Towards Sound Fresh Re-Keying with Hard (Physical) Learning Problems

Stefan Dziembowski[1], Sebastian Faust[2], Gottfried Herold[2], Anthony Journault[3], Daniel Masny[2], François-Xavier Standaert[3].

[1] University of Warsaw, Institute of Informatics, Poland.
[2] University of Bochum, Fakultät für Mathematik, Germany.
[3] Université catholique de Louvain, ICTEAM – Crypto Group, Belgium.

Abstract. Most leakage-resilient cryptographic constructions aim at limiting the information adversaries can obtain about secret keys. In the case of asymmetric algorithms, this is usually obtained by secret sharing (aka masking) the key, which is made easy by their algebraic properties. In the case of symmetric algorithms, it is rather key evolution that is exploited. While more efficient, the scope of this second solution is limited to stateful primitives that easily allow for key evolution such as stream ciphers. Unfortunately, it seems generally hard to avoid the need of (at least one) execution of a stateless primitive, both for encryption and authentication protocols. As a result, fresh re-keying has emerged as an alternative solution, in which a block cipher that is hard to protect against side-channel attacks is re-keyed with a stateless function that is easy to mask. While previous proposals in this direction were all based on heuristic arguments, we propose two new constructions that, for the first time, allow a more formal treatment of fresh re-keying. More precisely, we reduce the security of our re-keying schemes to two building blocks that can be of independent interest. The first one is an assumption of Learning Parity with Leakage, which leverages the noise that is available in side-channel measurements. The second one is based on the Learning With Rounding assumption, which can be seen as an alternative solution for low-noise implementations. Both constructions are efficient and easy to mask, since they are key homomorphic or almost key homomorphic.

1 Introduction

Side-channel attacks are an important concern for the security of cryptographic implementations. Since their apparition in the late 1990s, a large body of work has investigated solutions to prevent them efficiently, e.g. based on algorithmic and protocol ingredients. Masking (i.e., data randomization) and shuffling (i.e., operation randomization) are well studied representatives of the first category [41]. Leakage-resilient cryptography [29] is a popular representative of the second one. Interestingly, it has been shown recently that these approaches are

complementary. Namely, leakage-resilient cryptography brings strong (concrete) security guarantees for stateful primitives such as stream ciphers (where key evolution prevents attacks taking advantage of multiple leakages per key). However, these stateful primitives generally require to be initialized with some fresh data, for example new session keys [52].[1] In practice, this initialization typically involves a stateless primitive such as a Pseudo Random Function (PRF), for which leakage-resilience is significantly less effective, since nothing prevents the adversary to repeat measurements for the same plaintext and key in this case [10]. Hence, the state-of-the-art in leakage-resilient symmetric cryptography is torn between two contradicting observations. On the one hand, leakage-resilient PRFs (and encryption schemes) such as [57, 24, 58, 1] cannot be used for this initialization.[2] On the other hand, it seems that the execution of at least one stateless primitive (e.g., a PRF or a block cipher) is strictly needed for the deployment of leakage-resilient (symmetric) encryption and MACs [51]. This leaves the efficient protection of such stateless primitives with algorithmic countermeasures such as masking and shuffling as an important research goal.

In particular, masking appears as a promising solutions for this purpose, since it benefits from a good theoretical understanding [21, 37, 53, 26, 27]. Unfortunately, the secure masking of a block cipher like the AES also comes with significant drawbacks, especially when the number of shares increases (i.e., for so called higher-order masking schemes). First, it implies implementation overheads that are quadratic in the number of shares [33] (although some optimizations are possible for low number of shares, especially in hardware, e.g. [12]). Second, it has large randomness requirements (since the masked execution of non-linear operations at high-orders requires frequent refreshings of the shares). Third and probably most importantly, it assumes that the leakages of all these shares are independent, a condition that is frequently contradicted both in software (because of transition-based leakages [22, 6]) and hardware implementations (because of so-called glitches [42, 43]. Besides, standard algorithmic countermeasures able to deal with such independence issues usually imply additional implementation constraints, sometimes reflected by performance losses (e.g., the threshold implementations in [50, 13] prevent glitches by increasing the number of shares).

Quite naturally, an extreme solution to this problem is to take advantage of asymmetric cryptographic primitives, for which algebraic properties usually make the masking countermeasure much easier to implement, as suggested for El Gamal encryption [38] and pairing-based MACs [44]. While these solutions may indeed lead to better efficiency vs. security tradeoffs than the direct protection of a block cipher with masking in the long term, they still imply significant performance overheads that may not be affordable for low-cost devices, and can only be amortized for quite high-order masking schemes.

[1] Note that using the key-evolution approach for the session key derivation (i.e. computing session key K_i as an "evolved" session key K_{i-1}) is often impractical, since it requires synchronization between the sender and the receiver.

[2] Excepted if combined with additional heuristic assumptions such as in [47].

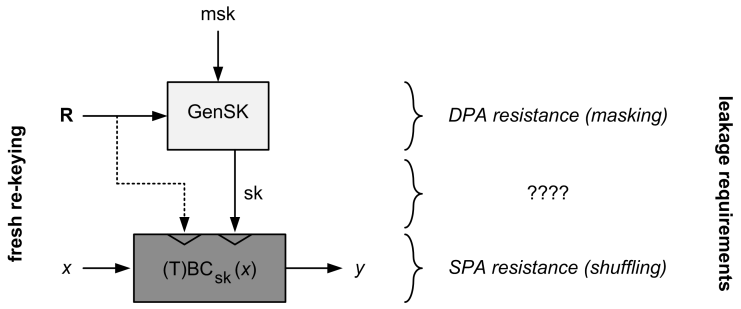

Fig. 1. Fresh re-keying and its leakage requirements.

Taking these challenges into account, an appealing intermediate path called *fresh re-keying* has been initiated by industrial and academic research [30, 46]. As illustrated in Figure 1, its main idea is to exploit a good "separation of duties" between a re-keying function *GenSK* and a block cipher or tweakable block cipher [40].[3] That is, the function *GenSK*, which is used to generate the fresh session keys sk, needs to resist Differential Power Analysis attacks (DPA), i.e. attacks exploiting multiple measurements per key. By contrast, the (possibly tweakable) block cipher only needs to resist Simple Power Analysis (SPA) attacks, i.e. attacks exploiting a single measurement per key (or DPA attacks with limited trace count if the key refreshing is amortized). Quite naturally, this solution is useless in case *GenSK* is also a (tweakable) block cipher (since it would then be equally difficult to protect with masking). So previous fresh re-keying schemes additionally came with heuristic arguments justifying that this function does not need to be cryptographically strong, and only has to fulfill a limited set of properties (e.g., good diffusion). On top of this, they suggested to exploit key homomorphic *GenSK*'s, so that their masked implementation is (much) simplified. Taking advantage of key homomorphism indeed reduces the computational overheads and randomness requirements of masking to the minimum (i.e., the refreshing of the secret master key and the computation of the key homomorphic for each share). Besides, it also allows avoiding issues related to the independent leakage assumption, since we can then compute *GenSK* on each share independently. A polynomial multiplication in (e.g., a ring) was finally proposed as a possible instance for such functions [30, 46].

Yet, and while conceptually elegant, Figure 1 also suggests the important caveat of existing fresh re-keying schemes. Namely, between the re-keying function *GenSK* that can be well protected against DPA thanks to masking, and the underlying (tweakable) block cipher that has to be secure against SPA (e.g., thanks to shuffling), one has to re-combine the shares to produce a fresh session

[3] As discussed in [23], using a tweakable block cipher allows obtaining beyond birthday security for this fresh re-keying scheme, while a standard block cipher only provides birthday security. Whether one or the other option is chosen will be essentially equivalent for the discussions in this paper.

key sk: an operation of which the leakage was essentially left out of the analysis so far. More precisely, the only (informal) guidelines were that it should be difficult to precisely extract hard information (e.g., bits) about sk, in order to avoid the algebraic attacks outlined in [45]. Recent results from Belaid et al. and Guo and Johansson then made it clear that a small noise may not be sufficient to secure *GenSK* against leakage on sk [9, 11, 35].

Our contribution. Based on this state-of-the-art, we initiate the first formal study of fresh re-keying functions that are at the same time easy to mask (since key homomorphic or almost key homomorphic [18]) and cryptographically strong. To this end, we propose new security models using the ideal/real world paradigm and show that our instantiations described below can be proven secure under reasonable assumptions. Informally, our security guarantees state that even given continuous leakage (for instance, probing leakage or noisy leakage), the adversary will not be able to attack the re-keying function any better than an adversary that just obtains uniformly random session keys. We prove the security of two different instantiations of a re-keying function in this model (making different assumptions on the type of leakage as outlined below).

On the one hand, we start from the observation that in the context of re-keying, the function *GenSK*'s output is in fact never given to the adversary completely. Instead, the adversary learns only some *partial* leakage information about *GenSK*. Taking advantage of this observation, we first introduce a new assumption of Learning Parity with Leakage (LPL), of which the main difference with the standard Learning Parity with Noise (LPN) problem is that it relies on additive Gaussian (rather than Bernoulli) noise [32, 15, 14]. Note that we use the name Learning Parity with Leakage (and not with Gaussian noise) to reflect the fact that the amount of noise can be much larger than in the standard LPN assumption (since in a re-keying scheme, the authorized parties only deal with noise-free information). Then, we show that our new LPL assumption can be reduced to the standard LPN assumption. Finally, we instantiate a re-keying scheme based on LPL that is trivial to mask (since key homomorphic) and provide the actual noise values required to reach different security levels against adversaries targeting the re-combination step of the fresh key sk in Figure 1. Conceptually, the main advantage of this construction is that it exploits the noise that is naturally available in side-channel leakages. However, our study also suggests that this physical noise may have to be increased by design to reach high security levels – see Section 6 for a discussion.

On the other hand, we consider the complementary context of small embedded devices with too limited noise for the previous LPL problem to be hard. In this case, we take advantage of the recently introduced Learning with Rounding (LWR) assumption [7], and describe a re-keying that is perfectly suitable for a low-noise environment. In order to make it most efficient, we instantiate it with computations in $\mathbb{Z}_q$ with $q = 2^b$, and a rounding function that can be simply implemented by dropping bits. This allows us to directly take advantage of standard arithmetic operators available in most computing platforms (e.g., recent ARM devices perform 32-bit multiplications in one cycle), without any

additional hassle due to complex reductions. We then show that this re-keying function based on the LWR assumption can be efficiently masked thanks to an additional error correction step, which makes it almost key homomorphic. We finally provide parameters to instantiate it for various security levels, including very aggressive choice of parameters for which the security is not proven (or at least it is not based on the standard assumptions). Conceptually, the main advantage of this construction is that it ensures stronger cryptographic properties (i.e., computational indistinguishability from uniform) and therefore may be of interest beyond the re-keying scenario considered here.

As a result, we obtain two cryptographic constructions that can be used for fresh re-keying in both low-noise and high-noise contexts, for which masked implementations have minimum overheads and randomness requirements, and can easily fulfill the independent leakage assumption.

Besides the formal modeling and the security proofs of our constructions, we also present preliminary implementation results for our re-keying functions. Concretely, we report in Section 6 on implementations of our re-keying function on a 32-bit ARM and an 8-bit Atmel device. We give a comparison with masked AES implementations and show that for certain choices of the parameters (and under reasonable noise assumptions for the LPL-based construction) , we can achieve improved efficiency.

Related works. Our two re-keying constructions are naturally connected to previous cryptographic primitives based on LPN and LWR, such as LAPIN [36] and SPRING [8]. Interestingly, when it comes to their resistance against side-channel attacks, these new constructions also bring a neat solution to the main drawbacks of LAPIN and SPRING. Namely, for LAPIN, it remained that the generation and protection of the Bernoulli noise was challenging [31]. But when relying on the LPL assumption, we gain the advantage that this noise does not have to be generated (since it corresponds to the leakage noise that anyway has to be available on chip for the masking of F to be effective – see again [21, 37, 53, 26, 27]). As for SPRING, the main challenge was to deal with the masking of the (non–linear) rounding operation [19]. But as described in Section 5, this masking is made easier with our re-keying function based on LWR.

A similar technique to our reduction from LPL to LPN was used in [11], who also analyze physical noise used as a countermeasure to leakage in the context of finite field multiplication and attack this by deriving LPN instances.

2 Preliminaries

Notations. We denote scalars u, v by small italic single characters. Vectors $\mathbf{u}, \mathbf{r}, \mathbf{k}$ are denoted by small bold letters. Matrices $\mathbf{R}, \mathbf{T}$ are denote by capital bold letters. We use capital letters R, U to denote scalars if we want to emphasize that we treat them as random variables and argue about probabilities.

