^\lambda, \mathsf{crs}_N, 1, (\rho_i)_{i=1}^N)$
  $Q1 \leftarrow Q1 \cup \{\ \rho'\}$
  return (crs', ρ')

if $\mathsf{intent} = \mathsf{2round}$
  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$
  if $b = 0$ return $\bot$
  $(\mathsf{crs}', \psi') \xleftarrow{\$}$
  $\mathsf{2RSUpdate}(1^\lambda, \mathsf{crs}_{N+M}, 2, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$
  $Q2 \leftarrow Q2 \cup \{\ \rho'\}$
  return (crs', ρ')

if $\mathsf{intent} = \mathsf{final}$
  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$
  if $b = 0$ or $Q1 \cap \{\ \rho_i\}_i = \emptyset$ or $Q2 \cap \{\ \psi_i\}_i = \emptyset$ return $\bot$
  set $\mathsf{crs} \leftarrow \mathsf{crs}_{N+M}$ and return crs
else return $\bot$

$\underline{\mathsf{S}\text{-}\mathcal{O}_\mathsf{s}(\mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)}$
if $\mathsf{crs} \neq \bot$ return $\bot$
$b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M) = 0$
if $b = 0$ return $\bot$
set $\mathsf{crs} \leftarrow \mathsf{crs}_{N+M}$ and return crs

Fig. 1: The left games define completeness, zero-knowledge (X-ZK), soundness (X-SND), and knowledge soundness (X-KSND). The right oracles define the notions of trusted, 2 round setup, and subvertible CRS setups. A complete game is constructed by using an oracle from the right side in the game on the left side.

5 The Setup Algorithms and some Helpful Lemmas

In this section we describe the setup algorithms used in Sapling. We also provide four simple Lemmas regarding the format of setup transcripts that verify. These will be helpful later on in our security proofs. The setup algorithms are given in Figures 2&3.

$\underline{\mathsf{2RSSetup}(bp, \mathsf{crs}_1, (\rho_i)_{i=1}^N)}$
if $N = 0$:
$\mathsf{string} = bp$
$x, \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs} \leftarrow \left(bp, \{G^{x^i}, H^{x^i}\}_{i=0}^{2n-2}, \{G^{\alpha x^i}, H^{\alpha x^i}, G^{\beta x^i}, H^{\beta x^i}\}_{i=0}^{n-1}\right)$
$\rho \leftarrow \mathsf{PoKProve}(bp, G^{\alpha}, \mathsf{string}), \mathsf{PoKProve}(bp, G^{\beta}, \mathsf{string}), \mathsf{PoKProve}(bp, G^{x}, \mathsf{string}))$
return (crs, ρ)
else:
$\mathsf{string} = bp + \sum_{i=1}^N \rho_i$
$\delta \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs} \leftarrow \left(\mathsf{crs}_1, G^{\delta}, H^{\delta}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right)$
$\psi \leftarrow \mathsf{PoKProve}(bp, G^{\delta}, \mathsf{string})$
return (crs, ψ)

$\underline{\mathsf{2RSUpdate}(1^\lambda, \mathsf{crs}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)}$
$\mathsf{string} = bp + \sum_{i=1}^N \rho_i + \sum_{i=1}^M \psi_i$

if $M = 0$:
$\left(bp, \{G^{x^i}, H^{x^i}\}_{i=0}^{2n-2}, \{G^{\alpha x^i}, H^{\alpha x^i}, G^{\beta x^i}, H^{\beta x^i}\}_{i=0}^{n-1}\right) \leftarrow$ parse crs
$x', \alpha', \beta' \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs}' \leftarrow \left(bp, \{G^{(xx')^i}, H^{(xx')^i}\}_{i=0}^{2n-2}, \{G^{\alpha\alpha'(xx')^i}, H^{\alpha\alpha'(xx')^i}, G^{\beta\beta'(xx')^i}, H^{\beta\beta'(xx')^i}\}_{i=0}^{n-1}\right)$
$(P_A, P_B, P_X) \leftarrow \mathsf{PoKProve}(bp, G^{\alpha'}, \mathsf{string}, \alpha'), \mathsf{PoKProve}(bp, G^{\beta'}, \mathsf{string}, \beta'), \mathsf{PoKProve}(bp, G^{x'}, \mathsf{string}, x'))$
$\rho' \leftarrow (G^{\alpha'}, G^{\beta'}, G^{x'}, A, B, X, G^{\alpha\alpha'}, G^{\beta\beta'}, G^{xx'})$
return crs', ρ'

else $M > 0$
$\left(\mathsf{crs}_1, G^{\delta}, H^{\delta}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right) \leftarrow$ parse crs
$\delta' \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs}' \leftarrow \left(\mathsf{crs}_1, G^{\delta\delta'}, H^{\delta\delta'}, \left\{G^{\frac{x^i t(x)}{\delta\delta'}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta\delta'}}\right\}_{i=\ell+1}^{m}\right)$
$P_D \leftarrow \mathsf{PoKProve}(bp, G^{\delta'}, \mathsf{string}, \delta')$
$\psi' \leftarrow (G^{\delta'}, P_D, G^{\delta\delta'})$
return crs', ψ'

Fig. 2: Two round setup process.

