LCPR: High Performance Compression Algorithm for Lattice-Based Signatures and Schnorr-like Constructions

Rachid El Bansarkhani and Johannes Buchmann

Technische Universität Darmstadt
Fachbereich Informatik
Kryptographie und Computeralgebra,
Hochschulstraße 10, 64289 Darmstadt, Germany
{elbansarkhani,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. We present a novel and generic construction of a lossless compression algorithm for Schnorr-like signatures utilizing publicly accessible randomness. This strategy is from a mathematical and algorithmic point of view very interesting, since it is closely related to vector quantization techniques used for audio and video compression. Conceptually, exploiting public randomness in order to reduce the signature size has never been considered in cryptographic applications. This opens new directions for improving existing signature schemes. We illustrate the applicability of our compression algorithm using the examples of current-state-of-the-art signature schemes such as the efficient constructions due to Lyubashevsky et al. and the GPV signature scheme instantiated with the efficient trapdoor construction from Micciancio and Peikert. Both schemes benefit from increasing the main security parameter n, which is positively correlated with the compression rate measuring the amount of storage savings. For instance, GPV signatures admit improvement factors of approximately $\lg n$ implying compression rates of about 65% for practical parameters without suffering loss of information or decrease in security, meaning that the original signature can always be recovered from its compressed state. Similarly, for signatures generated according to the scheme due to Güneysu et al. we achieve compression rates of approximately 60% and even 73%, when combining with previous compression algorithms. As a further interesting result, we propose a generic unrestricted aggregate signature scheme.

Keywords: Lattice-Based Cryptography, Aggregate Signatures, Schnorr-Signatures, Generic Compression Algorithm, Public Randomness

1 Introduction

In recent years, one observes an increasing interest in cryptography based on problems that are hard to break even in a post-quantum world. Ever since the seminal work of Shor [Sho97] in 1994, it is a well-known fact that quantum computers can break cryptographic schemes with security based on the hardness of the discrete log or factoring problems in probabilistic polynomial time. As a consequence, lattice-based cryptography emerged as a promising candidate to replace classical cryptography in case powerful quantum computers are built. As opposed to cryptography based on factoring, which can classically be broken in $2^{\tilde{O}(n^{1/3})}$ time, lattice problems are conjectured to withstand quantum attacks. The time complexity of the current best known attacks on lattice problems, for example [CN11], are exponential in the main security parameter n. A major contribution to lattice-based cryptography is due to Ajtai, whose hardness results [Ajt96] immediatley induced the construction of new cryptographic primitives. Besides of simple operations inherent to lattice-based cryptography such as additions and multiplications of small integers, Ajtai's worst-case to average-case reductions allow for simplified instantiations of cryptographic schemes while enjoying worst-case hardness. This does apparently not apply to classical cryptography, which still suffers from the generation of huge primes and complex operations such as exponentiations. In the last couples of years, a sequence of new and remarkable constructions appeared as a result of well-studied lattice problems such as the shortest vector problem dating back to Gauss, Hermite and Dirichlet. For instance, cryptographic applications include provably secure digital signature schemes [DDLL13,GLP12,Lyu12,GPV08,MP12], preimage sampleable trapdoor functions [GPV08,MP12,AP09,Pei10,SS11], encryption schemes [LP11,HPS98,SS11], oblivious transfer [PVW08], blind signatures [Rüc10] and much more. But also conceptually new applications have recently been discovered such as fully homomorphic encryption [BV11,Gen09,GSW13] and multilinear maps [GGH$^+$13b,GGH13a], just to name a few examples. The majority of these schemes additionally benefit from the ring variant as it admits smaller key sizes and faster processing capabilities while simultaneously providing similar worst-case to average-case reductions.

1.1 Related Work

Lattice Signatures Digital signature schemes belong arguably to the most commonly used cryptographic primitives in practice with a wide range of applications. Hence, digital signatures are subject to intense research. The same also holds for lattice-based cryptography resulting in new and efficient signature schemes. First attempts to build lattice-based signature schemes [GGH97,HNHGSW03] failed due to the vulnerability to statistical attacks as shown, for instance, in [NR06,DN12,GJSS01]. First provably secure constructions were independently introduced in [GPV08] and [LM08]. The former approach is based on the hash-and-sign principle, which initiated the design of improving preimage sampleable trapdoor functions [GPV08,AP09,Pei10,MP12], the main building block of the GPV signature scheme, which has recently become practical [MP12]. A recent result [BZ13] even shows that the scheme is secure in the quantum random oracle model (EUF-qCMA secure). The second approach, however, requires to build signatures in a Schnorr-like manner. Subsequent works aim at increasing the efficiency through conceptually new ideas [Lyu09,Lyu12] and efficient instantiations [GLP12]. The most recent construction [DDLL13] presented at Crypto 2013 is practically efficient taking advantage of an improved discrete Gaussian sampler.

Aggregate Signatures An aggregate signature (AS) scheme enables a group of signers to combine their signatures on messages of choice such that the combined signature is essentially as large as an individual signature. Aggregate signatures have many application areas such as secure routing protocols [Lyn99] providing path authentication in networks. The first aggregate signature scheme is due to [BGLS03], which is based on the hardness of the co-Diffie-Hellman problem in the random oracle model. Following this proposal, the aggregation mechanism can be accomplished by any third party since it relies solely on publicly accessible data and the individual signature shares. Conceptually, this scheme is based on bilinear maps. In [LMRS04] Lysyanskaya et al. proposed a new variant of AS, known as sequential aggregate signatures (SAS), which differs from the conventional AS schemes by imposing an order-specific generation of aggregate signatures. A charecteristical feature of this scheme is to include all previously signed messages and the corresponding public keys in the computation of the aggregate. In practice, one finds, for instance, SAS schemes applied in the S-BGP routing protocol or in certificate chains, where higher level CAs attest the public keys of lower level CAs. The generic SAS construction provided in [LMRS04] is based on trapdoor permutations with proof of security in the random oracle model. However, the SAS scheme suffers from the requirement of certified trapdoor permutations [BY96] for the security proof to go through. Subsequent works provide similar solutions or improve upon existing ones [BNN07,Nev08,EFG$^+$10,BGR12]. Interestingly, Hohenberger et al. present in [HSW13] the first unrestricted aggregate signature scheme that is based on leveled multilinear maps. The underlying hardness assumptions are, nevertheless, not directly connected to worst-case lattice problems. In this work, we also address the question whether it is possible to build (S)AS schemes that can be based on worst-case lattice problems.

1.2 Our Results and Contribution

In this work, we propose a generic and novel high performance compression algorithm for Schnorr-like signature schemes moving current state-of-the-art signature schemes towards practicality. This innovative concept is realized based on the existence of publicly accessible randomness. The notion of public randomness was firstly introduced in [HL93]. We revisit this feature in the context of lattice-based cryptography and show how it can be extended to Gaussian-like distributions. In particular, we provide mechanisms that make use of public randomness in order to decrease the signature size. We differentiate the randomness used to generate signatures into its public and secret portion. Randomness that is publicly accessible [HL93] can be read by all parties. The key idea underlying our lossless compression algorithm is to reduce the public share of a signature to a short uniform random seed. We exemplify the applicability of our new compression algorithm using the most efficient lattice-based signature schemes such as the GPV signature scheme employing the most recent PSF construction [MP12] and the efficient Schnorr-like signature schemes as provided in [DDLL13,GLP12,Lyu12,Lyu09]. To the best of our knowledge, such a compression strategy has never been used for cryptographic applications.

Methodology of Compressing Schnorr-like Signatures. In general, Schnorr signatures are constituted of $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$, where $f_{\mathbf{s}}(\mathbf{c}) = \mathbf{s} \cdot \mathbf{c}$ involves the secret key $\mathbf{s}$ and $\mathbf{y}$ is sampled from a proper distribution $\mathcal{Y}(\mathbf{x})$ serving as masking term in order to conceal the secret. In many signature schemes the magnitude of the entries in $\mathbf{y}$, when considering $\mathbf{y}$ as a vector with n independent entries, are very huge as compared with the entries in $f_{\mathbf{s}}(\mathbf{c})$ and thus leak information about $\mathbf{y}$. Consequently, a large difference allows to specify a narrow range $C = \mathbf{z} + [-h, h]^n$, from which the masking term $\mathbf{y} \in C$ was sampled. Here, $h = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$ denotes the maximum bound on the entries for any choice of $\mathbf{s}$ and $\mathbf{c}$. This range is publicly accessible and can be revealed by any party viewing the signature. Our goal is to exploit the public randomness introduced by $\mathbf{y}$ and hence the set C. Thus, let $\mathbf{z}$ be a fresh signature as above. We will show that arbitrary many other signers can reuse public randomness by secretly sampling their masking term $\mathbf{y}'$ from the same set C according to the conditional probability distribution $\mathcal{Y}(\mathbf{x})/P_{\mathbf{y}\sim\mathcal{Y}}[\ \mathbf{y} \in C\ ]$, where $\mathbf{x}$ is restricted to C. Since $\mathbf{y}$ is independently sampled, we have $\mathbf{y} \in C$ with probability $P_{\mathbf{y}\sim\mathcal{Y}}[\ \mathbf{y} \in C\ ]$ (or shortly $P[\ C\ ]$) for arbitrary fixed $\mathbf{z}$. Hence, exploiting public and secret randomness leads to a vector $\mathbf{y}'$ that is distributed as $P[\mathbf{y} \in\ C\ \wedge \mathbf{y}' = \mathbf{x}] = P[\ C\ ] \cdot \mathcal{Y}(\mathbf{x})/P[\ C\ ] = \mathcal{Y}(\mathbf{x})$, which exactly coincides with the required distribution using conditional probability rules. Following this approach, we derive an upper bound for the maximum distance of two signatures $\|\mathbf{z} - \mathbf{z}'\|_\infty = \|\mathbf{z} - \mathbf{s}' \cdot \mathbf{c}' - \mathbf{y}'\|_\infty \leq 2b$. A necessary condition for compression is given by $2h < \|\mathbf{z}\|_\infty$, which is typically satisfied for the current state-of-the-art signature schemes. A compressed signature is identified by the tupel $(\mathbf{z}, \mathbf{z} - \mathbf{z}')$, where $\mathbf{z}$ is called the centroid and serves to recover $\mathbf{z}'$. To prove security, we simulate the compression algorithm using an oracle for uncompressed signatures from the underlying signature schemes. As a consequence from this, the same centroid can be utilized by different other signers such that only one centroid is required to uncompress all other signatures. This obviously induces a conceptually new aggregate signature scheme, where a set of users participate in producing an aggregate signature. Going further, since the distribution $\mathcal{Z}$ of signatures $\mathbf{z}$ can always be simulated by a cryptographic hash function modeled as random oracle in combination with a rejection sampling algorithm, we forbear to store the centroid $\mathbf{z}$ and store a short seed $\mathbf{r} \in \{0,1\}^\mu$ that serves as input to a sampler for $\mathcal{Z}$ (typically discrete Gaussian or Uniform distribution). This strongly reduces the signature size, since the public share of the signature can deterministically be recovered by use of $\mathbf{r}$. Doing this, it is even possible to compress individual signatures to $(\mathbf{r}, \mathbf{z} - \mathbf{z}')$. Of course, the seed has always to be be refreshed for every new signature. The corresponding aggregate signature is subsequently represented by the tupel $(\mathbf{r}, \mathbf{z} - \mathbf{z}_1, \mathbf{z} - \mathbf{z}_2, \ldots, \mathbf{z} - \mathbf{z}_l)$, where $\mathbf{z}_i$ denotes the signature of the i-th signer.

Geometrically, the proposed compression algorithm is akin to vector quantization techniques [GG91,Gra84] applied for lossy video and audio compression (e.g. MPEG-4). But from an algorithmic point of view our scheme works differently as it requires the signers to sample signatures $\mathbf{z}'$ within short distance to a vector called centroid $\mathbf{z}$ (see Figure 1), which could be a signature or a vector sampled from the discrete Gaussian distribution using a short random seed as input. This allows for high compression rates without loss of quality because it is always possible to recover the signatures after compression. As a result, it suffices to store only the seed and the difference $\mathbf{z} - \mathbf{z}'$ of the signatures to the centroid. This apparently avoids the need to store complete signatures (see Figures 2 and 3). When employing GPV signatures, for instance, the implied storage savings amount to approximately 65 % for practical parameters (see Table 2) yielding a factor improvement of approximately $\lg n$ with n being the main security parameter.

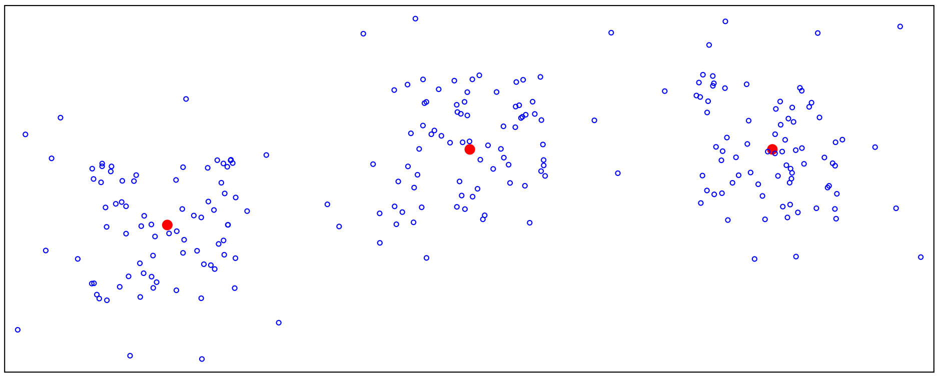

Fig. 1. Centroids (red circles) are surrounded by signatures from different signers (blue circles). Each signature belongs to one cluster defined by its centroid.

From this compression strategy we derive an aggregate signature scheme (see Figure 1) that allows an arbitrary number of signers to combine their signatures to an aggregate signature of reduced storage size.

		Signature size in [kB] before/after comp.		Compression rate [%]		Factor improvement		Entropy of Randomness $\approx$
n	k	Ring	Mat	Ring	Mat	Ring	Mat	$h_{public}(X) - h_{secret}(X)$
256	28	18 / 7	14 / 6	66	58	3.0	2.4	2^{12}
512	30	40 / 14	32 / 13	68	61	3.2	2.5	2^{13}
1024	35	100 / 33	79 / 29	70	63	3.4	2.7	2^{14}

Table 1. Compression rates in the ring and matrix variant for different parameter sets.

Compression of GPV Signatures. Ever since the seminal work [GPV08] the hash-and-sign approach for building signatures becomes more and more attractive for use in cryptographic applications. However, the construction of new and more efficient preimage sampleable trapdoor functions with tighter bounds, which are simpler to handle and to instantiate, appears to be a challenging task in lattice-based cryptography. One of the main goals of those constructions is to reduce the signature size while preserving security. Decreasing the parameter s governing the signature size is often not readily possible without affecting the security, since the security proof [GPV08] requires $s \geq \eta_\epsilon(\Lambda_q^\perp(\mathbf{A}))$ to be satisfied for a random matrix $\mathbf{A}$. Usually, the quality s is almost tight due to the construction of the public key $\mathbf{A}$. Thus, enhancing the quality always involves the construction of new trapdoor families. In our work, we provide a very different approach to reduce the signature size by exploiting large amounts of publicly accessible randomness stored in the construction of signatures.

To get an impression of how our compression algorithm works, we believe it is reasonable to first sketch the GPV signature scheme instantiated with the efficient trapdoor construction from [MP12]. The GPV signature scheme was a big move towards provably secure lattice-based signatures. Similar to the full-domain hash schemes and its variants in [BR93,BR96,Cor00], it is based on collision-resistant preimage sampleable (trapdoor) functions (PSF) $\mathsf{f}_{\mathbf{A}} : B_n \rightarrow R_n$, which enable a dedicated signer to sample preimages $\mathbf{z} \in B_n$ for arbitrary given target vectors $\mathbf{y} \in R_n$ such that $\mathsf{f}_{\mathbf{A}}(\mathbf{z}) = \mathbf{y}$ holds, but other than that signer none is capable of producing preimages. The security of this scheme consists in reducing the problem of finding collisions for $\mathsf{f}_{\mathbf{A}}(\cdot)$ to the hardness of forging signatures. In the course of years, several constructions of PSF families appeared [GPV08,AP09,Pei10], where the collision-resistance stems from the hardness of SIS, which is in turn believed to withstand quantum attacks for properly chosen parameters. The main drawback of all those schemes is the lack of efficiency due to complex procedures. Recently, Micciancio et al. [MP12] proposed an elegant trapdoor construction, that is characterized by efficient operations providing tighter bounds for all relevant quantities and thus improving upon previous constructions. But also in practice they appear to be efficient, which can also be attributed to the corresponding ring variant proposed in [BB13,MP12]. We now describe one instantiation of the digital signature scheme that is most

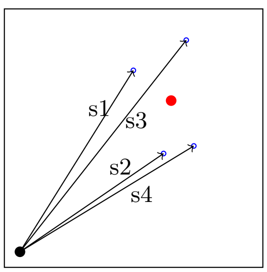

Fig. 2. Depicts signatures from different signers without compression. Complete signatures are stored.

