Practical Robust DKG Protocols for CSIDH

Shahla Atapoor[1], Karim Baghery[1], Daniele Cozzo[2,1], and Robi Pedersen[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
[2] IMDEA Software Institute, Madrid, Spain
`firstname.lastname@kuleuven.be`,
`daniele.cozzo@imdea.org`

Abstract. A Distributed Key Generation (DKG) protocol is an essential component of threshold cryptography. DKGs enable a group of parties to generate a secret and public key pair in a distributed manner so that the secret key is protected from being exposed, even if a certain number of parties are compromised. Robustness further guarantees that the construction of the key pair is always successful, even if malicious parties try to sabotage the computation. In this paper, we construct two efficient robust DKG protocols in the CSIDH setting that work with Shamir secret sharing. Both the proposed protocols are proven to be actively secure in the quantum random oracle model and use an Information Theoretically (IT) secure Verifiable Secret Sharing (VSS) scheme that is built using bivariate polynomials. As a tool, we construct a new piecewise verifiable proof system for structured public keys, that could be of independent interest. In terms of isogeny computations, our protocols outperform the previously proposed DKG protocols CSI-RAShi and Structured CSI-RAShi. As an instance, using our DKG protocols, 4 parties can sample a PK of size 4kB, for CSI-FiSh and CSI-SharK, respectively, 3.4 and 1.7 times faster than the current alternatives. On the other hand, since we use an IT-secure VSS, the fraction of corrupted parties is limited to less than a third and the communication cost of our schemes scales slightly worse with an increasing number of parties. For a low number of parties, our scheme still outperforms the alternatives in terms of communication.

Keywords: Distributed Key Generation · CSIDH · Isogenies · VSS.

1 Introduction

Key management is an important aspect of cryptography, as the security of cryptographic algorithms crucially relies on the safety of secret keys. If an attacker obtains the secret key, the security of the system is generally compromised. Threshold cryptography addresses this issue by dividing the secret key among a group of devices, in such a way that only subsets of these devices above a specific threshold size t can reconstruct the secret key. An attacker would then need to obtain shares from $t + 1$ parties to reconstruct the secret key. This effectively eliminates the single point of failure of cryptographic algorithms. At the basis of threshold schemes are Distributed Key Generation (DKG) protocols, that are run by the parties to produce a correct sharing of the secret key along with the associated public key.

Threshold protocols based on classical assumptions have been extensively studied and an initiative to standardize them has been taken by NIST.[3] However, due to the vulnerability of classical assumptions, like the Discrete Logarithm (DL) and factoring problems, against quantum computers [30], the cryptographic community has begun to work on a general migration of public key cryptography to algorithms that are based on post-quantum problems. Threshold cryptography will also need to adapt to these new problems.

One active research direction for designing post-quantum secure protocols is isogeny-based cryptography, which relies on the hard problem of finding a secret isogeny between two public elliptic curves. Isogeny-based cryptography was initially based on ordinary elliptic curves [13, 14, 28], but in the last decade, focus has almost entirely switched to supersingular elliptic curves due to security and/or efficiency reasons. One of the first protocols to propose using supersingular elliptic curves was SIDH (supersingular isogeny-based Diffie-Hellman) [17]. However, protocols based on SIDH need parties to reveal extra information along with their public key, which recently has been proven to be enough to recover the secret isogeny [10, 24, 27], thus effectively breaking the SIDH-based family of protocols. Fortunately, newer schemes such as SQISign [18] and CSIDH [11] (commutative SIDH) are unaffected by these attacks, as they do not need to reveal this extra information. The intrinsic commutativity of CSIDH has shown to provide a lot of flexibility when designing cryptographic protocols. As such, CSIDH provides a versatile toolbox, which allows to build more complex cryptographic protocols, such as threshold schemes [2, 6, 9, 15, 19].

Previous DKGs in the CSIDH Setting. The first CSIDH-based actively secure DKG was proposed as part of the distributed signature scheme called Sashimi [15]. However, their original DKG protocol is quite inefficient in practice. In follow-up work, Beullens, Disson, Pedersen and Vercauteren [6] proposed the first *robust* DKG protocol for CSIDH, which works with Shamir secret sharing and allows a set of parties to sample a single public key in a fully distributed manner. The robustness ensures that the parties are able to carry on the DKG even if the adversary tries to sabotage the protocol. Their construction, dubbed CSI-RAShi, consists of two phases. A Verifiable Secret Sharing (VSS) phase, where the parties create their shares of the secret key, and the Public Key (PK) computation phase, where the parties use their shares to compute the target PK. While in traditional Pedersen VSS [26], the verification of shares was easily done by exploiting the homomorphic properties of modular exponentiation, this is not possible with elliptic curves. To deal with this concern, [6] introduces Piecewise Verifiable Proofs (PVPs), a sort of Zero-Knowledge (ZK) proof that allows parties to verify that the share they got is consistent with respect to some public commitment. In the PK computation step, a different ZK proof is used.

[3] See `https://csrc.nist.gov/projects/threshold-cryptography`.

Both these proofs use a binary challenge space and as a result, the protocol is still computationally expensive, albeit faster than the Sashimi proposal. Moreover, as a result of the ZK proofs used in the PK computation step, the security of the DKG protocol needs to rely on a decisional assumption. As a result, the authors can only argue security when the target public key is uniformly random, rather than arbitrary.

CSI-RAShi is constructed to sample a single public key $[x]E_0$ in a distributed manner, and by repeating it k times, one can sample extended public keys of the form $([x_1]E_0, \ldots, [x_k]E_0)$ as they are used in the signature scheme CSI-FiSh [7]. Repeating CSI-RAShi k times is very inefficient, even if optimized [2]. To deal with that concern, the authors of [2] proposed CSI-SharK, as a variant of CSI-FiSh with Sharing-friendly Key, as well as Structured CSI-RAShi, as a variant of CSI-RAShi for Structured Public Keys (SPKs). Structured CSI-RAShi allows a set of parties to sample an SPK (introduced in [3]), i.e. a public key of the type $([c_1x]E_0, \ldots, [c_kx]E_0)$, where $c_1, \ldots, c_k$ define an exceptional set. For an SPK of the same size k, Structured CSI-RAShi is 4 times faster than generating an extended public key using the original CSI-RAShi.

Our Contribution. In this paper, we construct two efficient robust DKG protocols for CSIDH-based cryptographic primitives that work with Shamir secret sharing. Both protocols are proven to be actively secure in the Quantum Random Oracle Model (QROM). Our DKG protocols can be considered as an alternative to the DKG protocols CSI-RAShi [6] and its Structured variant [2]. We show that both our proposed DKG protocols outperform these previous proposals in terms of computational cost. Moreover, the VSS step of our DKG protocols does not rely on any decisional (or computational) assumption and does not use ZK proofs, but instead uses an efficient Information Theoretically (IT) secure VSS scheme. However, the latter comes at the cost of increased communication complexity and a higher number of needed honest parties, in comparison with the alternatives. Specifically, in the VSS step, more than two thirds of the parties must be honest rather than just the majority of them. But, as we are in the static corruption setting, for the PK computation phase still our protocols work in the majority honest setting, as in the original CSI-RAShi and its structured variant.

Technical Overview. To design our proposed DKG protocols, we have modified the construction of CSI-RAShi to minimize the number of isogeny computations required for each party, as they are the most computationally expensive parts in these protocols. Our first key modification is to change the secret sharing step of CSI-RAShi to an efficient VSS scheme based on bivariate polynomials, that was first proposed in [4, 33], but for a different purpose. The idea is that each party P_i can distribute a value $x^{(i)}$ by sampling a random polynomial $q^{(i)}(Z)$ of degree t subject to $q^{(i)}(0) = x^{(i)}$, then hiding $q^{(i)}(Z)$ in the bivariate polynomial $S^{(i)}(X, Y)$, also of degree t in both variables, in a way that $S^{(i)}(0, Z) = q^{(i)}(Z)$. Now, instead of sending evaluations of a univariate polynomial, the verification shares are two polynomials defined by $S^{(i)}(j, Y) = g_j(Y)$ and $S^{(i)}(X, j) = f_j(X)$, which are then sent to each other party P_j. Parties can now do pairwise checks to verify that the polynomials they got from P_i are correct by testing whether $f_j^{(i)}(k) = g_k^{(i)}(j)$ and $f_k^{(i)}(j) = g_j^{(i)}(k)$. The key point is that if there are at least t of these relations that are satisfied, then there exists a unique bivariate polynomial $S^{(i)}(X, Y)$ as above and P_i acted honestly.

After the VSS step, to compute the PK, parties use their secret shares and engage in a round-robin MPC protocol and prove that they did their computation correctly. As opposed to CSI-RAShi, parties can no longer use the ZK proof from [15] to prove correct execution, since they did not commit to their shares $x^{(i)}$ in the VSS step. Instead, we observe that as a result of our modifications, we can use the VSS as a distributed commitment scheme and prove the correctness of computations using a PVP scheme. With PVPs, parties are able to prove that they are updating the PK using $q^{(i)}(0) = x^{(i)}$, which was shared in the VSS step.

Using the bivariate polynomial-based VSS and our new modifications improves the efficiency of the final DKG protocol, when compared to previous results in the literature. The first source of improvement is that, instead of performing expensive PVPs and isogeny computations to verify the shares, as done in the VSS step of CSI-RAShi, now the parties just need to perform fast polynomial evaluations and pairwise comparisons. The second source of improvement is that we replace the ZK proofs in the PK computation step with a PVP scheme, which is more efficient. Moreover, we remove the need for a decisional assumption and also achieve IT security in the VSS step. Conveniently, removing the need for a decisional assumption, allows us to use the twist technique from [7], to extend the challenge space of the PVP scheme, and further improve the communication and computational costs in some cases. A downside resulting from our modifications is that the communication complexity increases, but outperforms previous designs for a low number of parties (e.g. below 26 for $k = 2^6$), and the fraction of corrupted parties allowed in the VSS reduces to less than a third, rather than half, which seems to be unavoidable for IT security. We also show how to build extended public keys with this protocol and discuss optimizations to reduce the overall runtime.

The second DKG protocol we propose allows a set of parties to sample an SPK. Note that since SPKs only have a single secret key, the VSS step is independent of the length of the SPK. In order to adapt the PK computation step to SPKs, we also construct a new PVP scheme, which is specific to SPKs, called Structured PVPs (SPVP), which we believe to be of independent interest. The SPVP scheme allows parties to prove that they are computing/updating the target SPK using different factors of the $x^{(i)}$ shared in the VSS step. By replacing the ZK proofs in the SPK computation with our new SPVPs, we also manage to improve the efficiency of the Structured CSI-RAShi scheme, proposed in [2].

At the end, it is worth mentioning that the idea of using an IT secure VSS based on bivariate polynomials within a DKG was already exploited by [37], in the DL setting. In that work, the authors plug the VSS scheme of [4, 33] into the threshold scheme by Gennaro, Jarecki, Krawczyk and Rabin [22] for distributing the key generation of the Schnorr signature. After the VSS, the

PK is then constructed by having the players aggregate all partial public keys as specified in [22]. However in the DL setting using the bivariate polynomial-based VSS [4,33] does not provide a big gain in terms of efficiency, in comparison with [22]. On the other hand, for isogenies, as we show the advantage of using a bivariate polynomial based VSS scheme in CSI-RAShi [6] and Structured CSI-RAShi [2] is significant. As we can save on expensive ZK proofs and on the number of isogeny computations (each taking around 35-40ms).

Efficiency. Our base protocol (for a single public key) is naturally about twice as fast as CSI-RAShi, currently the state-of-the-art. On the downside, communication cost scales quadratically in the number of parties n, instead of linearly, which leads to a noticeable increase when many parties are present but manageable for low to medium n. For low n, we get further reduction of these costs due to the higher impact of using the twist trick with fewer parties.

For larger public keys, we can use optimizations similar to the ones proposed in [2]. The gains depend on the number of parties n and public key sizes k, but we show that in terms of the number of isogeny computations, our protocols always outperform the results from [2]. Due to the twist trick, the most important gains are visible for low n. But even for $n \to \infty$, we outperform extended CSI-RAShi by a factor of 3 and structured CSI-RAShi with approximately a factor $1+k^{-1/2}$. The gain against the latter thus becomes negligible only for large k and large n.

As a numerical example, consider PKs with $k = 2^6$ elements (which have a size of about 4kB). For $n = 4$, we get a gain of 3.4 and 1.7 against extended and structured CSI-RAShi from [2], while for $n = 100$, this gain reduces to 3.0 and 1.5, respectively. Comparing the communications of our structured scheme and structured CSI-RAShi for $k = 2^6$, our scheme has about a third of the communication cost when $n = 4$, while for $n = 100$, we have about 4 times more communication, the break-even point being at $n = 25$ for this public key size. This break-even point increases with the number of parties, and is e.g. $n = 98$ for $k = 2^{15}$.

Outline. In Sec. 2, we review some preliminary concepts. In Sec. 3, we first discuss the VSS based on bivariate polynomials, then use it to construct a DKG protocol for a PK with a single curve and finally discuss its extension for generating multiple independent curves. Then, in Sec. 4, we present the second DKG protocol for sampling an SPK. We discuss and compare the efficiency of our DKG protocols in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Preliminaries

Notation. We use the assignment operator $\leftarrow$ to denote random sampling from a probability distribution D, e.g. $x \leftarrow D$, or uniform sampling from a set X, e.g. $x \leftarrow X$. We write λ to denote a security parameter. We call a function f *negligible in* X, if for any constant c, there exists some X_0, such that $f(X) < X^{-c}$ for $X > X_0$. We denote this as $\mathsf{negl}(X)$. We call a function simply *negligible* if it is negligible in λ. We write $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$ and $\log(x) := \log_2(x)$.