Standard Assumptions. Our constructions will be based on the Learning Parity with Noise (LPN) assumption and the Offset Learning with Uniform Noise assumption. To analyse these, we recall some relevant standard assumptions:

Definition 1 (LPN)**.** *Let $0 < \tau < \frac{1}{2}$ be fixed and $n \in \mathbb{N}$. For (unknown) $\mathbf{k} \in \mathbb{Z}_2^n$, the $\mathsf{LPN}_{n,\tau}$ sample distribution is is given by*

$$\mathcal{D}_{\mathsf{LPN},n,\tau} := (\mathbf{r}, \ell) \text{ for } \mathbf{r} \in \mathbb{Z}_2^n \text{ uniform, } e \leftarrow \mathcal{B}_\tau,\ \ell := \langle \mathbf{r}, \mathbf{k} \rangle + e \bmod 2,$$

where $\mathcal{B}_\tau$ denotes a Bernoulli distribution with $\Pr[e = 1] = \tau, \Pr[e = 0] = 1 - \tau$.

Given query access to $\mathcal{D}_{\mathsf{LPN},n,\tau}$ for uniformly random $\mathbf{k}$, the search $\mathsf{LPN}_{n,\tau}$-problem asks to find $\mathbf{k}$. The decision $\mathsf{LPN}_{n,\tau}$-problem asks to distinguish an oracle for $\mathcal{D}_{\mathsf{LPN},\tau}$ for uniformly random $\mathbf{k}$ from an oracle that outputs uniformly random values from $\mathbb{Z}_2^n \times \mathbb{Z}_2$. The search/decision $\mathsf{LPN}_{n,\tau}$-Assumption asserts that these problems are infeasible for PPT algorithms. (n, τ are given functions of the security parameter).

Definition 2 (LWE **[54]).** *Let Φ be some efficiently sampleable noise distribution on $\mathbb{Z}$, $n \in \mathbb{N}$ and $q > 0$ (often, but not necessarily prime). For (unknown) $\mathbf{k} \in \mathbb{Z}_q^n$, the $\mathsf{LWE}_{n,\Phi}$ sample distribution is given by*

$$\mathcal{D}_{\mathsf{LWE},n,q,\Phi} := (\mathbf{r}, \ell) \text{ for } \mathbf{r} \in \mathbb{Z}_q^n \text{ uniform, } e \leftarrow \Phi,\ \ell := \langle \mathbf{r}, \mathbf{k} \rangle + e \bmod q.$$

Given query access to $\mathcal{D}_{\mathsf{LWE},n,q,\Phi}$ for uniformly random $\mathbf{k} \in \mathbb{Z}_q^n$, the search $\mathsf{LWE}_{n,q,\Phi}$-problem asks to find $\mathbf{k}$. The decision $\mathsf{LWE}_{n,q,\Phi}$-problem asks to distinguish an oracle for $\mathcal{D}_{\mathsf{LWE},q,\Phi}$ for uniformly random $\mathbf{k}$ from an oracle that outputs uniformly random values from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The search/decision $\mathsf{LWE}_{n,q,\Phi}$-Assumption asserts that these problems are infeasible for PPT algorithms.

Usually, Φ is taken to be a discrete Gaussian, i.e. a probability distribution whose density $\Pr_{E \leftarrow \Phi}[E = x]$ is proportional to $\exp\left(-\frac{x^2}{2s}\right)$. In this case, one usually takes s as a paramter of the scheme rather than Φ. Note that LPN is an important special case of LWE with $q = 2$ and Φ a Bernoulli distribution.

Definition 3 (LWU **[25, 48]).** *Another important special case is when the error distribution Φ is uniform from some interval, say $\{0, \ldots, B-1\}$. In this paper, we call it the Learning with Uniform Noise distribution/problem/assumption $\mathsf{LWU}_{n,q,B}$. Note that only the length of the interval matters in this case, as the adversary can add a constant shift itself.*

Definition 4 (LWR **[7]).** *The Learning with Rounding (LWR) distribution/ problem/assumption is often seen as a deterministic variant of LWE, where instead of adding some random noise $e \leftarrow \Phi$ to perturb $\langle \mathbf{r}, \mathbf{k} \rangle$, we round $\langle \mathbf{r}, \mathbf{k} \rangle$.*

More precisely, for appropriately chosen integers $p < q$, the rounding function $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is $\lfloor x \rceil_p := \lfloor x \frac{p}{q} \rceil$, where $x \in \mathbb{Z}_q$ is represented as $x \in \{0, \ldots, q-1\}$. When applying $\lfloor \cdot \rceil_p$ to a vector in $\mathbb{Z}_q^n$, we apply it component-wise.

For (unknown) $\mathbf{k} \in \mathbb{Z}_q^n$, the $\mathsf{LWR}_{n,q,p}$ sample distribution is given by

$$\mathcal{D}_{\mathsf{LWR},n,q,p} := (\mathbf{r}, \ell) \text{ for } \mathbf{r} \in \mathbb{Z}_q^n \text{ uniform and } \ell = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p.$$

Again, given query access to $\mathcal{D}_{\mathsf{LWR},n,q,p}$ for uniformly random $\mathbf{k} \in \mathbb{Z}_q^n$, the search $\mathsf{LWR}_{n,q,p}$-problem asks to find $\mathbf{k}$. The decision problem asks to distinguish $\mathcal{D}_{\mathsf{LWR},n,q,p}$ for uniformly random $\mathbf{k}$ from an oracle that outputs samples $(\mathbf{r}, \lfloor u \rfloor_p)$ for $\mathbf{r} \in \mathbb{Z}_q^n, u \in \mathbb{Z}_q$ uniform. Note that $\lfloor u \rfloor_p$ is not uniform in $\mathbb{Z}_p$ unless $p \mid q$. The search/decision $\mathsf{LWR}_{n,q,p}$-Assumption asserts that the problem is infeasible for PPT algorithms.

3 General framework

3.1 Re-keying schemes

A re-keying scheme RK is a cryptographic primitive which generates session keys sk from a secret key msk and some public randomness $\mathbf{R}$. More precisely, $\mathsf{RK} = (\mathit{Gen}, \mathit{GenSK}, \mathit{CorSK}, \mathcal{D})$ consists of the following three PPT algorithms and an efficiently samplable distribution $\mathcal{D}$ from which the randomness is sampled:

- $\mathit{Gen}(1^\lambda)$: Outputs a secret key msk and d shares thereof that we denote with $(\mathsf{msk})_d$.
- $\mathit{GenSK}((\mathsf{msk})_d, \mathbf{R})$: Outputs a session key sk, new shares $(\mathsf{msk}')_d$ and potentially correction information v.
- $\mathit{CorSK}(\mathsf{msk}, \mathbf{R}, v)$: Outputs a session key sk.

Concretely, *GenSK* will be run by the chip to protect while *CorSK* will be run by the other party. RK is called correct iff for $(\mathsf{msk}, (\mathsf{msk})_d) \leftarrow \mathit{Gen}(1^\lambda)$, $\mathbf{R} \leftarrow \mathcal{D}$, $(\mathsf{sk}, (\mathsf{msk}')_d, v) \leftarrow \mathit{GenSK}((\mathsf{msk})_d, \mathbf{R})$, we have that $\mathit{CorSK}(\mathsf{msk}, \mathbf{R}, v) = \mathsf{sk}$ holds with overwhelming probability. Further, we require that $(\mathsf{msk}')_d$ and $(\mathsf{msk})_d$ follow the same distribution, conditioned on msk.

One may think of $(\mathsf{msk}')_d$ and $(\mathsf{msk})_d$ as some form of encoding that protects against side-channel attacks.[4] The correction information v may be needed in some constructions since the session key when computed from $(\mathsf{msk})_d$ by *GenSK* may be different when computed from msk by *CorSK*.

For the security definition of a re-keying scheme, we define three interactive PPT algorithms *Real*, *Ideal* and *Sim*. An adversary A will interact with them during a polynomially bounded amount of sessions (see Figure 2).

We denote this process with $A^{\mathit{Real}((\mathsf{msk})_d)}(1^\lambda)$ when A interacts with *Real* and $A^{(\mathit{Sim}^{\mathit{Ideal}_c}, \mathit{Ideal}_c)}(1^\lambda)$ when A interacts with *Sim* and *Ideal*. In the latter case, $(\mathit{Sim}^{\mathit{Ideal}_c}, \mathit{Ideal}_c)$ is the concatenation of their outputs. During each session A receives the following output from *Real*, *Ideal* and *Sim*:

- $\mathit{Real}((\mathsf{msk})_d)$: Takes $(\mathsf{msk})_d$ from the input and sample $\mathbf{R} \leftarrow \mathcal{D}$. Then, run $(\mathsf{sk}, (\mathsf{msk}')_d, v) \leftarrow \mathit{GenSK}((\mathsf{msk})_d, \mathbf{R})$. It outputs $(\mathbf{R}, \mathsf{sk}, v)$ to A. Further, it has additional, model specific inputs/outputs, e.g. probes which are leaked. It then overwrites $(\mathsf{msk})_d := (\mathsf{msk}')_d$ to be used in the next session.

[4] For the reader familiar with side-channel resistant implementations $(\mathsf{msk})_d$ denotes a masking of msk and $(\mathsf{msk}')_d$ denotes a refreshing of the shares of the masking. Indeed, in all our constructions, $(\mathsf{msk})_d$ will be d uniformly chosen values msk_i with $\sum_i \mathsf{msk}_i = \mathsf{msk}$. While we call $(\mathsf{msk})_d$ "shares" in the definition to match our later notation more closely, $(\mathsf{msk})_d$ could in principle be anything.

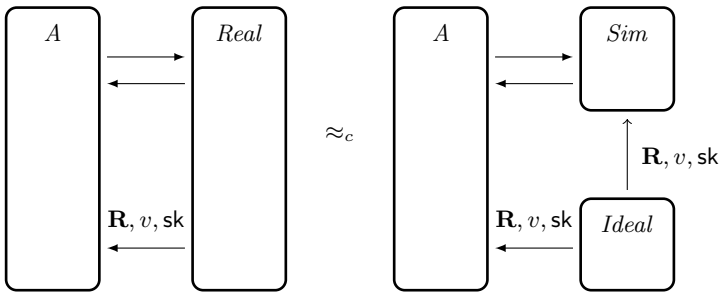

Fig. 2. An adversary A breaking the security of a re-keying scheme distinguishes the following cases: A interacts with *Real* or he interacts with *Sim* and *Ideal*. The session key is sk, the randomness for the session key generation $\mathbf{R}$ and some correction information v. Additionally A sees some model specific leakage.

$Ideal_c(1^\lambda)$: outputs in each session $(\mathbf{R}, v, \mathsf{sk})$ for uniform random sk, independent v and $\mathbf{R} \leftarrow \mathcal{D}$. Its random tape is c.
$Sim(1^\lambda)$: simulates model specific outputs while accessing the outputs of $Ideal_c$.

A re-keying scheme is called secure iff for any PPT A:

$$\left|\Pr\left[A^{Real((\mathsf{msk})_d)}(1^\lambda) = 1\right] - \Pr\left[A^{(Sim^{Ideal_c}, Ideal_c)}(1^\lambda) = 1\right]\right| \leq negl(\lambda),$$

where the probability is taken over the random tape of A, *Real*, *Sim*, c and $(\mathsf{msk}, (\mathsf{msk})_d) \leftarrow Gen(1^\lambda)$. It is easy to see that session keys sk need to be indistinguishable from uniform chosen keys to fulfill this security notion. This needs to hold even given the model specific leakage. The next two sections will describe two different leakage models that we consider in this work.

3.2 The leakage model for re-keying schemes

For the two different instantiations that we present in Section 4 and Section 5 we propose two different leakage models. The leakage model specifies what additional information the adversary can obtain in the real world. In the ideal world the leakage then has to be simulated in a consistent way by the simulator *Sim*.

Re-keying schemes in the t-probing model. An important model to analyze the security of side-channel countermeasures is a security proof in the t-probing model [37]. In the t-probing model, the adversary is allowed to learn up to t-intermediate values of the computation of *GenSK*, i.e. of the generation of the session key. Notice that the definition of an intermediate value typically depends on the underlying scheme and its implementation. Our schemes are naturally described using group operations and internal values are group elements, notably elements from $\mathbb{Z}_p$ for p being a power of 2 (this includes the case of bits with $p = 2$). This means that the adversary A specifies a set of t probes $\mathcal{P}$, where $|\mathcal{P}| \leq t$ and the adversary obtains back from $Real((\mathsf{msk})_d)$ the intermediate values $V = \{v_{w_i}\}$, where $w_i \in \mathcal{P}$ and v_{w_i} is the value carried on the intermediate

result labeled with w_i. Hence, if the computation was carried out over $\mathbb{Z}_p$, then the adversary obtains a set V with t elements, where each value in V corresponds to one of the intermediate values produced during the computation of $GenSK$. To show security of the re-keying function in the t-probing model, we need to construct an efficient simulator Sim that can simulate the replies of the adversary's probes V_i without probing access (i.e. from the values Sim obtains in the ideal world).

