A Verifiable Encryption Scheme for the Discrete Logarithm Relation with Applications to Distributed Key Generation

Hernán Darío Vanegas Madrigal

HashCloak Inc.
hernan@hashcloak.com

Abstract. We propose a non-interactive verifiable encryption scheme (NIVE) for the discrete logarithm relation using the ideas presented in [BHL24]. We also provide a proof of security for our NIVE according to the definition presented in [TZ24]. Finally, we present two protocols for distributed key generation (DKG): a protocol that generates unbiased keys in two rounds, and a protocol that generates biased keys in one round. For the second protocol, we show that although the resulting key pair may be biased, this does not allow the adversary to know about the discrete logarithm of the public key following the ideas from [Gen+07]. Both protocols utilize a bulletin-board functionality as a communication channel to abstract a setting where a set of parties publish and retrieve messages from a public, decentralized source, such as a blockchain. In that setting, we use a NIVE for the discrete logarithm relation to protect the published shares on the bulletin board and to guarantee the correctness of the shares.

Keywords: Non-Interactive Verifiable Encryption · Threshold Distributed Key Generation.

1 Introduction

Verifiable encryption (VE) schemes are protocols that prove that the encrypted plaintext satisfies a given relation or property. The concept was first proposed by [Sta96] and then generalized and formalized in [CD00]. In a VE scheme, a prover $\mathcal{P}$ can encrypt a witness w using a public key pk (which corresponding secret key sk is held by the receiver) and convince a third party verifier $\mathcal{V}$ that the secret plaintext w satisfies the relation R (which means that $R(x, w) = 1$), for some publicly known statement x. Intuitively, the VE scheme must fulfill three properties [TZ24]:

- Completeness: if $\mathcal{P}$, $\mathcal{V}$, and $\mathcal{R}$ are behave honestly, then $\mathcal{V}$ accepts after interacting with $\mathcal{P}$, and $\mathcal{R}$ obtains a w such that $R(x, w) = 1$ when it uses the secret key sk.
- Zero-knowledge: The verifier $\mathcal{V}$ learns nothing about the secret plaintext w.

– Validity: If $\mathcal{V}$ accepts after interacting with any adversarial prover $\mathcal{P}^*$, the receiver $\mathcal{R}$ is guaranteed to obtain a plaintext w such that $R(x, w) = 1$.

There is a particular case of VE schemes called *non-interactive verifiable encryption scheme* (NIVE) where the prover $\mathcal{P}$ does not require interacting with $\mathcal{V}$ to generate a proof. Instead, it can generate the proof locally and then send it to the verifier.

Verifiable encryption schemes can be useful in multiple applications. For example, VE schemes can be used in verifiable backup problems [TZ24]. Also, as mentioned in [CS03], verifiable encryption can be used in key escrow, optimistic fair exchange, publicly verifiable secret sharing and signature sharing [CD00], Universally composable commitments, confirmer signatures [CM00], and group signatures and anonymous credentials [CV02].

In this paper, we present a NIVE scheme for the discrete logarithm relation. The NIVE scheme relies on the ideas of [BHL24] in that the scheme uses a group pair where the co-DDH problem is hard. Then, we prove security of the proposed VE scheme according to the definition of a non-interactive encryption scheme presented in [TZ24]. Also, we present two distributed key generation (DKG) protocols using any secure NIVE for the discrete-logarithm relation. For the first protocol, we prove its security for a standard DKG ideal functionality using the Universal Composability (UC) framwework [Can01]. The first DKG has a communication complexity of two rounds, and it relies on a modified bulletin board with commitment ideal functionality [Gra+24] and a standard zero-knowledge proof of knowledge. The second protocol has a communication complexity of one round and uses a standard bulletin board functionality. This second protocol outputs biased key pairs; however, we prove that this bias does not allow the adversary to learn the secret key, as the bias is not significant from a security point of view. This kind of bias makes this protocol useful for some use cases.

Related work. There is some research on VE schemes [CD00; LN17; ASW98; Lee+19]. For the discrete logarithm relation, the work of [CS03] proposes a verifiable encryption scheme for discrete logarithms over a group where Paillier's decision composite residuosity problem is hard, and the public key encryption scheme is a variant of Paillier's scheme. The work of [Nic+20] constructs a VE scheme for discrete logarithms using an elliptic curve PRF proposed in the same work. The work of [TZ24] proposes a general framework that realizes VE protocols using zero-knowledge proof systems based on the MPC-in-the-head paradigm for any public-key encryption scheme. The resulting VEs guarantee the binding of the encryption when the ciphertext is used as a commitment. There is significant and extensive reseatch on distributed key generation for discrete logarithm cryptosystems [BK25], however, up to our knowledge, the only work that aligns with our goals appears in [Kat+23]. The difference is that we use the co-DDH problem which is a standard assumption rather than class groups and their specific-tailored hard problems, and we can achieve the security still without requiring range proofs as in prior works.

Organization. Section 2 presents some preliminaries necessary to understand the document's content. In Section 3, we present a specification of our NIVE and provide a proof of security. In Section 4, we present a distributed key generation (DKG) based on any NIVE for the discrete-logarithm relation with its proof of security, and we improve it to a one-round dkg in Section 5.

Acknowledgements. We want to thank the Serai DEX team and MAGIC Grants, in particular to Luke Parker who posited the initial protocol and NIVE specification, and Justin Ehrenhofer for motivating the execution of this project and bringing us the initial problem statement. They constantly gave us continuous help and discussion spaces for the project. We want to thank Yashvanth Kondi from Silence Laboratories, who kindly helped us review the DKG protocol and its proof, improve the protocol, and make some suggestions to correct some errors.

2 Preliminaries

Notation. For an integer $n \in \mathbb{N}$, we denote $[n] = \{1, \dots n\}$. We denote $\lambda \in \mathbb{N}$ as the public security parameter. We use the notation $x \overset{\$}{\leftarrow} S$ to denote that x was sampled from the set S independently and uniformly at random. Given a probability distribution $\mathcal{D}$, we use the notation $x \leftarrow \mathcal{D}$ to represent the sampling according to the distribution $\mathcal{D}$. We use the abbreviation PPT for "probabilistic polynomial-time" algorithms. We use the notation $\mathsf{negl}(\kappa)$ to represent a negligible function of the parameter κ. If X and Y are two random variables, we write $X \overset{\text{stat}}{\equiv} Y$ and $X \overset{\text{comp}}{\equiv} Y$ to denote that X and Y are statistically and computationally indistinguishable, respectively.

The NIVE proposed in this work relies on group pairs. Intuitively, the strategy consists of embedding one group into another, but both groups must have compatibility conditions on their domain and order, as we show in the following definition.

Definition 1 (Group pair ensemble - taken from [BHL24]). *We say that $(\mathcal{G}_T, \mathcal{G}_E)$ is a group pair ensemble if the following two requirements hold:*

- *$\mathcal{G}_T = \{(\mathbb{G}_{T,\lambda}, G_{T,\lambda}, p_\lambda)\}_{\lambda \in \mathbb{N}}$ and $\mathcal{G}_E = \{(\mathbb{G}_{E,\lambda}, G_{E,\lambda}, r_\lambda)\}_{\lambda \in \mathbb{N}}$ are group ensembles.*
- *For every $\lambda \in \mathbb{N}$, the group $\mathbb{G}_{E,\lambda}$ is a group of points on an elliptic curve defined over the field $\mathbb{Z}_{p_\lambda}$, where p_λ is the order of $\mathbb{G}_{T,\lambda}$.*

The security of the scheme relies on the co-Decisional Diffie-Hellman problem. This problem is formally stated in [BLS01] and is explicitly formalized in [GR04]. This problem is a generalization for two groups of the traditional DDH problem. We explicitly define this problem for a group pair following [BHL24, Definition 7].

Definition 2 (co-Decisional Diffie-Hellman problem [BHL24; GR04]). *We say that the co-DDH problem is hard with respect to a group-pair ensemble*

$(\mathcal{G}_T, \mathcal{G}_E)$, where $\mathcal{G}_T = \{(\mathbb{G}_{T,\lambda}, G_{T,\lambda}, p_\lambda)\}_{\lambda\in\mathbb{N}}$ and $\mathcal{G}_E = \{(\mathbb{G}_{E,\lambda}, G_{E,\lambda}, r_\lambda)\}_{\lambda\in\mathbb{N}}$, if for any PPT *non-uniform algorithm $\mathcal{D}$, there exists a negligible function such that*

$$\left|\Pr\left[\mathcal{D}\left(1^\lambda, k\cdot G_{T,\lambda}, X_{E,\lambda}, k\cdot X_{E,\lambda}\right)=1\right] - \Pr\left[\mathcal{D}\left(1^\lambda, k\cdot G_{T,\lambda}, X_{E,\lambda}, Y_{E,\lambda}\right)=1\right]\right| \leq \mathsf{negl}(\lambda)$$

for all λ, where the probability is taken over the choice of $X_{E,\lambda}, Y_{E,\lambda} \overset{\$}{\leftarrow} \mathbb{G}_{E,\lambda}$ and $k \overset{\$}{\leftarrow} \left[2^{\ell_\lambda+1}-1\right]$, for $\ell_\lambda = \lfloor \log_2 \min\{p_\lambda, r_\lambda\}\rfloor - 1$, and over the internal randomness of $\mathcal{D}$.

2.1 Verifiable encryption schemes

First, we present the definition of a non-interactive verifiable encryption scheme. This definition is taken from [TZ24, Appendix C.1].