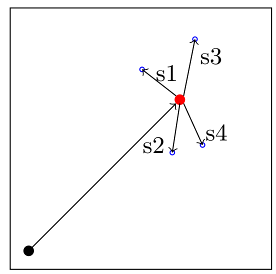

Fig. 3. Depicts signatures with compression. Signatures from different signers are stored in relation to the centroid (red circle).

suitable for GPV: The signer generates a random matrix $\bar{\mathbf{A}} \in \mathbb{Z}^{n \times n}$ and a secret matrix $\mathbf{R} \in \mathbb{Z}^{2n \times nk}$ with entries sampled from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\alpha q}$, where $\alpha q > \sqrt{n}$ and $q = 2^k$. The public key is given by $\mathbf{A} = [\mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\mathbf{G} \in \mathbb{Z}^{n \times nk}$ is called the gadget, a matrix of special structure $\mathbf{I}_n \otimes \mathbf{g}^\top$ with $\mathbf{g}^\top = (1, 2, \ldots, 2^{k-1})$, which allows to sample preimages more efficiently. In the signing step the signer computes $\mathbf{u} = H(m)$ for a message m of choice, samples a perturbation vector $\mathbf{p}$ and a preimage $\mathbf{x} \xleftarrow{\$} \mathcal{D}_{\Lambda^\perp_{\mathbf{v}}(\mathbf{G}),r}$ for $\mathbf{v} = \mathbf{u} - \mathbf{A} \cdot \mathbf{p}$ and $r > \eta_\epsilon(\Lambda^\perp_q(\mathbf{G}))$. The resulting signature $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{x} + \mathbf{p} = (\mathbf{R}\mathbf{x} + \mathbf{p}_1, \mathbf{x} + \mathbf{p}_2) \in \mathbb{Z}^{2n} \times \mathbb{Z}^{nk}$ is spherically distributed. Similar to the signature schemes [Lyu08,Lyu09,GLP12], the perturbation vector is used in order to keep the distribution of the signature independent from the secret key and thus not leaking information about its structure. By this means, strong attacks such as [NR09,DN12] are no longer feasible. Verification is performed by checking the validity of $\mathbf{A}\mathbf{z} \equiv H(m)$ and $\|\mathbf{z}\| \leq \sqrt{2n + nk}$.
We now turn our focus on the way $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2)$ is generated, since it plays an important role for our compression algorithm. Specifically, we sample $\mathbf{p} = \lceil \sqrt{\Sigma_p} \cdot \mathbf{d} \rfloor_a$, where $\lceil \cdot \rfloor_a$ denotes the randomized rounding operation from [Pei10] and $\mathbf{d}$ is sampled from the continuous Gaussian distribution $\mathcal{D}_1^{2n+nk}$ with parameter 1. As a result of [BB13], the perturbation matrix can be represented by $\sqrt{\mathbf{\Sigma_p}} = \begin{bmatrix}\frac{\mathbf{R}}{\sqrt{b}} & \mathbf{L}\\ \sqrt{b}\mathbf{I_{nk}} & 0\end{bmatrix}$ with $b = s^2 - 5a^2$, $a = r/2 = \sqrt{\ln\left(2n\left(1 + \frac{1}{\epsilon}\right)\right)/\pi}$ and $\mathbf{L}$ denoting the decomposition matrix from [BB13]. This immediately leads to the simplified representation of the perturbation vector $\mathbf{p}$ with $\mathbf{p}_2 =$

$\sqrt{b}\mathbf{d}_2+\mathcal{D}_{\mathbb{Z}^{n\cdot k}-\sqrt{b}\mathbf{d}_2,a}$ and $\mathbf{d}_2 \xleftarrow{\$} \mathcal{D}_1^{nk}$. Following the abstract form of a Schnorr-signature as above, the lower part of the signature is represented by $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{x},\mathbf{y}^{(2)})+\mathbf{y}^{(2)} = \mathbf{I}\cdot\mathbf{x}+\mathcal{D}_{\mathbb{Z}^{n\cdot k}-\mathbf{y}^{(2)},r}+\mathbf{y}^{(2)}$ with $\mathbf{y}^{(2)} = \sqrt{b}\mathbf{d}_2$ and $h = \max\left\|f_{\mathbf{I}}(\mathbf{x},\mathbf{y}^{(2)})\right\|_\infty \leq 4.7\cdot\sqrt{5}a$. By scaling the lower part of any signature to $\mathbf{z}^{(2)}/\sqrt{b}$ we extract large amounts of information about the continuous Gaussian $\mathbf{d}_2$ used for sampling the perturbation vector. This randomness is publicly accessible [HL93] and can be read by all parties without affecting the security. Indeed, the security level of cryptographic schemes should not be based on public random inputs according to [HL93], because any adversary can analyze public random strings and exploit them for potential attacks. In particular, we have $\mathbf{d}_2 \in C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + [-\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}]^{nk}$ except with negligible probability. Due to the huge value of $\sqrt{b}$ as compared to h the set C is of small width containing very little entropy. By use of rejection sampling, it is possible to sample a random variable $\mathbf{d}_2' \in C$ according to the probability density function $f(\ \mathbf{x} \mid \mathbf{x}\in C\ ) = e^{-\pi\|\mathbf{x}\|_2^2}/P[\ C\ ]$ in order to get a realization of a continuous Gaussian. More specifically, the first signer samples according to the basic signature scheme a continuous Gaussian $\mathbf{d}_2$, which lies in any set C with probability $P[\ C\ ]$ due to the independence of $\mathbf{d}_2$, and outputs the signature subvector $\mathbf{z}^{(2)}$. The second signer extracts the public randomness, namely the target range C of $\mathbf{d}_2$, and samples secretly $\mathbf{d}_2'$ according to $f(\ \mathbf{x} \mid \mathbf{x}\in C\ )$. Employing public and secret randomness results in a random vector $\mathbf{d}_2'$ following the probability density function $f(\ \mathbf{x} \mid \mathbf{x}\in C\ )\cdot P[\ C\ ] = f(\ \mathbf{x}\ ) = e^{-\pi\|\mathbf{x}\|_2^2}$, which is distributed just as $\mathcal{D}_1^{nk}$ applying conditional probability rules. As a consequence, by one of our main statements (Theorem 3) the difference $\left\|\mathbf{z}^{(2)} - \mathbf{z}_1^{(2)}\right\|_\infty \leq 2h$ requires at most 7 bits per entry with $\mathbf{z}_1 = (\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)})$ being the signature of the second signer. Any number of parties with different secret keys can utilize the same source of public randomness in the same manner. Hence, an abitrary signature $\mathbf{z}_1$ can be represented by the centroid $\mathbf{z}^{(2)}$ in combination with the compressed signature $(\mathbf{z}_1^{(1)}, \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)})$ (see Figure 3). This, however, requires $\mathbf{z}^{(2)}$ always to be fresh such that the stream of signatures $\mathbf{z}_1$ generated by a certain signer are uncorrelated. But one can do more when changing the way of generating the centroid $\mathbf{z}^{(2)}$. Since $\mathbf{z}^{(2)}$ serves as centroid for compression and is required in order to exctract public randomness, any signer can instead sample a fresh and short random seed $\mathbf{r} \in \{0,1\}^\mu$ as input to a discrete Gaussian sampler producing vectors being distributed just like signatures. By doing this, our construction even allows to compress individual signatures because storing the large centroid can now be omitted and is particularly replaced by this seed. From $\mathbf{r}$ one can deterministically recover the centroid and uncompress signatures. As a consequence, the verification costs increase due to an additional call to the discrete Gaussian sampler before checking the validity. Employing this strategy, we achieve storage improvement factors of $\lg n$.

		Signature size in [kB] before/after comp.			Compression rate [%]			Factor improvement			Entropy of Randomness in $\mathbf{y}$ $\approx$	
n	k	This work	[GLP12]	Combination	This work	[GLP12]	Combination	This work	[GLP12]	Combination	Public	Secret
512	2^{14}	1.9 / 0.8	1.9 / 1.1	1.9 / 0.55	60	42	71	2.5	1.7	3.4	2^{13}	2^{12}
1024	2^{15}	4 / 1.5	4 / 2.3	4/1.0	63	43	73	2.7	1.7	3.8	2^{14}	2^{13}

Table 2. Compression rates in the ring and matrix variant for different parameter sets.

Compression of Lyubashevsky Signatures. Previous compression algorithms as provided in [DDLL13], [GLP12] decrease the signature size by dropping the low order bits. Thus, the original signatures can no longer be recovered. This obviously impacts the security of the scheme, since it reduces to easier instances of SIS. By our generic construction, however, we are able to achieve higher compression rates without affecting the security of the scheme. In fact, the original signatures can always be recovered. Our algorithm is directly applicable to the line of signature schemes proposed in [DDLL13],[GLP12],[Lyu12]. For instance, if we consider the efficient instantiations from [GLP12], a signature has exactly the same representation as a Schnorr-signature. In particular, a signature is specified by a tupel $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ with $\mathbf{z}_i = \mathbf{s}_i \cdot \mathbf{c} + \mathbf{y}_i$, where $\|\mathbf{s}_i \cdot \mathbf{c}\|_\infty \leq 2^5$ and $\mathbf{y}_i$ denotes a polynomial with coefficients sampled uniformly at random from $[-k, k]$. Note that the compression algorithm of [GLP12] is only applied to $\mathbf{z}_2$, whereas our algorithm does not differ between $\mathbf{z}_1$ and $\mathbf{z}_2$. In fact, we will show that it is even possible to combine both algorithms in order to furtherly compress signatures. If we apply our compression algorithm on $\mathbf{z}_1$ and drop the low order bits from $\mathbf{z}_2$, we require approximately 4400 bits for $n = 512$ as compared to ≈ 15300 bits without compression. This implies an improvement factor of approximately 3.4.

Aggregation of Signatures (AS). The above-described compression algorithm represents the heart of our conceptually new aggregate signature scheme. An intuitive way of aggregating signatures is to let the signers independently compress their signatures before forwarding them to the verifier. As a disadvantage, such a strategy places a burden on the verifier as it is required to invoke the Gaussian sampler for each transmitted seed, which leads to unsatisfactory running times due to costly computations. To overcome this obstacle, the signers agree on a random seed prior to the actual scheme execution. As a result, the verifier calls the Gaussian sampler once, whose output is used as centroid for each compressed signature (see Figures 3 and Figure 1). This drastically reduces the verification costs as well as the number of seeds to be transmitted. We then show that for any desired number of signers l, we find an n_l such that the aggregate signature is at most as large as an individual signature for $n \geq n_l$ and thus satisfies the standard definition of an aggregate signature scheme. Furthermore, one notices as an additional benefit that an aggregate signature is not completely rejected, if one part of the aggregate fails to verify, which is apparently different from classical AS schemes. In fact, only the signature, that failed the checks, is considered not valid. A potentially interesting modification requires the seed to be made a shared secret among the participants, which are subsequently the only ones being capable of uncompressing and verifying the signatures. We exemplify the applicability of our construction within wireless sensor networks. We particularly show that it is conceivable to use a predistributed seed together with a counter maintained by each senor node. For any aggregation request, the actual counter is incremented before appending it to the seed, which in turn serves as input to the discrete Gaussian sampler.

1.3 Organization

The rest of this paper is structured as follows:

Section 2 Overviews the relevant background on our notations, compression rates and the Gaussian distribution.

Section 3 Explains the concept of public randomness and introduces the key features of our novel and generic compression algorithm. In particular, it aims at signatures following the abstract representation form $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ subsuming Schnorr-like signature schemes. Furthermore, it presents a description of the main steps required to compress individual signatures.

Section 4 Describes how to compress GPV signatures using the generic framework from Section 3, because GPV signatures are given by $\mathbf{z} = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{x} + \mathbf{p}$ and thus can be viewed as $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ with $\mathbf{s} = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}$.

Section 5 Illustrates the applicability of our framework on the signature schemes due to Lyubashevsky et al. and particularly focuses on the scheme presented in [GLP12]. Conceptually, all of the schemes are built in a Schnorr-like manner $\mathbf{z} = \mathbf{s} \cdot \mathbf{c} + \mathbf{y}$ with $f_{\mathbf{s}}(\mathbf{c}) = \mathbf{s} \cdot \mathbf{c}$.

Section 6 Highlights the possibility to exploit public randomness in order to build aggregate signature schemes. In fact, we provide a description for two generic constructions.

2 Preliminaries

2.1 Notation

We denote vectors by lower-case bold letters e.g. $\mathbf{x}$, whereas for matrices we use upper-case bold letters e.g. $\mathbf{A}$. Integers modulo q are denoted by $\mathbb{Z}_q$ and reals by $\mathbb{R}$. By $\vec{v}_i$ we denote a sequence of elements $v_1, \ldots, v_i$ such as vectors or bit strings. The matrix product $\mathbf{H} = \mathbf{B}\mathbf{B}^\top$ is positive definite for any non-singular matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$. A matrix $\mathbf{B}$ is called square root of $\mathbf{H}$, if $\mathbf{H} = \mathbf{B}\mathbf{B}^\top$. Therefore, we simply write $\sqrt{\mathbf{H}} = \mathbf{B}$. Any positive definite matrix has a square root and can be computed using Cholesky decomposition or related variants such as pivoted Cholesky decomposition. Let $\mathcal{R}_k^{q^n}$ denote a polynomial of degree $n-1$ with coefficients from $[-k, k]$.

2.2 Signature Sizes and Compression Rates

By $\mathsf{size}(\mathbf{z}_i)$ we define the individual signature size of signer i for $1 \leq i \leq l$ and $\mathsf{size}(\mathbf{z}_{AS})$ denotes the size of the compressed or aggregate signature. We derive the compression rate $\mathsf{rate}(\cdot)$ of an aggregate signature scheme as follows

$$\mathsf{rate}(l) = 1 - \tau(l), \quad \tau(l) = \frac{\mathsf{size}(\mathbf{z}_{AS})}{\sum_{i=0}^{l} \mathsf{size}(\mathbf{z}_i)}\,.$$

The size ratio $\tau(\cdot)$ represents the size of the aggregate or compressed signature measured as a percentage of the total size of all individual signatures. Hence, $\mathsf{rate}(l)$ returns the storage savings due to compression or aggregation. The scheme instantiation is said to be optimal in case $\mathsf{rate}(l) = 1 - 1/l$ or $\tau(l) = 1/l$ holds, meaning that in average the aggregate signature has the size of an individual signature. We use these notions of optimality both for SAS and AS schemes rather than for specific ones in order to make the schemes more comparable.

The following lemma provides a bound on the signature size.

Lemma 1. ***([BB13, Lemma 2])*** *Let $\mathbf{v} \in \mathbb{Z}^n$ be a vector with $\| \mathbf{v} \|_2 < b \cdot \sqrt{n}$. Then the maximal bit size needed to store this vector is bounded by $n \cdot (1 + \lceil \log_2(b) \rceil)$.*

2.3 Continuous and Discrete Gaussians

By $\rho : \mathbb{R}^n \to (0, 1]$ we define the n-dimensional Gaussian function $\rho(\mathbf{x}) = e^{-\pi \cdot \|\mathbf{x}\|_2^2}$. It follows $E[\mathbf{x} {\cdot} \mathbf{x}^\top] = \frac{\mathbf{I}}{2\pi}$. Applying a linear function $\mathbf{B}$ on $\mathbf{x}$ with $\mathbf{y} = \mathbf{B}\mathbf{x}$ leads to the following Gaussian function, where $\mathbf{B}$ is a $n \times n$-matrix with linearly independent columns

$$\rho_{\mathbf{B}}(\mathbf{y}) = \rho(\mathbf{B}^{-1}\mathbf{y}) = e^{-\pi \cdot <\mathbf{B}^{-1}\mathbf{y}, \mathbf{B}^{-1}\mathbf{y}>} = e^{-\pi \cdot \mathbf{y}^\top \mathbf{\Sigma}^{-1} \mathbf{y}}, \mathbf{\Sigma} = \mathbf{B}^\top \mathbf{B}\,.$$

One derives the probability density function $f_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) = \frac{\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x})}{\sqrt{\det \mathbf{\Sigma}}}$ of the continuous Gaussian distribution $\mathcal{D}_{\sqrt{\mathbf{\Sigma}}}$ by scaling $\rho_{\sqrt{\mathbf{\Sigma}}}$ by its total measure $\int_{-\infty}^{\infty} \rho_{\sqrt{\mathbf{\Sigma}}} d\mathbf{x} = \sqrt{\det \mathbf{\Sigma}}$. If $\mathbf{\Sigma} = s^2 \cdot \mathbf{I}$, we simply write $f_s(\mathbf{x}) = \rho_s(\mathbf{x})/s^n$. The conditional probability density function is defined by $f_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x} \mid \mathbf{x} \in C) = \frac{\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x})/\sqrt{\mathbf{\Sigma}}}{P[\ C\ ]/\sqrt{\mathbf{\Sigma}}} = \frac{\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x})}{P[\ C\ ]}$ for $\mathbf{x} \in C \subset \mathbb{R}^n$ and $P[\ C\ ] = \int_C \rho_{\sqrt{\mathbf{\Sigma}}} d\mathbf{x}$. The discrete Gaussian distribution $\mathcal{D}_{\Lambda+c, \sqrt{\mathbf{\Sigma}}}$ is defined to have support $\Lambda + c$, where $c \in \mathbb{R}$ and $\Lambda \subset \mathbb{R}^n$ is a lattice. For $\mathbf{x} \in \Lambda + c$, it basically assigns the probability

$$\mathcal{D}_{\Lambda+c, \sqrt{\mathbf{\Sigma}}}(\mathbf{x}) = \frac{\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x})}{\rho_{\sqrt{\mathbf{\Sigma}}}(\Lambda + c)}\,.$$

Definition 1 (Statistical distance). *The statistical distance of two distributions $\mathcal{X}_n$ and $\mathcal{Y}_n$ denoted by $\Delta(\mathcal{X}_n, \mathcal{Y}_n)$ over a countable set $\mathcal{C}$ is defined by $\Delta(\mathcal{X}_n, \mathcal{Y}_n) := \frac{1}{2}\sum_{s\in\mathcal{C}} |\mathcal{X}_n(s) - \mathcal{Y}_n(s)|$. The distribution $\mathcal{X}_n$ is said to be statistically close to the distribution $\mathcal{Y}_n$ if the statistical distance $\Delta(\mathcal{B}_1, \mathcal{B}_2)$ is negligible in n (related to the distributions).*

We recall the smoothing parameter introduced by Micciancio and Regev in [MR04]:

Definition 2. *For any n-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^*\backslash\{0\}) \leq \epsilon$*

For any orthogonalized basis $\tilde{\mathbf{B}}$ of a lattice Λ with basis $\mathbf{B}$, the following bound on the smoothing parameter holds.