Lemma 2. *Suppose that for* crs *there exists* $\{\rho_i\}_{i=1}^N, \{\psi_i\}_{i=1}^M$ *such that* $\mathsf{Verify}(1^\lambda, \mathsf{crs}, \{\rho_i\}_{i=1}^N, \{\psi_i\}_{i=1}^M) = 1$ *with non-negligible probability. Then there exists* $(\alpha, \beta, \delta, x)$ *such that*

$$\mathsf{crs} = \left(bp, \left\{G^{x^i}, H^{x^i}\right\}_{i=0}^{2n-2}, G^{\delta}, H^{\delta}, \left\{G^{\alpha x^i}, G^{\beta x^i}, H^{\alpha x^i}, H^{\beta x^i}\right\}_{i=0}^{n-1}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right).$$

.

Proof. This follows because the verifier checks that $1 \leq i \leq 2n-3$, $1 \leq j \leq n-2$, $\ell+1 \leq k \leq m$ and $v = \alpha, \beta, 1$ that

$\underline{\mathsf{SetupVerify}(1^\lambda, \mathsf{crs}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)}$

$\left(bp, \left\{G^{x^i}, H^{x^i}\right\}_{i=0}^{2n-2}, G^\delta, H^\delta, \left\{G^{\alpha x^i}, G^{\beta x^i}, H^{\alpha x^i}, H^{\beta x^i}\right\}_{i=0}^{n-1}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right) \leftarrow \text{parse } \mathsf{crs}$

for $1 \le i \le N$
  $\mathsf{string} = bp + \sum_{i=1}^{N-1} \rho_i$
  $(A_i, B_i, X_i, P_{A_i}, P_{B_i}, P_{X_i}, L_{A_i}, L_{B_i}, L_{X_i}) \leftarrow$ parse ρ_i
  for $V = A, B, X$
    check $\mathsf{PoKVerify}(bp, \mathsf{string}, V_i, P_{V_i})$
    $R_{V_i} \leftarrow \mathsf{Hash}(A_i, \mathsf{string})$
  if $i \neq 1$
    check $\mathsf{CheckRatio}(bp, (L_{V_{i-1}}, P_{V_i}), (L_{V_i}, R_{V_i})) = 1$

for $1 \le i \le M-1$
  $\mathsf{string} = bp + \sum_{i=1}^{N} \rho_i + \sum_{i=1}^{M-1} \psi_i$
  $(D_i, P_{D_i}, L_{D_i}) \leftarrow$ parse ψ_i
  check $\mathsf{PoKVerify}(bp, \mathsf{string}, D_i, P_{D_i})$
  $R_{D_i} \leftarrow \mathsf{Hash}(D_i, \mathsf{string})$
  if $i \neq 1$
    check $\mathsf{CheckRatio}(bp, (L_{D_{i-1}}, P_{D_i}), (L_{D_i}, R_{D_i})) = 1$

check $G^\alpha = A_N \wedge G^\beta = B_N \ \wedge \ G^x = X_N \ \wedge \ G^\delta = D_M$
for $v = \alpha, \beta, x, \delta$, check $e(G^v, H) = e(G, H^v)$

for $1 \le i \le 2n-3$ and $v = \alpha, \beta, 1$
  check $e(G^{vx^{i+1}}, H) = e(G^{vx^i}, H^x)$
  check $e(G, H^{vx^{i+1}}) = e(G^x, G^{vx^i})$
for $1 \le i \le n-2$
  check $e(G^{\frac{x^i t(x)}{\delta}}, H^\delta) = e(G^{x^i t(x)}, H)$
for $\ell+1 \le i \le m$
  check $e(G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}, H^\delta) = e(G^{(\beta u_i(x)+\alpha v_i(x)+w_i(x))}, H)$
return 1 if all checks pass, else return 0

Fig. 3: Verification of the setup transcript.

$$
\begin{aligned}
&e(G^{vx^{i+1}}, H) = e(G^{vx^i}, H^x)\\
&e(G, H^{vx^{i+1}}) = e(G^x, G^{vx^i})\\
&e(G^{\frac{x^j t(x)}{\delta}}, H^\delta) = e(G^{x^j t(x)}, H)\\
&e(G^{\frac{\beta u_k(x)+\alpha v_k(x)+w_k(x)}{\delta}}, H^\delta) = e(G^{(\beta u_k(x)+\alpha v_k(x)+w_k(x))}, H)
\end{aligned}
$$

Lemma 3. *Suppose that for $\{\rho_i\}_{i=1}^N, \{\psi_i\}_{i=1}^M$ there exists* crs *such that* $\mathsf{Verify}(1^\lambda, \mathsf{crs}, \{\rho_i\}_{i=1}^N, \{\psi_i\}_{i=1}^M) = 1$ *with non-negligible probability. Write* crs *as*

$$
\mathsf{crs} = \left(bp, \left\{G^{x^i}, H^{x^i}\right\}_{i=0}^{2n-2}, G^\delta, H^\delta, \left\{G^{\alpha x^i}, G^{\beta x^i}, H^{\alpha x^i}, H^{\beta x^i}\right\}_{i=0}^{n-1}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right).
$$

Then

$$
\alpha = \prod_{i=1}^N \log_H \hat{A}_i, \quad \beta = \prod_{i=1}^N \log_H \hat{B}_i, \quad x = \prod_{i=1}^N \log_H \hat{X}_i, \quad \delta = \prod_{i=1}^M \log_H \hat{D}_i.
$$