Re-keying schemes in the t-noisy probing model. For our first re-keying scheme, we will show security in a weaker model than the standard t-probing model. We will assume that each of the t-probes is perturbed with additive Gaussian noise. This is a common assumption in works on side-channel analysis and can for instance be guaranteed using a physical noise generator [34]. For simplicity we assume that all intermediate values are in $\mathbb{Z}_2$ and for a probe on a wire that carries the bit $b \in \mathbb{Z}_2 = \{0, 1\}$ the adversary learns $b + e$, where e corresponds to noise from a continuous Gaussian distribution. It is important to note that the addition of the noise is a normal addition in the reals. The above can be generalized but for ease of exposition we stick to these simplifications in the rest of the paper. Using this terminology, in the t-noisy probing model the adversary A specifies a set of t probes $\mathcal{P}$, but instead of obtaining the exact values of the intermediate wires, he obtains a noisy version of them. That is, the set of replies V is $V = \{v_{w_i} + e_i\}$. Since the adversary only sees a noisy version of the intermediate values, the t-noisy probing model offers a weaker security guarantee than the t-probing model. Trivially, the leakage obtained in the t-noisy probing model can be simulated by t-probing leakage.

Security against continuous (noisy) probes. In the above two section we considered an adversary that can specify a set of probes and obtains the corresponding intermediate values. In the continuous probing model, this notion is extended by letting the adversary specify for *each* execution adaptively a new set of probes $\mathcal{P}$. More precisely, during the execution in the real world at the beginning of the i-th session, the adversary specifies adaptively a set of probes $\mathcal{P}^i$, and obtains the (noisy) intermediate values V^i that correspond to the wires specified in $\mathcal{P}^i$. Notice that the choice of $\mathcal{P}^i$ can depend on all information that the adversary A has seen previously, i.e. on $\{(V^j, \mathsf{sk}^j, \mathbf{R}^j, v^j)\}_{j \in [i-1]}$. Observe that the continuous probing model is significantly stronger than the one-shot (noisy) probing model as the adversary obtains significantly more information that he can exploit in breaking the scheme.

3.3 Masking schemes

Masking schemes are a method to achieve security in the t-probing model (and hence also in the t-noisy-probing model since this model is weaker). In a masking scheme, each sensitive intermediate variable is split into d shares such that knowing only $(d-1)$ shares does not reveal information about the sensitive variable.

Consider for instance the Boolean maskings scheme [37] where a sensitive variable k is represented by d random shares $k_1, \ldots, k_d$ such that $k = \sum_i k_i$. Clearly, knowing only $d-1$ arbitrary shares does not allow to recover the secret value k. The main difficulty in designing secure masking schemes is in computing with shared variables in a secure way. To this end one needs to design masked operations. In traditional masking schemes one typically designs masked algorithms for the basic operations of the underlying group (e.g., for addition and multiplication). While linear operations can be masked very efficiently, masking the non-linear operations, e.g., the multiplication, is significantly more costly. For instance, a masked multiplication in the Boolean masking scheme results into an overhead of $O(d^2)$ for each masked multiplication used in the computation.

We will apply masking schemes to protect the re-keying function against t-(noisy) probing attacks. To obtain better efficiency when executed in the masked domain, we design in the next sections cryptographically strong re-keying functions that are almost linear. Concretely, for our construction *GenSK* is divided into d sub-computations where each sub-computation only takes as input one share msk_i of $(\mathsf{msk})_d$ (this is the linear part of the re-keying function *GenSK*). Only at the very end of the computation the outputs of this linear part are recombined to obtain the final session key. We emphasize that by following this approach our construction also obtains strong glitch resistance – unlike normal masked implementations. Glitches can occur in hardware implementations due to synchronization problems. Since in our construction the sub-computations only depend on individual shares, glitches are prevented as we do not use operations that access multiple shares jointly.

Masking in the continuous leakage model. To guarantee security of a masked implementation in the continuous leakage model, the secret shares used in the computation and in particular the shares of the key need to be refreshed frequently. Such a refreshing is typically done by a probabilistic $\mathsf{Refresh}$ algorithm. In our case, the refreshing is part of the *GenSK* algorithm and takes as input the shared master secret key $(\mathsf{msk})_d$ and produces a refreshed master secret key $(\mathsf{msk}')_d$. The correctness requirement of the refreshing algorithm says that both $(\mathsf{msk})_d$ and $(\mathsf{msk}')_d$ correspond to the same master secret key msk. If the underlying masking scheme is the Boolean masking scheme then this means that $\sum_i \mathsf{msk}_i = \sum_i \mathsf{msk}'_i$. Further, we require that the the distribution of (msk') is uniform among all possible such sharings of msk. Besides correctness, the refreshing algorithm also shall guarantee that side-channel information (i.e., the (noisy) probes) from different executions of the re-keying functions cannot be combined in an exploitable way. Informally, the refreshing schemes $\mathsf{Refresh}$ is said to be secure in the t-probing model against continuous attacks, if the leakage for probe sets $\mathcal{P}^j$ can be simulated without knowledge of the master secret key. Typically, the simulation is statistically indistinguishable from the continuous real execution of the $\mathsf{Refresh}$ algorithm.

To prove the security of our construction, we will need a secure instantiation of a refreshing algorithm. Several variants have been proposed in the literature with varying efficiency [4, 26, 37]. Since the focus of this work is not on designing

secure refreshing schemes, we mainly ignore them for the rest of this paper. The following lemma can be proven about the Refresh from [26].

Lemma 1. *For any set of probes $\mathcal{P}$ with cardinality t, there exists a PPT simulator $\mathcal{S}$ and a set $\mathcal{I}$ of cardinality t such that for any $k \in \mathcal{K}$ and $k_1, \ldots, k_d$, $k'_1, \ldots, k'_d$ chosen uniformly at random subject to the constraint that $k := \sum_i k_i = \sum_i k'_i$ we have:*

$$(\mathcal{P}(\mathbf{k}' \leftarrow \mathsf{Refresh}(\mathbf{k})), \mathbf{k}_{\mathcal{I}}, \mathbf{k}'_{\mathcal{I}}) \equiv (\mathcal{S}(\mathcal{P}, \mathbf{k}_{\mathcal{I}}, \mathbf{k}'_{\mathcal{I}}), \mathbf{k}_{\mathcal{I}}, \mathbf{k}'_{\mathcal{I}}),$$

where in the above "$\equiv$" denotes the statistical equivalence and for a vector $\mathbf{k}$ and a set $\mathcal{I} \subseteq [d]$ the vector $\mathbf{k}_{\mathcal{I}}$ is the vector $\mathbf{k}$ restricted to the positions in $\mathcal{I}$.

The refreshing algorithm from Lemma 1 has complexity $O(d^2)$ (where the hidden constant in the O-notation are small). Recently, an improved refreshing algorithm based on expander graphs was proposed which has complexity $O(d)$ [4]. We notice that while asymptotically better, the hidden constants in $O(d)$ are much larger and hence are of less practical relevance. In our applications, the keys and all shares are vectors $k, k_i \in \mathbb{Z}_p^n$ for some n and the sharing of each of the n coordinates can be done independently. Consequently, we assume that if we probe only a total of t individual coordinates of some k_i's, then there exists a subset $J \subset \{1, \ldots, n\}$ of coordinates with $|J| \leq t$, such that the above Lemma 1 holds when restricted to coordinates from J.

4 Fresh re-keying with physical noise

In this section, we instantiate the abstract re-keying scheme described above in an environment where sufficient physical noise is available. Our construction exploits the physical noise available in side-channel measurements in a constructive manner: the computation of the re-keying function is tailored in such a way that if the adversary obtains t-noisy probing leakage, he will not be able to break the re-keying scheme. While we believe that exploiting physical noise in a constructive way (i.e., for designing new cryptographic primitives) is an interesting conceptual contribution by itself, it also leads to potential efficiency improvements as we show in the implementation section (cfr. Section 6).

To show security of our re-keying scheme, we introduce a new learning assumption that we call the *Learning Parity with Leakage* (LPL) assumption. The LPL assumption says that inner product with physical noise cannot be distinguished from uniform samples. The main technical step is to show that the LPL assumption can be reduced to the classical LPN assumption (this is shown in Section 4.1). Notice that the most important difference between these two assumptions is that in LPL we add additive Gaussian noise (and no modular reduction is carried out), while in LPN the noise comes from the binomial distribution (and a modular reduction is carried out).

Of course, the requirement that the physical noise follows a Gaussian distribution is a strong assumption, and may not be perfectly fulfilled in practice:

physical noise indeed originates from a variety of sources (transistor noise, measurement noise, noise engines added by the cryptographic designers, ...). Yet, it has been observed in many practical settings that this assumption holds to a good extent [41]. More importantly, it is the starting point of most of the (e.g. template and regression-based) attacks that are usually considered in side-channel security evaluations [20, 56]. Besides, it has been shown recently how to verify empirically that deviations from this Gaussian assumption do not significantly impact the security level of an implementation [28], which can therefore be done for our primitives as well.[5]

Our re-keying function. We present our proposed LPN-based re-keying scheme $\Pi_{\text{noisy}} = (\mathit{Gen}, \mathit{GenSK}, \mathit{CorSK}, \mathcal{D})$, which will be proven secure when sufficient physical noise is available in the leakage measurements. Let n be the length of the master secret key, $m < n$ be the length of the session keys and d the number of shares used for the masking scheme. The distribution $\mathcal{D}$ from which the fresh randomness is sampled is defined as drawing uniformly at random $\mathbf{R} \leftarrow \mathbb{Z}_2^{m \times n}$. Let $H \colon \mathbb{Z}_2^m \to \mathbb{Z}_2^m$ be a hash function (modeled as a random oracle, see discussion below) and assume we have some secure refresh algorithm $\mathsf{Refresh}$ that satisfies the property of Lemma 1. Our re-keying function is then defined as follows:

$\mathit{Gen}(1^\lambda)$**:** Samples $\mathsf{msk} \leftarrow \mathbb{Z}_2^n$.

It creates d shares $(\mathsf{msk})_d = \mathsf{msk}_1, \ldots, \mathsf{msk}_d$ such that $\sum \mathsf{msk}_i = \mathsf{msk}$.

$\mathit{GenSK}((\mathsf{msk})_d, \mathbf{R})$**:** A probabilistic algorithm working as follows:

1. Compute $\mathbf{u}_i = \mathbf{R} \cdot \mathsf{msk}_i$
2. Compute $\mathbf{u} = \sum_i \mathbf{u}_i$ iteratively as $((\ldots(\mathbf{u}_1 + \mathbf{u}_2) + \mathbf{u}_3) + \ldots) + \mathbf{u}_d$. Notice that other ways of computing this sum are possible, but will make the analysis more involved.
3. The session key is computed as $\mathsf{sk} = H(\mathbf{u})$.
4. Finally refresh the shares $(\mathsf{msk}')_d \leftarrow \mathsf{Refresh}((\mathsf{msk})_d)$.
5. Output $(\mathbf{R}, \mathbf{u})$.

$\mathit{CorSK}(\mathsf{msk}, \mathbf{R})$**:** Output $H(\mathbf{R} \cdot \mathsf{msk})$.

From the above description it is clear that the additional value v (the correction term) is not used in this construction. It will be used in our construction from Section 5. Moreover, the reader may notice that the re-keying function is linear (except for the application of H at the end). The security against side-channel attacks comes from the fact that the adversary in the t-noisy probing model only obtains noisy intermediate values.

[5] Note that if significant deviations from the Gaussian assumptions were observed, it would not imply that our following constructions are directly broken – just that the parameters of our reductions below, and hence the parameters of our construction, will have to be changed, cfr. Remark 1 below.

On the use of the Random Oracle. Our construction outputs $\mathsf{sk} = H(\mathbf{u})$ as the session key rather than $\mathbf{u} = \mathbf{R} \cdot \mathsf{msk}$. The use of the random oracle H is only for simplifying and unifying the security analysis. In particular, in case the preliminary session key $\mathbf{u}$ is used directly in an accompanied block cipher – as is the typical application of a re-keying scheme – then the additional hash function execution is not needed. We notice that in such a case the analysis would (at least) require that the block-cipher is secure against related key attacks. One way to enforce this (and obtain a security proof) is to model the block cipher as an ideal cipher in the analysis. Finally, we want to mention that of course the adversary can learn noisy probes of the preliminary session key $\mathbf{u}$ – however, in any applications of the re-keying function one has to make sure that these values are never seen directly by the adversary (i.e., without the noise).