Definition 3 (Non-interactive verifiable encryption scheme - NIVE). *Let R be a relation, and $\mathcal{L}_\mathsf{R} = \{x \mid \exists w, (x,w)\in R\}$. A secure non interactive verifiable encryption scheme for the relation R consists of a tuple of algorithms $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ as follows:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: Outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathcal{P}(\mathsf{pk}, x, w) \to \mathsf{tr}$*: A prover algorithm that outputs a transcript* tr*.*
- $\mathcal{V}(\mathsf{pk}, x, \mathsf{tr}) \to b \in \{0,1\}$*: A verifier algorithm that outputs a bit b indicating whether $\mathcal{V}$ accepts or rejects.*
- $\mathsf{Enc}(x, \mathsf{tr}) \to c$*: A compression algorithm that outputs a ciphertext c.*
- $\mathsf{Dec}(\mathsf{sk}, c) \to w$*: A recovery algorithm that outputs a plaintext w.*

The algorithms must satisfy the following properties:

Completeness: For all $(x,w)\in R$,

$$\Pr\left[b \neq 1 \ \vee\ (x,w') \notin R \ :\ \begin{array}{r} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ \mathsf{tr} \leftarrow \mathcal{P}(\mathsf{pk}, x, w), \\ b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr}), \\ c \leftarrow \mathsf{Enc}(x, \mathsf{tr}), \\ w' \leftarrow \mathsf{Dec}(\mathsf{sk}, c). \end{array}\right] \leq \mathsf{negl}(\lambda) \qquad (1)$$

Validity: For all pairs of PPT *adversaries $(\mathcal{A}^*, \mathcal{P}^*)$,*

$$\Pr\left[b = 1 \ \wedge\ (x,w') \notin R \ :\ \begin{array}{r} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}), \\ \mathsf{tr} \leftarrow \mathcal{P}^*(\mathsf{pk}, \mathsf{sk}, x), \\ b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr}), \\ c \leftarrow \mathsf{Enc}(x, \mathsf{tr}), \\ w' \leftarrow \mathsf{Dec}(\mathsf{sk}, c). \end{array}\right] \leq \mathsf{negl}(\lambda) \qquad (2)$$

Computational zero-knowledge: There exists a simulator $\mathcal{S}$ such that for all PPT *distinguishers and all $(x, w) \in R$,*

$$\left| \Pr \left[i = i' \ : \ \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ \mathsf{tr}_0 \leftarrow \mathcal{P}(\mathsf{pk}, x, w), \\ \mathsf{tr}_1 \leftarrow \mathcal{S}(\mathsf{pk}, x), \\ i' \leftarrow \mathcal{D}(\mathsf{pk}, x, \mathsf{tr}_i), \\ i \xleftarrow{\$} \{0,1\} \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda) \tag{3}$$

2.2 Zero-knowledge proofs

For the instantiation of the proposed NIVE, we need non-interactive ZK-proof systems for the following relations:

– An extended discrete logarithm relation:

$$\begin{aligned} R = \{(A, B, K, H, G_E, G_T, k'; a, b) \mid A &= a \cdot G_E, \\ B &= b \cdot G_E, \\ K &= (k' \cdot (a \cdot H)_x + (b \cdot H)_x) \cdot G_T\}; \end{aligned} \tag{4}$$

– A traditional discrete logarithm relation:

$$R_{\mathsf{DL}} = \{(X, G_E; x) \in \mathbb{G}_E \times \mathbb{G}_E \times \mathbb{Z}_p \mid X = x \cdot G_E\}; \tag{5}$$

– For the security of the DKG protocol, we need to guarantee that the public key used during the computation of the proof in the NIVE is consistent with the secret key generated in the initialization phase. Let $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ be a NIVE. Let us define a relation as $R_{\mathsf{key}} = \big\{(\mathsf{pk}, (\mathsf{sk}, r)) \mid (\mathsf{pk}, \mathsf{sk}) = \mathsf{Gen}(1^\lambda; r)\big\}$.

It may be possible that the relation R could be instantiated using the proof system proposed by [BHL24, Section 6.2] using Bulletproofs expressing the relation R as a R1CS with the statement in the exponent. Such representation is not presented in this work, and for the rest of the work, we assume that there is a ZK-proof system for this relation. The relation R_{DL} can be instantiated using a Schnorr's discrete logarithm proof system.

3 Verifiable encryption scheme

We present the non-interactive verifiable encryption scheme for the relation R_{DL} in Scheme 1. The scheme is based on the ideas of the DDH-based verifiable random function (eVRF) presented in [BHL24] by using the group pair strategy. As in the mentioned citation, we used the Leftover Hash Lemma to avoid a bias in the encryptions. If the trick is not applied, the range of the ciphertext would be roughly half of $\mathbb{Z}_p$. To avoid this bias, we follow the idea of [BHL24] by using an extractor map $ext : \mathbb{Z}_p \times \mathbb{Z}_p^2 \to \mathbb{Z}_p$, such that $ext(k', (x_a, x_b)) = k' \cdot x_a + x_b$. If $\mathcal{S} \in \mathbb{Z}_p$ is the set of x-coordinates of points in $\mathbb{G}_T$, then, assuming that $(x_a, x_b) \xleftarrow{\$} \mathcal{S} \times \mathcal{S}$ and $k' \xleftarrow{\$} \mathbb{Z}_p$, then $(k, k' \cdot x_a + x_b)$ is statistically close to the uniform distribution over $\mathbb{Z}_p^2$ with negligible statistical distance over the security parameter.

Scheme 1: Verifiable DDH-based encryption scheme

Let $\mathbb{G}_T$ be an elliptic curve over $\mathbb{Z}_q$ with scalar field over $\mathbb{Z}_p$ and generator G_T, and $\mathbb{G}_E$ be an elliptic curve over $\mathbb{Z}_p$ with scalar field over $\mathbb{Z}_r$ and generator G_E.

- $\mathsf{Gen}(1^\lambda)$: Sample $h \xleftarrow{\$} \mathbb{Z}_r^*$ and compute $H \overset{\text{def}}{=} h \cdot G_E$. Generate a NIZK-PoK for the relation R_{DL} as $\pi_H \xleftarrow{\$} \mathcal{P}_{\mathsf{DL}}(H, G_E; h)$. Also, sample $k' \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{pk} \overset{\text{def}}{=} (H, \pi_H, k')$ and $\mathsf{sk} \overset{\text{def}}{=} (h, k')$.
- $\mathcal{P}(\mathsf{pk}, x, w \in \mathbb{Z}_p)$: The prover $\mathcal{P}$ parses $\mathsf{pk} = (H, \cdot, k')$. Choose a uniform $a, b \xleftarrow{\$} \mathbb{Z}_r^*$, and compute $A \overset{\text{def}}{=} a \cdot G_E$ and $B \overset{\text{def}}{=} b \cdot G_E$. Let $a \cdot H = (x_a, \cdot)$ and $b \cdot H = (x_b, \cdot)$. Define $k \overset{\text{def}}{=} k' \cdot x_a + x_b \in \mathbb{Z}_p$ and let $K \overset{\text{def}}{=} k \cdot G_T$. Let $s' \overset{\text{def}}{=} w + k$. Use the prover $\mathcal{P}_R$ to generate a NIZK proof for R as $\pi \xleftarrow{\$} \mathcal{P}_R(A, B, K, H, k'; a, b)$. Output $\mathsf{tr} \overset{\text{def}}{=} (\pi, A, B, K, s')$.
- $\mathcal{V}(\mathsf{pk}, x, \mathsf{tr})$: Parse $\mathsf{tr} = (\pi, A, B, K, s')$, $\mathsf{pk} = (H, \pi_H, k')$. The algorithm outputs $b = 1$ if (1) $A, B \in C_E$ and $K \in C_T$, (2) π is a valid proof for (A, B, K, H) with respect to the relation R using $\mathcal{V}_R$, (3) checks that π_H is a valid proof for (H, G_E) with respect to relation R_{DL} using $\mathcal{V}_{\mathsf{DL}}$, and (4) checks that $s' \cdot G_T = x + K$. Otherwise, the algorithm outputs $b = 0$.
- $\mathsf{Enc}(x, \mathsf{tr})$: Parse $\mathsf{tr} = (\pi, A, B, K, s')$. Output $c \overset{\text{def}}{=} (s', A, B)$.
- $\mathsf{Dec}(\mathsf{sk}, c)$: Parse $c = (s', A, B)$ and $\mathsf{sk} = (h, k')$. Check that $A, B \in C_E$, otherwise, output $\perp$. Output $w = s' - (k' \cdot (h \cdot A)_x + (h \cdot B)_x)$

3.1 Security

To prove the security of Scheme 1, we need to prove correctness, validity, and computational zero-knowledge properties presented in Definition 3. Specifically, we will prove the following theorem.

Theorem 1. *Let $(\mathcal{P}_{\mathsf{DL}}, \mathcal{V}_{\mathsf{DL}})$ be a non-interactive zero-knowledge proof system for the relation R_{DL}, and $(\mathcal{P}_R, \mathcal{V}_R)$ be a non-interactive zero-knowledge proof system for the relation R. Let $\mathbb{G}_T$ and $\mathbb{G}_E$ be a group pair of points of elliptic curves where co-DDH is hard. Then the scheme $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ described in Scheme 1 is a secure verifiable encryption scheme as in Definition 3 for the relation R_{DL}.*

Proof. We present a proof of each property presented Definition 3.

Completeness. To prove completeness, let $(x, w) \in R_{\mathsf{DL}}$. This means that $x = w \cdot G_T$. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{tr} \leftarrow \mathcal{P}(\mathsf{pk}, x, w)$, $b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr})$, $c \leftarrow \mathsf{Enc}(x, \mathsf{tr})$,

$w' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$, such that $\mathsf{pk} = (H, \pi_H, k')$, $\mathsf{sk} = (h, k')$, and $\mathsf{tr} = (\pi, A, B, K, s')$, following the execution of each algorithm.