2.1 Isogeny-based Cryptography

Isogenies are rational maps between elliptic curves, that are also surjective homomorphisms with respect to the natural group structure between these curves. In this work, we will only consider supersingular elliptic curves and separable isogenies defined over prime fields $\mathbb{F}_p$. We denote the set of such elliptic curves as $\mathcal{E}$. The endomorphisms of elliptic curves over $\mathbb{F}_p$ define a ring structure that is isomorphic to orders $\mathcal{O}$ in the quadratic imaginary field $\mathbb{Q}(\sqrt{-p})$. Separable isogenies are uniquely defined by their kernels, which in turn can be identified with the kernels of ideal classes in the class group $\mathrm{cl}(\mathcal{O})$. For efficiency reasons, the prime p is chosen, so that $p-1 = 4\prod_i \ell_i$ consists of the multiplication of small primes. By this choice, ideals of the type $\ell_i\mathcal{O}$ split into a prime ideal $\mathfrak{l}_i$ and its conjugate $\overline{\mathfrak{l}_i}$, uniquely defining an isogeny and its dual, both of small prime (and thus efficiently computable) prime degree ℓ_i.[4] Throughout this work, we assume the class group for the relevant order $\mathrm{cl}(\mathcal{O})$ to be known, so that arbitrary ideals can be transformed into efficient isogeny computations using the relation lattice, by translating them to a small number of consecutive degree-ℓ_i isogeny computations.[5]

We note that in general, class groups are of composite order. For any cyclic subgroup of the class group, of size $N \mid \#\mathrm{cl}(\mathcal{O})$ with generator $\mathfrak{g}$, we can then define isogenies through the action of elements in $\mathbb{Z}_N \subseteq \mathbb{Z}_{\#\mathrm{cl}(\mathcal{O})}$ as $[\,] : \mathbb{Z}_N \times \mathcal{E} \to \mathcal{E}$, where the action $[a]E \mapsto E'$ defines an isogeny with kernel isomorphic to $\ker \mathfrak{g}^a$, reduced modulo the relation lattice. In this way, the map $[\,]$ defines a free and transitive group action [14] by $\mathbb{Z}_N$ (as a proxy for the subgroup of $\mathrm{cl}(\mathcal{O})$) on the set $\mathcal{E}$. We refer the reader to [5,7,12,36] for more details on the explicit computations of isogenies. For a more thorough introduction to isogenies and isogeny-based cryptography, the authors recommend [12,16,32].

The fact that N can be composite implies that in general $\mathbb{Z}_N$ constitutes a ring (and not a field). This has to be handled with care, as in some applications, inverse elements are needed. Assuming that N has the prime decomposition $\prod_{i=1}^{n} N_i$, where, for later reference, we assume $N_1 < \cdots < N_n$, we can easily work in a subgroup of size $N' = \prod_{i\in S} N_i$ for $S \subset \{1,\ldots,N\}$ by using the generator $\mathfrak{g}^{N/N'}$.

2.2 DKG Protocols and Piecewise Verifiable Proofs

A DKG protocol mainly consists of secret sharing and PK computation. Secret sharing-based protocols are interactive schemes between n parties $P_1,\ldots,P_n$,

[4] The extra factor 4 is chosen so that $p \equiv 3 \mod 4$, which makes the particular curve $E_0 : y^2 = x^3 + x$ supersingular, and allows to work in the more efficient Montgomery coordinates, see [12] for more details.

[5] We note that this is not a trivial assumption, since computing large class groups is generally difficult using classical computers, cf. [8], which computed a 257-bit class group and associated lattice of relations for the CSIDH-512 parameter set from [11]. An alternative approach is discussed in [20], which strongly speeds up the class group computations, but unfortunately leads to much slower group action computations. We note however that there exist efficient quantum algorithms [23] for this purpose.

such that at the end of the protocol, each party holds a share of a common secret s in a way, that only specifically allowed sets (qualified sets) can join forces in order to efficiently reconstruct s (or equivalently act with the isogeny $[s]$), while this is unfeasible for any non-qualified set.

Throughout this work, we realize this sharing using Shamir Secret Sharing (SSS) [29] and variants thereof. In SSS, the secret is defined as the evaluation of some secret polynomial $q(X) \in \mathbb{Z}_N[X]$ at zero, i.e. $s = q(0) \in \mathbb{Z}_N$. The shares that the parties hold are then just evaluations of $q(X)$ at other positions than at (typically) zero. For simplicity we fix the share of party P_i to be $s_i = q(i)$ for $i \in \{1, \ldots, n\}$. The degree of the secret polynomial, $t = \deg q$, is then the determining factor of qualified sets, i.e. any set Q of size $|Q| > t$ can use Lagrange interpolation in order to reconstruct the secret as follows

$$s = q(0) = \sum_{i \in Q} q(i) L_i^Q = \sum_{i \in Q} q(i) \prod_{j \in Q \setminus \{i\}} \tfrac{j}{j-i} \mod N\,,$$

where L_i^Q are the Lagrange coefficients for the set Q. For a set of size $\leq t$, the value s cannot be reconstructed, and in fact is information-theoretically hidden. We note that since N might be a composite number in our case, we have to ensure that $n < N_1$ [19], where N_1 is the smallest (non-trivial) divisor of N so that any difference $j - i$ is guaranteed to be invertible mod N. In case we want to allow more than N_1 parties, we can work in the subgroup generated by $\mathfrak{g}^{N_1}$, as explained in Sec. 2.1. Throughout the rest of this work, for simplicity, we always assume N to be the size of the subgroup that we are working with, i.e. the largest subgroup $\mathbb{Z}_N \subseteq \mathbb{Z}_{\#\mathrm{cl}(\mathcal{O})}$, for which $N_1 > n$, where N_1 is the smallest non-trivial divisor of N.

In the easiest case, the shares s_i are produced by a trusted third party, called a dealer. However, a Verifiable Secret Sharing (VSS) can easily be turned into a DKG protocol, where the secret polynomial is generated by the parties themselves. A direct way to achieve this is outlined in [25], where each party P_i takes the role of a dealer and generates a polynomial $q^{(i)}(X) \in \mathbb{Z}_N[X]_t$ of the correct degree and privately sends the share $q^{(i)}(j)$ to the party P_j for $j \in \{1, \ldots, n\} \setminus \{i\}$. Then, every party can locally compute their own share by summing all the shares. This implicitly defines the polynomial $q(X) = \sum_{i=1}^{n} q^{(i)}(X)$ and each player's share $q(j) = \sum_{i=1}^{n} q^{(i)}(j)$ as the sum of their shares. The implicitly defined secret is $s = q(0)$, unknown to all players, assuming the honest-majority setting and that the protocol has been executed properly.

Since some (up to t) parties might behave maliciously, parties need a way to verify that the shares they get are correct, i.e. that these really are the shares of a polynomial of degree at most t. It is clear, that if a malicious adversary chooses a polynomial of a degree larger than t, the reconstruction will fail. We therefore define the functionality of a Shamir-based VSS as follows.

Definition 2.1 ([1, Functionality 5.5]). *We define the VSS-functionality*

$$F_{VSS}(q(X), n) = \begin{cases} q(1), \ldots, q(n) & , \ \textit{if } \deg q \leq t, \\ \perp, \ldots, \perp & , \ \textit{otherwise}, \end{cases}$$

taking as input a $q(X) \in \mathbb{Z}_N[X]_t$ of degree t, and each party $i \in \{1, \dots, n\}$ receiving a Shamir share $q(i)$. If $\deg q > t$, the parties output $\perp$ instead.

As mentioned before, the next step in a DKG is the PK computation. In isogeny-based protocols, this means generating one or several elliptic curves, e.g. $E = [s]E_0$, by jointly computing the action $[s]$ in a distributed manner. Since different elliptic curves cannot be combined, the distributed computation of $[s]$ has to be done in a round-robin fashion [19].

Security Requirements of DKG Protocols. Next, we recall the security requirements of DKG protocols based on SSS. We stick to the definitions outlined in [21], using the notation introduced in [6].[6] We denote by $A, B \leftarrow \langle \mathcal{A}^{\mathcal{O}_1}(X) \mid \mathcal{B}^{\mathcal{O}_2}(Y)\rangle$ the joint execution of the DKG protocol by (sets of) parties $\mathcal{A}$ with input X and oracle access to $\mathcal{O}_1$ and $\mathcal{B}$ with input Y and oracle access to $\mathcal{O}_2$, and yielding the output distributions A for $\mathcal{A}$ and B for $\mathcal{B}$, respectively. We define the output distributions of the DKG protocol as follows

$$\mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{O}_1}(X), \mathcal{B}^{\mathcal{O}_2}(Y)) = \left\{(A, B) \middle| A, B \leftarrow \langle \mathcal{A}^{\mathcal{O}_1}(X) | \mathcal{B}^{\mathcal{O}_2}(Y)\rangle \right\} ,$$

and drop the inputs, whenever the input string is empty.

Definition 2.2 (Robust correctness). *A DKG protocol based on SSS between n parties $P_1, \dots, P_n$ is called* correct*, if for any PPT adversary $\mathcal{A}$ and any subset $I \subseteq \{1, \dots, n\}$ of size $|I| > t$ and $n - |I| \leq t$ for any $t < n$, we have that*

$$\Pr\left[\begin{array}{c|c} \nexists f \in \mathbb{Z}_N[x]_t : & \\ E_1 = \dots = E_n = [f(0)]E_0 \,, & A, \{(E_i, s_i)\}_{i \in I} \leftarrow \langle \mathcal{A}^{\mathcal{O}} \mid \{P_i^{\mathcal{O}}\}_{i \in I}\rangle \\ \text{and } \forall i \in I : f(i) = s_i & \end{array}\right] \leq \mathsf{negl}(\lambda) \,.$$

Definition 2.3 (Secrecy). *Let $\mathcal{O}$ be a random oracle. A DKG protocol based on SSS between n parties $P_1, \dots, P_n$ satisfies* secrecy*, if for any PPT adversary $\mathcal{A}$, and any subset $I \subseteq \{1, \dots, n\}$ with $|I| > t$ and $n - |I| \leq t$, there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that the following distributions are indistinguishable*[7]

$$\mathsf{D}_{\mathit{out}}(\mathcal{A}^{\mathcal{O}} | \{P_i^{\mathcal{O}}\}_{i \in I}) \approx \mathsf{D}_{\mathit{out}}(\mathcal{A}^{\mathcal{S}_2} | \mathcal{S}_1(E)) \,.$$

Basically, robust correctness implies, that in the honest-majority setting, the protocol will end in a way that each honest party P_i for $i \in I$ will hold a tuple (E_i, s_i), for which there exists a polynomial $f(X) \in \mathbb{Z}_N[X]_t$, such that $E_i = [f(0)]E_0$ and $s_i = f(i)$, up to negligible probability. Note that the definition is for the case of a single PK, and it is straightforward to generalize it to the case of an extended (structured) PK. Secrecy on the other hand implies, that the adversary $\mathcal{A}$, controlling up to t malicious parties, cannot learn anything about s, other than what it can learn from the public key $E = [s]E_0$.

[6] We emphasize however, that our definition of secrecy is the same as the original one introduced in [21] and thus differs from the "weaker" version presented in [6].

[7] $\mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{O}} | \{P_i^{\mathcal{O}}\}_{i \in I}) = \left\{(A, E_{i^*}) \middle| A, \{(E_i, s_i)\}_{i \in I} \leftarrow \langle \mathcal{A}^{\mathcal{O}} | \{P_i^{\mathcal{O}}\}_{i \in I}\rangle \right\}$ for any $i^* \in I$.

Piecewise Verifiable Proofs. Next, we revisit Piecewise Verifiable Proofs (PVPs), introduced in [6] to make the secret sharing step in CSI-RAShi verifiable. PVPs are ZK proofs for a list of relations $R_0, \ldots, R_n$ with the *same* witness space, where individual statements can be verified independently. For a list of statements $x_0, \ldots, x_n$, a PVP allows one to prove the existence of a witness w such that $(x_i, w) \in R_i$ for any $i \in \{0, \ldots, n\}$. The proof is of the form $(\tilde{\pi}, \{\pi_i\}_{i \in \{0,\ldots,n\}})$, where $(\tilde{\pi}, \pi_0)$ allows verification of x_0 w.r.t. R_0 (called the *main* proof) and π_i for $i \in \{1, \ldots, n\}$ further allows verification of x_i w.r.t. R_i. In particular, in [6], the witnesses constitute elements $f(X) \in \mathbb{Z}_N[X]_t$, i.e. polynomials in the variable X with coefficients defined over $\mathbb{Z}_N$ and of degree at most t. The relations are given as

$$R_0 = \{((E_0, E_1), f(x)) \mid E_1 = [f(0)]E_0 \} \ \wedge \ R_i = \{(x_i, f(x)) | f(i) = x_i\}, \quad (1)$$

for $i = 1, \ldots, n$, and $E_0, E_1 \in \mathcal{E}$, i.e. supersingular elliptic curves defined over a prime field $\mathbb{F}_p$. Note that the witness is the same for all the relations.