Proof. This holds due to the checks that for $V = A, B, X$, $2 \le i \le N$

$$
\mathsf{CheckRatio}(bp, (L_{V_{i-1}}, P_{V_i}), (L_{V_i}, R_{V_i})) = 1
$$

and for $2 \le i \le M$

$$
\mathsf{CheckRatio}(bp, (L_{D_{i-1}}, P_{D_i}), (L_{D_i}, R_{D_i})) = 1.
$$

Lemma 4. *Suppose that there exists a generic adversary $\mathcal{A}$ that outputs ρ_r, ψ_s for some $1 \le r \le N$ and $1 \le s \le M$ such that there exists* crs*, ρ_i, ψ_i such that* $\mathsf{SetupVerify}(\mathsf{crs}, \{\rho\}_{i=1}^N, \psi_{i=1}^M) = 1$*. Then, by the B-KEA assumption, there exists a PPT extractor $\mathcal{X}$ that, given the random tape of $\mathcal{A}$ as input, outputs $(\alpha, \beta, \delta, x)$ such that $\log_H \hat{A}_r$, $\log_H \hat{B}_r$, $\log_H \hat{X}_r$, $\log_G \hat{D}_s = \alpha, \beta, x, \delta$.*

Proof. This holds directly from the security of the proof of knowledge for A_i, B_i, X_i and D_i.

Lemma 5. *Suppose there exists a generic adversary $\mathcal{A}$ that outputs* crs$, \rho, \psi$ *such that* $\mathsf{Verify}(1^\lambda, \mathsf{crs}, \rho, \psi) = 1$ *with non-negligible probability. Then, by the B-KEA assumption, there exists a PPT extractor $\mathcal{X}$ that, given the random tape of $\mathcal{A}$ as input, outputs $(\alpha, \beta, \delta, x)$ such that* $(\mathsf{crs}, \rho) = \mathsf{Setup}(1^\lambda; (\alpha, \beta, \delta, x))$.

Proof. By Lemma 2, for any reference string that passes verification there exist values $(\alpha, \beta, \delta, x) \in \mathbb{Z}_p^4$ such that crs contains

$$
\left(bp, \left\{G^{x^i}, H^{x^i}\right\}_{i=0}^{2n-2}, G^\delta, H^\delta, \left\{G^{\alpha x^i}, G^{\beta x^i}, H^{\alpha x^i}, H^{\beta x^i}\right\}_{i=0}^{n-1}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2}, \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\right).
$$

If ρ, ψ pass verification then by equality checks we additionally have that

$$
\rho = (G^\alpha, H^\alpha, G^\beta, H^\beta, G^x, H^x), \quad \psi = (G^\delta, H^\delta).
$$

Thus crs$, \rho, \psi$ are structured exactly as if it were computed by $\mathsf{Setup}(1^\lambda)$. Moreover, by Lemma 4, there exists a PPT extractor that outputs α, β, δ, x.

6 Subversion Zero-Knowledge

Subversion zero-knowledge implies that even if the entire setup process was subverted, the subverter still should not be able to deduce any information about the witness from the proof that it could not calculate anyway. The proof is simple and reflects techniques used in [Fuc18].

Theorem 2. *The system has subversion knowledge soundness by Lemma 5.*

Proof. To prove subversion zero-knowledge, we need to both show the existence of an extractor $\mathcal{X}$, and describe a Simulate algorithm that produces indistinguishable proofs when provided the extracted trapdoor (which it can compute given the randomness of both $\mathcal{A}$ and the honest algorithms). The simulator knows α, β, δ, x and picks $r \leftarrow \mathbb{Z}_p$ and sets $A = G^r$, $B = H^r$ and $C = G^{r^2 - \alpha\beta - \sum_{i=1}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}$. The simulated proof has the same distribution as a real proof, since A, B are both distributed uniformly at random and C is determined uniquely by A and B. Consequently, subversion zero-knowledge follows from the extraction of the trapdoor, which can be extracted by Lemma 5.

MAIN $\mathsf{Game}_2\mathcal{A}, \mathcal{X}_{\mathcal{A}}(\lambda)$

$\mathsf{crs} \leftarrow \bot$, $Q \leftarrow \emptyset$

$(\phi, \pi) \xleftarrow{r} \mathcal{A}^{\mathcal{O}_2}(1^\lambda)$

$w \xleftarrow{\$} \mathcal{X}_{\mathcal{A}}(\mathsf{crs}, r)$

return $\mathsf{Verify}(\mathsf{crs}, \phi, \pi) \ \wedge \ (\phi, w) \notin R$

$\mathcal{O}_2(\mathsf{intent}, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$

if $\mathsf{crs} \neq \bot$ return $\bot$

if $\mathsf{intent} = \mathsf{setup}$

  if $M = 0$:

    $(\mathsf{crs}_1, \rho_1) \xleftarrow{(\alpha_1, \beta_1, x_1)} 2\mathsf{RSSetup}(1^\lambda)$

    $\mathsf{crs}_f \leftarrow \mathsf{crs}_1, I \leftarrow 1$

    $Q1 \leftarrow \{\rho_1\}$

    return (crs_1, ρ_1)

  else:

    $(\mathsf{crs}_{N+1}, \psi_1) \xleftarrow{\delta_1} 2\mathsf{RSSetup}(1^\lambda, \mathsf{crs}_N, (\rho_i)_{i=1}^N)$