Allowed Probes. In the t-(noisy) probing model, we allow the adversary to select probes from the following intermediate values: individual bits of msk_i or msk'_i, internal values of the computation of the $\mathbf{u}_i = \mathbf{R}\mathsf{msk}_i$, internal wires of $\mathsf{Refresh}$ or H (unless in the ROM) and individual bits of any $\sum_{i=1}^{k} \mathbf{u}_i$ (as specified above). The adversary is not allowed to obtain multiple noisy probes from the same value during a single session. This assumption is the same as made in all works on the noisy leakage model [21, 53, 26]. Finally, we assume that the adversary never probes $\mathbf{R}$ and sk as these values he obtains for free anyway.

4.1 Security of our Construction based on Physical Noise

In this section, we prove the security of our construction under the LPL (Learning Parity with Leakage) assumption. To this end, we will first formally define our new assumption, and show that it can be reduced to the classical LPN assumption. We then generalize LPL to an assumption that also models leakage from intermediate values from the computation of the session key and show that this change in the assumption does not affect the reduction by much. The LPL assumption with noisy probes then allow us to prove the security of our construction in the above specified model. So to summarize we show:

$$\mathsf{LPN} \text{ is hard} \implies \mathsf{LPL} \text{ is hard} \implies \mathit{\Pi}_{\text{noisy}} \text{ is secure.}$$

Learning Parity with Leakage (LPL). We now give a formal definition of the LPL assumption, in which we model the physical noise distribution with a continuous Gaussian distribution Φ_s with density function $\Phi_s(x) := \frac{1}{\sqrt{2\pi}s} \exp(-\frac{x^2}{2s^2})$ and standard deviation s. First, the $\mathsf{LPL}_{n,s}$ sample distribution for secret $\mathbf{k} \in \mathbb{Z}_2^n$ is defined as

$$\mathcal{D}_{\mathsf{LPL}} := (\mathbf{r}, \ell = \langle \mathbf{r}, \mathbf{k} \rangle + e) \text{ for } \mathbf{r} \leftarrow \mathbb{Z}_2^n, e \leftarrow \Phi_s,$$

where $\langle \mathbf{r}, \mathbf{k} \rangle \in \{0, 1\}$ is computed over $\mathbb{Z}_2$ and $\langle \mathbf{r}, \mathbf{k} \rangle + e$ is taken over the reals. Similarly, we define a distribution $\mathcal{D}_{\mathsf{UniformL}}$ that outputs $(\mathbf{r}, \ell)$, where $\mathbf{r} \leftarrow \mathbb{Z}_2^n$ and $\ell = u + e \in \mathbb{R}$ with $u \leftarrow \{0, 1\}$ is a uniform bit and $e \leftarrow \Phi_s$.

The $\mathsf{LPL}_{n,s}$-search problem asks to find the secret, uniform $\mathbf{k}$, given query access to $\mathcal{D}_{\mathsf{LPL}}$.[6] The decision problem asks to distinguish $\mathsf{LPL}_{n,s}$ from $\mathcal{D}_{\mathsf{UniformL}}$. The search/decision LPL-Assumption is the assumption that these problems are hard for PPT adversaries.

Security proof for LPL. We now show that LPL is at least as hard as LPN for appropriate choices of parameters. For this, we first show that LPL is actually equivalent to a variant of LPN, where the error probability τ is per-sample random and known. Formally, for a (sampleable) distribution Ψ on $[0, \frac{1}{2}]$, the $\mathsf{LPN}_{n,\Psi}$ sample distribution for uniform secret $\mathbf{k}$ is given by

$$\mathcal{D}_{\mathsf{LPN},\Psi} := (\mathbf{r}, \langle \mathbf{r}, \mathbf{k} \rangle + e, \tau) \text{ for } \mathbf{r} \leftarrow \mathbb{Z}_2^n, \tau \leftarrow \Psi, e \leftarrow \mathcal{B}_\tau.$$

Given query access to the sample distributions for fixed, random $\mathbf{k}$, the search $\mathsf{LPN}_{n,\Psi}$-problem ask to find $\mathbf{k}$ and the decision $\mathsf{LPN}_{n,\Psi}$-problem asks to distinguish $\mathcal{D}_{\mathsf{LPN},\Psi}$ from $\mathcal{D}_{\mathsf{Uniform},\Psi}$, where $\mathcal{D}_{\mathsf{Uniform},\Psi}$ outputs samples $(\mathbf{r}, \ell, \tau)$ with $\tau \leftarrow \Psi$ and $(\mathbf{r}, \ell) \leftarrow \mathbb{Z}_2^{n+1}$.

Lemma 2. *The search resp. decision $\mathsf{LPL}_{n,s}$-problem is equivalent to the search resp. decision $\mathsf{LPN}_{n,\Psi}$-problem via a tight, sample-preserving reduction. Here, the distribution for Ψ is given by sampling $\widetilde{U} \leftarrow \{0,1\}, \widetilde{E} \leftarrow \Phi_s, \widetilde{L} = U + E$, $\widetilde{R}_{>1} := \exp\left(\frac{\left|L - \frac{1}{2}\right|}{s^2}\right)$ and outputting $\widetilde{\tau} = \left(\widetilde{R}_{>1} + 1\right)^{-1}$.*

A full proof is given in Appendix A. Here, we only give an intuition and explain the distribution of Ψ.

The key idea is to set Ψ in such a way that the amount of information learned about $\langle \mathbf{r}, \mathbf{k} \rangle$ from a single LPN sample is the same as the amount of information learned about $\langle \mathbf{r}, \mathbf{k} \rangle$ from a single LPL sample. To this end, we consider a quantity called R_{Bayes}, defined below, that measures exactly the amount of information learned about $\langle \mathbf{r}, \mathbf{k} \rangle$. We then compute this value for both the LPN case and for the LPL case. In the LPN case, R_{Bayes} is a function of τ. In the LPL-case, R_{Bayes} is a function of ℓ_{LPL}, where $(\mathbf{r}, \ell_{\mathsf{LPL}})$ is the output from LPL. Equating the values of R_{Bayes} will give us the involved definition of Ψ given above.

In fact, the proof in Appendix A uses this value R_{Bayes} to transform LPL samples into LPN samples and vice versa via the correspondence $\ell_{\mathsf{LPL}} \leftrightarrow R_{\text{Bayes}} \leftrightarrow \tau$. Intuitively, giving an LPN sample $(\mathbf{r}, \ell) = (\mathbf{r}, u + e)$ for $u := \langle \mathbf{r}, \mathbf{k} \rangle, e \leftarrow \mathcal{B}_\tau$ with $\mathbf{r} \neq \mathbf{0}$ is (information-theoretically) equivalent to giving out $(\mathbf{r}, P_0, P_1)$, where $P_i = \Pr_\Omega[u = i \mid \ell \text{ is observed}]$. Since $P_0 + P_1 = 1$, we consider the fraction $R = \frac{P_0}{P_1}$ instead, which uniquely determines P_0, P_1. The probability space Ω for the definition of P_i takes $u \in \mathbb{Z}_2$ uniform for simplicity. For a general "prior" distribution $\Pr[u = 0]$ of u, Bayes' rule gives

$$\frac{\Pr[u = 0 \mid \ell \text{ is observed}]}{\Pr[u = 1 \mid \ell \text{ is observed}]} = R_{\text{Bayes}} \cdot \frac{\Pr[u = 0]}{\Pr[u = 1]} \tag{1}$$

[6] We assume that for any value in $\mathbb{R}$, the adversary A receives an arbitrarily precise but polynomial representation. In particular A chooses how values in $\mathbb{R}$ are represented.

for the random variable R_{Bayes}, defined as a function of ℓ via

$$R_{\mathrm{Bayes}} = \frac{Q_0}{Q_1} \in [0,\infty], \quad Q_i = \Pr_{E\leftarrow\mathcal{B}_\tau, L=E+u}[L = \ell \mid u = i].$$

We have $R = R_{\mathrm{Bayes}}$ and the definition of R_{Bayes} does not depend on how u is chosen and completely captures what can be learned (in addition to any prior knowledge) from a given LPN sample about u via Eq. (1). For LPN, we easily compute $R_{\mathrm{Bayes}} = \frac{1-\tau}{\tau}$ for $\ell = 0$ and $R_{\mathrm{Bayes}} = \frac{\tau}{1-\tau}$ for $\ell = 1$. In the case $\ell = 0$, this means that $\tau = (R_{\mathrm{Bayes}} + 1)^{-1}$. For $\ell = 1$ we have $1 - \tau = (R_{\mathrm{Bayes}} + 1)^{-1}$. Similarly, an individual LPL sample $(\mathbf{r}, \ell_{\mathsf{LPL}}) = (\mathbf{r}, u + e_{\mathsf{LPL}})$ for continuous error $e_{\mathsf{LPL}} \leftarrow \Phi_s$ provides statistical information about u and we can define R_{Bayes} analogously. A simple computation yields $R_{\mathrm{Bayes}} = \frac{\Phi_s(\ell_{\mathsf{LPL}})}{\Phi_s(\ell_{\mathsf{LPL}}-1)} = \exp\big(-\frac{\ell_{\mathsf{LPL}}-\frac{1}{2}}{s^2}\big)$. The distribution of Ψ is constructed such that R_{Bayes} follows the same distribution in both the $\mathsf{LPN}_{n,\Psi}$ and the $\mathsf{LPL}_{n,s}$ case. Indeed, in the definition of Ψ, we mimic the distribution of R_{Bayes} by sampling $\widetilde{U}, \widetilde{E}$ and defining $\widetilde{R}_{\mathrm{Bayes},>1}$ in a similar way to R_{Bayes}. Taking the absolute value in $\widetilde{R}_{\mathrm{Bayes}}$ corresponds to normalizing τ into $1 - \tau$, if τ would otherwise be larger than $\frac{1}{2}$. The latter can be done by replacing ℓ_{LPN} by $1 - \ell_{\mathsf{LPN}}$ in LPN.

Note that this information-theoretic argument does not show how to efficiently transform $\mathsf{LPN}_{n,\Psi}$-samples into $\mathsf{LPL}_{n,s}$-samples and vice versa. This is done in the full reductionist proof in Appendix A.

Next, we reduce standard LPN with fixed noise rate τ' to $\mathsf{LPN}_{n,\Psi}$ with varying noise rate $\tau \leftarrow \Psi$. Clearly, if $\tau \geq \tau'$, this is very easy by just adding additional noise. If $\tau < \tau'$, the reduction fails, but any single sample with small noise rate can reveal at most 1 bit of information about $\mathbf{k}$. Hence, we need to bound the number of such outliers. Consequently, we have the following theorem:

Theorem 1. *Consider $s > 0$ and $0 < \tau' < \frac{1}{2}$. Then, provided s is sufficiently large, the $\mathsf{LPL}_{n,s}$ problem is at least as hard as the $\mathsf{LPN}_{n,\tau'}$ problem.*
More precisely, if $\mathsf{LPN}_{n-X,\tau'}$ is (t,ε,Q)-secure, then $\mathsf{LPL}_{n,s}$ is (t',ε',Q') secure with $Q = Q' - X, t \approx t', \varepsilon \approx \varepsilon'$.[7] *Here, X is a random variable measuring the loss of dimension. X follows a Bernoulli distribution on Q tries with success probability p, where $p = \Pr_{\tau\leftarrow\Psi}[\tau < \tau']$ and where Ψ is defined as in Lemma 2.*

Let $0 < \Delta n$ be a (small) real number measuring the acceptable loss of dimension. Then by setting s such that

$$s \ln\Big(\frac{1-\tau'}{\tau'}\Big) > \mathrm{Fc}^{-1}\Big(\frac{\Delta n}{2Q}\Big) + \frac{1}{2s}, \tag{2}$$

where $\mathrm{Fc}(x) = 1 - \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp(-\frac{t^2}{2})dt$ is the complementary cdf of the normal distribution, we can ensure $\mathbb{E}[X] = pQ < \Delta n$.

[7] Note that here the (known) dimension of the LPN secret is random as well. If $X > n$, the problem is considered trivial.

Proof. See Appendix A.

To make the above theorem useful, we ask for $\Delta n \leq 1$. In this case, X is approximately Poisson distributed with parameter Δn. By making Δn negligibly small, we can have $X = 0$ with overwhelming probability. For setting parameters, we consider $\Delta n = 1$ acceptable, which gives $s = \mathcal{O}\Big(\sqrt{\log Q}/\log\big(\frac{1-\tau}{\tau}\big)\Big)$.