CSI-RAShi [6] is an n-party honest-majority DKG protocol, that uses PVPs in its VSS step. The idea is that a party, called the dealer, is in possession of some polynomial $f(X) \in \mathbb{Z}_N[X]_t$, where $t \leq \lfloor \frac{n-1}{2} \rfloor$. The dealer publishes $E_1 = [f(0)]E_0$ and distributes the shares $f(i)$ to parties $P_1, \ldots, P_n$. To prove that these shares are correct, the dealer publishes a PVP for the relations $R_0, R_1, \ldots, R_n$ as described above, and each party P_i can then verify the main statement R_0 as well as the statement R_i related to their share. In the honest-majority setting, if all honest parties agree that their share is correct, they know that they each possess a share of a unique polynomial $f(X)$ of degree t, whose evaluation in 0 is in the commitment $E_1 = [f(0)]E_0$.

We describe the proof generation and verification of their PVP scheme in Algorithms 1 and 2, where $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is a random oracle and $\mathcal{C} : \{0,1\}^* \times \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a commitment scheme that is collapsing [35, Definition 12] and quantum computationally hiding [6, Definition 2]. The authors prove that their PVP scheme is correct, sound and ZK, therefore constitutes a ZK proof system for the individual relations $R_0, \ldots, R_n$. For more details, we refer the reader to the original CSI-RAShi paper [6].

The CSI-RAShi DKG protocol. We quickly outline the building blocks and underlying security rationale of CSI-RAShi [6]. We refer the reader to the original source for more details and an algorithmic description of the protocol. The CSI-RAShi protocol consists of two consecutive phases, the VSS step, and the PK computation. The VSS is executed as outlined in Sec. 2.2, using a Pedersen-type approach [25] for a distributed secret generation. The shares are then verified using PVPs and any inconsistencies are resolved using the complaint resolution protocol introduced in [22]. At the end of the VSS, the parties that have not been disqualified, define the qualified set Q, which also implicitly defines the secret $s = \sum_{i \in Q} f^{(i)}(0)$. In the PK computation step, the parties within the qualified set then engage in a round-robin MPC protocol to compute the PK $E = [s]E_0$ as successive computations of the type $F_i = [f^{(i)}(0)]F_{i-1}$, starting at $F_0 = E_0$ and ending at $F_{|Q|} = E$. Parties prove correctness of their action using

Algorithm 1: PVP.P: The prover of non-interactive PVP [6].

```
Input : A witness polynomial $f(X) \in \mathbb{Z}_N[X]_{\leq t}$,
        a statement $x = ((E_0, E_1), x_1, \cdots, x_n)$.
Output: A non-interactive piecewise proof $\pi$ of the relations in equation (1).
for $j = 1, \ldots, \lambda$ do
    $b_j \leftarrow \mathbb{Z}_N[X]_{\leq t}$ uniformly at random; Set $\hat{E}_j \leftarrow [b_j(0)]E_0$
Sample $y_0, y_0' \leftarrow \{0,1\}^\lambda$ uniformly at random, and set
 $\mathsf{C}_0 \leftarrow \mathcal{C}(\hat{E}_1 \parallel \cdots \parallel \hat{E}_\lambda, y_0)$, and $\mathsf{C}_0' \leftarrow \mathcal{C}(E_0, E_1, y_0')$.
for $i = 1, \ldots, n$ do
    $y_i, y_i' \leftarrow \{0,1\}^\lambda$ uniformly at random and set $\mathsf{C}_i \leftarrow \mathcal{C}(b_1(i) \parallel \cdots \parallel b_\lambda(i), y_i)$
     and $\mathsf{C}_i' \leftarrow \mathcal{C}(x_i, y_i')$
$\mathbf{d} = d_1 \ldots d_\lambda \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$, where $\mathsf{C} = (\mathsf{C}_0, \ldots, \mathsf{C}_n), \mathsf{C}' = (\mathsf{C}_0', \ldots, \mathsf{C}_n')$
for $j = 1, \ldots, \lambda$ do
    $r_j(x) \leftarrow b_j(x) - d_j f(x) \mod N$
return $\tilde{\pi} = (\mathsf{C}, \mathsf{C}', \mathbf{r})$ and $\{\pi_i = (y_i, y_i')\}_{i=0}^n$, where $\mathbf{r} = (r_1, \ldots, r_\lambda)$.
```

Algorithm 2: PVP.V: The verifier of non-interactive PVP [6].

```
Input : An index $i = 0, \ldots, n$, a statement piece $x_i$ of the form
        $x_0 = (E_0, E_1)$ if $i = 0$, or $x_i \in \mathbb{Z}_N$ if $i \neq 0$, as well as a proof piece
        $(\widetilde{\pi}, \pi_i) = ((\mathsf{C}, \mathsf{C}', \mathbf{r}), (y_i, y_i'))$.
Output: true or false
if $\mathsf{C}_i' \neq \mathcal{C}(x_i, y_i')$ then
    return false
$d_1 \ldots d_\lambda \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$
if $i == 0$ then
    for $j = 1, \ldots, \lambda$ do
        if $d_j == 0$ then $\widetilde{E}_j \leftarrow [r_j(0)]E_0$
        else $\widetilde{E}_j \leftarrow [r_j(0)]E_1$
    return $\mathsf{C}_0 == \mathcal{C}(\widetilde{E}_1, \cdots, \widetilde{E}_\lambda, y_0)$
else
    return $\mathsf{C}_i == \mathcal{C}(r_1(i) + d_1 x_i \parallel \cdots \parallel r_\lambda(i) + d_\lambda x_i, y_i)$
```

the ZK proofs introduced in [15, Sec. 3.1]. The share $f^{(i)}(0)$ of any party that misbehaves in the protocol can be reconstructed using Lagrange interpolation, so that the protocol always succeeds in the honest-majority setting.

CSI-RAShi satisfies the robust correctness property of Definition 2.2, relying on the soundness of PVPs. Furthermore, it satisfies a weaker variant of the secrecy property, which unlike Definition 2.3 assumes the input curve E to be uniformly sampled from $\mathcal{E}$, rather than being arbitrary [6, Definition 4]. This modification stems from the fact, that secrecy relies not only on the ZK property of PVPs, but also on the isogeny-based Decisional Diffie-Hellman assumption (DDH) [34, Problem 2.2], which only holds for random inputs.[8]

[8] We note that the authors of [6] use the description of *very hard homogeneous spaces* as introduced by Couveignes [14], where this assumption is called the decisional parallelization problem.

Costs of CSI-RAShi. We discuss the computational and communication costs of the CSI-RAShi protocol. The authors of [6] state the total sequential cost of the protocol as $n+2+(4n+1)\lambda$ isogeny computations, arguing that other costs are negligible in comparison. For direct comparison with the results in Tables 2 and 3, we summarize the costs of CSI-RAShi and its extended and structured versions in Table 1 below.

Table 1. Computational (top) and communication (bottom) cost of CSI-RAShi and its extended and structured versions. The computational costs in terms of isogenies are taken from [8] and [2], respectively. The communication cost for the basic protocol can be recovered from the extended case, by setting $k = 1$. Note that the factor $\frac{1}{2}$ in the C_C-communication cost comes from the λ-bit challenge in the zero-knowledge proofs of [8], which is half the size of a commitment scheme output ($C_C = 2\lambda$).

	Basic	Extended	Structured
T_E	$1+2n\lambda$	$(1+2n\lambda)k$	$1+2n\lambda$
T_I	$n+2+(4n+1)\lambda$	$(n+2)(k-1)+2n\lambda(3k-1)$	$2+n(k-1)+2\lambda\big(n(\sqrt{k-1}+1)^2+1\big)$
T_C	$6n-2$	$(6n-2)k$	$6n-2$
T_H	$3n$	$3nk$	$n(2k+1)$

	Extended	Structured
C_N	$k\lambda(t+2)$	$\lambda(t+1+\sqrt{k-1})$
C_E	$3k$	$k+2$
C_C	$(3n+2+\frac{1}{2})k$	$3n+2+\frac{1}{2}\sqrt{k-1}$

Security assumptions. Next, we recall the security definitions of isogeny-based Decisional Diffie-Hellman assumption (DDH) [34, Problem 2.2], and $(c_0, c_1, \cdots, c_{k-1})$-Vectorization Problem with Auxiliary Inputs ($\mathbb{C}_{k-1}$-VPwAI) [3], that are used in CSI-RAShi [6] and its structured variant [2], respectively.

Definition 2.4 (Decisional Diffie-Hellman assumption). *Distinguish with non-negligible advantage between the distributions* $([a]E, [b]E, [a+b]E)$ *and* $([a]E, [b]E, [c]E)$*, where* a*,* b *and* c *are chosen uniformly at random from* $\mathbb{Z}_N$*.*

Definition 2.5 ($(c_0, c_1, \cdots, c_{k-1})$-Vectorization Problem with Auxiliary Inputs ($\mathbb{C}_{k-1}$-VPwAI)). *Given an element* $E \in \mathcal{E}$ *and the pairs* $(c_i, [c_i x]E)_{i=1}^{k-1}$*, where* $\mathbb{C}_{k-1} = \{c_0 = 0, c_1 = 1, c_2, \ldots, c_{k-1}\}$ *is an exceptional set, find* $x \in \mathbb{Z}_N$*.*

3 Robust DKG for CSIDH Using Bivariate Polynomials

In this section, we introduce a new DKG protocol for CSIDH, based on a VSS using bivariate polynomials [4,33], an idea that was originally introduced in [4]. When comparing to CSI-RAShi [6], the main advantage of this approach is that we can replace the piecewise verifiable proofs in the VSS step with simple comparison operations and further replace the ZK proofs used in the PK computation step with the cheaper piecewise verifiable proofs. This results in a more efficient DKG protocol. On the downside, the protocol has increased the communication cost, mainly due to the nature of the IT-secure VSS, and the number of corrupted parties is restricted to $t < \frac{n}{3}$, rather than honest-majority (i.e. $t < \frac{n}{2}$), as is the case for CSI-RAShi. Throughout this section, we assume that $n < N_1$, where N_1 is the smallest divisor of $N \mid \#\mathrm{cl}(\mathcal{O})$.

3.1 A VSS Based on Bivariate Polynomials

We revisit the VSS approach from [1, 4, 33], using bivariate polynomials. This section follows closely along the lines of [1, Sec. 5]. Yet, since we are working with polynomials over rings, we have to slightly adapt the theorems and point out differences in the proofs, in order to account for this fact.

A bivariate polynomial $S(X, Y) \in \mathbb{Z}_N[X, Y]$ of degree t is a polynomial over variables X and Y, each of which has degree at most t, i.e. we can write it as

$$S(X, Y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} X^i Y^j \, .$$

We write $S(X, Y) \in \mathbb{Z}[X, Y]_t$. The idea behind using bivariate polynomials in the BGW VSS [4] is related to the following Theorem, adapted from [1, Claim 5.2].

Theorem 3.1. *Let $f_1(X), \ldots, f_{t+1}(X) \in \mathbb{Z}_N[X]$ be polynomials of degree t. Then there exists a unique bivariate polynomial $S(X, Y) \in \mathbb{Z}_N[X, Y]$ of degree t such that for every $k = 1, \ldots, t + 1$, it holds that $S(X, k) = f_k(X)$.*

Proof. The proof is identical to the proof of Claim 5.2 of [1], which uses Lagrange interpolation over a finite field to construct $S(X, Y)$ from the evaluations $S(X, k)$. Since $t + 1 < N_1$, the Lagrange interpolation polynomials are well-defined over the ring $\mathbb{Z}_N$ and the proof works analogously. $\square$

Thus every degree-t bivariate polynomial can be interpolated from $t+1$ univariate degree-t polynomials in the same way that every degree-t univariate polynomial can be interpolated from $t + 1$ points. Conversely, the evaluation of a bivariate polynomial in one of its variables defines a univariate polynomial (of the same degree), in the same way as evaluating a univariate polynomial defines a point. Similarly, if less than $t + 1$ univariate polynomials are known, the overarching bivariate polynomial is IT hidden as the following theorem suggests.

Theorem 3.2. *Let $I \subset \{1, \ldots, n\}$ with $|I| \leq t$ and let $f, g \in \mathbb{Z}_N[X]_t$ with $f(i) = g(i)$ for all $i \in I$. Then for any two bivariate polynomials $S_1(X, Y), S_2(X, Y) \in \mathbb{Z}_N[X, Y]_t$ with $S_1(0, Y) = f(Y)$ and $S_2(0, Y) = g(Y)$, the two sets*

$$\{i, S_1(X, i), S_1(i, Y)\}_{i \in I} \quad \text{and} \quad \{i, S_2(X, i), S_2(i, Y)\}_{i \in I}$$

are indistiguishable.

Proof. See proof of Claim 5.4 of [1]. $\square$

Bivariate polynomials can be used in VSSs as follows [4]. To share a secret x between n parties $P_1, \dots, P_n$, one party, called the *dealer*, samples a random polynomial $q(Z) \in \mathbb{Z}_N[Z]_t$ such that $q(0) = x$. Then the dealer samples a random bivariate polynomial $S(X, Y) \in \mathbb{Z}_N[X, Y]_t$ with the constraint that $S(0, Z) = q(Z)$, so in particular $S(0, 0) = x$. In order to distribute x among the n parties, the dealer sets

$$f_i(X) = S(X, i) \quad \text{and} \quad g_i(Y) = S(i, Y)\,,$$

then sends $f_i(X)$ and $g_i(Y)$ privately to party P_i for $i = 1, \dots, n$. Each party can now construct their share as $x_i = f_i(0) = S(0, i)$, which allows to recover $x = S(0, 0)$ via Lagrange interpolation. The true advantage of using bivariate polynomials manifests itself in the share verification step, which relies on the following theorem.