    $\mathsf{crs}_g \leftarrow \mathsf{crs}_{N+1}, J \leftarrow 1$

    $Q2 \leftarrow \{\psi_1\}$

    return $(\mathsf{crs}_{N+1}, \psi_1)$

if $\mathsf{intent} = 1\mathsf{round}$

  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_N, (\rho_i)_{i=1}^N)$

  if $b = 0$ return $\bot$

  if $Q1 = \emptyset$

    $(\mathsf{crs}', \rho') \xleftarrow{(\alpha_N, \beta_N, x_1)}$

    $2\mathsf{RSUpdate}(1^\lambda, \mathsf{crs}_N, 1, (\rho_i)_{i=1}^N)$

    $\mathsf{crs}_f \leftarrow \mathsf{crs}', I \leftarrow N$

  else

    $(\mathsf{crs}', \rho') \xleftarrow{\alpha_N, \beta_N, x_N} 2\mathsf{RSUpdate}(1^\lambda, \mathsf{crs}_f, 1, (\rho_i)_{i=1}^I)$

    $(\alpha, \beta, x) \leftarrow \bigotimes_{i=I+1}^{N+1} (\alpha_i, \beta_i, x_i)$

    $P_A, P_B, P_X \leftarrow \mathsf{PoKProve}(bp, G^\alpha, \mathsf{string}, \alpha), \ldots$

    $\rho' \leftarrow (G^\alpha, \ldots, P_A, \ldots, G^{\alpha_1 \cdots \alpha_I \alpha}, \ldots)$

  $Q1 \leftarrow Q1 \cup \{\ \rho'\}$

  return (crs', ρ')

if $\mathsf{intent} = 2\mathsf{round}$

  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$

  if $b = 0$ return $\bot$

  if $Q2 = \emptyset$

    $(\mathsf{crs}', \psi') \xleftarrow{\$}$

    $2\mathsf{RSUpdate}(1^\lambda, \mathsf{crs}_{N+M}, 2, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$

    $\mathsf{crs}_g \leftarrow \mathsf{crs}', J \leftarrow M$

  else

    $(\mathsf{crs}', \psi') \xleftarrow{\delta_M} 2\mathsf{RSUpdate}(1^\lambda, \mathsf{crs}_g, 2, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^J)$

    $\delta \leftarrow \prod_{i=J+1}^{M+1} \delta_i$

    $P_D \leftarrow \mathsf{PoKProve}(bp, G^\delta, \mathsf{string}, \delta)$

    $\psi' \leftarrow (G^\psi, P_D, G^{\delta_1 \cdots \delta_I \delta})$

  $Q2 \leftarrow Q2 \cup \{\ \rho'\}$

  return (crs', ρ')

if $\mathsf{intent} = \mathsf{final}$

  $b \leftarrow \mathsf{SetupVerify}(1^\lambda, \mathsf{crs}_{N+M}, (\rho_i)_{i=1}^N, (\psi_i)_{i=1}^M)$

  if $b = 0$ or $Q1 \cap \{\ \rho_i\}_i = \emptyset$ or $Q2 \cap \{\ \psi_i\}_i = \emptyset$ return $\bot$

  set $\mathsf{crs} \leftarrow \mathsf{crs}_{N+M}$ and return crs

else return $\bot$

Fig. 4: Game where all oracle updates are performed on the same query.

7 Knowledge Soundness

Here we show that an adversary cannot produce a valid proof unless it knows a valid witness. For our proof of knowledge soundness, we go beyond the knowledge soundness definition provided in [BGM17]. We begin with a Lemma that shows that an adversary participating in all but one of the updates has a strictly higher advantage in the knowledge soundness game then one that participates less. We then show that even this strengthened adversary cannot break knowledge soundness.

7.1 Single Update per Round Suffices to Show Knowledge Soundness

Lemma 6. *If there is no restricted adversary that breaks* 2RSUpdate *knowledge-soundness which can only query* 2RS-$\mathcal{O}_\mathsf{s}$ *on* 1round *at most once and on* 2round *at most once, then there is no adversary that breaks* 2RSUpdate *knowledge soundness.*

Proof. This is not immediately obvious because knowledge extractors are counter intuitive. Although it is possible to build an adversary that succeeds with less queries to the oracle from one that succeeds with more queries, it does not follow that an extractor can be built for the adverary with more queries. Instead we borrow from Lemma 6 of [GKM+18] which uses that the trapdoor components commute (which ours also do).

There are three games: the first is the standard 2RS-knowledge soundness game. The third game is the 2RS-knowledge soundness game where the adversary can only query 1round and 2round once. The second game is given in Figure 4. The adversary can make multiple queries, but every time they make an extra query in each round, the oracle instead just reupdates their first

query and simulates a 2RS proof. Provided the oracle has access to the adversaries state, the oracle can use the commutability of trapdoor components to simulate a proof.