It is immediate that π_H is a valid proof for $(H, G_E; h)$ for the relation R. Then $K = k \cdot G_T = (k' \cdot x_a + x_b) \cdot G_T = [k' \cdot (a \cdot H)_x + (b \cdot H)_x] \cdot G_T$. Hence, $(A, B, K, H, G_E, G_T; a, b, k') \in R$. Also, $s' \cdot G_T = (w+k) \cdot G_T = w \cdot G_T + k \cdot G_T = x + K$. In that way, we have shown that $b = 1$. On the other hand, if $c = (s', A, B)$, we have $w' = s' - (k' \cdot (h \cdot A)_x + (h \cdot B)_x)$. From the algorithms $\mathcal{P}$ and Enc, we have $s' = w + k = w + k' x_a + x_b$. Hence, $w' = w + k' x_a + x_b - k'(ha \cdot G_E)_x - k'(hb \cdot G_E)_x = w$. Therefore, $(x, w') \in R_{\mathsf{DL}}$. In conclusion, the previous arguments prove that $\Pr\left[b = 1 \;\wedge\; (x, w') \in R_{\mathsf{DL}}\right] = 1$, hence $\Pr\left[b \neq 1 \;\vee\; (x, w') \notin R_{\mathsf{DL}}\right] = 0$, which shows that the verifiable encryption scheme is complete.

Validity. For the proof of validity, we will assume the existence of adversaries whose success probability is overwhelming while attacking the validity of the NIVE, and we will construct an adversary attacking the soundness of the NIZK for the relation R (i.e. we are reducing the validity of the NIVE to the soundness of the NIZK for the relation R).

Assume that there exists $(\mathcal{A}^*, \mathcal{P}^*)$ such that

$$\Pr\left[b = 1 \;\wedge\; (x, w') \notin R \;:\; \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk}), \\ \mathsf{tr} \leftarrow \mathcal{P}^*(\mathsf{pk}, \mathsf{sk}, x), \\ b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr}), \\ c \leftarrow \mathsf{Enc}(x, \mathsf{tr}), \\ w' \leftarrow \mathsf{Dec}(\mathsf{sk}, c). \end{array} \right] > 1 - \mathsf{negl}(\lambda). \tag{6}$$

Let $\mathcal{A}$ be defined as presented next.

$\underline{\mathcal{A}(1^\lambda):}$ Perform the following steps:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- $x \leftarrow \mathcal{A}^*(\mathsf{pk}, \mathsf{sk})$.
- $(\pi, A, B, K, s') \leftarrow \mathcal{P}^*(\mathsf{pk}, \mathsf{sk}, x)$.
- Parse $\mathsf{pk} = (H, \pi_H, k')$.
- Output (A, B, K, H, k', π).

Let $\mathsf{tr} = (\pi, A, B, K, s')$, $b \leftarrow \mathcal{V}(\mathsf{pk}, x, \mathsf{tr})$, $c \leftarrow \mathsf{Enc}(x, \mathsf{tr})$, and $w' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. If $b = 1$, then, it holds that:

- $\mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1$.
- $\mathcal{V}_{\mathsf{DL}}(H, G_E, \pi_H) = 1$.
- $s' \cdot G_T = x + K$.

Notice that pk and sk were generated by the Gen algorithm, so $H = h \cdot G_T$. Combining all of these conclusions, we have that

$$\Pr\left[\begin{array}{r} b = 1 \;\wedge \\ (x, w') \notin R_{\mathsf{DL}} \end{array} \right] = \Pr\left[\begin{array}{r} \mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1 \;\wedge \\ s' \cdot G_T = x + K \;\wedge \\ (x, w') \notin R_{\mathsf{DL}} \end{array} \middle|\; H = h \cdot G_E \right].$$

Now, we have $w' = s' - [k'(h \cdot A)_x + (h \cdot B)_x]$, and hence $s' = w' + [k'(h \cdot A)_x + (h \cdot B)_x]$. If $s' \cdot G_T = x + K$ and $(x, w') \notin R_{\mathsf{DL}}$, so $w' \cdot G_T + [k'(h \cdot A)_x + (h \cdot B)_x] \cdot G_T = x + K$, which means that $[k'(h \cdot A)_x + (h \cdot B)_x] \cdot G_T \neq K$. Hence

$$\Pr\left[\begin{array}{r} \mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1 \wedge \\ s' \cdot G_T = x + K \wedge \\ (x, w') \notin R_{\mathsf{DL}} \end{array} \middle| \, H = h \cdot G_E \right] \leq \Pr\left[\begin{array}{r} \mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1 \wedge \\ (A, B, K, H, G_R, G_T, k) \notin \mathcal{L}_R \wedge \\ H = h \cdot G_E \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{r} \mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1 \wedge \\ (A, B, K, H, G_R, G_T, k) \notin \mathcal{L}_R \end{array}\right].$$

Finally, this shows that

$$\Pr\left[\begin{array}{r} \mathcal{V}_R(A, B, K, H, G_E, G_T, k', \pi) = 1 \wedge \\ (A, B, K, H, G_R, G_T, k) \notin \mathcal{L}_R \end{array}\right] > 1 - \mathsf{negl}(\lambda), \tag{7}$$

proving that $\mathcal{A}$ is a successful adversary for the soundness of $(\mathcal{P}_R, \mathcal{V}_R)$. This shows that the validity property holds for Scheme 1.

Computational zero-knowledge. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ such that $\mathsf{pk} = (H, \pi_H, k')$. Let us define the simulator $\mathcal{S}$ required by the definition.

$\underline{\mathcal{S}(\mathsf{pk}, x)}$ Perform the following steps:

- Sample $a, b \xleftarrow{\$} \mathbb{Z}_r^*$.
- Let $A = a \cdot G_E$ and $B = b \cdot G_E$.
- Let $a \cdot H = (x_a, \cdot)$ and $b \cdot H = (x_b, \cdot)$.
- Let $k = k' \cdot x_a + x_b$.
- Let $K = k \cdot G_T$.
- Sample $r \xleftarrow{\$} \mathbb{Z}_p$.
- Simulate the NIZK proof for R to obtain the transcript π.
- Output (π, A, B, K, r)

The proof proceeds to show that the output of $\mathcal{S}$ is computationally indistinguishable from the output trace of a real execution of the algorithm $\mathcal{P}$. Notice that the output $\mathcal{S}(\mathsf{pk}, x)$ is of the form (π, A, B, K, r) where $r \xleftarrow{\$} \mathbb{Z}_p$, and the output of a real execution of the prover is of the form $(\pi, A, B, K, w + k)$, where $k = k'x_a + x_b$. Hence, the proof is reduced to proving that $w + k$ is indistinguishable from the uniform distribution on $\mathbb{Z}_p$.

To prove that $w + k$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_p$, we will prove that (k', k) is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_p^2$. To do this, we will follow the same approach as in [BHL24] by using the Leftover Hash Lemma (see [ILL89]). First, notice that $h, a, b \xleftarrow{\$} \mathbb{Z}_r$, hence $a \cdot H$ and $b \cdot H$ are random elements in C_E^*. Let $S = \{x \in \mathbb{Z}_p \mid (x, y) \in C_E^*\}$. We proceed to use the Leftover Hash Lemma for $\mathsf{ext} : \mathbb{Z}_p \times \mathbb{Z}_p^2 \to \mathbb{Z}_p$, such that $\mathsf{ext}(k', (x_1, x_2)) = k'x_1 + x_2$. We have that $|C_E^*| \geq p - 2\sqrt{p}$, hence, $|S| \geq (p - 2\sqrt{p})/2$. The size of S allows you to compute the guessing probability of $(x_a, x_b) \in S^2$ as $\gamma = \frac{1}{|S|^2} \leq \frac{4}{p(\sqrt{p}-2)^2}$. It follows that the statistical distance between (k', k) and the uniform distribution over $\mathbb{Z}_p^2$ is $\Delta \leq \frac{1}{2}\sqrt{\gamma p} \leq \frac{1}{\sqrt{p}-2} \leq \mathsf{negl}(\lambda)$.

The previous argument proves that k is indistinguishable from the random distribution over $\mathbb{Z}_p$. Hence, if $r \overset{\$}{\leftarrow} \mathbb{Z}_p$, then $r \overset{\text{stat}}{\equiv} w + r \overset{\text{stat}}{\equiv} w + k$, which proves that $w + k$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_p$, and finally, this proves that the trace outputted by the simulator $\mathcal{S}$ is statistically indistinguishable from the trace output by a real execution of $\mathcal{P}$, which proves the computational zero-knowledge property.

4 Two-Round DKG From Verifiable Encryption

To prove the security of the proposed DKG protocol, we will use the stand-alone simulation technique presented in [Can00], but given that our proof has a straight-line black-box simulator, we can conclude that the protocol is UC-secure [KLR06]. In this technique, we first need to define an ideal functionality that captures the desired security and the expected result of our protocol. Then, to show the security of our protocol, we compare our real-world protocol with this ideal-world functionality using the simulation technique. The proof allows us to show that our protocol is as secure as the ideal functionality. All the ideal functionalities presented in this paper are adversarially delayed, which means that the ideal functionality will notify the adversary that the computation is ready to be sent to the honest parties; then, the adversay may choose which honest party will receive the output of the ideal functionality and the moment to deliver the output. The ideal functionality that we will consider for DKG is presented in $\mathcal{F}_{\mathsf{DKG},\mathcal{Q}}^{n,t}$ in Appendix B, and is taken from [BHL24].