Lemma 2. ***([GPV08, Theorem 3.1])*** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. We have*

$$\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{B}} \| \cdot \sqrt{\ln(2n(1+1/\epsilon))/\pi}\,.$$

In particular, for any function $\omega(\sqrt{\log n})$, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \leq \| \tilde{\mathbf{B}} \| \cdot \omega(\sqrt{\log n})$.

2.4 Lattices and the SIS-Problem

A lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$ with $m \geq 0$. Λ is the set containing all integer linear combinations of k linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ with $k \leq m$. Formally speaking, we have $\Lambda = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^k\}$. In this work, we are mostly concerened with q-ary lattices $\Lambda_q^\perp(\mathbf{A})$, where $q > 0$ denotes the modulus, which is bounded by a polynomial in n, and $\mathbf{A} \in \mathbb{Z}^{n\times m}$ is an arbitrary matrix. This lattice contains $q\mathbb{Z}^m$ as sublattice and is, hence, of full-rank. $\Lambda_q^\perp(\mathbf{A})$ is defined by

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{Ax} \equiv \mathbf{0} \mod q\}\,.$$

Any coset $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ can be rewritten as $\mathbf{x} + \Lambda_q^\perp(\mathbf{A})$ with $\mathbf{Ax} \equiv \mathbf{u}$.

For the SIS problem, we consider the full-rank m-dimesnional integer lattices $\Lambda_q^\perp(\mathbf{A})$ consisting of all vectors that belong to the kernel of the matrix $\mathbf{A}$. More specifically, $SIS_{q,n,\beta}$ is an average-case variant of the approximate shortest vector problem on $\Lambda_q^\perp(\mathbf{A})$ for $\beta > 0$. Given is a uniform random matrix $\mathbf{A} \in \mathbb{Z}^{n\times m}$ with $m = poly(n)$, the problem is to find a non-zero vector $\mathbf{x} \in \Lambda_q^\perp(\mathbf{A})$ such that $\| \mathbf{x} \| < \beta$. For $q \geq \beta\sqrt{n}\omega(\sqrt{\log n})$ finding a solution to this problem is at least as hard as probabilistically $\tilde{O}(\beta\sqrt{n})$-approximating the Shortest Independent Vector Problem on n-dimensional lattices in the worst-case [GPV08,Ajt96].

3 Generic Lossless Compression of Schnorr-like Signatures

In this section we introduce a novel compression algorithm for signatures following the Schnorr construction. Conceptually, Schnorr-like signature schemes are characterized by simple representations and efficient operations. After establishing a framework for lossless compression, we show how to derive customized compression algorithms for the signature schemes due to Lyubashevsky et al. and the GPV signature scheme [GPV08] in conjunction with [MP12,BB13].

In general, lossless compression of data aims at reducing the bits needed to identify a data unit by removing statistical redundancy without loss of quality. Vector quantization [GG91,Gra84] is a technique from signal processing that belongs to the class of lossy data compression algorithms. It divides a large set of data viewed as vectors into clusters. For each cluster, the algorithms heuristically determine a centroid such that the distance between any vector in the cluster and its centroid is minimized. The whole set of data points is then represented by the centroids. For instance, such algorithms are employed for audio and video compressions like the Twin vector quantization (VQF) for MPEG-4. In order to achieve lossless compression, it is essentially required to store the direction vectors, which preferably should have small entries. But in practice, lossless compression strategies based on vector quantization techniques are rather rare due to low compression rates as compared to other alternatives. However, the approach we propose, makes use of the fact that the centroids are known just before sampling the signatures, which is different to current vector quantization techniques. In particular, we exploit the structure and constituents of signatures in order to reduce the amount of information needed to recover the signatures. Conceptually, one defines the centroids in advance and each signer samples its signature around the centroids (see Figure 1). Doing this, one has only to store the direction vectors rather than all individual signatures as depicted in Figure 2 and Figure 3. We even show that storing the large centroid (see Figure 3) is in fact not needed due to the existence of simulators for random vectors or discrete Gaussians providing the required public randomness. By means of a short random seed, which serves as input to a RO or a discrete Gaussian sampler acting as simulator for signatures, one can deterministically recover the centroid. Following this strategy, we achieve storage improvement factors of about $2.5 - 3.8$ for practical parameters and approximately $\lg n$ for the general case. The compression factor is asymptotically optimal in the main security parameters.

3.1 Lossless Compression Algorithm

In the following, we introduce our generic compression algorithm. We call it the LCPR algorithm (Lattice-based Compression from Public Randomness). We consider two approaches. The first approach compresses signatures with respect to a given signature serving as a centroid. Therefore, we shortly write $\mathbf{v}$ is compressed wrt $\mathbf{w}$, when $\mathbf{w}$ is used as the source of public randomness and acts as the centroid for compression.

Algorithm 1: Generic Compression

Data: Fresh signature $\mathbf{z}_1 = f_{\mathbf{s}_1}(\mathbf{c}_1) + \mathbf{y}_1 \in \mathbb{Z}^m$ of Signer 1
with $\mathbf{y}_1 \sim \mathcal{Y}$ and $\mathbf{z}_1 \sim \mathcal{Z}$

Set $h = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
Set $C = \mathbf{z}_1 + [-h, h]^m,\ P[\ C\ ] := P_{\mathbf{y}_1 \sim \mathcal{Y}}(\mathbf{y}_1 \in C)$
Sample $\mathbf{y}_2 \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
$\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$
Output $\mathbf{z} = (\mathbf{z}_1,\ \mathbf{z}_1 - \mathbf{z}_2)$

Algorithm 2: Single Signature Compression

Data: Distribution of signatures $\mathcal{Z}$

Sample $\mathbf{r} \leftarrow \mathcal{U}(\{0,1\}^l)$
Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
Set $h = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
Set $C = \mathbf{z}_1 + [-h, h]^m, P[\ C\ ] := P_{\mathbf{y} \sim \mathcal{Y}}(\mathbf{y} \in C)$
Sample $\mathbf{y}_2 \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
$\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$
Output $\mathbf{z} = (\mathbf{r},\ \mathbf{z}_1 - \mathbf{z}_2)$

Fig. 4. Lossless compression algorithms

Informal Description

In Figure 4 we present two generic compression algorithms. We briefly describe the main steps required to compress a single signature with respect to a given fresh signature (Algorithm 1) or using a simulator for signatures (Algorithm 2) with a short input seed. Therefore, we consider Schnorr-like signatures in a more abstract representation $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$, where $f_{\mathbf{s}}(\mathbf{c})$ describes a function of the secret key and is, hence, kept secret within the process of signature generation. However, $\mathbf{y}$ is called the masking term required to conceal the secret key and to obtain the desired target distribution of the signature. In many schemes the magnitude of the entries in $\mathbf{y}$ are huge as compared to $f_{\mathbf{s}}(\mathbf{c})$. This offers the opportunity to read and reuse public randomness. In Algorithm 1 a fresh signature $\mathbf{z}_1$ of an arbitrary signer is given. By using only public parameters a second signer, that is different from the first signer, extracts public randomness by determining a narrow set C such that $\mathbf{y}_1 \in C$ with overwhelming probability. Subsequently, he samples its own masking term $\mathbf{y}_2$ secretly from the same set using the conditional probability distribution $\mathcal{Y}(\mathbf{x})/P[\, C\,]$, where $P[\, C\,]$ denotes the probability of the event $\mathbf{y} \in C$ under the distribution $\mathcal{Y}$. Finally, the signer outputs a compressed signature $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ with $\mathbf{z}_2 = f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2$. Algorithm 2 allows to compress individual signatures without involving any other party providing a signature. In fact, the distribution $\mathcal{Z}$ of a signature can be simulated by use of a random oracle $H : \{0,1\}^{\mu} \rightarrow \{0,1\}^{t}$ with $\mu < t$ in combination with a rejection sampling algorithm. Therefore, we replace a real signature by a sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ with input seed $\mathbf{r} \leftarrow_R \{0,1\}^{\mu}$. The remaining steps are identical to those in Algorithm 1. In the last step, however, the signer outputs the compression $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2)$ including the short seed rather than a huge signature $\mathbf{z}_1$. We note, that arbitrary many other signers can exploit the same public randomness using either of the algorithms. But the same signer is not allowed to reuse the same randomness twice, since the distribution of own signatures would otherwise change due to correlations. Uncompressing signatures is trivial and requires for Algorithm 2 to recover $\mathbf{z}_1$ using the seed $\mathbf{r}$.

3.2 Analysis

The authors of [HL93] were the first classifying the notion of randomness into its public and secret portion. Public accessible randomness is the part that can be read by all parties and particularly also by an adversary. The secret portion of the randomness, on the other hand, is only known to the party enacting the cryptographic primitive. This distinction is essential since a potential attacker can exploit public randomness in order to mount an attack on the particular cryptographic primitive. As a consequence, the security of any scheme should mainly depend on the secret portion of randomness. However, the authors made such a distinction only for uniformly random strings. In our work, we extend this notion also to other distributions such as Gaussians-like distributions and show how this allows to build a strong compression algorithm. The key idea underlying this construction is to reuse public randomness in order to sample signatures within short distance to the centroids.

We begin with a formal definition of public randomness.

Lemma 3. *Let $\mathcal{Z}$ be a distribution and $y \leftarrow \mathcal{Z}$ with $y \in C = z + [-h, h]$ for $h > 0$ and $z \in \mathbb{R}$. Then, there exists a bijective transformation $\phi : \{0,1\}^* \times [b_1, b_2) \rightarrow \mathbb{R}$ for $b_2, b_1 \in \mathbb{R}$ with $b_2 - b_1 = 1$ such that $\phi^{-1}(\frac{z}{2h} + [-0.5, 0.5]) = (a_0, \ldots, a_m) \times [b_1, b_2)$ for $m \in \mathbb{N}$. Moreover, we have $\phi^{-1}(\frac{y}{2h}) \in (a_0, \ldots, a_m) \times [b_1, b_2)$ and the probability of $(a_0, \ldots, a_m)$ to occur is $P_{y \sim \mathcal{Z}}[\, y \in C\,]$.*

We call the m-bit string $(a_0, \ldots, a_m)$ public randomness, that is publicly accessible and hence can be read by all parties. Of course, if the range of C gets wider, the number of public random bits decreases. In fact, we show in the following Lemma, that it is possible to reuse $(a_0, \ldots, a_m)$ or less bits of it.

Lemma 4. *Let $y_1 \leftarrow \mathcal{Z}$ with $y_1 \in C = z + [-h, h]$ for $h > 0$ and $z \in \mathbb{R}$. And let $\phi : \{0,1\}^* \times [b_1, b_2) \rightarrow \mathbb{R}$ be a bijective transformation as defined in Lemma 3. Then, we obtain a full realization y from $\mathcal{Z}$ by sampling a number according to the probability distribution $P_{y \sim \mathcal{Z}}[\, y = y_2 \mid y \in C\,]$.*

The proofs of Lemma 3 and Lemma 4 are given in Appendix B.1 resp. Appendix B.2.

The continuous case works similar and requires to consider the probability density function. The following theorem states that exploiting public randomness indeed does not change the distribution of signatures. Furthermore, it indicates a necessary condition for compression.

Theorem 1. *The compression algorithm provided in Figure 4 outputs signatures $\mathbf{z}_2$ distributed according to $\mathcal{Z}$ with $\max \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty \leq 2h$, for $h = \max_{\mathbf{s}} \|f(\mathbf{s})\|_\infty$. Hence, the size of a compressed signature $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2)$ is bounded by*

$$\lceil m \cdot \log_2 2h \rceil + \mu \text{ bits},$$

where $\mathbf{r}$ occupies μ bits of memory.

Proof. For simplicity, assume $m = 1$ and we are given a signature $\mathbf{z}_1 = f_{\mathbf{s}_1}(\mathbf{c}_1) + \mathbf{y}_1$ as in Figure 4, where $\mathbf{y}_1$ is independently sampled according to the distribution $\mathcal{Y}$. Then, we have $\mathbf{y}_1 \in C = \mathbf{z}_1 + [-c \cdot b, c \cdot b]$ for all $c \geq 1$. Thus, let $c = 1$. The probability of $\mathbf{y}_1 \in B = \mathbf{z} + [-h, h]$ for any fixed choice of $\mathbf{z}$ is $P[\ B\ ]$ under the distribution $\mathcal{Y}$, since $\mathbf{y}_1$ is independently sampled. Subsequently, the term $\mathbf{y}_2$ is secretly sampled from C according to the distribution $\mathcal{Y}_1 / P[\ C\ ]$ by reusing the publicly accessible randomness C induced by $\mathbf{y}_1$. We now analyze the distribution of $\mathbf{y}_2$, when exploiting public and secret randomness. Indeed, the probability of $\mathbf{y}_2 = \mathbf{x}$ for $\mathbf{x} \in C$ is given by $P[\mathbf{y}_1 \in C \wedge \mathbf{y}_2 = \mathbf{x}] = P[\ C\ ] \cdot \mathcal{Y}(\mathbf{x}) / P[\ C\ ] = \mathcal{Y}(\mathbf{x})$ according to Lemma 4, which exactly coincides with the required distribution. The continuous case works similar and requires to consider the probability density function. Thus, we obtain $\max \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty = \max \|\mathbf{z}_1 - f_{\mathbf{s}_2}(\mathbf{c}_2) + \mathbf{y}_2\|_\infty \leq 2h$. We observe that $\mathbf{z}_1$ is identified to be the source for public randomness and is subsequently required as a centroid for compression. With focus on compressing individual signatures, we can provide both features by a simulator for the distribution of signatures $\mathcal{Z}$ using a short random seed $\mathbf{r} \in \{0, 1\}^\mu$ as input to a cryptographic hash function modeled as random oracle in combination with a rejection sampler. Following this approach, $\mathbf{z}_1$ is replaced by $\mathbf{r}$ and can deterministically be recovered at any time by use of the simulator. Thus, the signature size is bounded by $\lceil n \cdot \log_2 2h \rceil + \mu$ bits, where μ denotes the bit size of $\mathbf{r}$. As a further advantage, the public randomness can be reused by all parties in an analogous manner. □

3.3 Security

The following Theorem essentially states that compressed signatures are as secure as uncompressed ones.

Theorem 2. *Compressed signatures are indistinguishable from uncompressed signatures.*

The proof of this Theorem is given in Appendix B.3. Via a sequence of games, we show that a challenger, who is given access to an oracle for uncompressed signatures, is able to compress signatures. In fact, we use Algorithm 1 as an oracle whose output vector is distributed like a signature and finally serves as a centroid.

3.4 Compression Rate of Individual Signatures

Let $h = \max_{\mathbf{s}, \mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$ and $\mathbf{z}$ be the centroid generated by use of the seed $\mathbf{r}$ of size μ bits serving as input to a simulator for signatures. The compression rate of an individual signature is given by

$$\mathsf{rate}(1) = 1 - \frac{\lceil n \cdot \log_2 2h \rceil + \mu}{\lceil n \cdot \log_2 \max \|\mathbf{z}_1\|_\infty \rceil},$$

where the denominator indicates the maximum bit size of an uncompressed signature. In many signature schemes [DDLL13,MP12,Lyu12,GLP12,Lyu09,GPV08], we have $\max \|\mathbf{z}\|_\infty = \tilde{O}(n)$ or $\tilde{O}(n^{1/2})$ dependend on the scheme and its instantiation with $\max \|\mathbf{z}-\mathbf{z}_1\|_\infty = o(n)$, when applying the compression algorithm from Section 3.1. Following this, we achieve compression rates of roughly

$$\tau(1) = 1 - \frac{o(\log_2 n)}{\tilde{O}(\log_2 n)}$$

implying asymptotically an improvement factor of $O(\log n)$.

4 Compression of GPV signatures

In the following section, we provide a detailed description of how to apply the proposed compression algorithm on GPV signatures that are produced by means of the trapdoor construction [MP12]. Signatures generated within this framework essentially resemble Schnorr signatures $(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \cdot \mathbf{x} + \mathbf{p}$ with $\mathbf{z}^{(1)} \in \mathbb{Z}^{2n}$ and $\mathbf{z}^{(2)} \in \mathbb{Z}^{nk}$. Hence, a signature is of the form $\mathbf{z} = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ in accordance to the abstract representation from Section 3 with $\mathbf{s} = \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}$. It is even possible to split the signature into $\mathbf{z} = (f_{\mathbf{R}}(\mathbf{c}) + \mathbf{y}^{(1)}, f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)})$. The exact specifications of $f(\cdot)$, $\mathbf{y}$ and $\mathbf{c}$ will be given below. However, we will focus particularly on the lower part of the signature $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)} = \mathbf{I}\cdot\mathbf{x} + \mathbf{p}^{(2)}$ due to the large difference of magnitudes associated to $\mathbf{x}$ and $\mathbf{p}^{(2)}$.

Suppose we have l parties $S_1, \ldots, S_l$ that want to sign individual messages $m_1, \ldots, m_l$. For the sake of simplicity, we restrict to the case, where $l = 2$ and both parties use in accordance to the basic signature scheme in Appendix C.1 the same signing parameter s, security parameter n and modulus $q = 2^k$, meaning that the trapdoor functions $f_{\mathbf{A}_1}$ and $f_{\mathbf{A}_2}$ have the same domain $B = \{\, \mathbf{z} \in \mathbb{Z}^{n(2+k)} \mid \|\mathbf{z}\| \le s\sqrt{n(2+k)} \,\}$ and range $R = \mathbb{Z}_q^n$. Our compression strategy focuses on the signature subvector $\mathbf{z}^{(2)} = \mathbf{x} + \mathbf{p}^{(2)} \in \mathbb{Z}^{n\cdot k}$, where $\mathbf{p}^{(2)}$ is distributed as $\tilde{\mathbf{p}}^{(2)} + \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \tilde{\mathbf{p}}^{(2)}, r}$ with $\tilde{\mathbf{p}}^{(2)} \leftarrow \sqrt{b}\cdot\mathcal{D}_1^{n\cdot k}$ and $\mathbf{x}$ is sampled from $\mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}), r}$ with $\mathbf{v}_i = H(m_i) - \mathbf{A}_i\mathbf{p}_i$ for $i = 1, 2$. This leads to the following representation of $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{c}) + \mathbf{y}^{(2)}$, where

- $\mathbf{y}^{(2)} \leftarrow \tilde{\mathbf{p}}^{(2)} = \sqrt{b}\cdot\mathbf{d}$ with $\mathbf{d} \leftarrow \mathcal{D}_1^{n\cdot k}$
- $f_{\mathbf{I}}(\mathbf{x}, \mathbf{y}) := \mathbf{I}\cdot\mathbf{x} + \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \mathbf{y}^{(2)}, r}$ with $\mathbf{c} = (\mathbf{x}, \mathbf{y})$.