We show if there does not exist an adversary that can win the third game then there does not exist an adversary that can win the second game. Suppose that for any adversary in the third game, there exists an extractor $\mathcal{X}_A$ such that $\mathcal{A}$ looses. Let $\mathcal{B}$ be an adversary playing the second game that outputs a valid instance and proof with respect to the final crs. We build an extractor for $\mathcal{B}$ as follows. First we construct $\mathcal{A}$ against the third game. The algorithm $\mathcal{A}$ runs $\mathcal{B}$. When $\mathcal{B}$ requests an update, $\mathcal{A}$ simulates the process - unless it is for the final query in the round, in which case $\mathcal{A}$ queries its own oracle and returns the output. Then $\mathcal{A}$ returns $\mathcal{B}$'s output. If $\mathcal{B}$'s output verifies, then $\mathcal{A}$'s output verifies, so there exists an extractor $\mathcal{X}_A$ that outputs a valid witness. The extractor $\mathcal{X}_B$ for $\mathcal{B}$ runs $\mathcal{A}$ and $\mathcal{X}_A$, and returns the output of $\mathcal{X}_A$. As $\mathcal{X}_A$'s witness is valid, so is $\mathcal{X}_B$'s, so $\mathcal{B}$ looses.

Let $\mathcal{A}$ and $\mathcal{X}$ be a PPT algorithm and extractor such that $\mathcal{A}$ wins the second game. We show that $\mathcal{A}$ wins the first game with respect to $\mathcal{X}$. To see this consider an alternative game where $\mathcal{B}$ is trying to guess whether it is interacting with the oracle in the first game or the second game. The responses of the two oracles are distributed identically, so this is a statistically impossible game. The adversary $\mathcal{B}$ simulates $\mathcal{A}$ and $\mathcal{X}$ and return 1 if and only if $\mathcal{A}$ succeeds. When $\mathcal{A}$ queries its oracle, $\mathcal{B}$ queries its oracle on the same input and returns the response. If $\mathcal{A}$ and $\mathcal{X}$ have different probabilities between the two games, then $\mathcal{B}$'s advantage will not equal $\frac{1}{2}$, contradicting the impossibility of the game.

We now show that a generic adversary in Game 3 from Lemma 6 cannot break knowledge soundness. This suffices to show that a generic adversary cannot break 2RS knowledge soundness.

Theorem 3. *The system has* 2RS *knowledge soundness in the generic group model.*

Proof. Imagine we have a generic adversary that queries 2RS-$\mathcal{O}_\mathsf{s}$ on either (setup) or $(1\mathsf{round}, \mathsf{crs}_{r-1}, \{\rho\}_{i=1}^{r-1})$, and then on $(2\mathsf{round}, \mathsf{crs}_{N+s-1}, \{\rho\}_{i=1}^{N}, \{\psi\}_{i=1}^{s-1})$, and then on $(\mathsf{final}, \mathsf{crs}_{N+M}, \{\rho\}_{i=1}^{N}, \{\psi\}_{i=1}^{M})$ and outputs an instance and proof (ϕ, π) that gets accepted; i.e. such that $\mathsf{SetupVerify}(R, \mathsf{crs}_{N+M}, \{\rho\}_{i=1}^{N}, \{\psi\}_{i=1}^{M}) = 1$ and $Q1$ and $Q2$ are non-empty and $\mathsf{Verify}(\mathsf{crs}_{N+M}, \phi, \pi) = 1$. By Lemma 6, if we can build an extractor for any such adversary then 2RS knowledge soundness is implied.

Since the final crs verifies, by Lemmas 2&3 it is of the form

$$bp,\ \left\{G^{x^i}\right\}_{i=0}^{2n-2},\ G^{\delta},\ \left\{G^{\alpha x^i}, G^{\beta x^i}\right\}_{i=0}^{n-1},\ \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2},\ \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}\ \left\{H^{x^i}\right\}_{i=0}^{2n-2},$$
$$H^{\delta},\ \left\{H^{\alpha x^i}, H^{\beta x^i}\right\}_{i=0}^{n-1},\ \left\{H^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2},\ \left\{H^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}$$

for some $x_1, \alpha_1, \beta_1, \ldots, x_N, \alpha_N, \beta_N$ such that

$$x = \prod_{j=1}^{N} x_j,\ \alpha = \prod_{j=1}^{N} \alpha_j,\ \beta = \prod_{j=1}^{N} \beta_j,\ \delta = \prod_{j=1}^{M} \delta_j.$$

Since the adversary outputs verifying proofs, by Lemma 4 there exist extractors $\mathcal{X}_i$ for $1 \le i \le N$, $i \ne r$, that output $\tau_{1,i} = (\alpha_i, \beta_i, x_i)$. Similarly, there exist extractors $\mathcal{X}_j$ for $1 \le j \le M$, $i \ne s$, that output $\tau_{2,i} = (\delta_i)$.

Values Queried by the Adversary

The adversary is limited solely in the responses from its oracle, since it gets to choose the final crs. If the adversary queries 2RS-$\mathcal{O}_\mathsf{s}$ on $(1\mathsf{round}, \mathsf{crs}_{r-1}, \{\rho\}_{i=1}^{r-1})$ with $r > 2$ it obtains the values

$$\left\{\textstyle\prod_{j=1}^{r} x_j)^i\right\}_{i=0}^{2n-2},\ \left\{\textstyle\prod_{j=1}^{r} \alpha_j) x_j^i,\ \prod_{j=1}^{r} \beta_j x_j^i\right\}_{i=0}^{n-1}.$$

in both groups and

$$\chi_{\alpha_r},\ \chi_{\alpha_r}\alpha_r, \chi_{\beta_r},\ \chi_{\beta_r}\beta_r,\ \chi_{x_r},\ \chi_{x_r}x_r$$

for random $\chi_{\alpha_r}, \chi_{\beta_r}, \chi_{x_r}$ in $\mathbb{H}$. Since $\mathcal{A}$ can divide through by $\prod_{j=1}^{r-1} x_j^i$, $\prod_{j=1}^{r-1} \alpha_j x_j^i$, $(\prod_{j=1}^{r-1} \beta_j) \prod_{j=1}^{r-1} x_j$, this is akin to being given the values

$$\left\{x_r^i\right\}_{i=0}^{2n-2},\ \left\{\alpha_r x_r^i, \beta_r x_r^i\right\}_{i=0}^{n-1}$$

in both groups. If the adversary queries 2RS-$\mathcal{O}_\mathsf{s}$ on (setup) it obtains the same values with $r = 1$ directly.