Remark 1. We observe that in Lemma 2, we showed that $\mathsf{LPL}_{n,s}$-samples $(\mathbf{r}, \ell)$ provide the more information about $\langle \mathbf{r}, \mathbf{k} \rangle$, the further away ℓ is from $\frac{1}{2}$. This is due to the superexponential decay of the Gaussian error function, which means that very large values of $\ell > 1$ are extremely more likely to have come from $\langle \mathbf{r}, \mathbf{k} \rangle = 1$ than to have come from $\langle \mathbf{r}, \mathbf{k} \rangle = 0$. Since we see a usually very large number Q of samples, it is the most extreme outliers for ℓ that determine the noise level τ' (and hence security) of LPN via the correspondence of Thm. 1. In particular, we care about outliers that appear with probability Q^{-1}. Note that this means that our reduction is sensitive to the behavior of the distribution tail of the physical noise, for which the assumption that this is Gaussian is much harder to verify. For example, a faster asymptotic decay than quadratic-exponential would hurt our reduction in terms of parameters, while a slower decay would lead to better parameters. Ideally, one would want a single-exponential decay rate. Also, since an adversary might choose to ignore all samples except for the outliers, there are actually attacks corresponding to the parameter loss of our reduction, provided the attacks do not use many samples.

LPL with Leakage of Intermediate Values To adequately model the fact that the adversary may probe intermediate values, we consider a variant $\mathsf{LPL}_{n,s,d}$ of the LPL-problem, which is tailored to our particular application. Note that this variant models a situation where the adversary is able to probe *all* $\langle \mathbf{r}, \mathsf{msk}_i \rangle$ and also *all* partial sums $\sum_{i=1}^{k} \langle \mathbf{r}, \mathsf{msk}_i \rangle$ for $k \geq 2$, without being restricted to t probes. We do not model probes on msk_i's, internal values of $\mathbf{R} \cdot \mathsf{msk}_i$ or internal wires of $\mathsf{Refresh}$ here. The latter will be justified in Lemma 4, which shows that these probes do not help the adversary much, provided their number is restricted (the restriction on the number of probes only appears there). We now define the $\mathsf{LPL}_{n,s,d}$ distribution:

For secret $\mathbf{k} \in \mathbb{Z}_2^n$, $d \geq 2$, the $\mathsf{LPL}_{n,s,d}$ sample distribution is given as follows:

1. First, sample $\mathbf{r} \in \mathbb{Z}_2^n$.
2. Set $u = \langle \mathbf{r}, \mathbf{k} \rangle \bmod 2$ and share u into d uniform values $u_i \in \mathbb{Z}_2$ conditioned on $u = \sum_{i=1}^{d} u_i \bmod 2$.
3. For any $2 \leq k \leq d$, we define u'_k as the partial sum $u'_k = \sum_{i=1}^{k} u_i$.
4. Sample independent noise e_k for $1 \leq k \leq d$ and e'_k for $2 \leq k \leq d$, where each e_k, e'_k independently follows Φ_s.
5. Output $\mathbf{r}$ and all $u'_k + e'_k$ and all $u_k + e_k$.

Given query access to $\mathsf{LPL}_{n,s,d}$ for unknown, uniform $\mathbf{k} \in \mathbb{Z}_2^n$, the corresponding $\mathsf{LPL}_{n,s,d}$ search problem is to find $\mathbf{k}$. The decision problem ask to distinguish

$\mathsf{LPL}_{n,s,d}$ for secret, uniform $\mathbf{k}$ from a distribution where u is chosen uniformly in the second step above. The decision/search $\mathsf{LPL}_{n,s,d}$-Assumption asserts that these problems are intractable for PPT algorithms.

Note that in the definition above, we split u into shares rather than $\mathbf{k}$ as in our scheme $\mathit{\Pi}_{\text{noisy}}$. Due due to linearity, this is equivalent, provided $\mathbf{R} \leftarrow \mathcal{D}$ selected in $\mathit{\Pi}_{\text{noisy}}$ is full-rank.

Lemma 3. *Let $d \geq 2$. Then the $\mathsf{LPL}_{n,\sqrt{2}s,d}$-search problem is at least as hard as the search-$\mathsf{LPL}_{n,s}$ problem. More precisely, if search-$\mathsf{LPL}_{n,s}$ is (t,ε)-hard with q samples, then $\mathsf{LPL}_{n,\sqrt{2}s,d}$ is (t',ε')-hard with q samples, where $t \approx t', \varepsilon \approx \varepsilon'$.*

Proof. Note that if the adversary knows all shares u_i for $1 \leq i < d$ (which contain no information about $\mathbf{k}$) except for the last, then the only useful data are $u_d + e_d$ and $u'_d + e'_d$. With the given data, u_d can be computed from u'_d and vice versa. Having two independent noisy samples for the same value u_d is equivalent to reducing the noise by a factor $\sqrt{2}$. See Appendix A for details.

Remark 2. The above shows that we only lose at worst a factor of $\sqrt{2}$ in the noise rate due to the probes for intermediate values. We remark that this is an upper bound on the parameter loss and is not matched by real attacks: the reduction assumes that the u_i for $1 \leq i < d$ are known in clear (without the noise), which in reality is not the case. In fact, the more precise parameter loss is determined by the tail distribution of the R_{Bayes} value for $\mathsf{LPL}_{n,s,d}$, defined as in the proof of Lemma 2. Unfortunately, this distribution is difficult to compute.

We now give an intuition why the parameter loss of $\sqrt{2}$ is an exaggeration. The security level of $\mathsf{LPL}_{n,s}$ and $\mathsf{LPL}_{n,s,d}$ is essentially determined by outliers in those leakages (cfr. Rmk. 1). If we assume in favor of the adversary that we know all u_i but u_d, u_{d-1}, then we know all intermediate values except for u'_{d-1} and $u = u'_d$. We are interested in what can be learned about the latter.

The leakages with noise rate s for u_{d-1} and u'_{d-1} are as good a single leakage for u'_{d-1} with noise rate $s/\sqrt{2}$ by an argument similar to Lemma 3. For the leakage of u_d, we get a noise rate of s, which gives us some information about $u'_d = u_d + u'_{d-1}$. However, by taking the sum, the amount of information (measured by some $R'_{d,\text{Bayes}}$ defined as in the proof of Lemma 2) that we learn about u'_d from the set of all leakages excluding that of u'_d is limited. Indeed, it can be at most as large as what we can learn about (the worse of) u_d and u'_{d-1}.

What we learn about u'_d from all leakages is then determined by R'_{Bayes} and what we learn from the leakage of u'_d directly. Since it is extremely more unlikely that the leakage of u'_{d-1} and u_d are *both* outliers than that the single measurement for u'_d is an outlier, the tail distribution for the information learned is mostly determined by the leakage of u'_d alone. Consequently, we expect to lose almost no security at all by revealing intermediate values. For that reason, we do not include the $\sqrt{2}$-factor in our concrete parameters.

The above definition only models probes on the computation of $\mathbf{u} = \sum_i \mathbf{u}_i$ in our re-keying scheme. It does not include probes for bits of shares $\mathsf{msk}_i, \mathsf{msk}'_i$ of the master key, probes for internal values of $\mathbf{R}\mathsf{msk}_i$'s or probes for interval

wires of Refresh. The following lemma shows that this is indeed adequate, as these additional possibilities do not help the adversary anyway.

Lemma 4. *Consider our re-keying scheme Π_{noisy} with parameters n, d, m. Assume $n - m > \lambda + d$, where λ is the security parameter. Let $2t < d$. Model H as a random oracle. Assume Π_{noisy} is secure in the continuous t-noisy probing model with Gaussian noise s, where the adversary is only allowed to probe bits of the inner products $u_i = \mathbf{R} \cdot \mathsf{msk}_i$ or of partial sums $u'_k = \sum_{i=1}^{k} u_i$ thereof for $k \geq 2$. Then Π_{noisy} is secure in the continuous t-noisy probing model with Gaussian noise $\sqrt{t+1}s$, but without the latter restriction, i.e. when we also allow probes on bits of master key shares msk_i, msk'_i, on bits of internal values of the computation of $u_i = \mathbf{R}\mathsf{msk}_i$ or probes on internal wires of* Refresh.

Proof. By assumption, we are given some simulator *Sim* that simulates answers to bits of $\mathbf{u}_i = \mathbf{R} \cdot \mathsf{msk}_i$ and to bits of partial sums $\sum_{i=1}^{k} \mathbf{u}_i$ thereof. We need to show that we can extend *Sim* to a simulator *Sim′* that also simulates probes on bits of msk_i's,msk'_i's and on internal wires of Refresh and the computations of $\mathbf{u}_i = \mathbf{R}\mathsf{msk}_i$.

Our simulator *Sim′* will use *Sim* for the probes to $\mathbf{u}_i$ or $\sum_{i=0}^{k} \mathbf{u}_i$. To simulate other probes, *Sim′* will fix appropriate bits of some msk_i's or msk'_j's and use these to simulate the missing queries (conditioned on $\mathbf{u}_i$ and $\sum_{i=0}^{k} \mathbf{u}_i$). Let $M_{i,j}$ resp. $M'_{i,j}$ be the jth bit of msk_i resp. msk'_i. For a query to the jth bit of msk_i or msk'_i, we fix the value of $M_{i,j}$ resp. $M'_{i,j}$. For t' probes on internal wires of Refresh, we can fix some $t' \times t'$ submatrix of both M and M' by the properties of Refresh guaranteed by Lemma 1, which allows us to perfectly simulate the desired probes. Further, some bits of $M_{i,j}$ might already be fixed from probes (on the then-called M') from the previous session. Since the number of probes per session is limited by t, all the fixed bits are contained in $2t \times 2t$ submatrices $M_{I,J}, M'_{I,J}$ for $I \subset \{1, \ldots, d\}, J \subset \{1, \ldots, n\}$. Since $|I| < d$, there exists a share on which nothing is fixed. Consequently, the real value of the bits that we fixed to uniform are uniform, and independent from msk. Since $n - m > \lambda + d$, we have that the rows of R, together with the unit vectors corresponding to J are linearly independent with overwhelming probability. Due to that, the fixed bits are independent from both msk and the $\mathbf{u}_i$'s. It follows that the simulation of the probes on M, M' and Refresh can be done perfectly, independent from *Sim*.

What remains is the probes on intermediate values of the computation of $\mathbf{u}_i = \mathbf{R}\mathsf{msk}_i$. Assume that the individual output bits are computed independently as scalar products $\langle \mathbf{R}_j, \mathsf{msk}_i \rangle$ where $\mathbf{R}_j$ is the j-th row of $\mathbf{R}$. Then for a natural implementation, intermediate values correspond to inner products $\langle \mathbf{w}, \mathsf{msk}_i \rangle$, where $\mathbf{w}$ is obtained from some row of $\mathbf{R}$ by zeroing bits. Now, we fix the inner products $\langle \mathbf{w}, \mathsf{msk}_i \rangle$ to a uniformly random value and use that to simulate the probes. (They are completely analogous to coordinates of msk_i). The only thing that may go wrong is that some $\mathbf{w}$ are linearly dependent to previously fixed coordinates of msk_i, where individually fixed bits of $M_{i,j}$ correspond to a unit vector for $\mathbf{w}$. In this case, we need to set the $\langle \mathbf{w}, \mathsf{msk}_i \rangle$ according to the linear combination. If this linear combination involves $\mathbf{w}$, we need to use $\mathbf{u}_i$, which we

do not know. Essentially, in this situation, the adversary is probing the same unknown value c times with independent Gaussian noise, which is equivalent to probing once with a noise width reduced by a factor $\sqrt{c}$.

Formally, we can still simulate the probe responses by using Lemma 7 from Appendix A. □

Theorem 2. *Consider our re-keying scheme Π_{noisy} with parameters n, d, m. Assume $n - m - d > \lambda, 2t < d$ and $\frac{n}{m} = \Theta(1)$. Model H as a random oracle. Then Π_{noisy} is secure in the continuous t-noisy probing model with Gaussian noise $\Phi_{\sqrt{t+1}s}$ under the Search-$\mathsf{LPL}_{n,s,d}$-Assumption.*

Clearly, using Lemma 3 together with Thm. 1, Thm. 2 proves our re-keying scheme Π_{noisy} secure under LPN.

Proof. See Appendix A.

4.2 Concrete parameters

In the previous section we proved our proposed scheme Π_{noisy} secure under the LPN assumption. We target our physical noise s such that the $\mathsf{LPL}_{n,s}$-assumption holds. Note that, as we argued in Rmk. 1, we expect the reduction in Thm. 1 relating $\mathsf{LPL}_{n,s}$ and $\mathsf{LPN}_{n,\tau'}$ to be matched by actual attacks, hence we really need these parameters. By contrast, the loss in the reduction of Thm. 2 is due to technical reasons and we do not expect there to be matching attacks. We argued in Rmk. 2 why the $\sqrt{2}$ loss factor for intermediate values that we obtained in Lemma 3 is far from tight. For the $\sqrt{t+1}$-factor from Lemma 4, one can actually show that it is not there if one uses a binary tree to carry out the computation of the sum in Π_{noisy}. Intuitively, it is better for the adversary to probe values as late in the computation as possible, as the best the adversary can hope is to learn some $\langle \mathbf{r}, \mathsf{msk} \rangle$, which is computed at the end. The $\sqrt{t+1}$-factor came from probes on internal values at the start of the computation. Unfortunately, we cannot prove Lemma 3 with our methods for a binary tree. The argument from Rmk. 2 becomes more complicated, but essentially still holds, which is why we ignore that $\sqrt{t+1}$ factor as well.