Based on this representation we can apply the tools developed in Section 3. In fact, we have

$$\mathbf{y}_2 \in \mathbf{z}^{(2)} + [-h, h]^{nk} \iff \mathbf{d} \in C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + \left[-\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}\right]^{nk}, \quad h = \max \|f_{\mathbf{I}}(\mathbf{c})\|_\infty .$$

Prior to stating the main Theorem of this section, which indicates an upper-bound for the size of a compressed signature, we prove some useful lemmas. In Lemma 5 we essentially show that we can sample any continuous Gaussian $d \leftarrow \mathcal{D}_1$ by first sampling a set B_i with probability $P[\, B_i \,]$ and then sampling a continuous Gaussian from B_i according to the probability densitity function $f(x \mid x \in B)$. In Lemma 6 we state a more general result than [Ban95, Lemma 2.4]. It is a very helpful instrument in order to bound sums of discrete Gaussians having different supports Λ_i, parameters s_i and centers c_i. It trivially subsumes Lemma [Ban95, Lemma 2.4]. By use of this result we give an upper-bound for h and hence for the compressed signature. In Theorem 3 we prove that an arbitrary signer, that is different from the first one, can reuse public randomness by sampling its own continuous Gaussian from C such that the difference of the lower part of its signature to the centroid $\mathbf{z}^{(2)}$ is sufficiently small.

Lemma 5. *Let X be distributed according to the countinous Gaussian distribution $\mathcal{D}_1$ with parameter $s = 1$ and center $\mu = 0$. Directly sampling $d \leftarrow \mathcal{D}_1$ is equivalent to first sampling a set B_i with probability $P[B_i] = \int_{B_i} e^{-\pi x^2} dx$ and then sampling a continuous Gaussian from B_i according to the probability density function $f(x \mid x \in B_i) = \frac{1}{P[B_i]} e^{-\pi x^2}$ for $x \in B_i$, where B_i depicts a partition of $\mathbb{R}$ for $1 \le i \le n$.*

Lemma 6. *Let $(\Lambda_i)_{1\leq i\leq n} \in \mathbb{R}^{n_i}$ be a sequence of n_i-dimensionial lattices. Then for any reals $s_i \neq s_j > 0$ such that $1 \leq i, j \leq n$ and $T > 0$, and $x_i \in \mathbb{R}^{n_i}$, we have*

$$\underset{d_i \sim \mathcal{D}_{\Lambda_i, c_i, s_i}}{Pr}[|\sum_{i=1}^{k} \langle \mathbf{x}_i, \mathbf{d}_i - c_i \rangle| \geq T \cdot \| (s_1\mathbf{x}_1, \ldots, s_k\mathbf{x}_k) \|] < 2e^{-\pi T^2}$$

The proofs of these lemmas are given in Appendix C.2 and Appendix C.3. In the following, we state our main theorem of this section, which enables an arbitrary group of signers to compress signatures.

Theorem 3. *Assume we have access to an oracle (e.g. a signing or discrete Gaussian oracle) providing a spherically distributed signature $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ with $\mathbf{z}^{(1)} \in \mathbb{Z}_q^{2n}$, $\mathbf{z}^{(2)} \in \mathbb{Z}_q^{nk}$ and parameter s according to the signing algorithm from Appendix C.1. Then, each signer S_i is capable of producing spherically distributed signatures $\mathbf{z}_i = (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)})$ such that the following bound on $\mathbf{z}^{(2)} - \mathbf{z}_i^{(2)}$ holds with overwhelming probability*

$$\log_2 \| \mathbf{z}^{(2)} - \mathbf{z}_i^{(2)} \|_\infty \leq 7\,.$$

The proof of this theorem is given in Appendix C.4 explaining the usage publicly accessible randomness in order to sample within short distance to the centroid $\mathbf{z}^{(2)}$. By the same arguments as in Section 3, it enables an arbitrary signer S_i to secretly sample a continuous Gaussian vector $\mathbf{d}_i$ from $C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + [\frac{h}{\sqrt{b}}, \frac{h}{\sqrt{b}}]^{nk}$ using the conditional rejection sampler as provided in Appendix C.5. The public randomness is supplied by $\mathbf{z}^{(2)}$ and can be reused by an unbounded number of users.

The following result shows that it is possible to leave out the first n entries from $\mathbf{z}$, which can always be recovered due to the existence of the identity submatrix in $\mathbf{A}$.

Lemma 7. *Suppose $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \mathbf{z}^{(3)})$ is a signature for a message with hash value $H(m)$ under public key $\mathbf{A} = [\ \mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\mathbf{z}^{(1)},\ \mathbf{z}^{(2)} \in \mathbb{Z}^n$. Then, the signer requires only to output $(\mathbf{z}^{(2)}, \mathbf{z}^{(3)}) \in \mathbb{Z}^{n(k+1)}$ in order to ensure correct verification.*

Proof. The verifier computes $\mathbf{t} = H(m)$ and defines $\mathbf{z}^{(1)} := \mathbf{t} - [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \cdot (\mathbf{z}^{(2)}, \mathbf{z}^{(3)}) \in \mathbb{Z}^{n(k+1)}$. Then, the verifier needs only to check the validity of $\|\mathbf{z}\| \leq s\sqrt{n(k+2)}$, since $\mathbf{A} \cdot \mathbf{z} = H(m)$ holds per definition of $\mathbf{t}$. ☐

4.1 Compression of Individual GPV Signatures ($l = 1$)

Our goal is to allow an individual signer to compress its own signature following the generic approach from Section 3.1. In particular, we aim at replacing the large centroid by a short uniform random string that is used to produce vectors being distributed just like GPV signatures. As a result, we have a way of simulating signatures such that the output vectors take over the role of the centroid. In fact, lattice-based GPV signatures are distributed just like discrete Gaussian vectors. Therefore, a discrete Gaussian sampler can be used as a simulator for signatures providing the required public randomness. We know from previous works that a discrete Gaussian can be generated by rejection sampling algorithms parametrized by sequences of uniformly distributed numbers [GPV08,Lyu12]. But it is also possible to produce discrete Gaussians by means of a continuous Gaussian sampler in combination with the technique from [Pei10]. Therefore, suppose we want to sample a vector being distributed negligibly close to a discrete Gaussian with parameter s representing the centroid as assumed by Theorem 3. According to the proof, we ouput a vector $\mathbf{z}^{(2)}$ distributed as $\mathbf{x}+\mathbf{c}+\mathcal{D}_{\mathbb{Z}^{n\cdot k}-\mathbf{c},a}$, where $\mathbf{c} = \sqrt{s^2 - 5a^2}\cdot\mathbf{d}$, $\mathbf{d} \xleftarrow{\$} D_1^{n\cdot k}$ and $\mathbf{x} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{n\cdot k},r}$ holds. Following [Pei10] this is equivalent to first generating a continuous Gaussian vector $\mathbf{d}$ with parameter 1, multiplying it with

$\sqrt{s^2-a^2}$ and rounding each component of the vector to a nearby integer using the randomized rounding operation with parameter a. This produces a vector distributed as $\lceil\sqrt{s^2-a^2}\rfloor_a = \sqrt{b}\cdot\mathbf{d}+\mathcal{D}_{\mathbb{Z}^{n\cdot k}-\sqrt{b}\cdot\mathbf{d},a}$ with $b = s^2-a^2$. Note that the randomized rounding operation outputs a random vector that is distributed just like a discrete Gaussian. Thus, for the scheme to work, a potential signer samples a fresh random seed $\mathbf{r}$ of size μ bits as input to a cryptographic hash function modeled as random oracle outputting a sequenence of random numbers that in turn serve as input to a discrete Gaussian sampler. Applying the compression algorithm (Algorithm 2) and using Theorem 3, the signer outputs the public seed $\mathbf{r}$ for $\mathbf{z}^{(2)}$ and a compressed signature $(\bar{\mathbf{z}}_1, \mathbf{z}^{(2)}-\mathbf{z}_1^{(2)})$, where $\bar{\mathbf{z}}_1$ contains only the last n entries of $\mathbf{z}_1^{(1)} \in \mathbb{Z}^{2n}$ as per Lemma 7. The size of the compressed signature amounts to approximately $n(\log_2(4.7\cdot s)+k\cdot 7)+\mu$ bits as compared to $n(k+2)\cdot\log_2(4.7\cdot s)$ bits without compression. The verifier receives the compressed signature and computes the discrete Gaussian $\mathbf{z}^{(2)}$ using the seed $\mathbf{r}$. He then uncompresses the signature to $(\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)})$ and verifies the GPV signature by invoking VerifyGPV (see Appendix A.1).

4.2 Analysis

In this section we analyze the compression rate of the signature scheme. A simple and practical way of comparing compressed signatures is to use the ratio of signature sizes $\mathsf{size}(\mathbf{z}_i) before and after compression \mathsf{size}(\mathbf{z}_{CS})$. We then define by

$$\mathsf{rate}(l) = 1 - \frac{\mathsf{size}(\mathbf{z}_{CS})}{\sum_i^l \mathsf{size}(\mathbf{z}_i)}$$

the compression rate, which represents the amount of storage that has been saved due to compression.

For analyzing the compression rate and its asymptotics, we first consider a lower bound on the compression rate starting with the individual case $l=1$. Let $\mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^{nk},s}$ be the centroid sampled by a simulator for signatures such as a discrete Gaussian sampler using a seed $\mathbf{r} \in \{0,1\}^\mu$ as input. A compressed signature $(\mathbf{r}, \bar{\mathbf{z}}_1, \mathbf{z}-\mathbf{z}_1^{(2)})$ consists of $\bar{\mathbf{z}}_1$ of size $n\cdot\lceil\log_2(4.7\cdot s)\rceil$ bits, $\mathbf{z}-\mathbf{z}_1^{(2)}$ of size $n\cdot k\cdot\lceil\log_2(2\cdot 4.7\cdot\sqrt{5}a)\rceil$ bits and a short seed $\mathbf{r}$ of size μ bits. Without compression, however, the size of an individual standard GPV signature amounts to $n\cdot(k+2)\cdot\lceil\log_2(4.7\cdot s)\rceil$ bits

$$\begin{aligned}
\mathsf{rate}(1) &= 1 - \frac{n\cdot\lceil\log_2(4.7\cdot s)\rceil + n\cdot k\cdot\lceil\log_2(\sqrt{448.8}a)\rceil+\mu}{n\cdot(k+2)\cdot\lceil\log_2(4.7\cdot s)\rceil} && (1)\\
&\geq 1 - \left(\frac{1}{k+2} + \frac{\lceil\log_2(\sqrt{448.8}a)\rceil+1}{\lceil\log_2(4.7\cdot s)\rceil}\right) && (2)\\
&= 1 - \left(\frac{1}{k+2} + \frac{o(\log_2(\ln n))}{O(\log_2(n))}\right). && (3)
\end{aligned}$$

The compression factor converges for increasing n towards $1-1/k+2$, if k is chosen to be constant. But in fact, since the parameter s grows with increasing n, it is required to increase k as a function of n for the scheme to be secure. Typically, one requires $q=2^k=Poly(n)$, which is equivalent to $k=O(\log_2(n))$, implying an improvement factor of approximately $\lg n$. In this case, the compression factor converges towards 1, which is asymptotically unbounded.

A more practical way of measuring the concrete compression rates is to consider the length of the compressed signature and subsequently derive from it the storage requirements. Therefore, we recall the representation of a compressed signature according to Theorem 3. Let $x \xleftarrow{\$} \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}+\sqrt{b}\mathbf{d}}(\mathbf{G}),r}$ and $\mathbf{v} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^{nk}-\sqrt{b}\mathbf{d},a}$, then it follows

$$\begin{aligned}\left\|\mathbf{z}-\mathbf{z}_1^{(2)}\right\|_2 &= \left\|\sqrt{b}\cdot\tfrac{\mathbf{z}}{\sqrt{b}}-\mathbf{x}-\sqrt{b}\cdot\mathbf{d}-\mathbf{v}\right\|_2 \\ &\le \left\|\sqrt{b}\cdot(\tfrac{\mathbf{z}}{\sqrt{b}}-\mathbf{d})-\mathbf{x}\right\|_2+\|\mathbf{v}\|_2\,.\end{aligned}$$

We now consider the expression $\|\mathbf{v}\|_2$, which can be rewritten as $\sqrt{nk}\sqrt{\frac{1}{n\cdot k}\sum_{i=0}^{n\cdot k} v_i^2}$. By the law of large numbers and due to the huge number of samples the estimator $\frac{1}{n\cdot k}\sum_{i=0}^{n\cdot k} v_i^2$ essentially equals to $E[v_i^2]=a^2$ such that $\|\mathbf{v}\|_2$ can be approximated by $\sqrt{n\cdot k}\cdot\sqrt{E[v_i^2]}=a\sqrt{n\cdot k}$. The first expression, however, was a little bit tricky to approximate, since the entries d_i lie in different sets dependend on the entries of $\mathbf{z}$. Signatures produced by the GPV framework basically follow the discrete Gaussian distribution. As a consequence, the random variables $T_i=(\frac{z_i}{\sqrt{b}}-d_i)^2$ with $z_i\sim\mathcal{D}_{\mathbb{Z},s}$ and $d_i\sim\mathcal{D}_1, d_i\in C_i$ are independent and identically distributed such that the law of large numbers applies. Moreover, the squared entries x_i^2 of $\mathbf{x}$ are of finite variance and independent from T_i. For large enough samples, we obtain

$$\begin{aligned}\tfrac{1}{n\cdot k}\sum_{i=1}^{nk}(\sqrt{b}(\tfrac{z_i}{\sqrt{b}}-d_i)-x_i)^2 &\to E[(\sqrt{b}(\frac{z_i}{\sqrt{b}}-d_i)-x_i)^2] \\ &= E[x_i^2]+b\cdot E[T_i^2] \\ &= r^2+b\cdot\sum_{y\in\mathbb{Z}}P[z_i=y]\cdot E[T_i^2\mid z_i=y] \\ &\le r^2+b\cdot\max_{y\in\mathbb{Z}}E[T_i^2\mid z_i=y]\,.\end{aligned}$$

In order to find the maximum conditional expectation value for each considered parameter selection n and k, we run experiments

$$g^2=b\cdot\max_{0\le i\le\lceil 4.7\cdot s\rceil}E\left[\left(\frac{i}{\sqrt{b}}-d_i\right)^2\mid d_i\xleftarrow{\$}\mathcal{D}_1, d_i\in C_i\right],\quad C_i=\frac{i}{\sqrt{b}}+[-\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}},\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}]\,.$$

The conditional expectation is computed as follows

$$\frac{1}{P[C_i]}\int_{C_i}(\frac{i}{\sqrt{b}}-x)^2e^{-\pi x^2}dx\,.$$

Surprisingly, we observed by our experiments that the rounded conditional expectations are all equal. As a result, we have $\left\|\sqrt{b}\cdot(\frac{\mathbf{z}}{\sqrt{b}}-\mathbf{d})\right\|_2\le g\sqrt{n\cdot k}$ and finally $\left\|\mathbf{z}-\mathbf{z}_1^{(2)}\right\|_2\le(\sqrt{g^2+r^2}+a)\cdot\sqrt{n\cdot k}$. Following this approach in combination with Lemma 1, we can estimate the compression rate more precisely. The compressed signature requires $\lceil n\cdot k(1+\log_2(\sqrt{g^2+r^2}+a))\rceil$ bits and the seed $\mathbf{r}$ occupies at most n bits of memory. A standard GPV signature requires $\lceil n(k+2)(1+\log_2(s))\rceil$ bits. It follows

$$\mathsf{rate}(1)=1-\frac{\lceil n(1+\log_2(s))\rceil+\lceil n\cdot k(1+\log_2(\sqrt{g^2+r^2}+a))\rceil+n}{\lceil n(k+2)(1+\log_2(s))\rceil},$$

where g^2 is a parameter dependend value extracted from our experiments. The storage improvement factor is simply the inverse of the fraction.

For $l > 1$ the compression factor is slightly higher, because only one seed of size μ bits is required instead of $l \cdot \mu$ bits. Furthermore, the computational costs decrease due to a single call of the discrete Gaussian sampler as opposed to l calls in case without aggregation. For any number of signers, we can find n_0 such that the size ratio satisfies $\tau(l) \leq 1/l$ for all $n \geq n_0$ or equivalently the aggregate signature is at most as large as an individual one.

The following table depicts the compression rates for individual signatures. It furtherly contains the parameter s and the factor improvement, which is simply defined by the inverse of the size ratio $\tau(1)$, for different parameter sets of the matrix and ring variant. Increasing the parameter n yields higher factor improvements and hence higher compression rates.

		Signature size in [kB] before/after comp.		Compression rate [%]		Factor improvement		Entropy of Randomness $\approx$
n	k	Ring	Mat	Ring	Mat	Ring	Mat	$h_{public}(X) - h_{secret}(X)$
256	25	16 / 6	13 / 5	65	58	2.9	2.4	2^{12}
256	28	18 / 7	14 / 6	65	58	2.9	2.4	2^{12}
512	25	34 / 12	27 / 11	68	60	3.1	2.5	2^{13}
512	28	38 / 14	30 / 12	68	61	3.1	2.5	2^{13}
512	30	40 / 14	32 / 13	68	61	3.2	2.5	2^{13}
1024	28	80 / 27	64 / 24	70	63	3.3	2.7	2^{14}
1024	35	100 / 33	79 / 29	70	63	3.4	2.7	2^{14}

Table 3. Compression rates for different parameter sets.