When the adversary queries $2\mathsf{round}$ on $\mathsf{crs}_{N+s-1}, \{\rho\}_{i=1}^{N}, \{\psi\}_{i=1}^{s-1})$ it obtains the values

$$\textstyle\prod_{j=1}^{s} \delta_j,\ \left\{\frac{x^i t(x)}{\prod_{j=1}^{s} \delta_j}\right\}_{i=0}^{n-2},\ \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\prod_{j=1}^{s} \delta_j}\right\}_{i=\ell+1}^{m}.$$

in both groups and

$$\chi_{\delta_s},\ \chi_{\delta_s}\delta_s$$

for random χ_{δ_s} in $\mathbb{H}$. Since $\mathcal{A}$ can divide through by $\prod_{j=1}^{s-1} \delta_j$, this is akin to being given the values

$$\delta_s,\ \left\{\frac{x^i t(x)}{\delta_s}\right\}_{i=0}^{n-2},\ \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta_s}\right\}_{i=\ell+1}^{m}$$

in both groups. If the adversary queries 2RS-$\mathcal{O}_\mathsf{s}$ on (setup) it obtains the same values with $s = 1$ directly.

Format of a Generic Advesary's Verifying Proof

Where $\mathcal{A}$ outputs a verifying proof $\pi = A, B, C \in \mathbb{G}^3$, A, B, C are a linear combination of elements in the obtained from the oracles. This means that $\mathcal{A}$ knows polynomials $A'(X_r), A'_\alpha(X_r), A'_\beta(X_r), A'_t(X)$ and of degree $2n-2$, $n-1$, $n-1$, $n-2$ respectively and values $A'_\delta, A'_{\ell+1}, \ldots, A'_m$ such that

$$\log_G(A) = A'(x_r) + \delta_s A'_\delta + \alpha_r A'_\alpha(x_r) + \beta_r A'_\beta(x_r) + \tfrac{t(x)}{\delta_s} A'_t(x) + \tfrac{1}{\delta_s} \textstyle\sum_{i=\ell+1}^{m} A'_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)).$$

Similarly

$$\begin{aligned}\log_H(B) = {} & B'(x_r) + \delta_s B'_\delta + \alpha_r B'_\alpha(x_r) + \beta_r B'_\beta(x_r) + \tfrac{t(x)}{\delta_s} B'_t(x) + \tfrac{1}{\delta_s} \textstyle\sum_{i=\ell+1}^{m} B'_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) \\ & + \chi_{\alpha_r} B_{\chi_\alpha} + \chi_{\beta_r} B_{\chi_\beta} + \chi_{x_r} B_{\chi_x} + \chi_{\delta_s} B_{\chi_\delta}.\end{aligned}$$

and

$$\log_G(C) = C(x_r)' + \delta_s C'_\delta + \alpha_r C'_\alpha(x_r) + \beta_r C'_\beta(x_r) + \tfrac{t(x)}{\delta_s} C'_t(x) + \tfrac{1}{\delta_s} \textstyle\sum_{i=\ell+1}^{m} C'_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)).$$

First observe that by dividing each of the coefficients of $A'(X_r)$ by powers of $\prod_{j \neq r} x_j$, $\mathcal{A}$ can express $A'(X_r)$ as a polynomial $A(X)$ in $X = X_1 \ldots X_n$.

$$A'(x_r) = \sum_{i=0}^{2n-2} f_i x_r^i = \sum_{i=0}^{2n-1} \frac{f_i}{(\prod_{j \neq r} x_j)^i} x^i = A(x).$$

The adversary finds the polynomial $A_\alpha(X)$ such that $\alpha A_\alpha(x) = \alpha_r A'_\alpha(x_r)$ as follows

$$\alpha_r A'_\alpha(x_r) = \alpha_r \sum_{i=0}^{2n-2} f_i x_r^i = \alpha \sum_{i=0}^{2n-2} \frac{f_i}{(\prod_{j \neq r} \alpha_j)(\prod_{j \neq r} x_j)^i} x = \alpha A_\alpha(X).$$

Likewise we can find $A_\beta(X)$ such that $\beta A_\beta(x) = \beta_r A'_\beta(x_r)$. The adversary additionally sets

$$A_t(X) = \frac{1}{\prod_{j \neq s} \delta_j} A'_t(X), \quad A_\delta = \frac{1}{\prod_{j \neq s} \delta_j} A'_\delta, \quad A_i = \frac{1}{\prod_{j \neq s} \delta_j} A'_i \text{ for } \ell + 1 \leq i \leq m.$$