4.1 Auxiliary Ideal Functionalities

First, we present the auxiliary functionality for the bulletin board based on [Gra+24] denoted $\mathcal{F}_{\mathsf{ComBB}}^{n}$. A bulletin board can be considered as a broadcast channel with memory in which the parties can post messages and other parties can retrieve these messages later while guaranteeing their consistency. The functionality $\mathcal{F}_{\mathsf{ComBB}}^{n}$ presented here has an additional modification with respect to the one presented in [Gra+24]: we added a commitment message so that when a party P_i writes something in the bulletin board, the functionality notifies to all parties that P_i sent a value to the functionality; then, when a party asks for the message sent by P_i, the functionality delivers the published message only if all messages from the parties have been commited. This feature is necessary in the DKG protocol to achieve security in presence of a rushing adversary, so that the parties first wait for the commitment notification before reading the message from the bulletin board. In this way, an adversary cannot choose the content of the message that he wants to publish based on previously published messages from honest parties.

Functionality 1: $\mathcal{F}^n_{\mathsf{ComBB}}$

- Initialize:
  - On first input $(\mathsf{init}, \mathsf{sid})$, initialize the set $C = \varnothing$ and the flag $\mathsf{com} = \mathsf{false}$.
  - On subsequent inputs $(\mathsf{init}, *)$, ignore the message.
- On input $(\mathsf{write}, \mathsf{sid}, \mathsf{id}, \mathsf{msg})$ from party P_i for which $(\mathsf{sid}, \mathsf{id}, i, \cdot)$ has not stored yet:
  - Store $(\mathsf{sid}, \mathsf{id}, i, \mathsf{msg})$, and for every party P_j, send the message $(\mathsf{commit}, \mathsf{id}, i)$. Also, replace $C \leftarrow C \cup \{i\}$.
  - If $C = [n]$, set $\mathsf{com} = \mathsf{true}$ and send $(\mathsf{sid}, \mathsf{complete})$ to all parties.
- On input $(\mathsf{sid}, \mathsf{readAll})$ from party P_i: if $\mathsf{com} = \mathsf{true}$, send all the messages stored internally $(\mathsf{sid}, \mathsf{id}, j, \mathsf{msg})$ to party P_i, otherwise send $(\mathsf{sid}, \perp)$.
- On input $(\mathsf{sid}, \mathsf{read}, \mathsf{id}, j)$ from party P_i: if $\mathsf{com} = \mathsf{true}$, send $(\mathsf{sid}, \mathsf{id}, i, \mathsf{msg})$ to party P_i.

In $\mathcal{F}^R_{\mathsf{zk}}$, we present the functionality for zero-knowledge proof of knowledge for arbitrary NP relations. This functionality will be used specifically with the relation R_{key}.

Functionality 2: $\mathcal{F}^R_{\mathsf{zk}}$

On input $(\mathsf{prove}, i, j, x, w)$ from party P_i, send $(\mathsf{prove}, i, j, x, 1)$ to P_j if $(x, w) \in R$, otherwise it sends $(\mathsf{prove}, i, j, x, 0)$.

4.2 Protocol for DKG

In this section, we present out protocol for DKG using an arbitrary NIVE for the discrete-logarithm relation. The protocol is presented in $\Pi^{n,t}_{\mathsf{DKG},\mathcal{Q}}$.

Protocol 2: $\Pi^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ – Based on [BHL24]

Let $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ be a NIVE for the discrete-logarithm relation for an elliptic curve group $\mathbb{G}$ with generator G and scalar field $\mathbb{Z}_p$.

- **Initialize – for all n parties**:
  1. Sample randomness $r_i \xleftarrow{\$} \{0,1\}^*$ uniformly.
  2. Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i)$.
  3. Send $(\mathsf{prove}, i, j, \mathsf{pk}_i, (\mathsf{sk}_i, r_i))$ as input to $\mathcal{F}^{R_{\mathsf{key}}}_{\mathsf{zk}}$ for $j \in [n]$.
  4. For $j \in [n]$, receive $(\mathsf{prove}, i, j, \mathsf{pk}_j, b_j)$. If $b_j = 0$ for some $j \in [n]$, the party P_i aborts and reports P_j as the culprit.
  5. Sample $a_i^\ell \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random, for $\ell = 0, \ldots, t$.
  6. Compute $A_i^\ell = a_i^\ell \cdot G$, for $\ell = 0, \ldots, t$.
  7. Send $(\mathsf{init}, \mathsf{sid})$ to the functionality $\mathcal{F}^n_{\mathsf{ComBB}}$.
- **Generate – for a quorum $\mathcal{Q}$ of $t+1$ parties:** Upon input $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q})$ whith a quorum $\mathcal{Q} \subseteq [n]$, each party P_i with $i \in \mathcal{Q}$:

1. Let $p_i(x) = \sum_{i=0}^{t} a_i^{\ell} \cdot x^{\ell}$.
2. Compute $k_{i\to j} = p_i(\alpha_j)$, for $j \in \mathbf{P}$.
3. Compute the encryption of the share $k_{i\to j}$, for $j \in \mathbf{P}$:
   (a) Compute $K_{i\to j} = \sum_{\ell=0}^{t} (\alpha_j)^{\ell} \cdot A_i^{\ell} = p_i(\alpha_j) \cdot G$.
   (b) Compute $\mathsf{tr}_{i\to j} \leftarrow \mathcal{P}(\mathsf{pk}_j, K_{i\to j}, k_{i\to j})$.
4. Send $(\mathsf{write}, \mathsf{pubinfo}_{i\to j} \| \mathsf{nonce} \| \mathcal{Q}, (\mathsf{tr}_{i\to j}, K_{i\to j}, \{A_i^{\ell}\}_{\ell=0}^{t}))$ to $\mathcal{F}_{\mathsf{ComBB}}^{n}$.

– **Output – for all n parties:**

1. For every $i \in \mathcal{Q}$ and $j \in \mathbf{P}$, wait to receive $(\mathsf{commit}, \mathsf{pubinfo}_{i\to j} \| \mathsf{nonce} \| \mathcal{Q}, i)$ and $(\mathsf{sid}, \mathsf{complete})$ from $\mathcal{F}_{\mathsf{ComBB}}^{n}$.
2. Input $(\mathsf{read}, \mathsf{pubinfo}_{j\to i} \| \mathsf{nonce} \| \mathcal{Q}, j)$ to $\mathcal{F}_{\mathsf{ComBB}}^{n}$, for all $j \in \mathcal{Q}$.
3. Wait to receive $(\mathsf{pubinfo}_{j\to i} \| \mathsf{nonce} \| \mathcal{Q}, j, (\mathsf{tr}_{j\to i}, K_{j\to i}, \{A_j^{\ell}\}_{\ell=0}^{t}))$ from $\mathcal{F}_{\mathsf{ComBB}}^{n}$, for all $j \in \mathcal{Q}$.
4. For all $j \in \mathcal{Q}$, check that $\mathcal{V}(\mathsf{pk}_i, K_{j\to i}, \mathsf{tr}_{j\to i}) \overset{?}{=} 1$, otherwise the party aborts and reports P_j as the culprit.
5. For all $j \in \mathcal{Q}$, check that $K_{j\to i} = \sum_{\ell=0}^{t} \alpha_i^{\ell} \cdot A_j^{\ell}$, otherwise the party aborts and reports P_j as the culprit.
6. Compute $k_{j\to i} = \mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(K_{j\to i}, \mathsf{tr}_{j\to i}))$.
7. Compute $k_i = \sum_{j\in\mathcal{Q}} k_{j\to i}$.
8. Compute $A_{\ell} = \sum_{j\in\mathcal{Q}} A_j^{\ell}$, for $\ell = 0, 1, \ldots, t$.
9. Compute $K = A_0$.
10. Output $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q}, k_i, \{A_{\ell}\}_{\ell=0}^{t}, K)$.

Theorem 2. *Let $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ be a NIVE for the discrete-logarithm relation for an elliptic curve group $\mathbb{G}$ with generator G and scalar field $\mathbb{Z}_p$. Protocol $\Pi_{\mathsf{DKG},\mathcal{Q}}^{n,t}$ securely implements $\mathcal{F}_{\mathsf{DKG},\mathcal{Q}}^{n,t}$ with computational security with abort in the $(\mathcal{F}_{\mathsf{ComBB}}^{n}, \mathcal{F}_{\mathsf{zk}}^{R_{\mathsf{key}}})$-hybrid model, in the presence of a static malicious adversary corrupting up to t parties.*

The proof of the above theorem follows the same pattern as [BHL24], and it is presented in Appendix A.

5 Robust One-Round Biased DKG from eVRF and Verifiable Encryption

In this section, we will present another version for the DKG protocol to reduce the number of rounds and make it robust at the cost of obtaining biased key pairs. This goal will be achieved by using the $\mathcal{F}_{\mathsf{eVRF}}$ functionality proposed in [BHL24] along with $\mathcal{F}_{\mathsf{BB}}$ as we did in the previous section. In this version of the protocol, we will not use the quorum approach as it was used in Section 5.

First, let us present the $\mathcal{F}_{\mathsf{eVRF}}$ functionality taken from [BHL24].

Functionality 3: $\mathcal{F}_{\mathsf{eVRF}}$ – Based on [BHL24]

Let $\mathcal{X}$ be a finite set and $\mathcal{Y}$ a set that defines a group of order q with generator G.