Theorem 3.3. *Let $K \subseteq \{1, \dots, n\}$ be a set of indices with $|K| \geq t + 1$ and let $\{f_k(X), g_k(X)\}_{k \in K}$ be a set of pairs of polynomials in $\mathbb{Z}_N[X]_t$. If $f_i(j) = g_j(i)$ holds for every $i, j \in K$, then there exists a unique bivariate polynomial $S(X, Y) \in \mathbb{Z}_N[X, Y]_t$, such that for every $k \in K$, $f_k(X) = S(X, k)$ and $g_k(Y) = S(k, Y)$.*

Proof. This proof works analogous to the proof of Claim 5.3 of [1], by using the uniqueness of $S(X, Y)$ guaranteed by Theorem 3.1 (instead of [1, Claim 5.2]). □

Thus, in order to verify the correctness of their shares, each pair of parties P_i, P_j simply checks that $f_i(j) = g_j(i)$ and $g_i(j) = f_j(i)$. If all these tests succeed, then parties know that their shares are consistent with a single bivariate polynomial and that the sharing was successful. In the converse case, if some of these checks do not succeed, the players engage in the following steps to resolve the conflict [1].

1. If for a player P_i, the checks $f_i(j) \overset{?}{=} g_j(i)$ or $g_i(j) \overset{?}{=} g_j(i)$ do not succeed, then P_i broadcasts a complaint by disclosing $(i, j, f_i(j), g_i(j))$. As a response, the dealer reveals $(i, f_i(X), g_i(Y))$. Then each other player P_k evaluates the complaints as below. Whenever players are satisfied with the complaint resolution, they broadcast `consistent`, otherwise they don't.
   (a) If, for every *joint complaint*, e.g. $(i, j, f_i(j), g_i(j))$ by P_i and $(j, i, f_j(i), g_j(i))$ by P_j, the dealer does not reveal $(i, f_i(X), g_i(Y))$ nor $(j, f_j(X), g_j(Y))$, jump to step 2. (without broadcasting `consistent`), otherwise continue.
   (b) If there exists a response $(k, f_k(X), g_k(X))$, P_k accepts this as its new shares, then jumps to step 2. (without broadcasting `consistent`), otherwise continue.
   (c) For any other response, verify if the polynomials revealed by the dealer indeed do not satisfy the necessary checks. If they don't, jump to step 2. (without broadcasting `consistent`), otherwise continue.
   (d) Broadcast the message `consistent`.
2. If at least $n - t$ parties have broadcasted `consistent`, then each party outputs its share $x_i = f_i(0)$, otherwise return $\perp$.

Theorem 3.4. *For $t < n/3$, the secret sharing protocol, along with the conflict resolution procedure described above, implements F_{VSS} in a correct and secure way, for a static malicious adversary (which might include the dealer).*

Proof. We refer the reader to the proof of Theorem 5.7 of [1]. The proof here works the same way, except that references to Claims 5.3 and 5.4 should be substituted with Theorems 3.3 and 3.2, respectively. Furthermore, the values $\alpha_1, \ldots, \alpha_n$, at which the polynomials are evaluated in these proofs, should be chosen from the subset $\{1, \ldots, N_1 - 1\}$. For simplicity, we choose $\{1, \ldots, n\}$. □

3.2 Robust Distributed Generation of $[x]E_0$

We can easily extend the protocol described in the previous section to a distributed VSS by using the approach from [25]. There, each party P_i involved in the protocol individually takes the role of the dealer and constructs and distributes shares of its secret $x^{(i)}$ to each other party. The final secret is then simply the sum of all the secrets and each party's share is the sum of all the shares. Yet, if parties misbehave (as dealers or as receiving parties), the checks described in the previous section will uncover this and parties will be disqualified. In the end, the secret and secret shares are then actually defined as the sum of the respective elements of all the *qualified players*. After a shared secret x was implicitly defined in the *VSS step*, the parties engage in a *PK computation step* to compute the PK $[x]E_0$. Thus our DKG protocol's steps follow a two step approach, along the lines of [22] or [6]. In contrast to those protocols however, the parties do not have to commit to their shares in the first phase, but rather resort to the pairwise comparisons described in the previous subsection. We present our protocol in Fig. 1 and prove the following theorem in App. A.

Theorem 3.5. *If PVPs are sound, then the DKG protocol of Fig. 1 is correct and robust. If PVPs are zero-knowledge, then the DKG protocol satisfies the secrecy property.*

Below, we explain some details of the steps in Fig. 1.

VSS Step. We assume each party P_i wants to share a secret $x^{(i)}$. The distribution of $x^{(i)}$ is achieved by sampling a random univariate polynomial $q^{(i)}(Z) \in \mathbb{Z}_N[Z]_t$ with $x^{(i)} = q^{(i)}(0)$ and a random bivariate polynomial $S^{(i)}(X,Y) \in \mathbb{Z}_N[X,Y]_t$ with $S^{(i)}(0,Z) = q^{(i)}(Z)$. Then P_i sets

$$f_j^{(i)}(X) = S^{(i)}(X,j) \quad \text{and} \quad g_j^{(i)}(Y) = S^{(i)}(j,Y)$$

for $j = 1, \ldots, n$ and sends $f^{(j)}(X)$ and $g^{(j)}(Y)$ privately to party P_j for all j. The parties engage in the protocol to verify the correctness of their shares. In case the checks fail, i.e. $\perp$ is returned while a player P_k is the dealer, then P_k will be disqualified and the protocol continues without the inputs by P_k. Thus, after the consistency check step, there will be a set $Q \subseteq \{1, \ldots, n\}$ of qualified parties which implicitly defines the shared secret x as

$$x = \textstyle\sum_{i \in Q} x^{(i)} = \sum_{i \in Q} S^{(i)}(0,0) = \sum_{i \in Q} q^{(i)}(0) = q(0)\,.$$

Verifiable Secret Sharing:

1. For $i = 1, \dots, n$, player P_i
   (a) samples $q^{(i)}(Z) \leftarrow \mathbb{Z}_N[Z]_t$ and sets $x^{(i)} = q^{(i)}(0)$,
   (b) samples $S^{(i)}(X,Y) \leftarrow \mathbb{Z}_N[X,Y]_t$ with $S^{(i)}(0,Z) = q^{(i)}(Z)$,
   (c) for $j = 1, \dots, n$, defines $f_j^{(i)}(X) = S^{(i)}(X,j)$ and $g_j^{(i)}(Y) = S^{(i)}(j,Y)$ and sends $\{f_j^{(i)}(X), g_j^{(i)}(Y)\}$ privately to party P_j.
2. For $k = 1, \dots, n$, each pair of players P_i, P_j checks that $f_i^{(k)}(j) = g_j^{(k)}(i)$ and $g_i^{(k)}(j) = f_j^{(k)}(i)$. Whenever one of these checks fails, the concerned player runs the conflict resolution procedure described in Sec. 3.1. In case the procedure outputs $\perp$, the dealer P_k of the concerned polynomials is disqualified, otherwise the protocol continues normally.
3. In the end, all the honest players agree on the same set of qualified players $Q \subseteq \{1, \dots, n\}$, and the shared secret key x is given as the sum of the individual secrets of the qualified players $x = \sum_{i \in Q} x^{(i)}$, while the parties' shares of x can be constructed as $x_j = \sum_{i \in Q} f_j^{(i)}(0)$.

Computing the Public Key:

4. Let for simplicity $Q = \{1, \dots, n'\}$. In a round-robin way, the qualified players now compute $F_i \leftarrow [x^{(i)}]F_{i-1}$ where $F_0 = E_0$. At each step, using Algorithm 1, player P_i further creates and publishes a PVP proof
$$\pi^{(i)} = (\tilde{\pi}^{(i)}, \pi_1^{(i)}, \dots, \pi_{n'}^{(i)}) \leftarrow \mathrm{PVP.P}\big((F_{i-1}, F_i); q^{(i)}(Z)\big)$$
which includes a main proof $(\tilde{\pi}^{(i)}, \pi_0^{(i)})$ as well as individual proof pieces $\pi_j^{(i)}$ for each other player P_j.
5. Using Algorithm 2, each other player P_j verifies both $\mathrm{PVP.V}(j, f_j^{(i)}(0), \tilde{\pi}^{(i)}, \pi_j^{(i)})$ and $\mathrm{PVP.V}(0, (F_{i-1}, F_i), \tilde{\pi}^{(i)}, \pi_0^{(i)})$. Whenever a verification of $\pi^{(i)}$ fails, the verifier P_j broadcasts $f_j^{(i)}(X)$. All other parties verify correctness of $f_j^{(i)}(X)$ as in step 2. If it is correct, since there are at least $t+1$ honest players, they will be able to reconstruct $q^{(i)}(0)$, compute F_i and proceed with the protocol (and potentially disqualify P_i). Otherwise, if the checks of $f_j^{(i)}(0)$ fail, the complaint can be ignored (or P_j disqualified). In the latter case, the shares of P_j can also be reconstructed by the at least $t+1$ honest players.
6. At the end of the round-robin, the parties return the public key $F_{n'} = [x]E_0$.

Fig. 1. The DKG protocol for a single public key $[x]E_0$.

Each party can derive their share x_j of x as

$$x_j = q(j) = \textstyle\sum_{i \in Q} q^{(i)}(j) = \sum_{i \in Q} f_j^{(i)}(0)\,,$$

and $x = q(0)$ can be recovered by any subset of at least $t+1$ parties.

PK Computation Step. In this step, the parties in Q engage in a round-robin protocol for computing the public key $[x]E_0$. For simplicity, we assume $Q = \{1, \dots, n'\}$. Then, at step i, party P_i will compute

$$F_{i-1} \mapsto F_i = [x^{(i)}]F_{i-1}\,,$$

where F_0 is some starting curve. At the end of the round-robin,

$$F_{n'} = [x^{(n')}] \cdots [x^{(1)}]F_0 = \left[\sum_{i \in Q} x^{(i)}\right] F_0 = [x]F_0$$

is the public key. The only thing left to do for each player P_i is to prove that they used the correct $x^{(i)} = q^{(i)}(0)$. Since the other parties P_j each possess a share $f_j^{(i)}(0) = S^{(i)}(0, j) = q^{(i)}(j)$, the dealer P_i can convince them of having used the correct action by proving the following relation

- $F_i = [q^{(i)}(0)]F_{i-1}$ and
- for $j \in Q$, player P_j possesses the share $q^{(i)}(j)$ of $q^{(i)}(0)$.

for the witness polynomial $q^{(i)}(Z) \in \mathbb{Z}_N$. It turns out that this is exactly the language that is proved using the PVPs from Sec. 2.2, originally introduced in [6]. Thus a player P_i can convince the other players of having acted with $q^{(i)}(0)$, by relating it to the shares $q^{(i)}(j) = f_j^{(i)}(0)$ distributed in the VSS step.

Extended Public Keys. The DKG protocol in Fig. 1 can be easily extended to sample k public keys, which is required in protocols like CSI-FiSh [7]. This can simply be done by generating multiple polynomials $S_1^{(i)}, \ldots, S_k^{(i)}$ for the respective secrets $x^{(1,i)}, \ldots, x^{(k,i)}$. In the PK computation step, parties then compute the curves $F_i^1 \leftarrow [x^{(1,i)}]F_{i-1}^1, \ \ldots, \ F_i^k \leftarrow [x^{(k,i)}]F_{i-1}^k$, and prove correctness at each step. In this case, it requires running k independent PVPs.

4 Robust DKG for Structured Public Keys

We recall that for a given secret $x \in \mathbb{Z}_N$, an SPK [2, 3] has the form $\{E_i = [c_i x]E_0\}_{i=1}^k$, where $c_i \in \mathbb{C}_k$ and $\mathbb{C}_k = \{c_1 = 1, c_2, \cdots, c_k\}$ is a public (super)exceptional set, see also Sec. 2.1. In this section, we present a new variant of the DKG protocol in Fig. 1, which would allow a set of parties to sample an SPK in a distributed manner.

VSS Step. Since an SPK has only one secret key, we need to execute the VSS step only once. This is done exactly as in Fig. 1.

SPK Computation Step. In the SPK computation step, for a given superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \ldots, c_k\}$, each party P_i has to compute

$$F_i^1 \leftarrow [x^{(i)}]F_{i-1}^1, \ F_i^2 \leftarrow [c_2 x^{(i)}]F_{i-1}^2, \quad \ldots, \quad F_i^k \leftarrow [c_k x^{(i)}]F_{i-1}^k \, ,$$

and give a proof that they updated all the curves in the SPK with correct factors of the secret key $x^{(i)} = q^{(i)}(0)$ shared with other parties. Since the other parties P_j each possess a share $f_j^{(i)}(0) = S^{(i)}(0, j) = q^{(i)}(j)$ from the VSS step, party P_i needs to convince them that it used the correct action by proving the following relation:

- For a public superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \dots, c_k\}$:
  $F_i^1 = [c_1 q^{(i)}(0)]F_{i-1}^1 \wedge \dots \wedge F_i^k = [c_k q^{(i)}(0)]F_{i-1}^k$, and
- for $j \in Q$, player P_j possesses the share $q^{(i)}(j)$ of $q^{(i)}(0)$,

where $q^{(i)}(Z) \in \mathbb{Z}_N$ is the witness polynomial. To prove the above relations, we propose a new PVP scheme, which we call Structured PVP (SPVP), that can be considered as an extension of the one proposed in [6, Algorithms 3 and 4], and which will allow us to prove the computation of SPKs in our DKG.