This means that by using the same manipulations for $\log_G B$ and $\log_G C$, the adversary knows coefficients such that

$$\log_G(A) = A(x) + \delta A_\delta + \alpha A_\alpha(x) + \beta A_\beta(x) + \tfrac{t(x)}{\delta} A_t(x) + \tfrac{1}{\delta} \textstyle\sum_{i=\ell+1}^{m} A_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)).$$

Similarly

$$\begin{aligned}\log_H(B) = {} & B(x) + \delta B_\delta + \alpha B_\alpha(x) + \beta B_\beta(x) + \tfrac{t(x)}{\delta} B_t(x) + \tfrac{1}{\delta} \textstyle\sum_{i=\ell+1}^{m} B_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) \\ & + \chi_\alpha B_{\chi_\alpha} + \chi_\beta B_{\chi_\beta} + \chi_x B_{\chi_x} + \chi_\delta B_{\chi_\delta}.\end{aligned}$$

and

$$\log_G(C) = C(x) + \delta C_\delta + \alpha C_\alpha(x) + \beta C_\beta(x) + \tfrac{t(x)}{\delta} C_t(x) + \tfrac{1}{\delta} \textstyle\sum_{i=\ell+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)).$$

We shall show that the values $a_{\ell+1}, \ldots, a_m$ used to calculate C are a witness for ϕ.

Verification Equation Constraints

Since $\mathcal{A}$'s instance verifies in can be parsed as $\phi = (a_0 = 1, a_1, \ldots, a_\ell) \in \mathbb{F}^{\ell+1}$. For the verifiers equation to be satisfied we must have that

$$e(A, B) = e(G^\alpha, G^\beta) e(G^{\sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}, G) e(C, G^\delta).$$

Taking logarithms, this implies

$$\log(A) \log(B) = \alpha\beta + \sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \delta \log C.$$

If we insert our equations for $\log(A)$, $\log(B)$, and $\log(C)$ we have

$$\begin{aligned}&\Big(A(x) + \delta A_\delta + \alpha A_\alpha(x) + \beta A_\beta(x) + \tfrac{t(x)}{\delta} A_t(x) + \tfrac{1}{\delta} \textstyle\sum_{i=\ell+1}^{m} A_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))\Big) \\ &\times \Big(B(x) + \delta B_\delta + \alpha B_\alpha(x) + \beta B_\beta(x) + \tfrac{t(x)}{\delta} B_t(x) + \tfrac{1}{\delta} \textstyle\sum_{i=\ell+1}^{m} B_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \chi_\alpha B_{\chi_\alpha} + \chi_\beta B_{\chi_\beta} + \chi_x B_{\chi_x} + \chi_\delta B_{\chi_\delta}\Big) \\ &= \alpha\beta + \textstyle\sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \\ &\delta C(x) + \delta^2 C_\delta + \alpha\delta C_\alpha(x) + \beta\delta C_\beta(x) + t(x) C_t(x) + \textstyle\sum_{i=\ell+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)).\end{aligned}$$

We instantly observe that χ_α, χ_β, χ_x and χ_δ must be 0.

7.2 Terms including δ^0

First we look at the constraints introduced by terms not including $\frac{1}{\delta}$ or $\frac{1}{\delta^2}$, i.e. terms with factors of α^2, $\alpha\beta$, and β^2. Matching terms with α^2 gives the constraint

$$A_\alpha(x)B_\alpha(x) = 0$$

meaning that either $A_\alpha(x) = 0$ or $B_\alpha(x) = 0$. Wlog set $B_\alpha(x) = 0$. Matching terms with $\alpha\beta$ gives the constraint

$$A_\alpha(x)B_\beta(x) + \cancel{A_\beta B_\alpha(x)} = 1$$

meaning that $B_\beta(x) = \frac{1}{A_\alpha(x)}$. Wlog set $A_\alpha(x) = 1$ (so $B_\beta(x) = 1$ as well) by cancelling through if necessary.[1] Matching terms with β^2 gives the constraint

$$A_\beta(x)B_\beta(x) = 0 \implies A_\beta(x) = 0.$$

Terms including δ^{-2}

Second we look at the constraints introduced by terms including $\frac{1}{\delta^2}$, i.e. terms with factors of $\frac{\alpha\beta}{\delta^2}$. Matching terms with $\alpha\beta$ gives the constraint

$$\left(\sum_{i=\ell+1}^{m} A_i u_i(x)\right)\left(\sum_{i=\ell+1}^{m} B_i v_i(x)\right) + \left(\sum_{i=\ell+1}^{m} A_i v_i(x)\right)\left(\sum_{i=\ell+1}^{m} B_i u_i(x)\right) = 0. \tag{1}$$

7.3 Terms including δ^{-1}

Third we look at the constraints introduced by terms including $\frac{1}{\delta}$ i.e. $\frac{\alpha^2}{\delta}$, $\frac{\beta^2}{\delta}$, $\frac{\alpha\beta}{\delta}$, $\frac{\alpha}{\delta}$, and $\frac{\beta}{\delta}$.

$$\begin{aligned}&(A(x)+\alpha)\left(t(x)B_t(x) + \textstyle\sum_{i=\ell+1}^{m} B_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))\right)\\ &+ (B(x)+\beta)\left(t(x)A_t(x) + \textstyle\sum_{i=\ell+1}^{m} A_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))\right) = 0.\end{aligned}$$