1. Upon receiving a message $(\mathsf{init}, i, *)$ from some P_i:
   (a) If P_i is honest, and the message is (init, i), then receive a value sid from the ideal adversary, verify that it is unique and store (sid, i).
   (b) If P_i is corrupted and the message is $(\mathsf{init}, i, \mathsf{sid}, f)$, where $f : \mathcal{X} \to \mathbb{Z}_q$ is the description of a deterministic polynomial-time computable function, and sid has not ben stored, then store (sid, i, f).
   (c) Send $(\mathsf{init}, i, \mathsf{sid})$ to all parties.
2. Upon receiving $(\mathsf{eval}, i, \mathsf{sid}, x)$ from P_i, where $x \in \mathcal{X}$:
   (a) If (sid, i) or (sid, i, f) is not stored, then ignore the message.
   (b) If P_i is honest:
      i. If there does not exist a stored tuple $(\mathsf{sid}, i, x, y, Y)$, with $x \in \mathcal{X}$, then choose $y \xleftarrow{\$} \mathbb{Z}_q$, compute $Y = y \cdot G$, and store $(\mathsf{sid}, i, x, y, Y)$.
      ii. Send $(\mathsf{eval}, i, \mathsf{sid}, x, y, Y)$ to party P_i and $(\mathsf{eval}, i, \mathsf{sid}, x, Y)$ to all parties.
   (c) If P_i is corrupted, then compute $Y = f(x) \cdot G$ and send $(\mathsf{eval}, i, \mathsf{sid}, x, Y)$ to all parties.

Also, we abstract a broadcast channel using the bulletin board functionality, $\mathcal{F}_{\mathsf{BB}}$, presented in works like [BDO14; Gra+24]. The bulletin board acts as a functionality that stores messages and delivers previously stored messages when a party sends a request query. The data on the bulletin board can be accessed by any party (honest and corrupt). Contrary to $\mathcal{F}^n_{\mathsf{ComBB}}$ which contains a commitment capability, the functionality $\mathcal{F}_{\mathsf{BB}}$ is a standard bulletin board functionality.

Functionality 4: $\mathcal{F}_{\mathsf{BB}}$ – Based on [BDO14; Gra+24]

1. On input $(\mathsf{write}, \mathsf{id}, i, \mathsf{msg})$ from P_i where id was not assigned yet, store $(\mathsf{id}, i, \mathsf{msg})$.
2. On input $(\mathsf{read}, \mathsf{id})$, from P_i, the functinality checks whether id was assigned already. If so, it returns $(\mathsf{id}, j, \mathsf{msg})$ to P_i. Otherwise, it returns $(\mathsf{id}, \perp, \perp)$.

The protocol presented in $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ is a robust version of the threshold DKG protocol from [BHL24] where we add the bulletin board functionality and a NIVE to protect the messages published in the bulletin board. Unfortunately, this protocol suffers from the same vulnerability as Pedersen's DKG protocol presented in [Ped92], which uses Feldman's verifiable secret sharing. In [Gen+07], the authors show an attack in which the adversary can bias the result of the protocol by choosing which corrupted parties contribute to the output of the protocol based on the contributions from the honest parties to the public key. As a result, the protocol may output a biased result, which makes the resulting key pair deviate from the uniform distribution. In standard DKG, the uniformity of the resulting key pair is essential for its use in other protocols, as those protocols assume the uniform distribution of the pair in their proofs. However, in [Gen+07], the authors show that the protocol may be strong for certain applications, as the output of the protocol could be biased, but the adversary can not compute the discrete logarithm of the output of the protocol, as the bias is not significant. For more details of the proof, we refer the reader to [Gen+07]. In this paper, we prove that the protocol $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ has the same property as Pedersen's DKG by adapting the proof presented in [Gen+07] to our case.

Protocol 3: $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ – Based on [BHL24]

Let $(\mathsf{Gen}, \mathcal{P}, \mathcal{V}, \mathsf{Enc}, \mathsf{Dec})$ be a NIVE for the discrete-logarithm relation for an elliptic curve group $\mathbb{G}$ with generator G and scalar field $\mathbb{Z}_p$. Assume that $n \geq t+1$ and that the adversary can corrupt up to t parties.

- **Initialize – for all n parties**:
  1. Send (init, i) to $\mathcal{F}_{\mathsf{eVRF}}$. If the party is corrupted, it may send $(\mathsf{init}, i, \mathsf{sid}, f)$ to $\mathcal{F}_{\mathsf{eVRF}}$.
  2. Wait to receive $(\mathsf{init}, j, \mathsf{sid}_j)$ from $\mathcal{F}_{\mathsf{eVRF}}$, for all $j \in \mathbf{P}$.
  3. Sample randomness $r_i \xleftarrow{\$} \{0,1\}^*$ uniformly.
  4. Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i)$.
  5. Send $(\mathsf{prove}, i, j, \mathsf{pk}_i, (\mathsf{sk}_i, r_i))$ as input to $\mathcal{F}^{R_{\mathsf{key}}}_{\mathsf{zk}}$ for $j \in [n]$.
  6. For $j \in [n]$, receive $(\mathsf{prove}, i, j, \mathsf{pk}_j, b_j)$. If $b_j = 0$ for some $j \in [n]$, the party P_i aborts and reports P_j as the culprit.
- **Generate** Upon input $(\mathsf{gen}, \mathsf{nonce})$, each party P_i with $i \in \mathbf{P}$ performs the following steps:
  1. For $\ell \in \{0, \ldots, t\}$, send $(\mathsf{eval}, i, \mathsf{sid}, \mathsf{nonce} \| \ell)$ to $\mathcal{F}_{\mathsf{eVRF}}$ and receive back $(\mathsf{eval}, i, \mathsf{sid}, \mathsf{nonce} \| \ell, a_i^\ell, A_i^\ell)$
  2. Let $p_i(x) = \sum_{i=0}^{t} a_i^\ell \cdot x^\ell$.
  3. Compute $k_{i \to j} = p_i(\alpha_j)$, for $j \in \mathbf{P}$.
  4. Compute the encryption of the share $k_{i \to j}$, for $j \in \mathbf{P}$:
     (a) Compute $K_{i \to j} = \sum_{\ell=0}^{t} (\alpha_j)^\ell \cdot A_i^\ell$.
     (b) Compute $\mathsf{tr}_{i \to j} \leftarrow \mathcal{P}(\mathsf{pk}_j, K_{i \to j}, k_{i \to j})$.
  5. Send $(\mathsf{write}, \mathsf{pubinfo}_{i \to j} \| \mathsf{nonce}, \{\mathsf{tr}_{i \to j}\}_{j \in \mathbf{P}})$ to $\mathcal{F}_{\mathsf{BB}}$.
- **Output – for all n parties:**
  1. Wait to receive $\{(\mathsf{eval}, j, \mathsf{sid}, \mathsf{nonce} \| \ell, A_j^\ell)\}_{\ell=0}^{t}$ from $\mathcal{F}_{\mathsf{eVRF}}$ for all parties P_j with $j \in \mathbf{P}$.
  2. For $j \in \mathbf{P}$, let $K_{j \to i} = \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_j^\ell$.
  3. For $j \in \mathbf{P}$, send $(\mathsf{read}, \mathsf{pubinfo}_{j \to i} \| \mathsf{nonce})$ to $\mathcal{F}_{\mathsf{BB}}$.
  4. Wait to receive $(\mathsf{pubinfo}_{j \to i} \| \mathsf{nonce}, j, \{\mathsf{tr}_{i \to j}\}_{j \in \mathbf{P}})$ from $\mathcal{F}_{\mathsf{BB}}$, for all $j \in \mathbf{P}$.
  5. Let $R = \{j \in \mathbf{P} \mid \mathcal{V}(\mathsf{pk}_i, K_{j \to i}, \mathsf{tr}_{j \to i}) \overset{?}{=} 1\}$.
  6. For $j \in R$, compute $k_{j \to i} = \mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(K_{j \to i}, \mathsf{tr}_{j \to i}))$.
  7. Compute $k_i = \sum_{j \in R} k_{j \to i}$.
  8. Compute $A_\ell = \sum_{j \in R} A_j^\ell$, for $\ell = 0, 1, \ldots, t$.
  9. Compute $K = A_0$.
  10. Output $(\mathsf{gen}, \mathsf{nonce}, k_i, \{A_\ell\}_{\ell=0}^{t}, K)$.

Theorem 3 (Based on [Gen+07]). *Let $\mathcal{A}$ be an adversary that corrupts at most t parties in the execution of $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$. Let $\mathcal{A}'$ be an adversary that runs over the view of $\mathcal{A}$, denoted as $\mathsf{view}(\mathcal{A})$, and the output of $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$. Let $(\mathbb{G}, G, p)$ be a group of order p and generator G where the discrete logarithm problem is hard and $\|p\| \geq \lambda$, for a security parameter $\lambda \in \mathbb{N}$. For every $\mathcal{A}, \mathcal{A}'$, every polynomial $p(x)$ and a large enough λ, $\Pr[\mathcal{A}'(\mathbb{G}, G, p, K, \mathsf{view}(\mathcal{A})) = k] \leq \frac{1}{p(\lambda)}$, where $(K = k \cdot G, \mathsf{view}(\mathcal{A})) \leftarrow \Pi^{n,t}_{\mathsf{RobBiasDKG}}$.*

Proof. The proof presented here is a direct adaptation from [Gen+07] with the respective variations for the bulletin board functionality and the NIVE. To prove the theorem, we will proceed by constructing a reduction to the hardness of the discrete logarithm problem. This means that we will assume that there is a pair of adversaries $\mathcal{A}, \mathcal{A}'$ for whom the conclusion of the statement in the theorem does not hold, and we will use both adversaries to construct an algorithm that breaks the discrete logarithm problem for the given group. The proof proceeds in two steps: first we create a simulator $\mathcal{S}$ for $\mathcal{A}$on inputs $(\mathbb{G}, G, p, K_T = k_T \cdot G)$, where $K_T \in \mathbb{G}$ is the the target element in the group for which we want to find the discrete logarithm; second, we use the view of of the simulated execution along with the output of $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ as inputs to $\mathcal{A}'$ so that he can come up with the discrete logarithm of the target K_T.