Structured PVP (SPVP). We define the following list of relations $R = (R_0, \dots, R_n)$, whose common witness space is $\mathbb{Z}_N[X]_{\leq t}$, the set of polynomials over $\mathbb{Z}_N$ of degree at most t:

$$R_0 = \{((\mathbb{C}_k, F_1, F_1', \dots, F_k, F_k'), f(x)) \mid (F_l' = [c_l f(0)]F_l)_{l=1}^k \},$$
$$\forall i = 1, \dots, n : R_i = \{(x_i, f(x)) | f(i) = x_i\}\,. \tag{2}$$

A statement x_0 for R_0 consists of a public superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \dots, c_k\}$, and a set of curves $(F_1, F_1', \dots, F_k, F_k') \in \mathcal{E}^{2k}$. Just as in the original PVPs, a statement x_i for the relations $\{R_i\}_{i=1,\cdots,n}$ is an element of $\mathbb{Z}_N$.

The description of non-interactive SPVP for relations of the above form are presented in Algorithms 3 and 4. The algorithms use a random oracle $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^\lambda$, and a non-interactive commitment scheme $\mathcal{C} : \{0,1\}^\star \times \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$, where λ is the security parameter. In App. A, we prove the following Theorem.

Theorem 4.1. *Assuming that the commitment scheme $\mathcal{C}$ is collapsing and quantum computationally hiding, the described non-interactive PVP for structured public keys (in Algorithms 3 and 4) is complete, sound, and ZK in the QROM for the list of relations given in equation (2).*

Efficiency. As it can be seen, by running $\mathsf{SPVP}.P$ a party needs to compute λk group actions, query once to the random oracle $\mathcal{H}$, and $2(n+1)$ times to the commitment scheme $\mathcal{C}$. Similarly, by running $\mathsf{SPVP}.V$ a verifier needs to query once to the random oracle $\mathcal{H}$, and twice to the commitment scheme $\mathcal{C}$, and only in the case $i = 0$ it needs to compute λk group actions.

Ternary Challenge Space. The challenge space of the (structured) PVP schemes is binary. As a result, the PVP scheme constructed in [6] and its new variant proposed above need to be repeated λ times to achieve the soundness error $\frac{1}{2^\lambda}$. Unlike in CSI-RAShi, in our DKG protocols, the first party generates a PVP proof starting from the base curve E_0. Since the class group enjoys a symmetry around the elliptic curve E_0, this allows us to use the twist trick introduced in [7] and extend the challenge space of the PVP schemes to $\{-1, 0, 1\}$, with minimal changes to the descriptions in Algorithms 3 and 4. The resulting soundness error rate becomes $\frac{1}{3}$ and the number of repetitions reduces to $\lambda' := \lceil \lambda / \log_2 3 \rceil$ to achieve the soundness error $\frac{1}{2^\lambda}$. This results in a noticeable gain, especially if the number of parties is small.

Algorithm 3: $\mathsf{SPVP}.P$: The prover of non-interactive Structured PVP.

```
Input  : A witness polynomial $f(X) \in \mathbb{Z}_N[X]_{\le t}$,
         a statement $x = ((\mathbb{C}_k, F_1, F_1', \dots, F_k, F_k'), x_1, \cdots, x_n))$.
Output: A non-interactive piecewise proof $\pi$ of the relations in equation (2).
Parse $\mathbb{C}_k = \{c_1, c_2, \dots, c_k\}$.
for $j = 1, \dots, \lambda$ do
    $b_j \leftarrow \mathbb{Z}_N[X]_{\le t}$ uniformly at random
    $\hat{F}_j^1 \leftarrow [c_1 b_j(0)]F_1$ , . . . , $\hat{F}_j^k \leftarrow [c_k b_j(0)]F_k$
Sample $y_0, y_0' \leftarrow \{0,1\}^\lambda$ uniformly at random, and set
$\mathsf{C}_0 \leftarrow \mathcal{C}(\hat{F}_1^1, \cdots, \hat{F}_1^k \parallel \cdots \parallel \hat{F}_\lambda^1, \dots, \hat{F}_\lambda^k, y_0)$,
$\mathsf{C}_0' \leftarrow \mathcal{C}(F_1, F_1' \parallel \cdots \parallel F_k, F_k', y_0')$.
for $i = 1, \dots, n$ do
    $y_i, y_i' \leftarrow \{0,1\}^\lambda$ uniformly at random; set $\mathsf{C}_i \leftarrow \mathcal{C}(b_1(i) \parallel \cdots \parallel b_\lambda(i), y_i)$;
     and $\mathsf{C}_i' \leftarrow \mathcal{C}(x_i, y_i')$
$\mathbf{d} = d_1 \dots d_\lambda \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$, where $\mathsf{C} = (\mathsf{C}_0, \dots, \mathsf{C}_n), \mathsf{C}' = (\mathsf{C}_0', \dots, \mathsf{C}_n')$
for $j = 1, \dots, \lambda$ do
    $r_j(x) \leftarrow b_j(x) - d_j f(x) \mod N$
return $\tilde{\pi} = (\mathsf{C}, \mathsf{C}', \mathbf{r})$ and $\{\pi_i = (y_i, y_i')\}_{i=0}^n$, where $\mathbf{r} = (r_1, \dots, r_\lambda)$.
```

Algorithm 4: $\mathsf{SPVP}.V$: The verifier of non-interactive Structured PVP.

```
Input  : An index $i = 0, \dots, n$, a statement piece $x_i$ of the form
         $x_0 = (\mathbb{C}_k, F_1, F_1', \cdots, F_k, F_k')$ if $i = 0$, or $x_i \in \mathbb{Z}_N$ if $i \neq 0$, as well as
         a proof piece $(\widetilde{\pi}, \pi_i) = ((\mathsf{C}, \mathsf{C}', \mathbf{r}), (y_i, y_i'))$.
Output: true or false
if $\mathsf{C}_i' \neq \mathcal{C}(x_i, y_i')$ then
    return false
$d_1 \dots d_\lambda \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$
if $i == 0$ then
    for $j = 1, \dots, \lambda$ do
        if $d_j == 0$ then $\widetilde{F}_j^1 \leftarrow [c_1 r_j(0)]F_1$, . . . , $\widetilde{F}_j^k \leftarrow [c_k r_j(0)]F_k$
        else $\widetilde{F}_j^1 \leftarrow [c_1 r_j(0)]F_1'$, . . . , $\widetilde{F}_j^k \leftarrow [c_k r_j(0)]F_k'$
    return $\mathsf{C}_0 == \mathcal{C}(\widetilde{F}_1^1, \cdots, \widetilde{F}_1^k \parallel \cdots \parallel \widetilde{F}_\lambda^1, \cdots, \widetilde{F}_\lambda^k, y_0)$
else
    return $\mathsf{C}_i == \mathcal{C}(r_1(i) + d_1 x_i \parallel \cdots \parallel r_\lambda(i) + d_\lambda x_i, y_i)$
```

Construction of the DKG Protocol. Fig. 2 describes the construction of our proposed robust DKG protocol for structured public keys. The protocol uses the distributed VSS scheme described in Sec. 3.1, and the non-interactive Structured PVP ($\mathsf{SPVP}.P, \mathsf{SPVP}.V$), given in Algorithms 3 and 4, as a subroutine. We prove the following Theorem in App. A.

Theorem 4.2. *If structured PVPs are sound, then the DKG protocol of Fig. 2 to generate structured public keys is correct and robust. If structured PVPs are ZK, then the DKG protocol satisfies the secrecy property.*

Verifiable Secret Sharing:
This is done in the same way as Fig. 1. The players furthermore agree on an exceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \ldots, c_k\}$.

Computing the Structured Public Key:

4. Let for simplicity $Q = \{1, \ldots, n'\}$. Given a superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \ldots, c_k\}$, a qualified set of parties engage in a round-robin protocol, and party P_i computes
$$F_i^1 \leftarrow [x^{(i)}]F_{i-1}^1,\ F_i^2 \leftarrow [c_2 x^{(i)}]F_{i-1}^2,\quad \ldots,\quad F_i^k \leftarrow [c_k x^{(i)}]F_{i-1}^k\,,$$
where $F_0^1 = F_0^2 = \cdots = F_0^k = E_0$. At each step, player P_i further uses the non-interactive Structured PVP $(\mathsf{SPVP}.P, \mathsf{SPVP}.V)$, given in Algorithms 3 and 4, and creates and publishes a structured PVP proof,
$$\pi^{(i)} = (\tilde{\pi}^{(i)}, \pi_1^{(i)}, \ldots, \pi_{n'}^{(i)}) \leftarrow \text{SPVP.P}\big(\mathbb{C}_k, (F_{i-1}^1, F_i^1, \cdots, F_{i-1}^k, F_i^k); q^{(i)}(Z)\big)$$
which includes a main proof $(\tilde{\pi}^{(i)}, \pi_0^{(i)})$ as well as individual proof pieces $\pi_j^{(i)}$ for each other player P_j.
5. Each other player P_j verifies both $\text{SPVP.V}(j, f_j^{(i)}(0), \tilde{\pi}^{(i)}, \pi_j^{(i)})$ and $\text{SPVP.V}(0, (\mathbb{C}_k, (F_{i-1}^1, F_i^1, \cdots, F_{i-1}^k, F_i^k)), \tilde{\pi}^{(i)}, \pi_0^{(i)})$. Whenever a verification of $\pi^{(i)}$ fails, the verifier P_j broadcasts $f_j^{(i)}(X)$. All other parties verify correctness of $f_j^{(i)}(X)$ as in step 2 of Fig. 1. If it is correct, since there are at least $t+1$ honest players, they will be able to reconstruct $q^{(i)}(0)$, compute F_i and proceed with the protocol (and potentially disqualify P_i). Otherwise, if the checks of $f_j^{(i)}(0)$ fail, the complaint can be ignored (or P_j disqualified). In the latter case, the shares of P_j can also be reconstructed by the at least $t+1$ honest players.
6. At the end of the round-robin, the parties return the *structured* public key
$$F_{n'}^1 = [x]E_0,\ F_{n'}^2 = [c_2 x]E_0,\ \cdots\ , F_{n'}^k = [c_k x]E_0\,.$$

Fig. 2. The DKG protocol for structured public keys.

5 Efficiency of the DKG Protocols and Optimizations

We summarize the computational and communication costs of our DKG protocols in Tables 2 and 3. We express the computational cost as the *sequential* runtime of the protocol steps, i.e. the total runtime from start to finish, including when some of the parties are idle. The communication cost is expressed in terms of *outgoing* communication per party. These costs are established in detail in the App. B, where we also discuss optimizations in order to minimize them.

Comparison of Extended and Structured Cases. We end this section with a comparison between DKGs building extended or structured public keys in terms of computational and communication costs. A direct comparison of Tables 2 and 3 reveals that while extended DKGs scale linearly with k in every term, structured DKGs only scale with k in the number of isogeny computations

Table 2. Computational costs of the basic, extended and structured DKG in terms of polynomial evaluations T_E, isogeny computations T_I and calls to the commitment scheme T_C and random oracle T_H. The cost represents the total sequential cost of the protocol, including idle times by the parties. For compactness, we do not consider the twist trick described in the previous section; it can easily be reinstated by substituting the terms of the form $n\lambda$ to $\lambda' + (n-1)\lambda$ in the factors of T_E and T_I. The impact of the twist trick is further discussed in App. B. We define $\chi_{n,k} = \lceil \frac{k}{n} \rceil - \lfloor \frac{k}{n} \rfloor$.

	Basic DKG	Extended DKG	Structured DKG
T_E	$2(n-1)^2 + n\lambda(n+2)$	$2(n-1)^2k + n\lambda(n\lceil \frac{k}{n} \rceil + k + \chi_{n,k})$	$2(n-1)^2 + n\lambda(2n+1)$
T_I	$2n\lambda + n$	$n\lambda(k + \chi_{n,k}) + n\lceil \frac{k}{n} \rceil$	$n\lambda(k + \chi_{n,k}) + n\lceil \frac{k}{n} \rceil$
T_C	$2n(n+3)$	$2n\big(\lceil \frac{k}{n} \rceil(n-1) + 2k + 2\chi_{n,k}\big)$	$2n(3n-1)$
T_H	$2n$	$n(k + \chi_{n,k})$	n^2

Table 3. Communication costs of the extended and structured DKG in terms of the information contained in elements of $\mathbb{Z}_N$ and $\mathcal{E}$, i.e. C_N and C_E respectively, and of the output of our commitment scheme C_C. The cost represents the outgoing cost per party. The cost of the basic DKG immediately follows by setting $k = 1$ in either case. We regain the twist trick by the substitution $\lambda \mapsto \lambda'$ in the costs of C_N.