Matching terms with α^2 gives the constraint

$$\sum_{i=\ell+1}^{m} B_i v_i(x) = 0.$$

Matching terms with β^2 gives the constraint

$$\sum_{i=\ell+1}^{m} A_i u_i(x) = 0.$$

Constraint 1 now simplifies to

$$\begin{aligned}&\cancel{\left(\textstyle\sum_{i=\ell+1}^{m} A_i u_i(x)\right)\left(\sum_{i=\ell+1}^{m} B_i v_i(x)\right)} + \left(\textstyle\sum_{i=\ell+1}^{m} A_i v_i(x)\right)\left(\sum_{i=\ell+1}^{m} B_i u_i(x)\right) = 0\\ &\implies \left(\textstyle\sum_{i=\ell+1}^{m} A_i v_i(x)\right)\left(\sum_{i=\ell+1}^{m} B_i u_i(x)\right) = 0\end{aligned}$$

meaning that either

$$\left(\sum_{i=\ell+1}^{m} A_i v_i(x)\right) = 0 \text{ or } \sum_{i=\ell+1}^{m} B_i u_i(x) = 0. \tag{2}$$

Matching terms with $\alpha\beta$ gives the constraint

$$\sum_{i=\ell+1}^{m} A_i v_i(x) + \sum_{i=\ell+1}^{m} B_i u_i(x) = 0.$$

This combined with constraint 2 implies that both the sums equal zero.

$$\sum_{i=\ell+1}^{m} A_i v_i(x) = 0 \text{ and } \sum_{i=\ell+1}^{m} B_i u_i(x) = 0.$$

[1] If $\log A \times \log B = f(\alpha, \beta, \delta, x)$ then $\frac{\log A}{A_\alpha} \times A_\alpha \log B = f(\alpha, \beta, \delta, x)$. Thus the extractor can divide all the coefficients in $\log A$ by A_α and multiply all the coefficients in $\log B$ by A_α.

Matching terms with α gives the constraint

$$\begin{array}{l}\left(t(x)B_t(x) + \sum_{i=\ell+1}^{m} B_i w_i(x)\right) + \cancel{A(x)\sum_{i=\ell+1}^{m} B_i v_i(x)} + \cancel{B(x)\sum_{i=\ell+1}^{m} A_i v_i(x)} = 0 \\ \implies \\ t(x)B_t(x) + \sum_{i=\ell+1}^{m} B_i w_i(x) = 0.\end{array}$$

The polynomials $\{w_i(X)\}_{i=\ell+1}^{m}$ have maximum degree $n-1$. The polynomial $t(X)B_t(X)$ is either equal to 0 or it has minimum degree n. Thus

$$t(x)B_t(x) = 0$$

which in turn means that

$$\sum_{i=\ell+1}^{m} B_i w_i(x) = 0$$

.

Matching terms with β gives the constraint

$$\begin{array}{l}\left(t(x)A_t(x) + \sum_{i=\ell+1}^{m} A_i w_i(x)\right) + \cancel{B(x)\sum_{i=\ell+1}^{m} A_i u_i(x)} + \cancel{A(x)\sum_{i=\ell+1}^{m} B_i u_i(x)} = 0 \\ \implies \\ t(x)A_t(x) + \sum_{i=\ell+1}^{m} A_i w_i(x) = 0.\end{array}$$

Thus

$$t(x)A_t(x) = 0$$

which in turn means that

$$\sum_{i=\ell+1}^{m} A_i w_i(x) = 0.$$

Reduced Verification Constraints

The equation for $\log(A)$, $\log(B)$, and $\log(C)$ is now reduced to

$$\begin{array}{l}(\alpha + A(x) + \delta A_\delta)(\beta + B(x) + \delta B_\delta) - \sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) \\ = \\ \delta C(x) + \delta^2 C_\delta + \alpha\delta C_\alpha(x) + \beta\delta C_\beta(x) + t(x)C_t(x) + \sum_{i=\ell+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))\end{array}$$

Return to Terms Including δ^0

Fourth, we look at the constraints introduced by terms not including δ or δ^2, i.e. α, β, 1. Matching terms with α gives the constraint

$$B(x) = \sum_{i=0}^{m} a_i v_i(x).$$

Matching terms with β gives the constraint

$$A(x) = \sum_{i=0}^{m} a_i u_i(x).$$

Matching remaining terms with no positive powers of α or β or δ gives the constraint

$$\begin{array}{l}A(x)B(x) = t(x)C_t(x) + \sum_{i=0}^{m} a_i w_i(x) \\ \implies \\ \left(\sum_{i=0}^{m} a_i u_i(x)\right)\left(\sum_{i=0}^{m} a_i v_i(x)\right) = t(x)C_t(x) + \sum_{i=0}^{m} a_i w_i(x).\end{array}$$

If $\mathcal{X}$ is the extractor that returns $w = a_{\ell+1}, \ldots, a_m$ then the QAP with $h(X) = C_t(X)$ is satisfied. Hence w is a witness to the truth of the statement $\phi \in L_R$. This completes the proof. □

Acknowledgements

The research leading to these results has received funding from the Zcash Company. We would like to thank Ariel Gabizon for helpful discussions.

References

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pages 3–33, 2017.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *IACR Cryptology ePrint Archive*, 2017:1050, 2017.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79, 2012.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge snarks. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, pages 315–347, 2018.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. *IACR Cryptology ePrint Archive*, 2018:280, 2018.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.