4.3 Entropy of public and secret randomness

Measuring the public and secret portion of randomness requires to consider the entropy of the relevant quantities. The entropy $h(X)$ represents a mass for the amount of uncertainty stored in a random variable X. We aim at comparing the secret and public randomness stored in the continuous Gaussian vectors sampled in the signing step. Therefore, we have to compute the differential entropy for the distinct randomness portions. The differential entropy of a multivariate continuous Gaussian vector $\mathbf{d}$ with $f(x_1, \ldots, x_n) = e^{-\pi\|\mathbf{x}\|_2^2}$ is determined as follows

$$\begin{aligned} h(\mathbf{d}) &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \ldots, x_m) \cdot \log_2 f(x_1, \ldots, x_m) dx_1 \ldots dx_m \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \ldots, x_m) \cdot \left(-\pi \|\mathbf{x}\|_2^2\right) dx_1 \ldots dx_m \\ &= \frac{1}{2} \log_2(e^m) \quad bits\,. \end{aligned}$$

By outputting the signature the entropy of the continuous Gaussian vector decreases due to the knowledge of C, which leaks information about $\mathbf{d}$. As a consequence, the reduced entropy of $\mathbf{d}$ is now associated to the set $C = \frac{\mathbf{z}^{(2)}}{\sqrt{b}} + [-\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}, \frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}]^{nk}$, which corresponds to the secret randomness. In fact, when sampling perturbations $\mathbf{p}_1$ in the signing step, we require an additional continuous Gaussian vector $\mathbf{d}_1 \xleftarrow{\$} \mathcal{D}_1^{2n}$ that remains completely hidden. We know from information theory that the conditional entropy of $\mathbf{d}$ given $\mathbf{d} \in C$ is given by

$$h(\mathbf{d} \mid C) = \sum_{i=1}^{nk} h(d_i \mid C_i) = -\sum_{i=1}^{nk} \int_{C_i} \frac{f(x_i)}{P[\ C_i\ ]} \cdot \log_2 \frac{f(x_i)}{P[\ C_i\ ]} dx_i$$
$$\leq \sum_{i=1}^{nk} \int_{C_i} \log_2(P[\ C_i\ ]) + \int_{C_i} f(x_i) \cdot \log_2 f(x_i) dx_i \approx n \cdot k \cdot \log_2(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}})\,.$$

The first equality follows from the independence of the entries in $\mathbf{d}_1$. The secret portion of randomness has entropy amounting to $h_{\mathsf{secret}} \approx \frac{1}{2}\log_2(e^{2n}) + n \cdot k \cdot \log_2(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}})$ bits, whereas the public randomness is lower bounded by $h_{\mathsf{public}} = \frac{1}{2}\log_2(e^{n(k+2)}) - h_{\mathsf{secret}} \approx n \cdot k \cdot (\ \frac{1}{2}\log_2(e) + \log_2(\frac{2 \cdot 4.7 \cdot \sqrt{5}a}{\sqrt{b}})\ )$ bits. One notices that the differential entropy can be negative.

5 Compression of Signatures due to Lyubashevsky et al.

In this section we analyze the applicability of our compression algorithm to the line of signature schemes provided in [DDLL13,GLP12,Lyu12,Lyu09]. Conceptually, all of them produce signatures in a Schnorr-like manner. Thus, we can directly apply the framework from Section 3. Previous compression algorithms [DDLL13,GLP12], however, work in a different way as compared to our proposal since the signature size is reduced by dropping the low order bits of certain polynomials, which are subsequently lost without any chance of recovery. As a consequence, one obtains instances of SIS that are easier to solve.
In what follows, we will particularly focus on the lattice-based construction presented in [GLP12] that gained attractiveness due to its simplicity and high efficiency (see Appendix D). As already shown in Table 2 our lossless compression algorithm outperforms the previous one without changing the original hardness assumptions. That is, the compression algorithm uses only the structure of the signature polynomials and ignores the context. Following [GLP12] (see Appendix D), a signature mainly consists of the polynomials $\mathbf{z}_1 = (\mathbf{z}_1^{(1)}, \mathbf{z}_1^{(2)}) = f_{\mathbf{s}}(\mathbf{c}) + \mathbf{y}$ and $\mathbf{c}$ with $\mathbf{z}_1^{(1)} = \mathbf{s}_1^{(1)}\mathbf{c} + \mathbf{y}_1^{(1)}$ and $\mathbf{z}_1^{(2)} = \mathbf{s}_1^{(2)}\mathbf{c} + \mathbf{y}_1^{(2)}$. Applying the compression algorithm on $\mathbf{z}_1$ with $\max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty = \max_{\mathbf{s},\mathbf{c}} \|\mathbf{s} \cdot \mathbf{c}\|_\infty \leq 32$, we obtain the compressed signature $(\mathbf{r}, \mathbf{z}^{(1)} - \mathbf{z}_1^{(1)}, \mathbf{z}^{(2)} - \mathbf{z}_1^{(2)}, \mathbf{c})$, where the centroid $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ is deterministically recovered using a rejection sampling algorithm in combination with a cryptographic hash function modeled as random oracle with input seed $\mathbf{r}$. The maximum bit size required to store a compressed signature is bounded by $2n \cdot \log_2 2\,\|\mathbf{s} \cdot \mathbf{c}\|_\infty + \mu + 2n$ bits as compared to $2n \cdot \log_2(2k+1) + 2n$ bits without compression.

We further note, that it is also possible to combine the compression algorithm Comp_1 from [GLP12] with our proposal Algorithm 2. Indeed, both algorithms work independently since Comp_1 is used to compress $\mathbf{z}_2$, whereas Algorithm 2 is applied on $\mathbf{z}_1$. This greatly improves the sole usage of Comp_1 (see Table 2) leading to compression rates of approximately 70%. But as with the original compression algorithm, combining both algorithms implies loss of data and hence easier instances of SIS. For the other signature schemes, one takes similar steps. One achieves even higher compression rates in case the one bases the security of the scheme on worst-case lattice problems.

5.1 Entropy of Public and Secret Randomness

In the signature scheme from the previous section signature polynomials leak information about $\mathbf{y}_1$ or $\mathbf{y}_2$. These polynomials are selected uniformly at random from $\mathcal{R}_k^{q^n}$. After publishing the signature, any party can read the public randomness. We measure the public portion and secret portion of randomness in terms of entropy for discrete random variables. Due to $\mathbf{y}_1 \in \mathcal{R}_k^{q^n}$ and $\|\mathbf{z} - \mathbf{z}_1\|_\infty \leq 2^6$ the conditional entropy of secret randomness is determined by

$$h_{secret} = \sum_{i=1}^{2n} \log_2 65 = 2n \log_2 65 \,.$$

Hence, the entropy of publicly randomness is given by

$$h_{public} = -\sum_{i=1}^{2n} \log_2 \frac{1}{2k+1} - h_{secret} = 2n \log_2 \frac{(2k+1)}{65} \,.$$

This shows that signatures contain huge amounts of public randomness.

6 Aggregate Signatures

Prior to the description of our novel aggregate signature scheme constructions from public randomness, we start by some definitions. In principal, one differentiates sequential aggregate signature schemes from general aggregate signature schemes.

Definition 3. *In an aggregate signature (AS) scheme l signatures $\mathbf{z}_i$ on l distinct messages m_i from l distinct signers are combined into a single signature $\mathbf{z}_{AS}$ with $1 \leq i \leq l$. The resulting aggregate signature $\mathbf{z}_{AS}$ has essentially the size of an individual signature. Moreover, the aggregate signature will convince the verifier that the l signers indeed signed the l messages.*

Construction 4 *The generic construction of an aggregate signature scheme (AS) involves 4 algorithms.*

$\mathsf{KGen}(1^n, i with\ 1 \leq i \leq l)$*: Output secret key sk_i and public key pk_i to signer i.*

$\mathsf{Sign}(sk_i, m_i)$*: Output individual signature $\mathbf{z}_i$ and message m_i of signer i.*

$\mathsf{AggSign}(\overrightarrow{\mathbf{pk}}, \vec{\mathbf{z}}, \vec{\mathbf{m}})$*: Output aggregate signature $\mathbf{z}_{AS}$.*

$\mathsf{Verify}(\overrightarrow{\mathbf{pk}}, \mathbf{z}_{AS}, \vec{\mathbf{m}})$*: Verify the aggregate signature $\mathbf{z}_{AS}$ with the aid of the public keys $\overrightarrow{\mathbf{pk}} = (pk_1, \ldots, pk_l)$ and messages $\vec{\mathbf{m}} = (m_1, \ldots m_l)$. Output 1 if $\mathbf{z}_{AS}$ is valid, else output 1.*

Definition 4. *In a sequential aggregate signature (SAS) scheme l distinct signers, that are ordered in a chain, sequentially put their signature on messages m_i of choice to the aggregate signature such that the resulting aggregate signature $\mathbf{z}_{SAS}$ has the size of an individual signature. Moreover, the aggregate signature will convince the verifier that the l signers indeed signed the l messages.*

Construction 5 *The generic construction of an sequential aggregate signature scheme (SAS) involves 3 algorithms.*

$\mathsf{KGen}(1^n, i with\ 1 \leq i \leq l)$*: Output secret key sk_i and public key pk_i to signer i.*

$\mathsf{AggSign}(sk_i, \mathbf{z}_{SAS}^{(i-1)}, m_i, \overrightarrow{\mathbf{pk}}_{i-1}, \vec{\mathbf{m}}_{i-1})$*: Output aggregate signature $\mathbf{z}_{AS}^{(i)}$.*

$\mathsf{Verify}(\overrightarrow{\mathbf{pk}}, \mathbf{z}_{SAS} = \mathbf{z}_{SAS}^{(l)}, \overrightarrow{\mathbf{m}})$: *Verify the aggregate signature* $\mathbf{z}_{AS}$ *with the aid of the public keys* $\overrightarrow{\mathbf{pk}} = (pk_1, \ldots, pk_l)$ *and messages* $\overrightarrow{\mathbf{m}} = (m_1, \ldots m_l)$*. Output 1 if* $\mathbf{z}_{SAS}$ *is valid, else output 1.*

These definitions of (sequential) aggregate signature schemes are too strict with regard to the aggregate signature size. But for the lattice-based setting we require a more appropriate definition that accounts for the differing domains and ranges as compared to classical (S)AS schemes. This is due to the compressing property of lattice-based hash functions. The following definition is more flexible and hence allows for different aggregation types.

Definition 5. *We say x-(S)AS, if the aggregate signature* $\mathbf{z}_{(S)AS}$ *is as large as an individual signature extended by c bits of overhead with* $c = (1-x) \cdot \sum_i \textsf{size}(\mathbf{z}_i)$ *and* $x \in [0,1]$*, where* $\textsf{size}(\mathbf{z}_i)$ *denotes the signature size of signer i.*

For $x = 1$, we immediately obtain the aforementioned definition of the classical (S)AS scheme.

As already observed, the generic compression algorithms from Section 3 naturally induce aggregate signature schemes (AS), since all parties are allowed to consume the same source of public randomness. In fact, we only focus on how to generate the masking term $\mathbf{y}$ rather than modifying the underlying signature scheme. We hereby present two different approaches to construct an aggregate signature scheme (see Figure 5 and Figure 6), where a group of signers agree to add signature shares resulting in an aggregate signature. The security of the schemes inherently stems from Theorem 2.

Algorithm 3: AS Scheme: $\mathsf{AggSign}$

Data: Distribution of signatures $\mathcal{Z}$, seed $\mathbf{r} \in \{0,1\}^{\mu}$

for $i = 1$ *to* k **do**
    Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
    Set $b = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
    Set $C = \mathbf{z}_1 + [-b, b]^m, P[\ C\ ] := P_{\mathbf{y} \sim \mathcal{Y}}(\mathbf{y} \in C)$
    Sample $\mathbf{y}_i \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
    $\mathbf{z}_i = f_{\mathbf{s}_i}(\mathbf{c}_i) + \mathbf{y}_i$
end
Output $(\mathbf{r},\ \mathbf{z}_1 - \mathbf{z}_2, \ldots,\ \mathbf{z}_1 - \mathbf{z}_i)$

Algorithm 4: Verification: $\mathsf{AggVerify}$

Data: Aggregate signature $(\mathbf{r},\ \mathbf{z}_1 - \mathbf{z}_2, \ldots,\ \mathbf{z}_1 - \mathbf{z}_l)$, messages $\overrightarrow{\mathbf{m}}$

Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
for $j = 2$ *to* l **do**
    $\mathbf{a}_j = \mathbf{z}_1 - \mathbf{z}_j$
    $\mathbf{z}_j = -\mathbf{a}_j + \mathbf{z}_1$
    if *Verify*$(\mathbf{z}_j, m_j) == 1$ **then**
        $\mathbf{out}_j := 1$
    else
        $\mathbf{out}_j := 0$
    end
end
Output **out**

Fig. 5. Aggregate Signature Scheme

A straight forward approach to realize an AS scheme is to let every signer simply apply the compression algorithm to its own signature as with individual signatures (Algorithm 2) and subsequently assemble them together. This requires to append the random seeds to the aggregate. In addition to the increasing number of seeds to be transmitted, the verifier has to compute the corresponding centroids in order to uncompress the signatures and subsequently check their validity. Hence, the computation costs grow linearly in the number of participating signers. Applying Theorem 1 one resolves both problems in one step since all participants are allowed to use the same centroid for compression (see Figure 1 and Figure 2). Therefore, we require only one fresh random seed per aggregation request. As a result, the verifier has only to compute one discrete Gaussian vector for an arbitrary number of signers, which improves the computation complexity. As opposed to other aggregate signature schemes, the proposed one verifies each compressed signature on its own. Consequently, the aggregate signature need not to be rejected completely in case some signatures fail to verify, which is different from other classical aggregate signature schemes.

But it is also possible to realize a sequential aggregate signature scheme (SAS), where a set of signers ordered in a chain sequentially add their signatures to the aggregate signature. We briefly sketch a potential instantiation of a SAS scheme. For instance, the first signer uses a seed $\mathbf{r} \in \{0,1\}^{\mu}$ in order to compress its signature, which is then handed over together with his public key and message to the next signer, who in turn uncompresses and verifies the signature $\mathbf{z}_1$. Subsequentially, he generates its own signature $\mathbf{z}_2$ using $\mathbf{z}_1$ as a centroid and source of public randomness (see Figure 6). One proceeds analogously for the remaining participants. Finally, the aggregate signature $\mathbf{z}_{SAS} = (\mathbf{r}, \mathbf{z} - \mathbf{z}_1, \ldots, \mathbf{z}_{l-1} - \mathbf{z}_l)$ is output by the aggregate signature algorithm. In order to verify any signature $\mathbf{z}_i$ in the chain, we have to start at the beginning of $\mathbf{z}_{SAS}$ and successively uncompress and verify all signatures of the predecessors. The signature size of the aggregate signature is in both schemes bounded by $l \cdot \lceil n \cdot \log_2 2h \rceil + \mu$ bits.

Indeed, the above described aggregate signature schemes allow the verifier to recover all individual signatures from $\mathbf{z}_{(S)AS}$. The centroid associated to $\mathbf{r}$ connects all the individual signatures together. Furthermore, if the seed is made a shared secret, the aggregate signature can only be recovered and verified by the holders of $\mathbf{r}$. A very interesting application scenario for aggregate signature schemes is given in Appendix E, where cluster-based sensor networks are considered. Sensor networks are characterized by constrained ressources such that one observes an inherent need for aggregate signature schemes reducing the traffic.

Algorithm 5: SAS Scheme: AggSign

Data: Distribution of signatures $\mathcal{Z}$, seed $\mathbf{r} \in \{0,1\}^{\mu}$, AS $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2, \ldots, \mathbf{z}_{i-2} - \mathbf{z}_{i-1})$

Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
Set $b = \max_{\mathbf{s},\mathbf{c}} \|f_{\mathbf{s}}(\mathbf{c})\|_\infty$
for $j = 2$ *to* $i-1$ **do**
  $\mathbf{a}_j = \mathbf{z}_{j-1} - \mathbf{z}_j$
  $\mathbf{z}_j = -\mathbf{a}_j + \mathbf{z}_{j-1}$
end
Set $C_{i-1} = \mathbf{z}_{i-1} + [-b,b]^m$, $P[\ C\ ] := P_{\mathbf{y}\sim\mathcal{Y}}(\mathbf{y} \in C)$
Sample $\mathbf{y}_i \leftarrow \mathcal{Y}(\mathbf{x})/P[\ C\ ],\ \mathbf{x} \in C$
$\mathbf{z}_i = f_{\mathbf{s}_i}(\mathbf{c}_i) + \mathbf{y}_i$
Output $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2, \ldots, \mathbf{z}_{i-1} - \mathbf{z}_i)$

Algorithm 6: Verification: AggVerify

Data: SAS $(\mathbf{r}, \mathbf{z}_1 - \mathbf{z}_2, \ldots, \mathbf{z}_{l-1} - \mathbf{z}_l)$, messages $\vec{\mathbf{m}}$

Sample $\mathbf{z}_1 \leftarrow \mathcal{Z}$ using input seed $\mathbf{r}$
for $j = 2$ *to* l **do**
  $\mathbf{a}_j = \mathbf{z}_{j-1} - \mathbf{z}_j$
  $\mathbf{z}_j = -\mathbf{a}_j + \mathbf{z}_{j-1}$
  if *Verify*$(\mathbf{z}_j, \vec{\mathbf{m}}_j, \vec{\mathbf{a}}_j) == 0$ **then**
    Output **invalid**
  end
end
Output **valid**

Fig. 6. (Sequential) Aggregate Signature Scheme

6.1 Aggregate Signature Scheme in the GPV Setting ($l > 1$)

As already detailed in the previous section, we obtain an aggregate signature scheme from the aforementioned compression algorithm when involving more then one party $l > 1$.