Hence, we believe that setting parameters such that the $\mathsf{LPL}_{n,s}$-Problem becomes hard is sufficient for our scheme to be secure.

We follow the proposal of Bogos, Tramèr and Vaudenay [17] of parameter choices (n, τ') for $\epsilon := 2^{-80}$-hard LPN. We can then use the relationship between the number of samples Q, the Gaussian width s and the Bernoulli noise τ' to determine s via. Thm. 1. Concretely, for $Q = 2^{80}$ and $\Delta n = 1$, we have $\mathrm{Fc}^{-1}\left(\frac{\Delta n}{2Q}\right) \approx 10.2846$, which allows us to relate s and τ'.

Note that in the context of side-channel analysis, a further reduction of the data complexity parameter could be considered. A choice of $Q = 2^{40}$ would already imply the capture of $\approx 2^{40}$ leakage traces, which corresponds to weeks of measurements with current acquisition hardware [49]. Since $\mathrm{Fc}^{-1}\left(\frac{\Delta n}{2Q}\right) \approx 7.1436$, this reduces the require noise level by a factor of approximately $\frac{10.28}{7.14} \approx 1.43$. Altogether, this approach leads to the choice of parameters given in Table 1.

Dimension (n)	1280	640	512	448	384	256
LPN noise (τ')	0.05	0.125	0.25	0.325	0.4	0.45
LPL noise (s) for $Q = 2^{80}$	≈ 3.52	≈ 5.31	≈ 9.37	≈ 14.1	≈ 25.4	≈ 51.3
LPL noise (s) for $Q = 2^{40}$	≈ 2.46	≈ 3.70	≈ 6.52	≈ 9.79	≈ 17.6	≈ 35.6

Table 1. Standard deviations required for LPL with 80-bit hardness based on LPN with parameter τ. We assume a bound Q on the number of LPL-samples the adversary may see. To obtain these parameters, we numerically solved the (slightly) better Eq. (3) from Appendix A rather than Eq. (2) from Thm. 1 with $\Delta n = 1$.

Note also that the 80-bit security level of Table 1 corresponds to security against side-channel attacks. Of course, it remains that if no leakage is provided to the adversary, then the security of the re-keying scheme directly relates to the key size of the underlying (tweakable) block cipher.

5 Fresh re-keying without physical noise

For settings where no physical noise is given, or when it is not sufficient to achieve the desired security level, we now give an alternative solution for a fresh re-keying scheme, based on a variant of Learning with Rounding (LWR) assumption that we call Offset Learning with Rounding (OLWR). We will show in Thm. 3 below that for an unbounded amount of samples (OLWR) is at least as hard as Learning with uniform Errors (LWU). Before providing our OLWR-based re-keying scheme Π_{LWR}, we recall our rounding function and the OLWR assumption. For appropriately chosen integers $p < q$, the rounding function $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is $\lfloor x \rceil_p := \lfloor x \frac{p}{q} \rceil$, where $x \in \mathbb{Z}_q$ is represented as $x \in \{0, \ldots, q-1\}$. When applying $\lfloor \cdot \rceil_p$ to a vector in $\mathbb{Z}_q^n$, we apply it component-wise. OLWR samples for dimension n and secret $\mathbf{k} \leftarrow \mathbb{Z}_q^n$ and an adversarialy chosen offset $\mathbf{o} \in \mathbb{Z}_q^n$, which is freshly (and adaptively) chosen for each sample, follow the distribution

$$\mathcal{D}_{\mathsf{OLWR},n,q,p}(\mathbf{o}) := (\mathbf{r}, \lfloor \langle \mathbf{r}, \mathbf{k} + \mathbf{o} \rangle \rceil_p \mid \mathbf{r} \leftarrow \mathbb{Z}_q^n).$$

As usual, given query access to $\mathcal{D}_{\mathsf{OLWR},n,q,p}$ for uniform $\mathbf{k}$, the search $\mathsf{OLWR}_{n,q,p}$ problem asks to find $\mathbf{k}$. The search problem asks to distinguish this distribution from the uniform distribution $\mathcal{D}_{\mathsf{Uniform}}$. Note that the uniform distribution does not depend on the input $\mathbf{o}$. The search/decision $\mathsf{OLWR}_{n,q,p}$ Assumption asserts that this is infeasible for PPT algorithms.

We can define similar offset variants OLWU, OLWE etc. of LWU, LWE etc. where the adversary is allowed to add an (adaptively chosen) offset to $\mathbf{k}$.

For Learning with Errors (LWE) or Learning with Uniform Noise (LWU) it is easy to see that their offset variants are still as hard as LWE, LWU respectively. An adversary can simply compute samples with an arbitrary offset itself using the linearity of LWE and LWU. Since LWR is not linear this does not work for LWR. The reduction given in [7] from LWE to LWR also works from offset LWE to OLWR. Unfortunately it does not work for the parameters proposed in this work. Assuming the hardness of LWU for an unbounded amount of samples, OLWR will be also hard, as we show in the following theorem. A similar statement for their non-offset variants using similar techniques was shown by Bogdanov et. al. [16].

Theorem 3 (Relationship between $\mathsf{LWU}, \mathsf{OLWU}$ and OLWR).

(a) For both the search and decision variants, $\mathsf{LWU}_{n,q,B}$ and $\mathsf{OLWU}_{n,q,B}$ are equivalent via a tight, sample-preserving reduction.

(b) Assume $p \mid q$. Then hardness of search resp. decision $\mathsf{LWU}_{n,q,B}$ implies hardness of search resp. decision $\mathsf{OLWR}_{n,q,p}$, where $B = \frac{q}{p}$.
More precisely, if we can solve the search resp. decision $\mathsf{OLWR}_{n,q,p}$-problem with advantage ε in time T, using Q samples, then we can solve the search resp. decision $\mathsf{LWU}_{n,q,B}$ problem with advantage $\varepsilon' = \varepsilon$ in expected time $T + \mathcal{O}(QB)$ using an expected $Q' = QB$ samples.

Proof. (a) follows immediately by linearity, as the offset $\mathbf{o}$ just adds $\langle \mathbf{R}, \mathbf{o} \rangle$, which is known. For (b), we show that we can transform $\mathsf{OLWU}_{n,q,B}$-samples for secret $\mathbf{k}$ into $\mathsf{OLWR}_{n,q,p}$ samples for the same unknown secret $\mathbf{k}$. By (a), this implies the claim.

To this end, suppose our reduction has to produce a (simulated) $\mathsf{OLWR}_{n,q,p}$ sample with offset $\mathbf{o}$. Then we repeatedly query $(\mathbf{r}, \ell_{\mathsf{OLWU}}) \leftarrow \mathsf{OLWU}_{n,q,B}(\mathbf{o})$ until $\ell_{\mathsf{OLWU}} + 1$ is divisble by B. We then output $(\mathbf{r}, \lfloor \ell_{\mathsf{OLWU}} \rceil_p)$.

To analyse the output distribution, recall that $\ell_{\mathsf{OLWU}} = \langle \mathbf{r}, \mathbf{k} + \mathbf{o} \rangle + e$ for $0 \leq e < B$. It follows that $\langle \mathbf{r}, \mathbf{k} + \mathbf{o} \rangle \in \{\ell_{\mathsf{OLWU}}, \ell_{\mathsf{OLWU}} - 1, \ldots, \ell_{\mathsf{OLWU}} - B + 1\}$. The condition that $\ell_{\mathsf{OLWU}} + 1$ is divisble by B is equivalent that *all* possible values for $\langle \mathbf{r}, \mathbf{k} + \mathbf{o} \rangle$ map to the same value under $\lfloor . \rceil_p$.

Furthermore, note that the probability to reject an $\mathsf{OLWU}_{n,q,B}$ sample is always $\frac{1}{B}$, independent from $\mathbf{r}, \mathbf{k}, \mathbf{o}$, as it can be viewed as a condition on $e \leftarrow \{0, \ldots, B-1\}$ alone. It follows that we output the correct distribution. □

Unfortunately Theorem 3 does not work the ring version $\mathsf{Ring\text{-}OLWR}$ of OLWR. In this paper we choose parameter for $\mathsf{Ring\text{-}OLWR}$ such that ring LWR and ring LWU would be hard, even though there is no reduction to $\mathsf{Ring\text{-}OLWR}$ known for a polynomial modulus and their decisional variant which we will use.

5.1 Offset LWR-based Re-Keying

Our proposed re-keying scheme Π_{LWR} based on $\mathsf{OLWR}_{n,q,p}$ is defined as follows: The session keys are in $\mathbb{Z}_{p'}^m$ for $p' \mid p$ and the distribution $\mathcal{D}$ for $\mathbf{R}$ is the uniform distribution over $\mathbb{Z}_q^{m \times n}$.

Gen(1^λ): Samples $\mathsf{msk} \leftarrow \mathbb{Z}_q^n$. Create d shares $(\mathsf{msk})_d := \mathsf{msk}_1, \ldots, \mathsf{msk}_d$ such that $\mathsf{msk} = \sum_{i=1}^d \mathsf{msk}_i$, uniformly among all possibilities. Output msk and $(\mathsf{msk})_d$.

GenSK$((\mathsf{msk})_d, \mathbf{R})$: For each of the shares msk_i, compute $\mathsf{sk}_i := \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p$. Then set $\mathsf{sk} := \left\lfloor \sum_{i=1}^d \mathsf{sk}_i \right\rceil_{p'}$. Define the error correction information as $v := \sum_{i=1}^d \mathsf{sk}_i \bmod p/p'$. Finally, use a secure refresh operation $\mathsf{Refresh}$ as in Lemma 1 to refresh the shares $(\mathsf{msk}')_d \leftarrow \mathsf{Refresh}((\mathsf{msk})_d)$. Output sk, $(\mathsf{msk}')_d$ and v.

CorSK$(\mathsf{msk}, \mathbf{R}, v)$: Computes $y := \lfloor \mathbf{R} \cdot \mathsf{msk} \rceil_p$ and $z := y + (v - y \bmod p/p') \bmod p$. Output $\lfloor z \rceil_{p'}$.

Theorem 4. *Let $n, q, p, p', d \in \mathbb{N}$ such that $q > p > p'$, $p/p' > d$ and $p' \mid p$, $p \mid q$. Then the $\mathsf{OLWR}_{n,q,p}$-based re-keying scheme is correct.*

Proof. First notice for $p/p' > d$ that the error term

$$e := \sum_{i=1}^{t} \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p - \lfloor \mathbf{R} \cdot \mathsf{msk} \rceil_p$$

is bounded in each component by: $0 \leq |e| \leq p/p' - 1$. This follows directly from the fact that a value strictly smaller than 1 is rounded away per round operation, and $\mathsf{msk} = \sum_i \mathsf{msk}_i$, which implies $\mathbf{R} \cdot \mathsf{msk} = \sum_i \mathbf{R} \cdot \mathsf{msk}_i$. Next, $p' \mid p$ guarantees that any potential error component corresponds to a uniquely determined coset mod p/p'. Hence, we further have:

$$\begin{aligned} v - y \bmod p/p' :&= (\sum_i \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p \bmod p/p') - (\lfloor \mathbf{R} \cdot \mathsf{msk} \rceil_p \bmod p/p') \bmod p/p' \\ &= (\lfloor \langle \mathbf{R} \cdot \mathsf{msk} \rangle \rceil_p + e \bmod p/p') - (\lfloor \mathbf{R} \cdot \mathsf{msk} \rceil_p \bmod p/p') \bmod p/p' \\ &= e. \end{aligned}$$

Therefore, we have:

$$\begin{aligned} z' :&= y + (v - y \bmod p/p') \bmod p = \mathbf{R} \cdot \mathsf{msk} + e \bmod p \\ &= \sum_i \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p \bmod p = \mathsf{sk}. \end{aligned}$$

□

Theorem 5. *For moduli $p, q \in \mathbb{N}$ with $p \mid q$ and dimension $n \in \mathbb{N}$, the proposed re-keying scheme Π_{LWR} is secure under the $\mathsf{OLWR}_{n,q,p}$ assumption in the $2t < d$ probing model.*

Proof. We assume that *GenSK* just outputs $\sum_{i=1}^{d} \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p$ which is equivalent to outputting $\mathsf{sk} = \left\lfloor \sum_{i=1}^{d} \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p \right\rceil_{p'}$ and $v = \sum_{i=1}^{d} \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p \mod p/p'$. Therefore *Ideal* will simply output $\mathbf{R}, \mathsf{sk}'$ where $\mathsf{sk}' \leftarrow \mathbb{Z}_p^m$. *Sim* simulates t probes as follows:

- Let msk_i denote the shares at the beginning of the session for $i \in [d]$. *Sim* receives t probe requests and forwards the probes which affect the refresh procedure to the simulator for the refresh procedure. This simulator will w.l.o.g. respond with the $d - 1$ input shares msk_i of the refresh procedure and the probes which were targeted within the refreshing procedure. The output shares of the refresh procedure are not accessed by *Sim* during this session, but the next session after the probes were made.