The simulator is defined as it is presented next.

Simulator 1: Simulator $\mathcal{S}$ for $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ and the adversary $\mathcal{A}$

Input: The input to the simulator is the tuple $(\mathbb{G}, G, p, K_T = k_T \cdot G)$ where k_T is unknown to the simulator S.

Output: The output of the simulator is $k_T \in \mathbb{Z}_p$ which aims to be the discrete logarithm of K_T.

Notation: Let $\mathbf{P}$ be the set of parties participating in the protocol, $\mathcal{C}$ be the set of corrupted parties with $|\mathcal{C}| \leq t$, and $\mathcal{H}$ the set of honest parties.

1. Simulate $\mathcal{F}_{\mathsf{eVRF}}$ by sending $\{\mathsf{init}, j, \mathsf{sid}_j\}_{i \in \mathcal{H}}$ to $\mathcal{A}$ and receive back $\{\mathsf{sid}_j\}$ as part of the simulation of $\mathcal{F}_{\mathsf{eVRF}}$.
2. Simulate $\mathcal{F}_{\mathsf{eVRF}}$ by sending $(\mathsf{init}, j, \mathsf{sid}_j)$. to all parties for $j \in \mathcal{H}$.
3. For all $i \in \mathcal{C}$, receive $(\mathsf{init}, i, \mathsf{sid}_i, f_i)$ from $\mathcal{A}$, as sent from P_i to $\mathcal{F}_{\mathsf{eVRF}}$, and send back $(\mathsf{sid}, i, \mathsf{sid}_i)$ to $i \in \mathcal{C}$.
4. The simulator receives $(\mathsf{prove}, i, j, \mathsf{pk}_i, (\mathsf{sk}_i, r_i))$ from $\mathcal{A}$, for $i \in \mathcal{C}$, as if the inputs were sent from P_i to $\mathcal{F}^{R_{\mathsf{key}}}_{\mathsf{zk}}$. The simulator checks if $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda; r_i)$. If the check passes, the simulator sends $(\mathsf{prove}, i, j, \mathsf{pk}_j, 1)$ to $j \in \mathcal{C}$. Otherwise, the simulator outputs $(\perp, \mathsf{view}(\mathcal{A}))$.
5. Let $j' \in \mathcal{H}$ be an arbitrary fixed honest party. Notice that this party does exist as $|\mathcal{C}| \leq t$ and $|\mathbf{P}| > t$.
6. For $j \in \mathcal{C}$, and for $\ell = 0, \ldots, t$, the simulator receives $(\mathsf{eval}, j, \mathsf{sid}, \mathsf{nonce}\|\ell)$. Then, the simulator computes $a_i^\ell = f_i(\mathsf{nonce}\|\ell)$ and $A_i^\ell = a_i^\ell \cdot G$. Also, the simulator computes $p_i(x) = \sum_{\ell=0}^{t} a_i^\ell \cdot x^\ell$.

7. For $\mathcal{H}\backslash\{j'\}$, sample a random polynomial $p_i(x) = \sum_{\ell=0}^{t} a_i^\ell \cdot x^\ell$, and compute $A_i^\ell = a_i^\ell \cdot G$.
8. For $i \in \mathcal{C}$, the simulator samples $k_{j'\to i} \xleftarrow{\$} \mathbb{Z}_p$ at random. Notice that there is a unique polynomial $p_{j'}(x)$ such that $p_{j'}(0) = k_T$ and $p_{j'}(\alpha_i) = k_{j'\to i}$. However, as k_T is unknown to the simulator at this point, the coefficients of the polynomial are unknown too. Suppose that $p_{j'}(x) = \sum_{\ell=0}^{t} a_{j'}^\ell \cdot x^\ell$. Hence, $a_{j'}^\ell = L_{j'}^0 \cdot k_T + \sum_{i\in\mathcal{C}} L_{j'}^i \cdot k_{j'\to i}$, where $L_{j'}^i$ are the appropriate Lagrange coefficients. Therefore, the simulator can compute $A_{j'}^\ell = a_{j'}^\ell \cdot G = L_{j'}^0 \cdot K_T + \sum_{i\in\mathcal{C}} (L_{j'}^i \cdot k_{j'\to i}) \cdot G$.
9. Simulate $\mathcal{F}_{\mathsf{eVRF}}$ by sending $(\mathsf{eval}, i, \mathsf{sid}, \mathsf{nonce}\|\ell, a_i^\ell, A_i^\ell)$ to $\mathcal{A}$, for $\ell = 0, \ldots, t$ and $i \in \mathcal{C}$, along with $(\mathsf{eval}, j, \mathsf{sid}, \mathsf{nonce}\|\ell, A_j^\ell)$ for $j \in \mathcal{H}$.
10. For $j \in \mathcal{H} \setminus \{j'\}$ and $i \in \mathcal{C}$, the simulator computes $k_{j\to i} = p_j(\alpha_i)$.
11. For $j \in \mathcal{H}$ and $i \in \mathcal{C}$, compute $K_{j\to i} = k_{j\to i} \cdot G$ and $\mathsf{tr}_{j\to i} = \mathcal{P}(\mathsf{pk}_j, K_{j\to i}, k_{j\to i})$.
12. On input $(\mathsf{write}, \mathsf{pubinfo}_{i\to j}\|\mathsf{nonce}, \mathsf{tr}_{i\to j})$ for $i \in \mathcal{C}$ and $j \in \mathbf{P}$ from $\mathcal{A}$, simulate $\mathcal{F}_{\mathsf{BB}}$ by storing $(\mathsf{pubinfo}_{i\to j}\|\mathsf{nonce}, i, \mathsf{tr}_{i\to j})$.
13. On input $(\mathsf{read}, \mathsf{pubinfo}_{i\to j}\|\mathsf{nonce}, i, \mathsf{tr}_{i\to j})$ from $i, j \in \mathbf{P}$, if $\mathsf{tr}_{i\to j}$ is stored in $\mathcal{S}$'s internal memory, then send $(\mathsf{pubinfo}_{i\to j}\|\mathsf{nonce}, i, \mathsf{tr}_{i\to j})$ to $\mathcal{A}$, otherwise, send $(\mathsf{pubinfo}_{i\to j}\|\mathsf{nonce}, \perp, \perp)$ to $\mathcal{A}$.
14. The simulator computes the set $R = \{j \in \mathbf{P} \mid \mathcal{V}(\mathsf{pk}_i, K_{j\to i}, \mathsf{tr}_{j\to i}) \stackrel{?}{=} 1\}$.
15. For $j \in R \cap \mathcal{C}$ and $i \in \mathbf{P}$, the simulator computes $k_{j\to i} = \mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(K_{j\to i}))$.
16. For $i \in \mathcal{C}$, the simulator computes $k_i = \sum_{j\in R} k_{j\to i}$. Also, for $\ell = 0, \ldots, t$, the simulator computes $A_\ell = A_j^\ell$, and $K = A_0$.
17. The simulator writes $K = K_T + K_H + K_C$, where K_H is the contribution from the honest parties in $\mathcal{H} \setminus \{j'\}$, and K_C is the contribution from the honest parties in $\mathcal{C} \cap R$. Let $K_H = k_H \cdot G$ and $K_C = k_C \cdot G$. Notice that both k_H and k_C are known to the simulator: k_H is known as the simulator samples all the honest information itself; for k_C the simulator holds k_i for $i \in \mathcal{C}$, hence it can be reconstructed using interpolation.
18. The simulator runs $\mathcal{A}'$ on input $(K, \mathsf{view}(\mathcal{A}))$ and obtains k as an output from $\mathcal{A}'$.
19. The simulator outputs $k_T = k - k_C - k_H$.

Observe that if k is computed correctly by $\mathcal{A}'$ as the discrete logarithm of K, then k_T is the correct discrete logarithm for K_T. The correctness of both discrete logarithms depends whether the view of $\mathcal{A}$ is identical to a real-world execution of $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$.

We proceed to prove that the view of $\mathcal{A}$ is identical to a real-world execution. First, the simulated view of the execution of parties in $\mathcal{H} \setminus \{j'\}$ is identical to a real-world execution, as the sampling follows from the protocol $\Pi^{n,t}_{\mathsf{RobBiasDKG}}$ directly. Second, the simulation for j' differs from the protocol execution, but the distribution is the same. Notice that the $k_{j'\to i}$ is sampled at random, which is the same distribution of the evaluations of a random uniform polynomial with a fixed constant term. Hence, the values of $A_{j'}^\ell = a_{j'}^\ell$ are uniformly distributed as the polynomial $p_{j'}$ is also a uniformly distributed polynomial. Also, although the adversary can query data from the $\mathcal{F}_{\mathsf{BB}}$ without restriction, the information that he can access is limited only to the information that the adversary can decrypt using the secret key of the corrupt parties. The proofs for the relation R_{DL} delivered to the honest parties do not reveal any information about the honest parties' shares of the secret key. Hence, the adversary can not change its view according to the honest parties' secret shares.

References

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. “Optimistic Fair Exchange of Digital Signatures (Extended Abstract)”. In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer Berlin Heidelberg, Germany, May 1998, pp. 591–606. DOI: 10.1007/BFb0054156.

[BDO14] Carsten Baum, Ivan Damgård, and Claudio Orlandi. “Publicly Auditable Secure Multi-Party Computation”. In: *SCN 14: 9th International Conference on Security in Communication Networks*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. Lecture Notes in Computer Science. Amalfi, Italy: Springer, Cham, Switzerland, Sept. 2014, pp. 175–196. DOI: 10.1007/978-3-319-10879-7_11.