A usable and practical way of instantiating the scheme is to agree on a random string in advance. This is attained, for example, if each signer samples a random salt $\mathbf{r}_i$ and broadcasts it to the remaining parties in order to produce the ultimate seed $\mathbf{r} = H(\mathbf{r}_1, \ldots, \mathbf{r}_l)$ using a cryptographic hash function modeled as random oracle. Each signer maintains a counter that is increased for every aggregation request. The counter is appended to the random string and serves as input to a second hash function, whose output

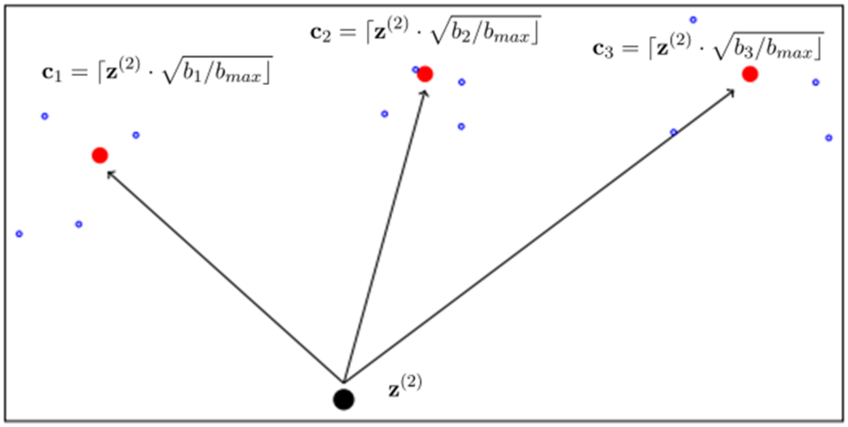

Fig. 7. Red circles are centroids derived from $\mathbf{z}^{(2)}$ and blue circles denote signatures.

sequence is used to sample the centroid. At this point we have to explain how to sample continuous Gaussians in the case when the signers' parameters n_i and k_i differ.

Our goal is to keep the scheme as efficient as with constant parameters. The computation complexity and the number of transmitted seeds should not change. Therefore, one starts by defining the maximum Gaussian parameter $s_{max} = \max_i \; s_i$ and the maximum number of entries $e = \max\{\, n_i \cdot k_i\}$ with s_i, n_i and k_i denoting the parameters of the $i-$th signer. Analogously, we define $b_i := s_i^2 - 5a^2$ and $b_{max} = s_{max}^2 - 5a^2 \geq b_i$. According to Theorem 3 each signer samples a continuous Gaussian from a set of proper width. This can be achieved by sampling $\mathbf{d}_i \xleftarrow{\$} \mathcal{D}_1^{n_i \cdot k_i}$, $\mathbf{d}_i \in C_i = [\frac{\mathbf{z}^{(2)}}{\sqrt{b_{max}}} - \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}, \frac{\mathbf{z}^{(3)}}{\sqrt{b_{max}}} + \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}]^{n_i \cdot k_i}$, where $\mathbf{z}^{(2)}$ is a discrete Gaussian vector with parameter s_{max} and $a = \sqrt{\ln\left(2n_{max}\left(1 + \frac{1}{\epsilon}\right)\right)/\pi}$. The choice of s_{max} implies $C_{max} \subseteq C_i$ and thus provides the required interval width.

Algorithm 7: AS Scheme: $\mathsf{AggSign}$

Data: Seed $\mathbf{r}$

$c_{act} = j$
$\mathbf{t} \leftarrow H(\mathbf{r}, c_{act})$
$\mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^e, s_{max}}(r)$
for $i = 1 \rightarrow l$ **do**
  $\mathbf{w}_i = \lceil \frac{\mathbf{z}}{\sqrt{b_{max}}} \cdot \sqrt{b_i} \rfloor$
  Define $C_i = [\frac{\mathbf{z}}{b_{max}} - \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}, \frac{\mathbf{z}}{b_{max}} + \frac{4.7 \cdot \sqrt{5}a}{\sqrt{b_i}}]$
  $\mathbf{d_1} \leftarrow \mathcal{D}_1^{2n_i}, \mathbf{d_2} \leftarrow \mathcal{D}_1^{n_i \cdot k_i} \;\wedge\; \mathbf{d_2} \in C_i$
  $\mathbf{z}_i = (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{z}_i^{(3)}) \leftarrow f_{A_i}^{-1}(H(m_1))$, (Section C.1)
  $\mathbf{y}_i = (\mathbf{z}_i^{(2)}, \mathbf{w}_i - \mathbf{z}_i^{(3)})$, (Lemma 7, Theorem 3)
end
Output Aggregate $(c_{act}, \mathbf{y}_1, \ldots, \mathbf{y}_l)$

Algorithm 8: Verification: $\mathsf{AggVerify}$

Data: Seed $\mathbf{r}$, $(c_{act}, \mathbf{y}_1, \ldots, \mathbf{y}_l)$, parameters s, n_{max}, k_{max}

$\mathbf{t} \leftarrow H(\mathbf{r}, c_{act})$
$\mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^e, s_{max}}(\mathbf{t})$
for $i = 1 \rightarrow l$ **do**
  $\mathbf{w}_i = \lceil \frac{\mathbf{z}}{\sqrt{b_{max}}} \cdot \sqrt{b_i} \rfloor$
  $\mathbf{y}_i = (\mathbf{z}_i^{(2)}, \mathbf{w}_i - \mathbf{z}_i^{(3)}),\; \mathbf{z}_i = (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{z}_i^{(3)})$
end
$\mathbf{out} = (0, \ldots, 0)$
for $i = 1 \rightarrow l$ **do**
  if $f_{\mathbf{A}_i}(\mathbf{z}_i) = H(m_i) \wedge \;\| \mathbf{z}_i \| < s\sqrt{m_i}$
    $\mathbf{out}_i = 1,\; \mathbf{z}_i$ is valid
end
output $\mathbf{out}$

Fig. 8. Aggregate signature scheme derived from Compression strategy

Due to differing parameters, the number of signature entries varies among the signers such that $n_i \cdot k_i \leq e$. Therefore one takes as many entries as required from $\mathbf{z}^{(2)}$ starting from the first component of $\mathbf{z}^{(2)}$. Since each signer operates with different parameters b_i and s_i when sampling signatures, we have to derive individual centroids from $\mathbf{z}^{(2)}$ efficiently. The most reasonable way of doing this is to use $\mathbf{w}_i = \lceil \frac{\mathbf{z}^{(2)}}{\sqrt{b_{max}}} \cdot \sqrt{b_i} \rfloor$ for signer i as the centroid such that its difference to the center of the scaled sets $\sqrt{b_i} \cdot C_i$ is smaller than 0.5 in each entry. As a result we have $\log_2 \parallel \mathbf{z}^{(2)} - \mathbf{z}_i^{(2)} \parallel_\infty \leq 7$ except with negligible probability. In case we have $b_i = b_{max}$ for all i, $\mathbf{z}^{(2)}$ is obtained as the centroid for all signers. In Figure 8 we provide the main steps of the aggregate signature scheme. Here $\mathcal{D}_{\mathbb{Z}^e, s_{max}}(\mathbf{r})$ denotes the discrete Gaussian sampler simulating signatures with parameter s_{max}, input seed $\mathbf{r}$ and $e = \max\{\ n_i \cdot k_i\}$.

References

[Ajt96] Miklós Ajtai, *Generating hard instances of lattice problems (extended abstract)*, 28th Annual ACM Symposium on Theory of Computing, ACM Press, May 1996, pp. 99–108.

[Ajt99] ———, *Generating hard instances of the short basis problem*, ICALP, LNCS, Springer, 1999, pp. 1–9.

[AP09] Joël Alwen and Chris Peikert, *Generating shorter bases for hard random lattices*, STACS, LIPIcs, vol. 3, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009, pp. 75–86.

[Ban95] W. Banaszczyk, *Inequalities for convex bodies and polar reciprocal lattices in r^n*, Discrete Computational Geometry **13** (1995), no. 1, 217–231.

[BB13] Rachid El Bansarkhani and Johannes Buchmann, *Improvement and efficient implementation of a lattice-based signature scheme*, Selected Areas in Cryptography (Lange Tanja, Kristin Lauter, and Petr Lisonek, eds.), LNCS, Springer, 2013.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham, *Aggregate and verifiably encrypted signatures from bilinear maps*, Advances in Cryptology – EUROCRYPT 2003 (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer, May 2003, pp. 416–432.

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin, *Sequential aggregate signatures with lazy verification from trapdoor permutations*, ASIACRYPT'12, Springer-Verlag, 2012, pp. 644–662.

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven, *Unrestricted aggregate signatures*, ICALP 2007: 34th International Colloquium on Automata, Languages and Programming(Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, eds.), Lecture Notes in Computer Science, vol. 4596, Springer, July 2007, pp. 411–422.

[BR93] Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, ACM CCS 93: 1st Conference on Computer and Communications Security(V. Ashby, ed.), ACM Press, November 1993, pp. 62–73.

[BR96] ———, *The exact security of digital signatures: How to sign with RSA and Rabin*, Advances in Cryptology – EUROCRYPT'96 (Ueli M. Maurer, ed.), Lecture Notes in Computer Science, vol. 1070, Springer, May 1996, pp. 399–416.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, 52nd Annual Symposium on Foundations of Computer Science(Rafail Ostrovsky, ed.), IEEE Computer Society Press, October 2011, pp. 97–106.

[BY96] Mihir Bellare and Moti Yung, *Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation*, Journal of Cryptology **9** (1996), no. 3, 149–166.

[BZ13] Dan Boneh and Mark Zhandry, *Secure signatures and chosen ciphertext security in a quantum computing world*, Advances in Cryptology CRYPTO 2013 (Ran Canetti and JuanA. Garay, eds.), LNCS, vol. 8043, Springer, 2013, pp. 361–379.

[CN11] Yuanmi Chen and Phong Q. Nguyen, *BKZ 2.0: Better lattice security estimates*, Advances in Cryptology – ASIACRYPT 2011 (Dong Hoon Lee and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 7073, Springer, December 2011, pp. 1–20.

[Cor00] Jean-Sébastien Coron, *On the exact security of full domain hash*, Advances in Cryptology – CRYPTO 2000 (Mihir Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer, August 2000, pp. 229–235.

[DDLL13] Lo Ducas, Alain Durmus, Tancrde Lepoint, and Vadim Lyubashevsky, *Lattice signatures and bimodal gaussians*, CRYPTO 2013 (Ran Canetti and JuanA. Garay, eds.), LNCS, vol. 8042, Springer Berlin Heidelberg, 2013, pp. 40–56.

[DN12] Léo Ducas and PhongQ. Nguyen, *Learning a zonotope and more: Cryptanalysis of ntrusign countermeasures*, ASIACRYPT 2012 (Xiaoyun Wang and Kazue Sako, eds.), LNCS, vol. 7658, Springer, 2012, pp. 433–450.

[EFG+10] Oliver Eikemeier, Marc Fischlin, Jens-Fabian Götzmann, Anja Lehmann, Dominique Schröder, Peter Schröder, and Daniel Wagner, *History-free aggregate message authentication codes*, SCN 10: 7th International Conference on Security in Communication Networks(Juan A. Garay and Roberto De Prisco, eds.), Lecture Notes in Computer Science, vol. 6280, Springer, September 2010, pp. 309–328.

[Gen09] Craig Gentry, *Fully homomorphic encryption using ideal lattices*, 41st Annual ACM Symposium on Theory of Computing(Michael Mitzenmacher, ed.), ACM Press, May / June 2009, pp. 169–178.

[GG91] Allen Gersho and Robert M. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi, *Public-key cryptosystems from lattice reduction problems*, Advances in Cryptology – CRYPTO'97 (Burton S. Kaliski Jr., ed.), Lecture Notes in Computer Science, vol. 1294, Springer, August 1997, pp. 112–131.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi, *Candidate multilinear maps from ideal lattices*, EUROCRYPT 2013 (Thomas Johansson and PhongQ. Nguyen, eds.), LNCS, vol. 7881, Springer, 2013, pp. 1–17.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters, *Attribute-based encryption for circuits from multilinear maps*, CRYPTO 2013 (Ran Canetti and JuanA. Garay, eds.), LNCS, vol. 8043, Springer, 2013, pp. 479–499.

[GJSS01] Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo, *Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001*, Advances in Cryptology – ASIACRYPT 2001 (Colin Boyd, ed.), Lecture Notes in Computer Science, vol. 2248, Springer, December 2001, pp. 1–20.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann, *Practical lattice-based cryptography: A signature scheme for embedded systems*, Cryptographic Hardware and Embedded Systems – CHES 2012 (Emmanuel Prouff and Patrick Schaumont, eds.), Lecture Notes in Computer Science, vol. 7428, Springer, September 2012, pp. 530–547.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, 40th Annual ACM Symposium on Theory of Computing(Richard E. Ladner and Cynthia Dwork, eds.), ACM Press, May 2008, pp. 197–206.

[Gra84] R.M. Gray, *Vector quantization*, IEEE ASSP Mag., 1984, pp. 4–29.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters, *Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based*, CRYPTO 2013 (Ran Canetti and JuanA. Garay, eds.), LNCSe, vol. 8042, Springer, 2013, pp. 75–92.

[HL93] Amir Herzberg and Michael Luby, *Pubic randomness in cryptography*, Advances in Cryptology – CRYPTO'92 (Ernest F. Brickell, ed.), Lecture Notes in Computer Science, vol. 740, Springer, August 1993, pp. 421–432.

[HNHGSW03] Jeffrey Hoffstein, Jill Pipher Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte, *NTRUSIGN: Digital signatures using the NTRU lattice*, Topics in Cryptology – CT-RSA 2003 (Marc Joye, ed.), Lecture Notes in Computer Science, vol. 2612, Springer, April 2003, pp. 122–140.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and JosephH. Silverman, *Ntru: A ring-based public key cryptosystem*, Algorithmic Number Theory, LNCS, vol. 1423, Springer Berlin Heidelberg, 1998, pp. 267–288.

[HSW$^+$00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, *System architecture directions for networked sensors*, SIGPLAN Not. **35** (2000), no. 11, 93–104.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters, *Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures*, CRYPTO 2013 (Ran Canetti and JuanA. Garay, eds.), LNCS, vol. 8042, Springer, 2013, pp. 494–512.

[LM08] Vadim Lyubashevsky and Daniele Micciancio, *Asymptotically efficient lattice-based digital signatures*, TCC 2008: 5th Theory of Cryptography Conference(Ran Canetti, ed.), Lecture Notes in Computer Science, vol. 4948, Springer, March 2008, pp. 37–54.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham, *Sequential aggregate signatures from trapdoor permutations*, Advances in Cryptology – EUROCRYPT 2004 (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science, vol. 3027, Springer, May 2004, pp. 74–90.

[LP11] Richard Lindner and Chris Peikert, *Better key sizes (and attacks) for lwe-based encryption*, CT-RSA, LNCS, Springer, 2011, pp. 319–339.

[Lyn99] Charles Lynn, *Secure border gateway protocol (s-bgp)*, ISOC Network and Distributed System Security Symposium – NDSS'99, The Internet Society, February 1999.

[Lyu08] Vadim Lyubashevsky, *Towards practical lattice-based cryptography*, 2008.

[Lyu09] ———, *Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures*, Advances in Cryptology – ASIACRYPT 2009 (Mitsuru Matsui, ed.), Lecture Notes in Computer Science, vol. 5912, Springer, December 2009, pp. 598–616.

[Lyu12] ———, *Lattice signatures without trapdoors*, Advances in Cryptology – EUROCRYPT 2012 (David Pointcheval and Thomas Johansson, eds.), Lecture Notes in Computer Science, vol. 7237, Springer, April 2012, pp. 738–755.

[MP12] Daniele Micciancio and Chris Peikert, *Trapdoors for lattices: Simpler, tighter, faster, smaller*, Advances in Cryptology – EUROCRYPT 2012 (David Pointcheval and Thomas Johansson, eds.), Lecture Notes in Computer Science, vol. 7237, Springer, April 2012, pp. 700–718.

[MR04] Daniele Micciancio and Oded Regev, *Worst-case to average-case reductions based on Gaussian measures*, 45th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, October 2004, pp. 372–381.

[Nev08] Gregory Neven, *Efficient sequential aggregate signed data*, Advances in Cryptology – EUROCRYPT 2008 (Nigel P. Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer, April 2008, pp. 52–69.

[NR06] Phong Q. Nguyen and Oded Regev, *Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures*, Advances in Cryptology – EUROCRYPT 2006 (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 4004, Springer, May / June 2006, pp. 271–288.

[NR09] ———, *Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures*, Journal of Cryptology **22** (2009), no. 2, 139–160.

[Pei07] Chris Peikert, *Limits on the hardness of lattice problems in lp norms*, In IEEE Conference on Computational Complexity, 2007, pp. 333–346.

[Pei10] ______, *An efficient and parallel gaussian sampler for lattices*, Advances in Cryptology – CRYPTO 2010 (Tal Rabin, ed.), Lecture Notes in Computer Science, vol. 6223, Springer, August 2010, pp. 80–97.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters, *A framework for efficient and composable oblivious transfer*, Advances in Cryptology – CRYPTO 2008 (David Wagner, ed.), Lecture Notes in Computer Science, vol. 5157, Springer, August 2008, pp. 554–571.

[Rüc10] Markus Rückert, *Lattice-based blind signatures*, Advances in Cryptology – ASIACRYPT 2010 (Masayuki Abe, ed.), Lecture Notes in Computer Science, vol. 6477, Springer, December 2010, pp. 413–430.

[Sho97] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing **26** (1997), no. 5, 1484–1509.