– *Sim* computes for the $d-1$ shares $\mathsf{sk}_i := \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p$. Let msk_j be the share that is missing and that has not been directly targeted by any probe request. *Sim* defines $\mathsf{sk}_j = \mathsf{sk}' - \sum_{i \neq j} \mathsf{sk}_i$. Since all sk_i are known, *Sim* can answer any probe request on any the intermediate values which arise when computing $\sum_{i=1}^{d} \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p$.

Now we show the following statement using a reduction to OLWR: for the given simulator *Sim*, for any PPT A with

$$|\Pr[A^{Real(\mathsf{msk})}(1^\lambda) = 1] - \Pr[A^{(Sim^{Ideal_c}, Ideal_c)}(1^\lambda) = 1]| = \varepsilon,$$

there is an algorithm D distinguishing OLWR with probability ε from uniform.

During each session, A requests t probes to D. It calls the simulator of the refresh scheme and receives $d-1$ shares msk_i, but not msk_j as it has been the case for *Sim*. Note that A has already sent all probe requests for the shares msk_i at the beginning of this and the previous session. Therefore D can identify a index j of a share msk_j which will never be requested by A. D requests a sample $(\mathbf{R}, \ell)$ for offset $-\sum_{i \neq j} \mathsf{msk}_i$ and has to decide in the end whether all of the samples $(\mathbf{R}, \ell)$ that D collects over the sessions are OLWR or uniformly distributed. D defines $\mathsf{sk}_j = \ell$ and computes $\mathsf{sk}_i = \lfloor \mathbf{R} \cdot \mathsf{msk}_i \rceil_p$ for all $i \neq j$. Now D can similar to *Sim* respond to all the probe requests using sk_i. D computes the session key sk and v simply by computing $\sum_{i=1}^{d} \mathsf{sk}_i$. Afterwards he outputs $\left(\mathbf{R}, \sum_{i=1}^{d} \mathsf{sk}_i\right)$ to finish the current session. After finishing all sessions D outputs the output bit of A.

Let us assume that the samples $\mathbf{R}, \ell$ are OLWR distributed. The offset is $-\sum_{i \neq j} \mathsf{msk}_i$ such that $\ell := \left\lfloor \mathbf{R}(\mathsf{msk} - \sum_{i \neq j} \mathsf{msk}_i) \right\rceil_p = \left\lfloor \mathbf{R} \cdot \mathsf{msk}_j \right\rceil_p$. Hence D successfully simulates *Real*.

If that $(\mathbf{R}, \ell)$ is uniform, then D simulates the output of *Ideal* by outputting $\left(\mathbf{R}, \ell + \sum_{i \neq j} \mathsf{sk}_i\right)$. This is clearly uniformly random for uniform ℓ. Further $\mathsf{sk}_j := \ell$ used by D is the same as *Sim* would have computed: $\mathsf{sk}_j = \mathsf{sk}' - \sum_{i \neq j} \mathsf{sk}_i = \ell + \sum_{i \neq j} \mathsf{sk}_i - \sum_{i \neq j} \mathsf{sk}_i = \ell$. All other sk_i are derived from the $d-1$ outputs of the simulator of the key refreshing and hence have the same distribution. Therefore D simulates *Ideal* and *Sim* perfectly. □

We emphasize that the presented results directly translate to the ring setting with Ring-OLWR. The reason for this is that the error correction of Π_{LWR}, the rounding $\lfloor \cdot \rceil_p$, and the addition in the ring are carried out component-wise which is sufficient for both correctness and security.

5.2 Concrete parameters

Historically, LWR has always been understood as a deterministic variant of LWE where the noise is rounded away applying the rounding function $\lfloor \cdot \rceil_p$ to a LWE sample. This technique was used by Banerjee, Peikert and Rosen to reduce LWE to LWR [7]. Unfortunately their reduction only holds for a superpolynomial modulus q. This was improved by Alwen et al. [3] by using lossy pseudorandom

samples. They achieve a reduction for a polynomial modulus q but with the drawback of a polynomially bounded amount of samples.

Because of these issues, the previous PRF construction by Banerjee et al. (called SPRING) [8] is based on parameters which are not obtained by choosing parameters for a hard instance of LWE and using one of the known reductions to LWR to get appropriate parameters for LWR. Interestingly, and despite their choice of parameters is not based on LWE, it seems that the best way to solve LWR for such a choice of parameters is still to exploit algorithms designed to solve LWE. We follow a similar approach, but using a different choice of parameters. Banerjee et al. propose two parameter choices for Ring-LWR, namely for $n = 128$ and $p = 2$ they choose either $q = 257$ or $q = 514$. For our application $p = 2$ is not sufficient, since we need $p/p' = \lceil \log(d) \rceil$ to correct errors in our rekeying scheme Π_{LWR} and $p > p'$. This means that for masking with up to 4 shares, we need at least $p \geq 4$.

Table 2 shows our choices of parameters for LWR, LWU and Ring-LWR, Ring-LWU which we will also use to instantiate OLWR and Ring-OLWR. The absolute noise level $\log q - \log p$ is large compared with modulus q. Furthermore, our secret msk is picked uniformly from $\mathbb{Z}_q^n$. Together, this will rule out the BKW algorithm and its variants [14, 39]. The same holds for the algorithm of Arora and Ge [5]. Besides, for comparably small dimensions n, good lattice reductions exist, but since the noise is large, shortest vector sieving requires to find a vector of very small norm which seems to be hard. The size of n is compensated by the large noise to modulus ratio. Such a LWR or OLWR instance can also be seen as a hidden number problem.

Assumption	Dimension n	Modulus $\log q$	Modulus $\log p$
LWR,LWU	128	16	4
LWR,LWU	128	32	10
Ring-LWR, Ring-LWU	128	16	2
Ring-LWR, Ring-LWU	128	32	3

Table 2. 128-bit security parameters for our re-keying scheme Π_{LWR}. For the ring version we use a irreducible polynomial $f \in \mathbb{Z}_q[X]/f$ with $\mathsf{deg} = n$. $\log q - \log p$ corresponds to the absolute noise level. The uniform noise for LWU is bounded by q/p. All our moduli are powers of two, which will guarantee that the output of the rounding function $\lfloor \cdot \rceil_p$ is uniform in $\mathbb{Z}_p$ for a uniform input in $\mathbb{Z}_q$.

For other concrete parameters, Albrecht, Player and Scott survey how algorithms for solving LWE perform and give estimated running times [2]. These estimates on the running times affirm our choice of parameters. In the LWR setting, the LWE standard deviation corresponds roughly to $\frac{q}{p}$. As for Ring-LWR, we choose the parameters more conservatively, since for our ring over $\mathbb{Z}_{2^{16}}$ or $\mathbb{Z}_{2^{32}}$, it is likely that the ring product $\mathbf{R}$ msk of secret msk and public sample vector $\mathbf{R}$ lies within an ideal of the ring. This might leak some information about the noise vector and the product might not depend on all the bits of msk.

Note finally that we chose a 128-bit security level which seem most relevant for general purpose applications. Reducing the security level to 80 could be ac-

ceptable for low-cost applications of Π_{LWR}. A simple (and conservative) solution for this purpose would be to reduce n from 128 to 80. By contrast, reducing the security parameters further (e.g., down to 40 bits) is not possible as in Section 4.2. Indeed, there is no guarantee that the attacks against our construction would require large data complexity (which is the only quantity that can be reasonably reduced in the context of side-channel attacks).

6 Implementation results

In order to confirm the efficiency of our constructions, we implemented them on a 32-bit ARM7 device and on an 8-bit Atmel AVR device, for the standard parameters that we would select for concrete applications. For the LPL-based re-keying, we choose $n = 512$ and for the wPRF-based re-keying we choose $n = 128$, $q = 2^{32}$, $p = 2^{10}$ and p' ranging from 4 to 16 depending on the number of shares considered for masking. We then compared our implementation results with the ones obtained for the AES in [33], where higher-order masked implementations are evaluated on an Atmel AVR device. For illustration, we further extrapolated the cycle counts of the masked AES on the ARM7 device as 4 times lower than for the Atmel ones (since moving from an 8-bit to a 32-bit architecture). Note that in all cases, we refreshed the master key with a simple (linear) refreshing algorithm based on the addition of a vector of shares summing to zero, and we assumed a cost of 10 clock cycles to generate each byte of fresh randomness. This is consistent with the approach used in earlier re-keying papers [30, 46]. More generally, and as for our selection of security parameters, such implementations correspond to the best tradeoff between security against state-of-the-art attacks and efficiency, that we suggest for further concrete investigation.

These performance evaluations are reported in Figure 3 from which we can extract a number of interesting observations. First, and as expected, the cycle counts of our new constructions scale linearly in the number of shares, with small discontinuities for the wPRF-based re-keying (corresonding to the addition of one bit for error correction, each time $\lceil \log(d) \rceil$ increases). Second, the performances of the LPL-based re-keying and wPRF-based re-keying are similar on a 32-bit ARM device (where the multiplication is easy both in $\mathbb{Z}_2$ and $\mathbb{Z}_q$). They more significantly differ in the Atmel AVR case, because inner product operations in $\mathbb{Z}_2$ only involve simple (AND and XOR) operations, and directly lead to efficient implementations on this platform. By contrast, the wPRF-based re-keying on the Atmel device implies additional overheads for the 32-bit multiplication based on 8-bit operations (which takes approximately 40 clock cycles). Third, comparisons with the AES shows that (as expected as well), the interest of our new constructions increases with the number of shares (hence security level). In this respect, it is important to note that a masked software implementations of the AES protected with Boolean masking will generally suffer from independence issues, e.g. the recombination of the shares due to transition-based leakages [22]. Since our (almost) key homomorphic conctructions do not suffer

from this risk (because we can manipulate the shares independently), comparisons with this curve are overly conservative. As a quite optimistic comparison point, we can observe the cost of the "glitch-free" software implementations proposed in [55] which is already higher than the one of our primitives for 2 shares.[8] Alternatively, one can also use the simple reduction in [6] and double the number of shares of the masking scheme to obtain similar security, which also makes our masked re-keying schemes more efficient than the corresponding masked AES with 3 shares. Quite naturally, these comparisons are only informal. Yet, they illustrate the good implementation properties of fresh-rekeying. In this respect, the simplicity of the implementations, and limited constraints regarding the need of independent leakages, are certainly two important advantages.

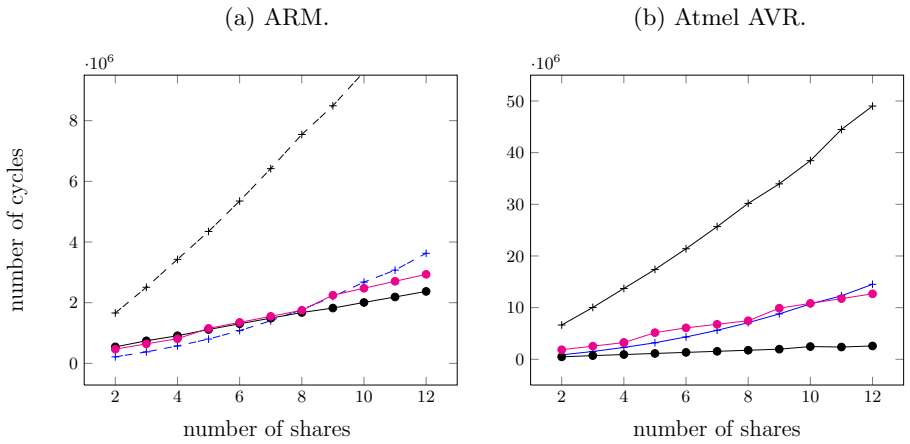

Fig. 3. Performance comparisons on ARM and Atmel AVR devices assuming 10 clock cycles per random byte. LPL-based re-keying —●—, wPRF-based re-keying: —●—, AES Boolean masking: –+–, AES glitch-free masking: –+–. Dashed ARM curves are extrapolated by scaling down the corresponding Atmel AVR performances by 4.