	Extended DKG	Structured DKG
C_N	$2k(n-1)(n+t-1) + k\lambda(t+1)$	$2(n-1)(n+t-1) + \lambda(t+1)$
C_E	k	k
C_C	$k(3n+2)$	$3n+2$

T_I and the number of shared elliptic curves C_E. We summarize the trends for varying k and n in Figs. 3 and 4 below, and discuss more details in the App. B. Figs 3 and 4 also compare our results with CSI-RAShi [6] with extended public keys (using the optimizations from [2, Sec. 4.3]), as well as the recently proposed *structured CSI-RAShi* from [2, Sec. 5.3].[9] Both of these schemes are honest-majority Shamir secret sharing based DKG protocols in the CSIDH setting. To our current knowledge, Structured CSI-RAShi represents the most efficient isogeny-based DKG in the literature for (structured) public keys of size $k > 1$.

[9] We note that [2] also analyzes extended and structured versions of the Sashimi DKG [15], which is a full-threshold DKG. The authors in [2] show that the communication and computational costs of this DKG are basically the same as for CSI-RAShi, up to some barely noticeable constant factors. We therefore omit their analysis here.

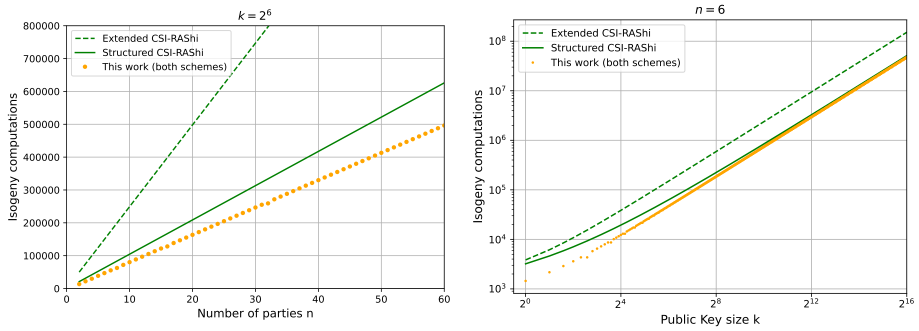

Fig. 3. Computational cost in terms of isogeny computations for the CSIDH-512 parameter set, shown as a function of the number of parties n (left; for $k = 2^6$) and as a function of the public key size k (right; for $n = 6$). We can see that our protocols outperform the protocols from [2,6], most notably for large n or low k. For asymptotically large k, our protocols coincide with structured CSI-RAShi in the number of isogeny computations.

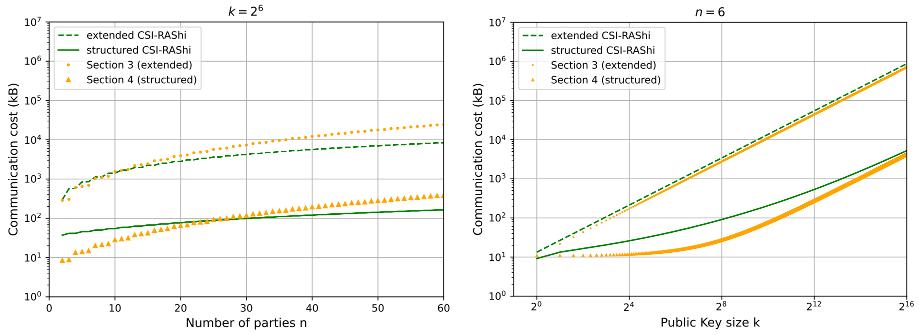

Fig. 4. Communication cost in kilobytes for the CSIDH-512 parameter set, shown as a function of the number of parties n (left; exemplified for $k = 2^6$) and as a function of the public key size k (right; for $n = 6$).
We compare our results to the extended and structured CSI-RAShi from [2]. We choose t as the largest integer $< n/3$ in the bivariate case and $< n/2$ for (structured) CSI-RAShi. The communication cost of our schemes is lower for $n < 10$ (extended case) and $n < 25$ (structured case), but the worse asymptotic behavior becomes noticeable above those values.

6 Conclusion

In this paper, we presented two efficient robust DKG protocols in the CSIDH setting, based on secret sharing with bivariate polynomials, which outperform current alternatives [2,6] in terms of computational cost.

The first protocol allows a set of parties to sample a public key like $[x]E_0$ or $([x_1]E_0, \cdots, [x_k]E_0)$ and obtain a Shamir share of each of the secret keys. Such public keys are used in isogeny-based cryptosystems (e.g. [19,28]) and signature schemes (e.g. [7]), respectively. The second protocol allows a set of parties to sample a structured public key, e.g. a PK of the form $([c_1x]E_0, \cdots, [c_kx]E_0)$, as used in the CSI-SharK signature scheme [2], and obtain a Shamir share of x. Both protocols are secure in the QROM and achieve IT security in the VSS step. However, compared to current alternatives, our protocols generally require more communication between parties (for a high number of parties) and a higher number of honest participants (more than two-thirds) during the VSS step.

In Sec. 5 we showed that for generating a single PK, $[x]E_0$, our first protocol is at least two times faster than the state-of-the-art scheme CSI-RAShi [6]. In cases where the number of parties is small, the improvement is even higher, e.g. for $n = 2$, our first DKG protocol is about 3 times faster and also requires 27% less communication. When generating extended or structured public keys, our protocols are also faster than the currently most efficient ones [2,6]. For instance, using our protocol, 4 parties can sample a PK of size 4kB (i.e., $k = 2^6$), for CSI-FiSh [7] and CSI-SharK [2], respectively 3.4 and 1.7 times faster than the current DKG protocols.

As a building block for our second DKG protocol, in Sec. 4, we presented a new PVP scheme specifically for SPKs, which we think can be of independent interest for future protocols using SPK.

Acknowledgments

This work has been supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract No. HR001120C0085, by the FWO under an Odysseus project GOH9718N, by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101020788 - Adv-ERC-ISOCRYPT), by CyberSecurity Research Flanders with reference number VR20192203, by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), by the Spanish Government under project PRODIGY (TED2021-132464B-I00), and by the Madrid Regional Government under project BLOQUES (S2018/TCS-4339). The last two projects are co-funded by European Union EIE, and Next Generation EU/PRTR funds.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ERC, DARPA, the US Government, the Spanish Government, Cyber Security Research Flanders or the FWO. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

1. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, January 2017.

2. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. CSI-SharK: CSI-FiSh with sharing-friendly keys. Cryptology ePrint Archive, Report 2022/1189, 2022. https://eprint.iacr.org/2022/1189.
3. Karim Baghery, Daniele Cozzo, and Robi Pedersen. An isogeny-based ID protocol using structured public keys. In M.B. Paterson, editor, *Cryptography and Coding - 18th IMA International Conference, IMACC 2021, Oxford, UK, December 14-15, 2021, Proceedings*, volume 13129 of *Lecture Notes in Computer Science*, pages 179–197. Springer, 2021.
4. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.
5. Daniel Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *arXiv preprint arXiv:2003.10118*, 2020.
6. Ward Beullens, Lucas Disson, Robi Pedersen, and Frederik Vercauteren. CSI-RAShi: Distributed key generation for CSIDH. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*, volume 12841 of *Lecture Notes in Computer Science*, pages 257–276. Springer, 2021.
7. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.
8. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: efficient isogeny based signatures through class group computations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 227–247. Springer, 2019.
9. Fabio Campos and Philipp Muth. On actively secure fine-grained access structures from isogeny assumptions. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 375–398. Springer, 2022.
10. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH (preliminary version). Cryptology ePrint Archive, Report 2022/975, 2022. https://eprint.iacr.org/2022/975.
11. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 395–427. Springer, 2018.
12. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.
13. Denis X Charles, Kristin E Lauter, and Eyal Z Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, 2009.
14. Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptol. ePrint Arch.*, 2006:291, 2006.

15. Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 169–186, Paris, France, April 15–17, 2020. Springer, Heidelberg, Germany.
16. Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.
17. Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Report 2011/506, 2011. `https://eprint.iacr.org/2011/506`.
18. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.
19. Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 187–212, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.
20. Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: scaling the csi-fish. *IACR Cryptol. ePrint Arch.*, page 58, 2023.
21. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and efficient sharing of RSA functions. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 157–172, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.
22. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007.
23. Alexei Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *Electron. Colloquium Comput. Complex.*, TR96-003, 1996.
24. Luciano Maino and Chloe Martindale. An attack on SIDH with arbitrary starting curve. Cryptology ePrint Archive, Report 2022/1026, 2022. `https://eprint.iacr.org/2022/1026`.
25. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany.
26. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.
27. Damien Robert. Breaking SIDH in polynomial time. Cryptology ePrint Archive, Report 2022/1038, 2022. `https://eprint.iacr.org/2022/1038`.
28. Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptol. ePrint Arch.*, 2006:145, 2006.
29. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

30. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
31. Carl Siegel. Über die classenzahl quadratischer zahlkörper. *Acta Arithmetica*, 1(1):83–86, 1935.
32. Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009.
33. Douglas R. Stinson and Ruizhong Wei. Unconditionally secure proactive secret sharing scheme with combinatorial structures. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999: 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 200–214, Kingston, Ontario, Canada, August 9–10, 1999. Springer, Heidelberg, Germany.
34. Anton Stolbunov. Cryptographic schemes based on isogenies. 2012.
35. Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 497–527, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
36. Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.
37. Qian Wu, Huiyan Chen, Zichen Li, and Cheng Jia. On a practical distributed key generation scheme based on bivariate polynomials. In *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, 2011.

A Security Proofs

A.1 Proof of Theorem 3.5

We prove the security of the new DKG protocol shown in Fig. 1 by the following theorem.

Theorem 3.5. *If PVPs are sound, then the DKG protocol of Fig. 1 is correct and robust. If PVPs are zero-knowledge, then the DKG protocol satisfies the secrecy property.*

Proof. **Robust correctness:** By Theorem 3.4, it is clear that the execution of the VSS step is correct. At the end of this step, there is a non-empty qualified set Q, containing at least $n-t$ honest parties. Each party in Q holds a correct share of the secret x, as specified by Theorem 3.4.

Correctness of the public-key computation step arises from the correct execution of the round-robin computation, the latter being proven and verified using PVPs. We now show that the correctness of our protocol reduces to the soundness of the PVPs. To this end, define $\mathcal{A}$ as an adversary against the correctness of our protocol. We then construct a PPT algorithm $\mathcal{B}^{\mathcal{A}}$ that breaks the soundness of the PVP, simulating the honest parties and interacting in the protocol with $\mathcal{A}$. Let $Q_{\mathcal{B}}$ denote the set of the honest parties controlled by $\mathcal{B}$ and let $Q_{\mathcal{A}} = Q \backslash Q_{\mathcal{B}}$. At the end of the protocol, $\mathcal{B}$ samples a random index i^*

in $Q_{\mathcal{A}}$ and returns the statements $((F_{i^*-1}, F_{i^*}), \{q^{(i^*)}(j)\}_{j \in Q_{\mathcal{B}}})$ and the proofs $(\widetilde{\pi}^{(i^*)}, \pi_0^{(i^*)}, \{\pi^{(i^*)}(j)\}_{j \in Q_{\mathcal{B}}})$.

If no party in $Q_{\mathcal{A}}$ successfully proved an invalid statement of the PVP, then we know that for each i, there exists $q^{(i)}(Z)$, such that $F_i = [q^{(i)}(0)]F_{i-1}$ and $q^{(i)}(j) = f_j^{(i)}(0)$ for all j. For any proof that fails, the honest parties can reconstruct this polynomial using the shares received in the VSS, thus they will be able to compute $F_i = [q^{(i)}(0)]F_{i-1}$. Therefore, in this case, the honest parties are guaranteed to be able to output the public key $F_{n'} = [x]E_0$, making the output correct and satisfying robustness.

Assume now that a party in $Q_{\mathcal{A}}$ was able to successfully prove an invalid statement using the PVP, which can lead to an incorrect protocol output. If the index of this party coincides with the index i^* selected by $\mathcal{B}$, then $\mathcal{B}$ has successfully output a statement and proof violating the soundness of the PVPs. It is clear that the probability of $\mathcal{B}$ choosing the correct index is $1/|Q_{\mathcal{A}}|$. With $|Q_{\mathcal{A}}| \leq t$, we can express $\mathsf{Adv}_{\mathcal{A}}^{\text{correctness}} \leq t \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{soundness}}$. Thus, if $\mathcal{A}$ is a PPT-adversary, which has a non-negligible advantage against the correctness of the DKG protocol, then $\mathcal{B}$ will be a PPT-adversary with a non-negligible advantage against the soundness of the PVP.

Secrecy: Let $\mathcal{A}$ be an adversary against the secrecy of the protocol. On a given input $E^* \in \mathcal{E}$, we construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ (with shared state), which simulates the honest parties interacting with $\mathcal{A}$ as well as the random oracle, so that the simulation is indistinguishable from the real protocol, and the output curve is E^*. Similar to [6], we do this in incremental steps, here as a sequence of three simulators $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$, which we describe below. We again denote by $Q_{\mathcal{A}} \subset Q$, the subset of qualified parties controlled by $\mathcal{A}$ and $Q_{\mathcal{S}} = Q \backslash Q_{\mathcal{A}}$.