[SS11] Damien Stehl and Ron Steinfeld, *Making ntru as secure as worst-case problems over ideal lattices*, Advances in Cryptology EUROCRYPT 2011, LNCS, vol. 6632, Springer, 2011, pp. 27–47.

A Appendix

A.1 Trapdoor functions and the full-domain hash scheme

In the following, we recall some basic definitions and properties of trapdoor functions [GPV08], that are required in our security proof. Later, we will particularly focus on collision-resistant preimage sampleable trapdoor functions (PSTF) that allow any signer knowing the trapdoor to create signatures in full domain hash schemes such as the GPV signature scheme. According to [GPV08,AP09,Pei10,MP12] there exists a polynomial-time algorithm $\mathsf{TrapGen}$ that on input the security parameter 1^n outputs public key $\mathbf{A}$ and the corresponding trapdoor $\mathbf{T}$ such that the trapdoor function $f_{\mathbf{A}} : B_n \longrightarrow R_n$ can efficiently be evaluated and satisfies the following properties:

1. The output distribution of $f_{\mathbf{A}}(x)$ is uniform at random over R_n given x is sampled from the domain B_n according to $\mathsf{SampleDom}(1^n)$, e.g. $\mathcal{D}_{\mathbb{Z}^n,s}$ with $s = \omega(\sqrt{\log n})$ by [GPV08, Lemma 5.2].
2. Anyone knowing the trapdoor can efficiently sample preimages $x \longleftarrow \mathsf{SamplePre}(\mathbf{T}, y)$ for a given syndrome $y \in R_n$ such that $f_{\mathbf{A}}(x) = y$, where x is distributed as $\mathsf{SampleDom}(1^n)$. By the one-way property the probability to find a preimage $x \in f_{\mathbf{A}}^{-1}(y) \subseteq B_n$ of a uniform syndrome $y \in R_n$ without the knowledge of the trapdoor is negligible.
3. The conditional min-entropy property of $\mathsf{SampleDom}(1^n)$ for a given syndrome $y \in R_n$ implies that two preimages x', x distributed as $\mathsf{SampleDom}(1^n)$ differ with overhelming property. This is due to the large conditional min-entropy of at least $\omega(\log n)$.
4. The preimage sampleable trapdoor functions are collision resistant, meaning that it is infeasible to find a collision $f_{\mathbf{A}}(x_1) = f_{\mathbf{A}}(x_2)$ such that $x_1, x_2 \in B_n$ and $x_1 \neq x_2$.

Due to the results of [Ajt99,GPV08] a lot of research has been made on the construction of preimage sampleable trapdoor functions in recent years resulting in a sequence of improving works [GPV08,AP09], [Pei10,MP12]. These constructions often satisfy these properties only statistically, meaning that the statistical distance between the claimed distributions and the provided ones are negligible. As a result the security proofs of cryptographic schemes involving concrete constructions hold only statistically, which is quite enough for practice. One cryptograhic scheme is the GPV-signature scheme, which is secure in the random oracle model and exploits the properties of collision-resistant trapdoor functions. Furthermore, it is stateful, meaning that it does not generate new signatures for messages already signed. This can be attributed to the fact that a potential attacker could otherwise use two signatures of the same message in order to construct an element of the kernel, which solves $SIS_{q,n,\beta}$ and hence allows the attacker to provide a second preimage of any message. One can remove the need for storing message and signature pairs by employing the probabilistic approach [GPV08], where the signer samples an extra random seed, which is appended to the message when calling $H(\cdot)$.

The GPV signature scheme consists mainly of sampling a preimage from a hash function endowed with a trapdoor. It involves the following 3 algorithms:

$\mathsf{KeyGenGPV}(1^n)$ On input 1^n the algorithm $\mathsf{TrapGen}(1^n)$ outputs a key pair $(\mathbf{A}, \mathbf{T})$, where $\mathbf{T}$ is the secret key or trapdoor and $\mathbf{A}$ is the public key describing the preimage sampleable trapdoor function $f_{\mathbf{A}}$.

$\mathsf{SignGPV}(\mathbf{T}, m)$ The signing algorithm computes the hash value $H(m)$ of the message m and looks up $H(m)$ in its table, where $H(\cdot)$ is modeled as a random oracle. If it finds an entry, it outputs σ_m. Otherwise, it samples a preimage $\mathbf{z} \leftarrow \mathsf{SamplePre}(\mathbf{T}, H(m))$ of $H(m)$ and outputs $\mathbf{z}$ as the signature.

$\mathsf{VerifyGPV}(\mathbf{z}, m)$ The verification algorithm checks the satisfaction of $H(m) = f_{\mathbf{A}}(\mathbf{z})$ and $\mathbf{z} \in B_m$. If both conditions are valid it, outputs 1, otherwise 0.

Probabilistic Full-Domain Hash scheme

The probabilistic approach additionally requires the signer to generate a random seed r (e.g. $r \in \{0,1\}^n$) which is appended to the message m. Doing this, we can sign the same message m several times, since the r-part always differs except with negligible probability. Thus, we can consider $m||r$ as the extended message to be signed.

B Generic Lossless Compression of Schnorr-like Signatures

B.1 Proof of Theorem 3

Proof. We can always write a real number r as $r = x+t$ with $x \in \mathbb{Z}$ and $t \in [b_1, b_2)$ such that $b_2 - b_1 = 1$ and r can bijectively be mapped back to x and t. Intuitively, we fill the gap between two consecutive integers with reals modulo 1. Any integer x can now be transformed into its binary representation $(a_0, \ldots, a_m)$. Let $b_1 = -0.5 + c$ and $b_2 = 0.5 + c$, where $c = \frac{z}{2h} - \lceil \frac{z}{2h} \rfloor \in (-0.5, 0.5)$, then any element $r \in \frac{z}{2h} + [-0.5, 0.5]$ satisfies $\phi^{-1}(r) \in (a_0, \ldots, a_m) \times [b_1, b_2)$ with $(a_0, \ldots, a_m)$ being the binary representation of $\lceil \frac{z}{2h} \rfloor$, since $r \in \sum_{i=1}^{m} a_i 2^i + [b_1, b_2) = \lceil \frac{z}{2h} \rfloor + [c - 0.5, c + 0.5) = \frac{z}{2h} + [-0.5, 0.5]$. But indeed, we have also $\phi^{-1}(\frac{y}{2h}) \in (a_0, \ldots, a_m) \times [b_1, b_2)$. As a result, $(a_0, \ldots, a_m)$ is the same for all elements in that range. Therefore, the bit string $(a_0, \ldots, a_m)$ is called the public randomness induced by C and can be extracted by any party viewing C. Let $\mathcal{X}$ denote the distribution $\phi^{-1}(\mathcal{Z}/2h)$, where a vector $\phi^{-1}(\frac{y}{2h})$ sampled according to this distribution involves $y \leftarrow \mathcal{Z}$. We know that the probability is invariant with respect to bijective transformations and hence obtain $(a_0, \ldots, a_m)$ with probability

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ (\mathbf{x}, t) \in (a_0, \ldots, a_m) \times [b_1, b_2)\ \right] &= P_{\phi(\mathbf{x},t)\sim\mathcal{Z}/2h}\left[\ \phi(\mathbf{x}, t) \in \frac{z}{2h} + [-0.5, 0.5]\ \right] \\ &= P_{y\sim\mathcal{Z}}\left[\ y \in C\ \right] \text{ with } y = \phi(x) \cdot 2h\,. \end{aligned}$$

Note, that the support of $\mathcal{Z}$ can differ from $\mathbb{R}$. In fact, the proof works for any distribution over a subset of $\mathbb{R}$.

B.2 Proof of Theorem 4

Proof. From Lemma 3, we deduce that $\phi^{-1}(\frac{z}{2h} + [-0.5, 0.5]) = (a_0, \ldots, a_m) \times [b_1, b_2)$. Hence, the bit string $\mathbf{x} = (a_0, \ldots, a_m)$ occured with probability $P_{y_1\sim\mathcal{Z}}[\ y_1 \in C\ ]$. Suppose first, that $\mathcal{Z}$ is a discrete distribution and $\mathcal{X}$ denotes the distribution $\phi^{-1}(\mathcal{Z}/2h)$, where a vector $\phi^{-1}(\frac{y}{2h})$ sampled according to this distribution involves $y \leftarrow \mathcal{Z}$. Then, the term $t \in [b_1, b_2)$ is sampled according to the probability distribution

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ (\mathbf{x}, t) = (\mathbf{x}, t_1) \mid \mathbf{x} = (a_0, \ldots, a_m)\ \right] &= P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ t = t_1 \mid \mathbf{x} = (a_0, \ldots, a_m)\ \right] \\ &= P_{y\sim\mathcal{Z}}\left[\ y = y_2 \mid y \in C\ \right] \text{ with } y_2 = \phi(\mathbf{x}, t_1) \end{aligned}$$

Once having sampled t according to this probability distribution, we obtain a full realization $(\mathbf{x}, t)$ that is distributed as

$$\begin{aligned} P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ \mathbf{x} = (a_0, \ldots, a_m)\ \right] \cdot P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ t = t_1 \mid \mathbf{x} = (a_0, \ldots, a_m)\ \right] &= P_{(\mathbf{x},t)\sim\mathcal{X}}\left[\ (\mathbf{x}, t) = (a_0, \ldots, a_m, t_1)\ \right] \\ &= P_{y\sim\mathcal{Z}}\left[\ y = y_2\ \right]. \end{aligned}$$

In case, we consider a continuous distribution, we use the probability density function in the same way. □

B.3 Security Proof of Theorem 2

Proof. We prove security for the individual case $l = 1$. In order to prove that the scheme is as secure as standard individual signatures (e.g. standard GPV signatures, Lyubashevsky signatures etc.), we proceed via a sequence of games. The challenge compressed signature is given by $(\mathbf{z}_1^*, \mathbf{z}_1^* - \mathbf{z}_2^*)$, where $\mathbf{z}_1^*$ denotes the centroid for compression.

Game 0

The game $\mathbf{G_0}$ is exactly the attack with the original compression scheme. The challenger is given access to a signing oracle producing compressed signatures $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ in combination with the corresponding centroids $\mathbf{z}_1$ for compression. The centroids follow the same distribution $\mathcal{Z}$ as signatures. In addition, the challenger is given access to a random oracle H_0 and an oracle H_1, where H_0 is queried on messages of choice producing uniform random vectors. For a vector $\mathbf{c}$ distributed as $\mathcal{Z}$ as input, H_1 produces in accordance to the generic construction in Figure 4 a compressed vector $(\mathbf{c}, \mathbf{c} - \mathbf{x})$, where $\mathbf{x}$ is distributed as $\mathcal{Z}$ and the centroid is given by $\mathbf{c}$.

Game 1

In game $\mathbf{G_1}$, we change the way the signing oracle responds to signature requests and the challenge compressed signature $(\mathbf{z}_1^*, \mathbf{z}_1^* - \mathbf{z}_2^*)$ is produced, but in a way that it introduces only a $negl(n)$ statistical distance to $\mathbf{G_0}$. The signing oracle now outputs only uncompressed signatures (standard signatures). The signing oracle from $\mathbf{G_0}$, which generates compressed signatures together with the corresponding centroids, is now simulated as follows. The signing oracle is queried in order to obtain an uncompressed signature $\mathbf{z}_2$. Subsequently, H_1 is called on input $\mathbf{z}_2$, which then returns a compressed vector $(\mathbf{z}_2, \mathbf{z}_2 - \mathbf{z}_1)$ with $\mathbf{z}_2$ being its centroid. Finally, the compressed signature $(\mathbf{z}_1, \mathbf{z}_1 - \mathbf{z}_2)$ is output, where $\mathbf{z}_1$ acts as centroid. Since $\mathbf{z}_1$ and $\mathbf{z}_2$ are distributed according to $\mathcal{Z}$, the attacker can not distinguish between the games $\mathbf{G_0}$ and $\mathbf{G_1}$.

□

C Compression of GPV Signatures

C.1 Basic signature scheme

In what follows, we give a short description of the signature scheme [GPV08,MP12] when instantiated computationally, meaning that the matrix $\mathbf{A}$ is an instance of the LWE distribution and therefore pseudorandom when ignoring the identity submatrix. A computational instantiation of the trapdoor construction is preferred with regard to the GPV-signature scheme, since it allows to decrease the number of columns in the public key $\mathbf{A}$ as opposed to a statistical instantiation.

Construction 6

KeyGen(1^n): *Sample* $\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$, *and each entry of the secret key* $\mathbf{R} \in \mathbb{Z}^{2n \times n \cdot k}$ *from* $\mathcal{D}_{\mathbb{Z},\alpha q}$, *where* $q = 2^k$ *and* $\alpha q \geq 2\sqrt{n}$. *Output the signing key* $\mathbf{R}$, *the verification key* $\mathbf{A} = [\ \mathbf{I}_n \mid \bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$ *and parameter* s *such that* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *with* $m = 2n + n \cdot k$ *and* $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top \in \mathbb{Z}^{n \times n \cdot k}$ *is the primitive matrix consisting of* n *copies of the vector* $\mathbf{g}^\top = [1, 2, \ldots, 2^{k-1}]$.

Sign$(msg, \mathbf{R})$**:** *Compute the syndrome* $\mathbf{u} = H(msg)$*, sample* $\mathbf{p} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sqrt{\boldsymbol{\Sigma}_\mathbf{p}}}$ *with* $\sqrt{\boldsymbol{\Sigma}_\mathbf{p}} = \begin{bmatrix} \frac{\mathbf{R}}{\sqrt{b}} & \mathbf{L} \\ \sqrt{b}\mathbf{I_{nk}} & 0 \end{bmatrix}$.

Generating perturbations: [GPV08,MP12,BB13]

Sample $\mathbf{d}_1 \leftarrow D_1^{2n}, \mathbf{d}_2 \leftarrow D_1^{n \cdot k}$*, where* $\mathcal{D}_1$ *is the continuous Gaussian distribution with parameter 1.*

Compute $\tilde{\mathbf{p}} = (\tilde{\mathbf{p}}_\mathbf{1}, \tilde{\mathbf{p}}_2)$*, where* $\tilde{\mathbf{p}}_\mathbf{1} = \frac{1}{\sqrt{b}}\mathbf{R}\mathbf{d}_2 + \mathbf{L}\mathbf{d}_1$ *and* $\tilde{\mathbf{p}}_\mathbf{2} = \sqrt{b} \cdot \mathbf{d}_2$ *according to [BB13].*

Sample pertrubation $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}) = \tilde{\mathbf{p}} + \mathcal{D}_{\mathbb{Z}^{n \cdot (2+k)} - \tilde{\mathbf{p}}, a}$.

$(\eta_\epsilon(\mathbb{Z}) \leq a = \sqrt{\ln\left(2n\left(1 + \frac{1}{\epsilon}\right)\right)/\pi},\ \mathbf{p}_1 = \tilde{\mathbf{p}}_\mathbf{1} + \mathcal{D}_{\mathbb{Z}^{2n} - \tilde{\mathbf{p}}_\mathbf{1}, a},\ \mathbf{p}_2 = \tilde{\mathbf{p}}_\mathbf{2} + \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \tilde{\mathbf{p}}_\mathbf{2}, a})$

Signing:

Determine the adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A}\mathbf{p} \in \mathbb{Z}^n$*. Sample vector* $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp_\mathbf{v}(\mathbf{G}), r}$ *with* $r = 2a$.

Output signature $\mathbf{z} \leftarrow \mathbf{p} + \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{x}$.

Verify$(msg, \mathbf{z}, (H, \mathbf{A}))$**:** *Check whether* $\mathbf{A} \cdot \mathbf{z} \equiv H(msg)$ *and* $\|\mathbf{z}\|_2 \leq s\sqrt{m}$ *is satisfied. If so, output* 1 *(accept), otherwise* 0 *(reject).*

Fo the purpose of sampling $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp_\mathbf{v}(\mathbf{G}), r}$ in the signing step, the authors provide an efficient and simple algorithm (see Algorithm 5), which is derived from the randomized nearest plane algorithm.

Algorithm 5

The algorithm initializes a_0 with an entry $v_i \in \mathbb{Z}_q$ of the adjusted syndrome $\mathbf{v}$ and outputs a vector $\mathbf{b} \in \Lambda^\perp_{v_i}(\mathbf{g}^\top)$ distributed as $\mathcal{D}_{\Lambda^\perp_{v_i}(\mathbf{g}^\top), r}$.

for $i = 0, \ldots, k-1$ do:

1. $b_i \leftarrow \mathcal{D}_{2\mathbb{Z} + a_i, r}$
2. $a_{i+1} = (a_i - b_i)/2$

Output: $\mathbf{b} = (b_0, \ldots, b_{k-1})^\top$

When applying this algorithm sucessively on each component of $\mathbf{v}$ and concatenating the resulting vectors, we obtain a sample $\mathbf{x}$ distributed as $\mathcal{D}_{\Lambda^\perp_\mathbf{v}(\mathbf{G}), r}$.