[BHL24] Dan Boneh, Iftach Haitner, and Yehuda Lindell. *Exponent-VRFs and Their Applications*. Cryptology ePrint Archive, Report 2024/397. 2024. URL: https://eprint.iacr.org/2024/397.

[BK25] Renas Bacho and Alireza Kavousi. “SoK: Dlog-Based Distributed Key Generation”. In: *2025 IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society Press, May 2025, pp. 614–632. DOI: 10.1109/SP61157.2025.00127.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing”. In: *Advances in Cryptology – ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. Lecture Notes in Computer Science. Gold Coast, Australia: Springer Berlin Heidelberg, Germany, Dec. 2001, pp. 514–532. DOI: 10.1007/3-540-45682-1_30.

[Can00] Ran Canetti. “Security and Composition of Multiparty Cryptographic Protocols”. In: *Journal of Cryptology* 13.1 (Jan. 2000), pp. 143–202. DOI: 10.1007/s001459910006.

[Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[CD00] Jan Camenisch and Ivan Damgård. “Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes”. In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Kyoto, Japan: Springer Berlin Heidelberg, Germany, Dec. 2000, pp. 331–345. DOI: 10.1007/3-540-44448-3_25.

[CM00] Jan Camenisch and Markus Michels. "Confirmer Signature Schemes Secure against Adaptive Adversaries". In: *Advances in Cryptology – EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Bruges, Belgium: Springer Berlin Heidelberg, Germany, May 2000, pp. 243–258. DOI: 10.1007/3-540-45539-6_17.

[CS03] Jan Camenisch and Victor Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms". In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, Aug. 2003, pp. 126–144. DOI: 10.1007/978-3-540-45146-4_8.

[CV02] Jan Camenisch and Els Van Herreweghen. "Design and Implementation of The Idemix Anonymous Credential System". In: *ACM CCS 2002: 9th Conference on Computer and Communications Security*. Ed. by Vijayalakshmi Atluri. Washington, DC, USA: ACM Press, Nov. 2002, pp. 21–30. DOI: 10.1145/586110.586114.

[Gen+07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *Journal of Cryptology* 20.1 (Jan. 2007), pp. 51–83. DOI: 10.1007/s00145-006-0347-3.

[GR04] Steven D. Galbraith and Victor Rotger. *Easy decision-Diffie-Hellman groups*. Cryptology ePrint Archive, Report 2004/070. 2004. URL: https://eprint.iacr.org/2004/070.

[Gra+24] Mike Graf et al. "Accountable Bulletin Boards: Definition and Provably Secure Implementation". In: *CSF 2024: IEEE 37th Computer Security Foundations Symposium*. Enschede, The Netherlands: IEEE Computer Society Press, July 2024, pp. 201–216. DOI: 10.1109/CSF61375.2024.00013.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "Pseudo-random Generation from one-way functions (Extended Abstracts)". In: *21st Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA: ACM Press, May 1989, pp. 12–24. DOI: 10.1145/73007.73009.

[Kat+23] Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. *Non-interactive VSS using Class Groups and Application to DKG*. Cryptology ePrint Archive, Report 2023/451. 2023. URL: https://eprint.iacr.org/2023/451.

[KLR06] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. "Information-theoretically secure protocols and security under composition". In: *38th Annual ACM Symposium on Theory of Computing*. Ed. by Jon M. Kleinberg. Seattle, WA, USA: ACM Press, May 2006, pp. 109–118. DOI: 10.1145/1132516.1132532.

[Lee+19] Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. *SAVER: Snark-friendly, Additively-homomorphic, and Verifiable Encryption and decryption with Rerandomization.* Cryptology ePrint Archive, Report 2019/1270. 2019. URL: https://eprint.iacr.org/2019/1270.

[LN17] Vadim Lyubashevsky and Gregory Neven. "One-Shot Verifiable Encryption from Lattices". In: *Advances in Cryptology – EUROCRYPT 2017, Part I.* Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Paris, France: Springer, Cham, Switzerland, Apr. 2017, pp. 293–323. DOI: 10.1007/978-3-319-56620-7_11.

[Nic+20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. "MuSig-DN: Schnorr Multi-Signatures with Verifiably Deterministic Nonces". In: *ACM CCS 2020: 27th Conference on Computer and Communications Security.* Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. Virtual Event, USA: ACM Press, Nov. 2020, pp. 1717–1731. DOI: 10.1145/3372297.3417236.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology – CRYPTO'91.* Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[Sta96] Markus Stadler. "Publicly Verifiable Secret Sharing". In: *Advances in Cryptology – EUROCRYPT'96.* Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer Berlin Heidelberg, Germany, May 1996, pp. 190–199. DOI: 10.1007/3-540-68339-9_17.

[TZ24] Akira Takahashi and Greg Zaverucha. "Verifiable Encryption from MPC-in-the-Head". In: *IACR Communications in Cryptology (CiC)* 1.1 (2024), p. 3. DOI: 10.62056/a3wa3zl7s.

A Proof of Theorem 2

Proof. Let $\mathcal{A}$ be an adversary corrupting a set of parties $\mathcal{C}$ such that $|\mathcal{C}| \leq t$. Let $\mathcal{H}$ be the set of honest parties and $\mathbf{P}$ be the set of parties participating in the protocol, hence $\mathbf{P} = \mathcal{C} \cup \mathcal{H}$. We present next a simulator for the protocol $\Pi^{n,t}_{\mathsf{DKG},\mathcal{Q}}$.

Simulator 2: Simulator for $\Pi^{n,t}_{\mathsf{DKG},\mathcal{Q}}$

Perform the following steps:

1. Initialize:
   (a) The simulator $\mathcal{S}$ invokes $\mathcal{A}$ on input 1^λ and receives $((\mathsf{sk}_i, r_i), \mathsf{pk}_i)$, for $i \in \mathcal{C}$, from the adversary $\mathcal{A}$ as if it was sent to $\mathcal{F}^{R_{\mathsf{key}}}_{\mathsf{zk}}$ from the corrupt parties.

(b) For all $i \in \mathcal{C}$, $\mathcal{S}$ checks if $(\mathsf{pk}_i, \mathsf{sk}_i) = \mathsf{Gen}(1^\lambda, r_i)$. If some check fails, the simualtor sends an abort signal to $\mathcal{F}^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ and halts, otherwise $\mathcal{S}$ continues.

(c) $\mathcal{S}$ computes $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for $i \in \mathcal{H}$. For $i \in \mathcal{H}$ and $j \in \mathcal{C}$, $\mathcal{S}$ sends internally $(\mathsf{prove}, i, j, \mathsf{pk}_i, 1)$ to the adversary as if the message was sent by the honest parties.

(d) The simulator $\mathcal{S}$ receives $(\mathsf{init}, \mathsf{sid})$ from the adversary as if it was sent from the corrupt parties to the functionality $\mathcal{F}^n_{\mathsf{ComBB}}$. At that point, $\mathcal{S}$ initializes the set $C = \mathcal{H}$ and the flag $\mathsf{com} = \mathsf{false}$.

2. Generate and Output:

(a) For each $i \in \mathcal{Q} \cap \mathcal{C}$ and $j \in \mathbf{P}$, $\mathcal{S}$ receives

$$(\mathsf{write}, \mathsf{pubinfo}_{i\to j} \| \mathsf{nonce} \| \mathcal{Q}, (\mathsf{tr}_{i\to j}, K_{i\to j}, \{A_j^\ell\}_{\ell=0}^t))$$

as a simulation of $\mathcal{F}^n_{\mathsf{ComBB}}$. $\mathcal{S}$ sends $(\mathsf{commit}, \mathsf{pubinfo}_{i\to j} \| \mathsf{nonce} \| \mathcal{Q}, i)$ all the corrupted parties and updates $C \leftarrow C \cup \{i\}$. If $C = \mathbf{P}$, then $\mathcal{S}$ updates $\mathsf{com} = \mathsf{true}$ and sends $(\mathsf{sid}, \mathsf{complete})$ to all parties.

(b) For $i \in \mathcal{Q} \cap \mathcal{H}$ and $j \in \mathbf{P}$, $\mathcal{S}$ sends $(\mathsf{commit}, \mathsf{pubinfo}_{i\to j} \| \mathsf{nonce} \| \mathcal{Q}, i)$ to the adversary as a simulation of the honest responses to the corrupt parties.

(c) $\mathcal{S}$ sets the flag abort to false.

(d) For $i \in \mathcal{Q} \cap \mathcal{C}$ and $j \in \mathbf{P}$, $\mathcal{S}$ verifies the following checks:

i. $\mathcal{S}$ computes $b_{i\to j} = \mathcal{V}(\mathsf{pk}_j, K_{i\to j}, \mathsf{tr}_{i\to j})$. If there exists some $b_{i\to j} = 0$, then $\mathcal{S}$ sets the flag abort to true, otherwise, it continues.

ii. For $i \in \mathcal{Q} \cap \mathcal{C}$ and $j \in \mathbf{P}$, $\mathcal{S}$ verifies that $K_{i\to j} = \sum_{\ell=0}^t \alpha_j^\ell \cdot A_i^\ell$. If there exists some check that does not hold, then $\mathcal{S}$ sets the flag abort to true, otherwise, it continues.