- $\mathcal{S}^{(0)}$: $\mathcal{S}_1^{(0)}$ simulates the honest parties correctly, while $\mathcal{S}_2^{(0)}$ simulates a random oracle by maintaining a list of queries.
- $\mathcal{S}^{(1)}$: $\mathcal{S}_1^{(1)}$ picks a honest party P_{i^*} for $i^* \in Q_{\mathcal{S}}$, which behaves exactly as in the real protocol with one exception. Instead of computing the proof $\pi^{(i^*)}$ honestly, $\mathcal{S}_1^{(1)}$ calls the simulator $\mathcal{S}^{PVP}$ to construct $\pi^{(i^*)}$. $\mathcal{S}_2^{(1)}$ still takes the role of the random oracle, but forwards the queries of the PVP simulation to the random oracle simulator of $\mathcal{S}^{PVP}$.
- $\mathcal{S}^{(2)}$: In order to force the element E^* as output of the protocol, $\mathcal{S}_1^{(2)}$ first reconstructs the value $x^* = \sum_{i=i^*+1}^{n'} x^{(i)}$. For $i \in Q_{\mathcal{S}}$, these values are known, while for $i \in Q_{\mathcal{A}}$, these can be computed by interpolation of $q^{(i)}$ (since $|Q_{\mathcal{S}}| > t$). Then instead of computing F_{i^*} in the standard way, $\mathcal{S}_1^{(2)}$ computes $F^* = [-x^*]E^*$ and uses the same procedure as $\mathcal{S}^{(1)}$ for constructing the proof. $\mathcal{S}_2^{(3)}$ behaves in the same way as $\mathcal{S}_2^{(2)}$.

We can see that with the change by $\mathcal{S}^{(2)}$, due to the correctness and robustness of the protocol the round-robin will necessarily lead to the computation and output of $\left[\sum_{i=i^*+1}^{n'} x^{(i)}\right]F^* = [x^* - x^*]E^* = E^*$. We are left to prove that the view of $\mathcal{A}$ is indistinguishable from the real execution of the protocol. We argue the following identification step-by-step, using the notation introduced in Sec. 2.2.

$$\mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{O}}, \{P_i^{\mathcal{O}}\}_{i\in Q_{\mathcal{S}}}) \overset{(1)}{=} \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(0)}}, \mathcal{S}_1^{(0)}) \overset{(2)}{\approx_c} \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(1)}}, \mathcal{S}_1^{(1)}) \overset{(3)}{=} \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(2)}}, \mathcal{S}_1^{(2)}) \,.$$

(1) Since $\mathcal{S}^{(0)}$ performs the simulation faithfully, it is clear that the distribution is identical to the real protocol.

(2) Distinguishing these two distributions means distinguishing the real execution of the PVP protocol from the simulated one, thus breaking the zero-knowledge property of the PVPs. Therefore, since PVPs are computationally zero-knowledge, these two distributions are computationally indistinguishable.

(3) The real and simulated distributions here are identical. This is due to the fact, that the information which $\mathcal{A}$ holds, tells it nothing about how the action of $x^{(i)}$ should look like. In fact, $x^{(i)}$ is information-theoretically hidden by Theorem 3.2. Therefore, there is no way for $\mathcal{A}$ to distinguish between the real and simulated executions of the protocol.

□

A.2 Proof of Theorem 4.1

We prove the security of new non-interactive Structured PVP, described in Algorithms 3 and 4 by the following theorem.

Theorem 4.1. *Assuming that the commitment scheme $\mathcal{C}$ is collapsing and quantum computationally hiding, the described non-interactive PVP for the structured public keys (in Algorithms 3 and 4) is complete, sound, and ZK in the QROM for the list of relations given in equation (2).*

Proof. The proof of this theorem is analogous to the proof of [6, Theorem 2], with a few differences, which we will highlight in this proof.

A key peculiarity of PVPs (and SPVPs) is that they use a *weak* version of the Fiat–Shamir transform, i.e. where the random oracle is called with commitments as inputs rather than commitments and statements. In [6], the consequences regarding the security of this are treated in detail and PVPs could be proven secure, even with this modification. We refer the reader to [6, App. A] for more details. We note that these results also still apply to our case, so we will omit proving them again. Rather, let us point out, where the differences between SPVPs and PVPs lie. These are mainly in the definition of the relation R_0, which in the original CSI-RAShi paper is defined as

$$\{(x_0 = (F_1, F_1'), f(x)) \mid (F_1' = [f(0)]F_1)\}\,,$$

and thus represents the special case $k = 1$ and $c_1 = 1$ of the definition in equation (2).[10] Similarly, the commitments as represented in Algorithms 3 and 4 also reduce to the case $k = 1$.

[10] Regarding security assumptions, the fact that $f(0)$ cannot be obtained from $(F_1, F_1' = [x]F_1)$ relies on the GAIP, while we additionally rely on $\mathbb{C}_k$-Vectorization

As a result of this different structure, the proofs for completeness and soundness are adapted below for the case $i = 0$. The case $i \neq 0$ remains unchanged, as also here the relation is unchanged. We note that zero-knowledge immediately follows from the properties of the commitment scheme $\mathcal{C}$ and is therefore analogous to the proof in [6].

Completeness. For any $j = 1, \dots, \lambda$, if $d_j = 0$, then $r_j = b_j$ and hence $\widetilde{F}_j^l = [c_l r_j(0)]F_l = [c_l b_j(0)]F_l = \hat{F}_j^l$ for $l = 1, \cdots, k$. If $d_j = 1$, then $r_j(0) = b_j(0) - f(0)$, so again we have $\widetilde{F}_j^l = [c_l r_j(0)]F_l' = [c_l b_j(0) - c_l f(0)][c_l f(0)]F_l = [c_l b_j(0)]F_l = \hat{F}_j^l$, for $l = 1, \cdots, k$. Thus both C_0 are equal and the verifier will accept.

Soundness. Let $I \subseteq \{0, 1, \cdots, n\}$ with $|I| > t$. Given two accepting transcripts with different challenges (e.g. $d_j = 0$ and $d_j' = 1$, without loss of generality), if $0 \in I$ and any of $[c_1 r_j(0)]F_1 \neq [c_1 r_j'(0)]F_1'$, $[c_2 r_j(0)]F_2 \neq [c_2 r_j'(0)]F_2', \cdots, [c_k r_j(0)]F_k \neq [c_k r_j'(0)]F_k'$, then we found a collision in $\mathcal{C}$. Similarly, if for some non-zero $i \in I$ we have $r_j(i) \neq r_j'(i) + x_i$ then we also have a collision for $\mathcal{C}$. If there is no collision, then

$$r_j(i) = r_j'(i) + x_i \text{ for all } i \in I, i > 0\,, \text{ and}$$
$$[c_l r_j(0)]F_l = [c_l r_j'(0)]F_l' \quad \text{for } l = 1, 2, \cdots, k \quad (\text{if } 0 \in I)\,,$$

so we can extract a valid witness as $r_j(X) - r_j'(X)$. □

A.3 Proof of Theorem 4.2

Theorem 4.2. *If structured PVPs are sound, then the DKG protocol of Fig. 2 to generate structured public keys is correct and robust. If structured PVPs are ZK, then the DKG protocol satisfies the secrecy property.*

Proof. The proof follows the same steps as Theorem 3.5, with the exception of using the Structured PVP (SPVP) scheme and the expansion of the public key computation step to incorporate the structured public keys.

Robust correctness: The VSS step is identical to the one used in the single DKG protocol (given in Fig. 1), and its correctness can be argued as before. Once the VSS step is complete, each party in a qualified set Q, with at least $n - t$ honest parties, will hold a Shamir share of the secret key x. The correctness of SPK computation step arises from the correct execution of the round-robin computations and verifications, which is ensured by the soundness of the new SPVP scheme (given in Algorithms 3 and 4). Let, $\mathcal{A}$ be an adversary against the correctness of our SPVP scheme. We then construct a PPT algorithm $\mathcal{B}^{\mathcal{A}}$ that breaks the soundness of the SPVP, simulating the honest parties and interacting in the

Problem with Auxiliary Inputs ($\mathbb{C}_k$-VPwAI) to ensure that $f(0)$ cannot be obtained from the structured public key $(\mathbb{C}_k, F_1, F_1', \cdots, F_k, F_k')$.

protocol with $\mathcal{A}$. Let $Q_{\mathcal{B}}$ denote the set of the honest parties controlled by $\mathcal{B}$ and let $Q_{\mathcal{A}} = Q \backslash Q_{\mathcal{B}}$. At the end of the protocol, $\mathcal{B}$ samples a random index i^* in $Q_{\mathcal{A}}$ and returns the statements $((F^1_{i^*-1}, F^1_{i^*}), \cdots, (F^k_{i^*-1}, F^k_{i^*})), \{q^{(i^*)}(j)\}_{j \in Q_{\mathcal{B}}})$ and the proofs $(\widetilde{\pi}^{(i^*)}, \pi_0^{(i^*)}, \{\pi^{(i^*)}(j)\}_{j \in Q_{\mathcal{B}}})$.

If no party in $Q_{\mathcal{A}}$ successfully proved an invalid statement of the SPVP, then we know that for each i, there exists $q^{(i)}(Z)$, such that $F^1_i = [c_1 q^{(i)}(0)]F^1_{i-1}, \cdots, F^k_i = [c_k q^{(i)}(0)]F^k_{i-1}$ and $q^{(i)}(j) = f^{(i)}_j(0)$ for all j, and the superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \cdots, c_k\}$. For any proof that fails, the honest parties can reconstruct this polynomial using the shares received in the VSS, thus they will be able to compute $F^1_i = [q^{(i)}(0)]F^1_{i-1}, F^2_i = [c_2 q^{(i)}(0)]F^2_{i-1}, \cdots, F^k_i = [c_k q^{(i)}(0)]F^k_{i-1}$, given the superexceptional set $\mathbb{C}_k = \{c_1 = 1, c_2, \cdots, c_k\}$. Therefore, in this case, the honest parties are ensured to be able to return the SPK $(F^1_{n'} = [x]E_0, F^2_{n'} = [c_2 x]E_0, \cdots, F^k_{n'} = [c_k x]E_0)$, making the output correct and satisfying robustness. Let a party in $Q_{\mathcal{A}}$ successfully prove an invalid statement using the SPVP, which can lead to incorrect output. If the index of this party coincides with the index i^* selected by $\mathcal{B}$, then $\mathcal{B}$ has successfully output a statement and proof violating the soundness of the SPVP scheme. The probability of $\mathcal{B}$ choosing the correct index is $1/|Q_{\mathcal{A}}|$, thus, with $|Q_{\mathcal{A}}| \leq t$, we can express $\mathsf{Adv}^{\text{correctness}}_{\mathcal{A}} \leq t \cdot \mathsf{Adv}^{\text{soundness}}_{\mathcal{B}}$. Thus, if $\mathcal{A}$ is a PPT adversary, which has a non-negligible advantage against the correctness of the structured DKG protocol, then $\mathcal{B}$ will be a PPT adversary with a non-negligible advantage against the soundness of the SPVP.

Secrecy: Let $\mathcal{A}$ be an adversary against the secrecy of the structured DKG protocol. On a given input $(E^{*,1}, \cdots, E^{*,k}) \in \mathcal{E}^k$, we construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ (with the shared state), which simulates the honest parties interacting with $\mathcal{A}$ as well as the random oracle so that the simulation is indistinguishable from the real protocol, and the output curves are $(E^{*,1}, \cdots, E^{*,k})$. Similar to [6] and Theorem 3.5, we do this in incremental steps, here as a sequence of three simulators $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}$, and $\mathcal{S}^{(2)}$, which we describe next. We again denote by $Q_{\mathcal{A}} \subset Q$, the subset of qualified parties controlled by $\mathcal{A}$ and $Q_{\mathcal{S}} = Q \backslash Q_{\mathcal{A}}$.

- $\mathcal{S}^{(0)}$: $\mathcal{S}^{(0)}_1$ simulates the honest parties correctly, while $\mathcal{S}^{(0)}_2$ simulates a random oracle by maintaining a list of queries.
- $\mathcal{S}^{(1)}$: $\mathcal{S}^{(1)}_1$ picks an honest party P_{i^*} for $i^* \in Q_{\mathcal{S}}$, which behaves exactly as in the real protocol with one exception. Instead of computing the proof $\pi^{(i^*)}$ honestly, $\mathcal{S}^{(1)}_1$ calls the simulator $\mathcal{S}^{SPVP}$ to construct $\pi^{(i^*)}$. $\mathcal{S}^{(1)}_2$ still takes the role of the random oracle, but forwards the queries of the SPVP simulation to the random oracle simulator of $\mathcal{S}^{SPVP}$.
- $\mathcal{S}^{(2)}$: To force the elements $E^{*,1}, \cdots, E^{*,k}$, as the output of the protocol, $\mathcal{S}^{(2)}_1$ first reconstructs the value $x^* = \sum_{i=i^*+1}^{n'} x^{(i)}$. For $i \in Q_{\mathcal{S}}$, these values are known, while for $i \in Q_{\mathcal{A}}$, these can be computed by interpolation of $q^{(i)}$ (since $|Q_{\mathcal{S}}| > t$). Then instead of computing $(F^{i^*,1}, \cdots, F^{i^*,k})$ in the standard way, $\mathcal{S}^{(2)}_1$ computes $(F^{*,1} = [-x^*]E^{*,1}, F^{*,2} = [-c_2 x^*]E^{*,2}, \cdots, F^{*,k} =$

$[-c_k x^*]E^{*,k})$ and uses the same procedure as $\mathcal{S}^{(1)}$ for constructing the proof. $\mathcal{S}_2^{(3)}$ behaves in the same way as $\mathcal{S}_2^{(2)}$.