C.2 Proof of Lemma 6

Proof. One can easily verify that $\mathcal{D}_{\Lambda_i, \mathbf{c}_i, s_i}$ and $s_i \cdot \mathcal{D}_{\Lambda'_i, \mathbf{c}'_i, 1}$ define the same distribution, where Λ'_i and $\mathbf{c}'_i$ denote the scaled lattice Λ_i/s_i and center $\mathbf{c}_i/s_i$ respectively. In the rest of the proof, we will use this equivalence when considering the distribution on the lattice Λ_i. The cartesian product $\mathcal{L} = \Lambda_1/s_1 \times \cdots \times \Lambda_k/s_k$ of lattices is again a $(\sum_i n_i)$-dimensional lattice since we can always construct basis vectors for $\mathcal{L}$ using the basis vectors of Λ_i. For any countable set $A = A_1 \times \cdots \times A_k \subset \mathcal{L}$ the probability measure on it is defined by $\rho_{(\mathbf{c}'_1, \ldots, \mathbf{c}'_k)}(A) = \prod_i \rho_{c'_i}(A_i)$. Let $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_k})$ define the vector composed by k subvectors $\mathbf{x_i} \in \mathbb{R}^{n_i}$ and $\mathbf{c}' = (\mathbf{c}'_1, \ldots, \mathbf{c}'_k)$ respectively. Then we obtain the following equalities:

$$
\begin{aligned}
&\langle \mathbf{x}, (\mathcal{D}_{\Lambda_1, \mathbf{c}_1, s_1} - \mathbf{c}_1, \ldots, \mathcal{D}_{\Lambda_k, \mathbf{c}_k, s_k} - \mathbf{c}_k) \rangle && (4) \\
&= \langle \mathbf{x_1}, \mathcal{D}_{\Lambda_1, c_1, s_1} - \mathbf{c}_1 \rangle + \cdots + \langle \mathbf{x_k}, \mathcal{D}_{\Lambda_k, \mathbf{c}_k, s_k} - \mathbf{c}_k \rangle && (5) \\
&= \langle \mathbf{x_1}, s_1 \cdot (\mathcal{D}_{\Lambda'_1, \mathbf{c}'_1, 1} - \mathbf{c}'_1) \rangle + \cdots + \langle \mathbf{x_k}, s_k \cdot (\mathcal{D}_{\Lambda'_k, \mathbf{c}'_k, 1} - \mathbf{c}'_k) \rangle && (6) \\
&= \langle s_1 \cdot \mathbf{x_1}, \mathcal{D}_{\Lambda'_1, \mathbf{c}'_1, 1} - \mathbf{c}'_1 \rangle + \cdots + \langle s_k \cdot \mathbf{x_k}, \mathcal{D}_{\Lambda'_k, \mathbf{c}'_k, 1} - \mathbf{c}'_k \rangle && (7) \\
&= \langle (s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k}), (\mathcal{D}_{\Lambda'_1, \mathbf{c}'_1, 1} - \mathbf{c}'_1, \ldots, \mathcal{D}_{\Lambda'_k, \mathbf{c}'_k, 1} - \mathbf{c}'_k) \rangle && (8) \\
&= \langle (s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k}), \mathcal{D}_{\mathcal{L}, \mathbf{c}', 1} - \mathbf{c}' \rangle\,. && (9)
\end{aligned}
$$

The claim now follows from equation 9 and [Pei07, Lemma 5.1] with unit vector $\frac{(s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k})}{\|(s_1 \cdot \mathbf{x_1}, \ldots, s_k \cdot \mathbf{x_k})\|}$ $\square$

If we set $T \approx 4.69$ the probability of that inequality to hold is less than 2^{-100}.

C.3 Proof of Lemma 5

Proof. The probability densitity function of a sample distributed according to $\mathcal{D}_1$ is $f(x) = e^{-\pi x^2}$. Using conditional probability rules we have

$$P[B_i] \cdot f(x \mid x \in B_i) = P[B_i] \cdot \frac{1}{P[B_i]} e^{-\pi x^2} = e^{-\pi x^2} \text{ for } x \in B_i,\ P[B_i] = \int_{B_i} e^{-\pi x^2} dx,$$

which exactly coincides with the probability density function of a continuous Gaussian with parameter 1. $\square$

C.4 Proof of Theorem 3

Proof. Consider the subvector $\mathbf{z}^{(2)} = f_{\mathbf{I}}(\mathbf{x}, \mathbf{y}) + \mathbf{y}^{(2)} \in \mathbb{Z}^{n \cdot k}$ consisting of the last $n \cdot k$ entries of $\mathbf{z} \in \mathbb{Z}^{n(k+2)}$, that is generated according to the basic signature scheme from Appendix C.1. As stated in [Pei10] the subvector $\mathbf{z}^{(2)}$ can be written as $\mathbf{z}^{(2)} = \mathbf{x} + \lceil \mathbf{c} \rfloor_a = \mathbf{x} + \mathbf{c} + \mathcal{D}_{\mathbb{Z}^{n \cdot k} - \mathbf{c}, a}$, where $\lceil \cdot \rfloor_a$ denotes the randomized rounding operation and $\mathbf{c} = \sqrt{s^2 - 5a^2} \cdot \mathbf{d},\ \mathbf{d} \xleftarrow{\$} D_1^{n \cdot k}$ with a as above [BB13]. By [MP12] the parameter s can be as small as $\sqrt{\mathbf{s}_1(R)^2 + 1} \cdot \sqrt{6} \cdot a \gtrsim \mathbf{s}_1(R) \cdot \sqrt{6} \cdot a$ and when applying [MP12, Lemma 2.9], one sets $\mathbf{s}_1(R)$ to at least $1/\sqrt{2\pi} \cdot (\sqrt{2n} + \sqrt{nk}) \cdot \alpha q$.

Using $b = s^2 - 5a^2$ [BB13], one can deduce a range C from which the continuous Gaussian vector used in $\mathbf{z}^{(2)}$ originates. Thereto, we have to compute $h = \max \|f_{\mathbf{I}}(\mathbf{c})\|_\infty$ providing a bound to the sum $x_i + \mathcal{D}_{\mathbb{Z} - c_i, a}$. One notices that $\mathcal{D}_{\mathbb{Z} - c_i, a}$ and $-c_i + \mathcal{D}_{\mathbb{Z}, c_i, a}$ are identically distributed. As per Lemma 6 the sum is in the range $[-4.7 \cdot \sqrt{r^2 + a^2}, 4.7 \cdot \sqrt{r^2 + a^2}]$, except with negligible probability. As a result, it is possible to determine a concrete range for the continuous Gaussian vector $\mathbf{d}$ by employing only public data. In fact, we have

$$\mathbf{d} \in C = \tfrac{\mathbf{z}^{(2)}}{\sqrt{b}} + \left[-\tfrac{h}{\sqrt{b}}, \tfrac{h}{\sqrt{b}}\right]^{nk} \text{ with } h = 4.7 \cdot \sqrt{5}a\,.$$

The set C is publicly accessible and can, thus, be read by all parties. A complete secretly sampled continuous Gaussian $\mathbf{d}$ implies $C = \mathbb{R}^{nk}$, whereas $C = \mathbf{d}$ in case $\mathbf{d}$ is completely accessible to the public.

On the one hand, one observes that public randomness induced by the set C can be viewed and exploited by a potential adversary (and anyone else) in order to launch an attack against the underlying cryptographic primitive. Consequently, the security of any cryptosystem should only be based on secretly sampled random strings that are not visible to the public. In fact, we prove in Appendix B.3 that compressed signatures, that employ public randomness, are indistinguishable from standard signatures. On the other hand, arbitrary many other signers can take advantage of the available public randomness utilizing it for building own signatures. Since each signer operates with its own secret key, that is independently generated, exploiting public randomness has no impact on security (see Appendix B.3). On the contrary, the generation of public random strings can be delegated to other institutions providing the desired distributions on demand. Specifically, in Section 4.1 we highlight the usage of a short random seed $\mathbf{r}$ serving as input to a discrete Gaussian sampler acting as a simulator for signatures. The output vector is used in order to extract the required public randomness and more importantly to replace the large centroid $\mathbf{z}^{(2)}$. As a result, it suffices to store the seed instead of the large centroid.

The continuous Gaussian $\mathbf{d}$ was independently sampled and lies in C with probability $P[\ C\ ]$ (see Lemma 4). We say C occured, if $\mathbf{d} \in C$. Any other signer S_i can now secretly sample a continuous Gaussian

$\mathbf{d}_i \xleftarrow{\$} \mathcal{D}_1^{nk}$ conditioned on $\mathbf{d}_i \in C$ according to the probability density function $f(\ \mathbf{x} \mid \mathbf{x} \in C\ )$. Reusing public randomness causes the random vectors $\mathbf{d}_i$ to be distributed following the probability density function $f(\ \mathbf{x} \mid \mathbf{x} \in C\ ) \cdot P[\ C\ ] = e^{-\pi \|\mathbf{x}\|_2^2}$, which perfectly coincides with the required distribution $\mathbf{d}_i \xleftarrow{\$} \mathcal{D}_1^{nk}$. Following this approach, each signer S_i needs only to secretly generate its own continuous Gaussian vector $\mathbf{d}_i$ by sampling from the provided range C, for example with rejection sampling, such that $\mathbf{d}_i \sim \mathcal{D}_1^{nk}$ is satisfied. Intuitively, this strategy causes the vectors $\mathbf{z}_i^{(2)}$ to be distributed around the centroid $\mathbf{z}^{(2)}$ (see Figure 1). Per construction we have $\mathbf{d}_i \in C$ for all $1 \leq i \leq l$. Applying Lemma 6, we are capable of bounding the infinity norm on $\mathbf{z}^{(2)} - \mathbf{z}_i^{(2)}$, where $\mathbf{x} \xleftarrow{\$} \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),r}$ and $\mathbf{v}_i$ behave as described in the signing algorithm (see Appendix C.1)

$$\begin{aligned}
\|\ \mathbf{z}^{(2)} - \mathbf{z}_i^{(2)}\ \|_\infty &= \|\ \sqrt{b} \cdot \tfrac{\mathbf{z}^{(2)}}{\sqrt{b}} - \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),r} - \sqrt{b} \cdot \mathbf{d}_i - \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \sqrt{b}\cdot \mathbf{d}_i, a}\ \|_\infty \\
&\leq \|\ \sqrt{b} \cdot (\tfrac{\mathbf{z}^{(2)}}{\sqrt{b}} - \mathbf{d}_i)\ \|_\infty + \|\ \mathcal{D}_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),r} - \mathcal{D}_{\mathbb{Z}^{n\cdot k} - \sqrt{b}\cdot \mathbf{d}_i, a}\ \|_\infty \\
&\leq 2 \cdot 4.7 \cdot \sqrt{5}a < 128\,.
\end{aligned}$$

Each entry of $\mathbf{z}^{(2)} - \mathbf{z}_i^{(2)}$ occupies for $n \leq 2^{70}$ at most 7 bits of memory, except with negligible probability. This value is almost independent of n, which increases the incentive to use higher security parameters and thus causing larger compression factors. On the other hand, a signature is distributed according to a discrete Gaussian with parameter s. Each entry has magnitude of at most $4.7 \cdot s$ except with probability of at most 2^{-100}. □

C.5 Conditional Rejection Sampling

In this section we briefly discuss how to perform the rejection sampling step based on conditional probabilities. Specifically, we want to sample a vector $\mathbf{d}$ from $C = \frac{\mathbf{z}}{\sqrt{b}} + [-\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}, \frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}]^{nk}$ according to probability density function $f(\mathbf{x} \mid \mathbf{x} \in C) = e^{-\pi\|\mathbf{x}\|_2^2}/P[\ C\ ] = \prod_{i=1}^{nk} e^{-\pi x_i^2}/P[\ C_i\ ]$ with $C_i = \frac{z_i}{\sqrt{b}} + [-\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}, \frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}]$ and

$$P[C_i] = \int_{C_i} e^{-\pi x^2} dx\,.$$

By means of a simple rejection sampling algorithm, we can sample each entry of $\mathbf{d}$ independently from $C_i = \frac{z_i}{\sqrt{b}} + [-\frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}, \frac{4.7\cdot\sqrt{5}a}{\sqrt{b}}]$. For example, one samples a uniform random element d_i from C_i and a second random element u_i from the interval $[\ 0, 1\ ]$. We accept d_i, if $u_i < e^{-\pi d_i^2}$, otherwise we reject and resample. Due to the compact intervals of small width, the rejection sampling algorithm is very fast.

D Compression of Lyubashevsky Signatures

D.1 Basic Signature Scheme

Let $\mathcal{R}_k^{q^n}$ denote a polynomial of degree $n-1$ with coefficients from $[-k, k]$. Furthermore, let the range $D_{32}^n = \{\mathbf{c} \,:\, \mathbf{c} \in \{-1,0,1\}^n,\ \|\mathbf{c}\|_1 \le 32\}$ of the cryptographic hash function $H(\cdot)$ consist of all polynomials with at most 32 non-zero coefficients. The scheme

Signature Scheme [GLP12]	
Signing	**Verification**
Given : $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} \mathcal{R}_1^{q^n}$ $\mathbf{a} \xleftarrow{\$} \mathcal{R}^{q^n}$, $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$ Hash function $H : \{0,1\}^* \to D_{32}^n$	**Given** : $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} \mathcal{R}_1^{q^n}$ $\mathbf{a} \xleftarrow{\$} \mathcal{R}^{q^n}$, $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$ Hash function $H : \{0,1\}^* \to D_{32}^n$
$\mathsf{Sign}(\mu, \mathbf{a}, \mathbf{s}_1, \mathbf{s}_2)$ $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k^{q^n}$ $\mathbf{c} = H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$ $\mathbf{z}_1 = \mathbf{s}_1\mathbf{c} + \mathbf{y}_1,\ \mathbf{z}_2 = \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$ if $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-32}^{q^n}$, then restart. **Output** : $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$	$\mathsf{Verify}(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$ accept iff: $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-32}^{q^n}$ $\mathbf{c} = H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_1 - \mathbf{t}\mathbf{c}, \mu)$

Table 4. Signature Scheme [GLP12]

When transforming the scheme from above into an aggregate signature scheme using the generic constructions from Section 6, we observe, that the rejection sampling step in the last line of the signature scheme is completely omitted by all other participants, in case the set C used for extracting public randomness lies in $\mathcal{R}_{k-64}^{q^n}$. That is, the other participants sample polynomials $\mathbf{y}_1$ and $\mathbf{y}_2$ from C, which immediately yield $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{R}_{k-32}^{q^n}$. In this case, the protocol requires only one iteration for each participant without restarting, because each participant uses the same set C. This obviously does not hold, when each signer samples his polynomials independently from $\mathcal{R}_k^{q^n}$.

E Application Scenario - Cluster-based Aggregation in Wireless Sensor Networks

A very interesting application scenario for aggregate signature schemes are wireless sensor networks. In recent years, many research efforts have been spent on minimizing data transmissions within WSNs due to the resource constrained devices forming the topology. At the same time there is a claim for securing the communication flow against adversaries that could attack the network to gather information or to manipulate them. Therefore, a lot of theoretical research has been made on secure data aggregation protocols, which aim at providing a certain level of security as well as mechanisms that improve the lifetime of sensor networks by minimizing the number of transmitted messages. It is a well-known fact [HSW+00] that the transmission of a single bit consumes as much battery power as executing 800-1000 instructions. So it is more convenient to reduce the number bits to be sent at the cost of an increased computation complexity.

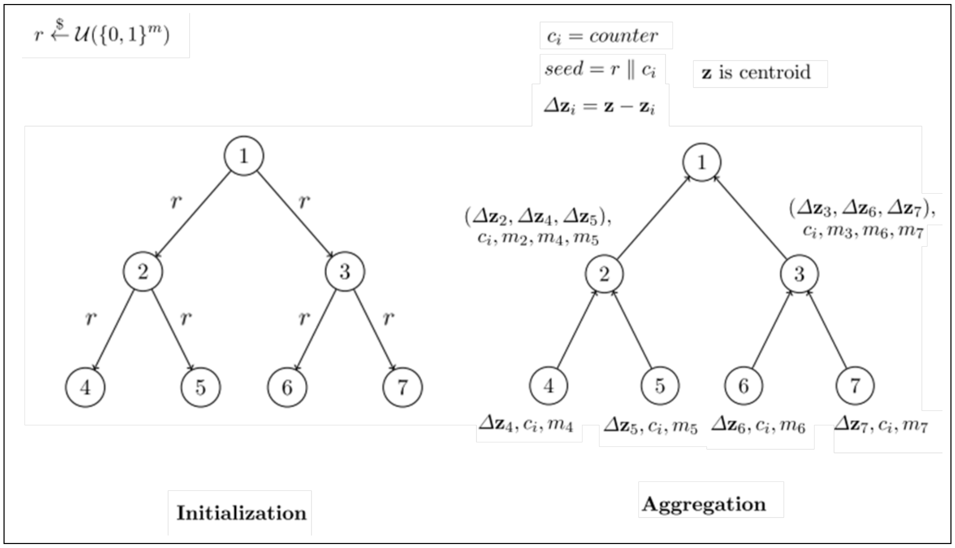

Fig. 9. Aggregate signatures in cluster based WSNs.

A cluster-based sensor network is an appropriate topology to support most of the aggregation schemes. Following this approach, one splits the network into clusters, each of similar size. Typically, one finds such topologies at Logistic Service Providers that monitor the supplied goods on their way from the manufacturer to the client. The products have different features (e.g. the temperature and humidity) that are of interest and thus need to be monitored for the whole delivery period. For instance, the temperature of perishables should remain constant on a reasonable level. Therefore, one should always keep an eye on the temperature preventing the spoiling of goods. A plausible way to design the topology for goods transported via trucks is to let each truck form a cluster with a small number of sensor nodes. Each cluster is characterized by its cluster head (CH), a dedicated node that acts as a subentity of an aggregator node (similar to a cluster head) or the root. At the top of the topology we have the root that sends its requests to the cluster heads, which in turn forward the request to the cluster participants. After sensing the required values, the cluster participants send them via the cluster heads back to the base station. There are many different ways to implement the signature aggregation schemes from above. For the sake of simplicity, we consider the generic aggregate signature scheme from Section 6. Therefore, the base station sends a fresh random salt securely to all nodes within the WSN in the setup phase. Whenever the nodes sense the required values they increase an internal counter which serves together with the random salt as the input seed to a discrete Gaussian sampler. Afterwards, they transmit their compressed signatures via the cluster heads back to the base station (see Figure 9). The usage of a salt in combination with a counter for creating signatures furtherly allows to capture Replay attacks, that can be launched by any adversary eavesdropping the message stream. One can furtherly allow only privileged members to be capable of verifying signatures. This can be achieved by making the random salt a secret key shared among the privileged members. By doing this, only those who share the secret key can recover the centroid and subsequently verify the uncompressed signature.