(e) $\mathcal{S}$ receives the input $(\mathsf{read}, \mathsf{pubinfo}_{j\to i} \| \mathsf{nonce} \| \mathcal{Q}, j)$ for $i \in \mathcal{C}$ and $j \in \mathcal{Q}$ simulating $\mathcal{F}^n_{\mathsf{ComBB}}$. If $\mathsf{com} = \mathsf{false}$, then send $(\mathsf{sid}, \perp)$ to $\mathcal{A}$. Otherwise, if the abort flag is true, $\mathcal{S}$ executes the following steps:

i. For $j \in \mathcal{Q} \cap \mathcal{H}$ choose a random polynomial $p_j(x) = \sum_{\ell=0}^t a_j^\ell x^\ell$, and compute $A_j^\ell = a_j^\ell \cdot G$ for $\ell = 0, \ldots, t$.

ii. For $j \in \mathcal{Q} \cap \mathcal{H}$ and $i \in \mathcal{C}$, compute $k_{j\to i} = p_j(\alpha_i)$ and $K_{j\to i} = k_{j\to i} \cdot G$.

iii. For $j \in \mathcal{Q} \cap \mathcal{H}$ and $i \in \mathcal{C}$, compute $\mathsf{tr}_{j\to i} = \mathcal{P}(\mathsf{pk}_i, K_{j\to i}, k_{j\to i})$.

iv. For $j \in \mathcal{Q} \cap \mathcal{H}$, and $i \in \mathcal{C}$, send

$$(\mathsf{pubinfo}_{j\to i} \| \mathsf{nonce} \| \mathcal{Q}, j, (\mathsf{tr}_{j\to i}, K_{j\to i}, \{A_j^\ell\}_{\ell=0}^t))$$

to $\mathcal{A}$ simulating the response from $\mathcal{F}^n_{\mathsf{ComBB}}$ to the corrupt parties.

v. $\mathcal{S}$ sends an abort signal to $\mathcal{F}^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ and halts.

(f) If the abort flag is false, the simulator $\mathcal{S}$ sends externally $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q})$ to the ideal functionality $\mathcal{F}^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ to obtain

$$\{k_i, A_0, \ldots, A_t, K\}_{i\in\mathcal{C}} .$$

(g) $\mathcal{S}$ executes the following steps:

i. For $i \in \mathcal{Q} \cap \mathcal{C}$ and $j \in \mathbf{P}$, $\mathcal{S}$ computes $k_{i\to j} = \mathsf{Dec}(\mathsf{sk}_j, \mathsf{Enc}(\mathsf{pk}_j, K_{i\to j}, \mathsf{tr}_{i\to j}))$.

ii. Let $j' \in \mathcal{Q} \cap \mathcal{H}$ a fixed honest party. This party must exist because $|\mathcal{Q}| = t + 1$ and $|\mathcal{C}| \leq t$.

iii. For $j \in \mathcal{Q} \cap \mathcal{H}$ with $j \neq j'$, choose a random polynomial $p_j(x) = \sum_{\ell=0}^{t} a_j^\ell x^\ell$, and compute $A_j^\ell = a_j^\ell \cdot G$ for $\ell = 0, \ldots, t$.

iv. For $j \in \mathcal{Q} \cap \mathcal{H}$ with $j' \neq j$, and for $i \in \mathcal{C}$, compute $k_{j \to i} = p_j(\alpha_i)$.

v. For $i \in \mathcal{C}$ compute $k_{j' \to i} = k_i - \sum_{j \in \mathcal{Q} \setminus \{j'\}} k_{j \to i}$. With this, we have that $k_i = \sum_{j \in \mathcal{Q}} k_{j \to i}$.

vi. For $\ell = 0, \ldots, t$, $A_{j'}^\ell = A_\ell - \sum_{j \in \mathcal{Q} \setminus \{j'\}} A_j^\ell$. Hence, $A_\ell = \sum_{j \in \mathcal{Q}} A_j^\ell$.

vii. For $j \in \mathcal{Q} \cap \mathcal{H}$, and $i \in \mathcal{C}$, compute $K_{j \to i} = k_{j \to i} \cdot G$, and $\mathsf{tr}_{j \to i} = \mathcal{P}(\mathsf{pk}_i, K_{j \to i}, k_{j \to i})$.

viii. For $j \in \mathcal{Q} \cap \mathcal{H}$, and $i \in \mathcal{C}$, send

$$(\mathsf{pubinfo}_{j \to i} \| \mathsf{nonce} \| \mathcal{Q}, j, (\mathsf{tr}_{j \to i}, K_{j \to i}, \{A_j^\ell\}_{\ell=0}^{t}))$$

to $\mathcal{A}$ simulating the response from $\mathcal{F}_{\mathsf{ComBB}}^{n}$ to the corrupt parties.

(h) Let $\mathcal{O} \subseteq \mathcal{H}$ be the set of honest parties such that the key pairs are valid according to $\mathcal{F}_{\mathsf{zk}}^{R_{\mathsf{key}}}$, that the adversary $\mathcal{A}$ allowed them to obtain the information from $\mathcal{F}_{\mathsf{ComBB}}^{n}$, and that successfully verified Steps 2(d)i and 2(d)ii.

(i) Send the set $\mathcal{O}$ to the ideal functionality $\mathcal{F}_{\mathsf{DKG},\mathcal{Q}}^{n,t}$.

(j) Output whatever $\mathcal{A}$ outputs.

First, we see that $\mathcal{S}$ simulates the abort points of the protocol, which means that the simulator aborts in the real execution with the same probability as the aborts in the real-world execution.

Second, if all the checks pass, notice that for $i \in \mathcal{C}$ it holds that $k_i = \sum_{j \in \mathcal{Q}} k_{j \to i}$, and $A_\ell = \sum_{j \in \mathcal{Q}} A_j^\ell$, for $\ell = 0, \ldots, t$. Now, let us prove that for $i \in \mathcal{C}$ and $j \in \mathcal{Q} \cap \mathcal{H}$, $k_{j \to i} \cdot G = K_{j \to i} = \sum_{\ell=0}^{t} (\alpha_i)^\ell \cdot A_j^\ell$ with overwhelming probability. It is clear that for $j \in \mathcal{Q} \cap \mathcal{H}$ with $j \neq j'$, it holds that $k_{j \to i} \cdot G = p_j(\alpha_i) \cdot G = \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_j^\ell$. Now, for $j' \in \mathcal{Q} \cap \mathcal{H}$, we have that

$$\begin{aligned}
k_{j' \to i} \cdot G &= \left(k_i - \sum_{j \in \mathcal{Q} \setminus \{j'\}} k_{j \to i} \right) \cdot G = k_i \cdot G - \sum_{j \in \mathcal{Q} \setminus \{j'\}} k_{j \to i} \cdot G \\
&= \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_\ell - \sum_{j \in \mathcal{Q} \setminus \{j'\}} k_{j \to i} \cdot G = \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_\ell - \sum_{j \in \mathcal{Q} \setminus \{j'\}} K_{j \to i} \\
&= \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_\ell - \sum_{\ell=0}^{t} \sum_{j \in \mathcal{Q} \setminus \{j'\}} \alpha_i^\ell \cdot A_{j'}^\ell = \sum_{\ell=0}^{t} \alpha_i^\ell \cdot \left(A_\ell - \sum_{j \in \mathcal{Q} \setminus \{j'\}} A_j^\ell \right) \\
&= \sum_{\ell=0}^{t} \alpha_i^\ell \cdot A_{j'}^\ell .
\end{aligned} \tag{8}$$

This equality holds with overwhelming probability given that for $i \in \mathcal{Q} \cap \mathcal{C}$ and $j \in \mathbf{P}$, $k_{i \to j} \cdot G = K_{i \to j}$ with overwhelming probability conditioned to the fact that the checks of Steps 2(d)i and 2(d)ii pass. This shows that the ideal response of the simulator to the adversary $\mathcal{A}$ simulating $\mathcal{F}^{n}_{\mathsf{ComBB}}$ is computationally indistinguishable from the real world interaction. Hence, the output of the real and ideal executions are computationally indistinguishable which proofs the theorem. Given that the simulator is a straight-line simulator, we can conclude that the $\Pi^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ UC-realizes $\mathcal{F}^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ in the $(\mathcal{F}^{n}_{\mathsf{ComBB}}, \mathcal{F}^{R_{\mathsf{key}}}_{\mathsf{zk}})$-hybrid model.

B Ideal Functionality for Distributed Key Generation

Functionality 5: $\mathcal{F}^{n,t}_{\mathsf{DKG},\mathcal{Q}}$ [BHL24]

Let $\mathbb{G}$ be a group of order q and generator G, let $\alpha_1, \dots \alpha_n$ be fixed distinct elements in $\mathbb{Z}_q$, and let $t < n$. Let $P_1, \dots, P_n$ be parties participating in the protocol, a quorum $\mathcal{Q} \subseteq [n]$ of $t+1$ online parties, and corrupted parties $\mathcal{C} \subseteq [n]$, with $|\mathcal{C}| \leq t$.

1. The functionality waits to receive $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q})$ from each party in $\mathcal{Q}$.
2. If $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q}, p(x))$ was already stored, the functionality executes the following steps:
   - Retrieve $p(x)$,
   - Store $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q}, p(x))$.
3. Otherwise, the functionality executes the following steps:
   - Choose a random degree-t polynomial $p(x) \xleftarrow{\$} \mathbb{F}_q[x]$, such that $p(x) = \sum_{i=0}^{t} a_i x^i$.
   - Store $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q}, p(x))$.
4. For $j \in [n]$, the functionality computes $k_j = p(\alpha_j)$.
5. For each $j \in [t]$, the functionality computes $A_i = a_i \cdot G$.
6. The functionality computes $k = p(0)$ and $K = k \cdot G$.
7. For each $i \in [n]$, the functionality send $(\mathsf{gen}, \mathsf{nonce}, \mathcal{Q}, k_i, A_0, \dots, A_t, K)$ to P_i.