We can see that with the change by $\mathcal{S}^{(2)}$, due to the correctness and robustness of the protocol the round-robin will necessarily lead to the computation and output of $\left[\sum_{i=i^*+1}^{n'} c_l x^{(i)}\right] F^{*,l} = [c_l x^* - c_l x^*]E^{*,l} = E^{*,l}$ for $l = 1, \cdots, k$. To prove that the view of $\mathcal{A}$ is indistinguishable from the real execution of the protocol, next, we will proceed step-by-step, using the notation introduced in Sec. 2.2.

$$\mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{O}}, \{P_i^{\mathcal{O}}\}_{i \in Q_{\mathcal{S}}}) \overset{(1)}{=} \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(0)}}, \mathcal{S}_1^{(0)}) \overset{(2)}{\approx}_c \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(1)}}, \mathcal{S}_1^{(1)}) \overset{(3)}{=} \mathsf{D}_{\mathsf{out}}(\mathcal{A}^{\mathcal{S}_2^{(2)}}, \mathcal{S}_1^{(2)}) \,.$$

(1) Since $\mathcal{S}^{(0)}$ performs the simulation faithfully, it is clear that the distribution is identical to the real protocol.
(2) Distinguishing these two distributions means distinguishing the real execution of the SPVP protocol from the simulated one, thus breaking the zero-knowledge property of the SPVP scheme. Therefore, since SPVPs are computationally ZK, these two distributions are computationally indistinguishable.
(3) The real and simulated distributions here are identical. This is due to the fact, that the information which $\mathcal{A}$ holds, tells it nothing about how the action of $x^{(i)}$ should look like. In fact, $x^{(i)}$ is information-theoretically hidden by Theorem 3.2. Therefore, there is no way for $\mathcal{A}$ to distinguish between the real and simulated executions of the protocol.

□

B Computational and Communication Costs of Our Protocols

In this section, we establish the computational and communication costs of our DKG protocols. We express the *sequential* costs τ of the protocol steps, i.e. the total runtime from start to finish, including when some of the parties are idle and discuss optimizations that minimize these idle times. We denote by T_I, T_E, T_C and T_H the cost of isogeny computations, polynomial evaluations, calls to the commitment scheme and calls to the random oracle, respectively. We ignore the cost of other operations, such as sampling and addition and multiplication over the ring $\mathbb{Z}_N$, as they are negligible in comparison. We express the communication cost in terms of *outgoing* communication cost γ per party. Let C_E and C_N denote the information content of an elliptic curve in $\mathcal{E}$ and an element in $\mathbb{Z}_N$, respectively. A monovariate polynomial of degree t can be represented by $t+1$ elements in $\mathbb{Z}_N$. We first determine the costs of the individual building blocks of our protocol, before we put them together and compute the full costs.

VSS. We can easily see that in the VSS step from Figure 1, each party first evaluates and sends out $2(n-1)$ monovariate polynomials. Then, in the verification step, parties further evaluate and share $2(n-1)(n-2)$ polynomial evaluations. We note that the evaluations can be done in parallel, thus this amounts to a total of $2(n-1)^2$ sequential evaluations and $2(n-1)(n+t-1)$ elements in $\mathbb{Z}_N$ sent out to the other parties. We find

$$\tau_{vss}(n) = 2(n-1)^2 T_E \quad \text{and} \quad \gamma_{vss}(n,t) = 2(n-1)(n+t-1)C_N \, .$$

Proof step. In the public key computation step of Figure 1, parties have to compute one isogeny and run the proof in Algorithm 1. By carefully counting the operations in the latter, we find the total cost of

$$\tau_{proof}(n,\lambda) = \lambda(n+1)T_E + (\lambda+1)T_I + 2(n+1)T_C + T_H \, .$$

After this step, the party has to publish the computed curve and the main proof and send the individual proof pieces to each other player. We can easily check that the proof pieces are 2λ bits each and that the main proof consists of $2(n+1)$ commitments, each for 2λ bits and of the response, for $\lambda(t+1)C_N$.

We note that both the computational and communication cost change when we use the twist trick. Remember that in this case, the challenge space increases from size 2 to 3, resulting in the number of repetitions being reduced to $\lambda' := \lceil \lambda / \log_2 3 \rceil$. In this case, the proof simply cost becomes $\tau_{proof}(n, \lambda')$. Regarding communication, we point out that the size of the proof pieces, determined by the security parameter λ, does not change when using the twist trick. To avoid confusion, we simply denote the cost of a commitment, or of a proof piece as $C_C = 2\lambda$, which is fixed. We can then express the total communication cost in the proof step as

$$\gamma_{proof}(n,t,\lambda) = C_E + (3n+2)C_C + \lambda(t+1)C_N \, .$$

Verification step. For simplicity, we look at the upper bound $|Q| = n$. The verification step is reduced to the evaluation of Algorithm 2 by $n-1$ parties, in parallel, once for $i = 0$ and once for $i \neq 0$. Note that the hash computation remains the same in both cases, and so only has to be computed once. By counting the different steps, we find the total of

$$\tau_{verif}(\lambda) = \lambda(T_E + T_I) + 4T_C + T_H \, .$$

If all the checks succeed, parties do not have to communicate anything in this step. In the converse case, per failed verification, parties have to broadcast one polynomial and verify at most n by evaluating them. This happens at most t times. We will ignore these costs in the interest of more realistic estimates.

Basic DKG protocol. We can finally compute the full cost of the protocol in Figure 1. This protocol simply consists of a VSS, and n consecutive proof and verification steps in the round-robin. We note that in the first round, we can use the twist trick. We find

$$\tau_{DKG}(n,\lambda) = \tau_{vss}(n) + \tau_{proof}(n,\lambda') + \tau_{verif}(\lambda')$$

$$+ (n-1)(\tau_{proof}(n,\lambda) + \tau_{verif}(\lambda))\,.$$

and $\gamma_{DKG}(n,t,\lambda) = \gamma_{vss}(n,t) + \gamma_{proof}(n,t,\lambda)$, where in the first round, we can substitute λ for λ'. By looking at the individual terms, we find the results summarized in Tables 2 and 3.

Extended DKG protocol. In the case of extended (non-structured) public keys discussed at the end of Sec. 3.2, the VSS step has to be repeated k times and the cost of a round-robin step naively increases by a factor k. This cost can be greatly improved by *staggering* the proofs and verifications, as was proposed in [19] and analyzed in more detail in [2]. Roughly, the idea is to compute the first proof and then publish it, so that other parties can verify it during the creation of the second proof and so on. As a result, the sequential cost of a round-robin step is reduced to the cost of k consecutive proofs plus one extra verification. But we can even do better, using the idea from [2, Sec. 6]: Since all the different round-robins are independent computations, we can permute the players for each of them, and run multiple round-robins in parallel. This means, that while P_1 computes the k first curves for one secret and creates the PVP, P_2 does the same but for a different secret etc. Then, all of the verifications are performed, before moving onto the second step of all of the round-robins. In that way, we minimize idle time.

For n players with k secrets, the lowest attainable sequential runtime in this way is composed of $\lceil \frac{k}{n} \rceil$ proof steps and $k - \lfloor \frac{k}{n} \rfloor$ sequential verification steps, *per round-robin step*. Including the twist trick, we find the total cost

$$\tau_{DKG}^{ext.}(n,k,\lambda) = k\tau_{vss}(n) + \left(\lceil \tfrac{k}{n} \rceil \tau_{proof}(n,\lambda') + \left(k - \lfloor \tfrac{k}{n} \rfloor\right)\tau_{verif}(\lambda')\right)$$
$$+ (n-1)\left(\lceil \tfrac{k}{n} \rceil \tau_{proof}(n,\lambda) + \left(k - \lfloor \tfrac{k}{n} \rfloor\right)\tau_{verif}(\lambda)\right). \quad (3)$$

The communication costs are not changed by changing the order, so that we simply find $\gamma_{DKG}^{ext.}(n,k,t,\lambda) = k\gamma_{DKG}(n,t,\lambda)$. The individual terms are again summarized in Tables 2 and 3.

Structured DKG protocol. If we use the DKG for structured public keys (given in Fig. 2), the VSS does not have to be repeated k times as we only have a single secret. Furthermore, in the public key computation step, proofs and verifications are done with SPVPs, which are introduced in Algorithms 3 and 4. Some scrutiny reveals

$$\tau_{proof}^{SPVP}(n,k,\lambda) = \lambda(n+1)T_E + k(\lambda+1)T_I + 2(n+1)T_C + T_H\,,$$
$$\tau_{verif}^{SPVP}(k,\lambda) = \lambda T_E + k\lambda T_I + 4T_C + T_H\,.$$

Note that τ_{proof}^{SPVP} also includes the computation of the curves in the round-robin step. In comparison to the cost of the standard PVPs established earlier, only the isogeny computations increase by a factor k, while the other terms remain unchanged. Regarding communication cost, we can easily see that an SPVP has the same size as a PVP, independent of k. The difference to the basic case is that we publish k curves instead of one, resulting in the cost per proof step of $\gamma_{proof}^{SPVP}(n, k, t, \lambda) = kC_E + (3n + 2)C_C + \lambda(t + 1)C_N$. We end up with the total

$$\gamma_{DKG}^{SPK}(n, k, t, \lambda) = \gamma_{vss}(n, t) + \gamma_{proof}^{SPVP}(n, t, \lambda)\,,$$

where again, we can substitute $\lambda \mapsto \lambda'$ in the first round. In the protocol from Figure 2, we can use a similar approach as for the extended DKG protocol, in the sense that we can run multiple round-robins in parallel. A difference here, is that each player does not run k individual PVPs, but instead batches them into SPVPs. This allows to run an initial round of n SPVPs in parallel, each with $\lfloor \frac{k}{n} \rfloor$ elements, and a second round with $k \bmod n$ PVPs in parallel. The first round has $n-1$ subsequent verifications to be performed and the second $k \bmod n$ more, again all in parallel by the individual players. The cost per round-robin step can therefore be expressed as

$$\begin{aligned} R(n, k, \lambda) = {} & \tau_{proof}^{SPVP}(n, \lfloor \tfrac{k}{n} \rfloor, \lambda) + (n-1)\tau_{verif}^{SPVP}(\lfloor \tfrac{k}{n} \rfloor, \lambda) \\ & + \chi_{n,k}(\tau_{proof}(n, \lambda) + (k \bmod n)\tau_{verif}(\lambda))\,, \end{aligned}$$

where we define $\chi_{n,k} = \lceil \frac{k}{n} \rceil - \lfloor \frac{k}{n} \rfloor$, i.e. $\chi_{n,k} = 0$, if $n \mid k$ or $k \mid n$, and 1 otherwise. These steps are repeated n times, where at the first step we can use the twist trick. Together with the VSS, we find the total cost of

$$\tau_{DKG}^{str.}(n, k, \lambda) = \tau_{vss}(n) + R(n, k, \lambda') + (n-1)R(n, k, \lambda)\,. \tag{4}$$

Again, the individual terms are summarized in Tables 2 and 3.

Comparison of extended and structured case. Finally, we establish some of the background related to Figures 3 and 4.

Communication. Using the fact that $N \approx \sqrt{p}$ [31] and choosing the security parameter $\lambda \approx \sqrt[4]{p}$ (reflecting the classical security, see [2, 12]), we can easily identify $2\lambda \approx C_C \approx C_N \approx \frac{1}{2}C_E$. By plugging this into the terms in Table 3, setting $t = n/3$ and dropping some of the constant terms, we can see, that the communication cost of the extended DKG asymptotically scales with $\sim k(8n^2\lambda + n\lambda^2 + 6\lambda + 9n)$, while the structured case scales with $\sim (8n^2\lambda + 2n\lambda^2 + 6k\lambda + 9n)$. For $n \to \infty$, the latter is k times smaller, while for $k \to \infty$, the latter is $8n^2 + n\lambda + 6 + 9n/\lambda$ times smaller, both considerable gains. We depict these trends in Figure 4. The asymptotic quadratic trend in n and linear trend in k of our schemes are clearly visible in the figure.

Computation. The results in Table 2 show that using structured public keys removes the dependency on k in all cases but isogeny computations. It is clear that the number of calls to commitment schemes or random oracles becomes the same around $k \approx n$. For the number of polynomial evaluations, this behavior becomes a bit more complex, and the structured case always outperforms the extended case for some $k \leq n$. This is due to the fact that the VSS in the extended case scales with k, while it is independent of k in the structured case.

We note that in general, isogeny computation costs will strongly dominate the full protocol cost. We restate the full isogeny costs of both protocols here, in the most general case, using the twist trick. For the latter, we define $\lambda' = \lceil \lambda / \log_2 3 \rceil$. By looking at the isogeny cost terms of equations (3) and (4), we find, after some arithmetic, that they are both equal to

$$I(n, k, \lambda) = (\lambda' + (n-1)\lambda)\big(k + \chi_{n,k}\big) + n\left\lceil \tfrac{k}{n} \right\rceil .$$

We compare this with the results from [2] in Figure 3. Below, we also summarize the gains we get by using the twist trick for low n.

n	2	3	4	5	6	8	10	20	50
Gain	18.3%	12.2%	9.2%	7.3%	6.1%	4.6%	3.6%	1.8%	0.7%