Finally, a key difference between the LPL- and wPRF-based re-keying is that the latter one offers significantly stronger guarantees (since it is secure even in front of noise-free leakages), which explains its lower performances. By contrast, the LPL-based re-keying implementations have to include noise in the adversary's measurements. Hence, we end this section with a brief discussion of the noise levels required for secure LPL implementations, and how to generate them. For this purpose, a conservative estimate is to assume that this noise will be generated thanks to additive algorithmic noise. Typically, this could imply implementing a parasitic Linear Feedback Shift Register (LFSR) in parallel to the inner product computations to which we have to add noise. Since the noise variance corresponding to 1 bit equals 0.25, we typically need an LFSR of size

[8] Note that this curve is linear which corresponds to the amortized complexity of the best "packed secret sharing" at each order.

$N = \lceil 4 \times \sigma^2 \rceil$ to reach our estimated security levels.[9] For illustration, some numbers are given in Table 3, where we can see a tradeoff between the cost of computing the inner products and the cost of generating the noise. So already for these estimates, we see that the $n = 640$ and $n = 512$ instances should allow efficient implementations. Yet and importantly, in case more efficient noise engines are embedded on chip (based on supply noise, clock jitter, shuffling, ...), these figures can only become more positive for our re-keying, and the same holds if some parallelism is considered for the inner product computations (in which case the cost of noise generation will be amortized).

Dimension (n)	1280	640	512	448	384	256
Bits of additive noise ($Q = 2^{80}$)	49	112	361	807	2601	10744
Bits of additive noise ($Q = 2^{40}$)	24	55	177	384	1272	5269

Table 3. Concrete parameters for noise generation.

So overall, LPL-based re-keying is conceptually interesting since it leverages the intrinsic noise that is anyway present in side-channel measurements. But the noise levels that we require to reach high security levels are admittedly larger than these intrinsic noise levels. So it leads to interesting design challenges regarding the tradeoff between the cost of noise generation vs. the cost of inner product computations. By contrast, wPRF-based re-keying is more conservative, since based on a stronger cryptographic primitive that requires less assumptions, at the cost of reasonable performance overheads. The combination of these solutions therefore brings an interesting toolbox to cryptographic engineers, for secure and efficient cryptographic implementations in software and hardware.

Acknowledgements

Stefan Dziembowski is supported by the Foundation for Polish Science WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy Operational Programme. Sebastian Faust is funded by the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG). Gottfried Herold is funded by the ERC grant 307952 (acronym FSC). Anthony Journault is funded by the INNOVIRIS project SCAUT. Daniel Masny is supported by the DFG Research Training Group GRK 1817/1. François-Xavier Standaert is a research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). His work was funded in parts by the ERC project 280141 (acronym CRASH) and the ARC project NANOSEC.

[9] This assumes that the computation of every bit requires a similar amount of energy, which is usually observed in practice [56], and certainly holds to a good extent when considering blocks of bits as we do. The proposed values should anyway only be taken as an indication that generating the required amount of noise is feasible with existing hardware. Besides, note that for such an "algorithmic" noise generated by LFSR, the Gaussian distribution is ensured by design which avoids any risk related to faster decreasing tails.

References

[1] M. Abdalla, S. Belaïd, and P. Fouque. "Leakage-Resilient Symmetric Encryption via Re-keying". In: *CHES*. 2013.

[2] M. R. Albrecht, R. Player, and S. Scott. "On the concrete hardness of Learning with Errors." In: *J. Mathematical Cryptology* 3 (2015).

[3] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. "Learning with Rounding, Revisited - New Reduction, Properties and Applications". In: *CRYPTO*. 2013.

[4] M. Andrychowicz, S. Dziembowski, and S. Faust. "Circuit Compilers with $O(1/\log(n))$ Leakage Rate". In: *EUROCRYPT*. 2016.

[5] S. Arora and R. Ge. "New Algorithms for Learning in Presence of Errors". In: *ICALP*. 2011.

[6] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F. Standaert. "On the Cost of Lazy Engineering for Masked Software Implementations". In: *CARDIS*. 2014.

[7] A. Banerjee, C. Peikert, and A. Rosen. "Pseudorandom Functions and Lattices". In: *EUROCRYPT*. 2012.

[8] A. Banerjee, H. Brenner, G. Leurent, C. Peikert, and A. Rosen. "SPRING: Fast Pseudorandom Functions from Rounded Ring Products". In: *FSE*. 2014.

[9] S. Belaïd, P. Fouque, and B. Gérard. "Side-Channel Analysis of Multiplications in GF(2128) - Application to AES-GCM". In: *ASIACRYPT*. 2014.

[10] S. Belaïd, V. Grosso, and F. Standaert. "Masking and leakage-resilient primitives: One, the other(s) or both?" In: *Cryptography and Communications* 1 (2015).

[11] S. Belaïd, J. Coron, P. Fouque, B. Gérard, J. Kammerer, and E. Prouff. "Improved Side-Channel Analysis of Finite-Field Multiplication". In: *CHES*. 2015.

[12] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. "A More Efficient AES Threshold Implementation". In: *AFRICACRYPT*. 2014.

[13] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. "Higher-Order Threshold Implementations". In: *ASIACRYPT*. 2014.

[14] A. Blum, A. Kalai, and H. Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model". In: *ACM STOC*. 2000.

[15] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. "Cryptographic Primitives Based on Hard Learning Problems". In: *CRYPTO*. 1993.

[16] A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen. "On the Hardness of Learning with Rounding over Small Modulus". In: *TCC*. 2016.

[17] S. Bogos, F. Tramèr, and S. Vaudenay. "On Solving Lpn using BKW and Variants". In: *IACR Cryptology ePrint Archive* (2015).

[18] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. "Key Homomorphic PRFs and Their Applications". In: *CRYPTO*. 2013.

[19] H. Brenner, L. Gaspar, G. Leurent, A. Rosen, and F. Standaert. "FPGA Implementations of SPRING - And Their Countermeasures against Side-Channel Attacks". In: *CHES*. 2014.

[20] S. Chari, J. R. Rao, and P. Rohatgi. “Template Attacks”. In: *CHES*. 2002.
[21] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. “Towards Sound Approaches to Counteract Power-Analysis Attacks”. In: *CRYPTO*. 1999.
[22] J. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala. “Conversion of Security Proofs from One Leakage Model to Another: A New Issue”. In: *COSADE*. 2012.
[23] C. Dobraunig, F. Koeune, S. Mangard, F. Mendel, and F. Standaert. “Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security”. In: *CARDIS* (2015).
[24] Y. Dodis and K. Pietrzak. “Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks”. In: *CRYPTO*. 2010.
[25] N. Döttling and J. Müller-Quade. “Lossy Codes and a New Variant of the Learning-With-Errors Problem”. In: *EUROCRYPT*. 2013.
[26] A. Duc, S. Dziembowski, and S. Faust. “Unifying Leakage Models: From Probing Attacks to Noisy Leakage”. In: *EUROCRYPT*. 2014.
[27] A. Duc, S. Faust, and F. Standaert. “Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device”. In: *EUROCRYPT*. 2015.
[28] F. Durvaux, F. Standaert, and N. Veyrat-Charvillon. “How to Certify the Leakage of a Chip?” In: *EUROCRYPT*. 2014.
[29] S. Dziembowski and K. Pietrzak. “Leakage-Resilient Cryptography”. In: *IEEE FOCS*. 2008.
[30] B. Gammel, W. Fischer, and S. Mangard. *Generating a Session Key for Authentication and Secure Data Transfer*. US Patent App. 14/074,279. Nov. 2013.
[31] L. Gaspar, G. Leurent, and F. Standaert. “Hardware Implementation and Side-Channel Analysis of Lapin”. In: *CT-RSA*. 2014.
[32] O. Goldreich, H. Krawczyk, and M. Luby. “On the Existence of Pseudorandom Generators”. In: *SIAM J. Comput.* 6 (1993).
[33] V. Grosso, F. Standaert, and S. Faust. “Masking vs. multiparty computation: how large is the gap for AES?” In: *J. Cryptographic Engineering* 1 (2014).
[34] T. Güneysu and A. Moradi. “Generic Side-Channel Countermeasures for Reconfigurable Devices”. In: *CHES*. 2011.
[35] Q. Guo and T. Johansson. *A New Birthday-Type Algorithm for Attacking the Fresh Re-Keying Countermeasure*. Cryptology ePrint Archive, Report 2016/225. 2016.
[36] S. Heyse, E. Kiltz, V. Lyubashevsky, C. Paar, and K. Pietrzak. “Lapin: An Efficient Authentication Protocol Based on Ring-LPN”. In: *FSE*. 2012.
[37] Y. Ishai, A. Sahai, and D. Wagner. “Private Circuits: Securing Hardware against Probing Attacks”. In: *CRYPTO*. 2003.
[38] E. Kiltz and K. Pietrzak. “Leakage Resilient ElGamal Encryption”. In: *ASIACRYPT*. 2010.
[39] P. Kirchner and P. Fouque. “An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices”. In: *CRYPTO*. 2015.

[40] M. Liskov, R. L. Rivest, and D. Wagner. "Tweakable Block Ciphers". In: *J. Cryptology* 3 (2011).
[41] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. 2007.
[42] S. Mangard, T. Popp, and B. M. Gammel. "Side-Channel Leakage of Masked CMOS Gates". In: *CT-RSA*. 2005.
[43] S. Mangard, N. Pramstaller, and E. Oswald. "Successfully Attacking Masked AES Hardware Implementations". In: *CHES*. 2005.
[44] D. P. Martin, E. Oswald, M. Stam, and M. Wójcik. "A Leakage Resilient MAC". In: *IMACC*. 2015.
[45] M. Medwed, C. Petit, F. Regazzoni, M. Renauld, and F. Standaert. "Fresh Re-keying II: Securing Multiple Parties against Side-Channel and Fault Attacks". In: *CARDIS*. 2011.
[46] M. Medwed, F. Standaert, J. Großschädl, and F. Regazzoni. "Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices". In: *AFRICACRYPT*. 2010.
[47] M. Medwed, F. Standaert, and A. Joux. "Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs". In: *CHES*. 2012.
[48] D. Micciancio and C. Peikert. "Hardness of SIS and LWE with Small Parameters". In: *CRYPTO*. 2013.
[49] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. "Pushing the Limits: A Very Compact and a Threshold Implementation of AES". In: *EUROCRYPT*. 2011.
[50] S. Nikova, V. Rijmen, and M. Schläffer. "Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches". In: *J. Cryptology* 2 (2011).
[51] O. Pereira, F. Standaert, and S. Vivek. "Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives". In: *ACM CCS*. 2015.
[52] C. Petit, F. Standaert, O. Pereira, T. Malkin, and M. Yung. "A block cipher based pseudo random number generator secure against side-channel key recovery". In: *ASIACCS*. 2008.
[53] E. Prouff and M. Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *EUROCRYPT*. 2013.
[54] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *ACM STOC*. 2005.
[55] T. Roche and E. Prouff. "Higher-order glitch free implementation of the AES using Secure Multi-Party Computation protocols - Extended version". In: *J. Cryptographic Engineering* 2 (2012).
[56] W. Schindler, K. Lemke, and C. Paar. "A Stochastic Model for Differential Side Channel Cryptanalysis". In: *CHES*. 2005.
[57] F. Standaert, O. Pereira, Y. Yu, J. Quisquater, M. Yung, and E. Oswald. "Leakage Resilient Cryptography in Practice". In: *Towards Hardware-Intrinsic Security - Foundations and Practice*. 2010.

[58] Y. Yu and F. Standaert. "Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness". In: *CT-RSA 2013*. 2013.

A Details for the Security Proofs for LPL

In this section, we will in the missing details from the security proofs of Sec. 4.1, notably Lemma 2, Thm. 1, Thm. 2 and Thm. 3.

We begin with the proof of Lemma 2, which we repeat here for reference:

Lemma 5. *The search resp. decision* $\mathsf{LPL}_{n,s}$*-problem is equivalent to the search resp. decision* $\mathsf{LPN}_{n,\Psi}$*-problem via a tight, sample-preserving reduction. Here, the distribution for* Ψ *is given by sampling* $\widetilde{U} \leftarrow \{0,1\}, \widetilde{E} \leftarrow \Phi_s, \widetilde{L} = U+E$, $\widetilde{R}_{>1} := \exp\Big(\frac{\left|L-\frac{1}{2}\right|}{s^2}\Big)$ *and outputting* $\widetilde{\tau} = \big(\widetilde{R}_{>1}+1\big)^{-1}$.

Proof. To relate $\mathsf{LPL}_{n.s}$ and $\mathsf{LPN}_{n,\Psi}$, it is convenient to introduce common probability spaces, modeling both LPL and LPN. We define two probability spaces Ω, Ω' for fixed $\mathbf{k}$. We use capital letters for random variables (except for τ). The definition is done in such a way that Ω models both LPL and LPN, whereas Ω' models the corresponding $\mathcal{D}_{\mathsf{UniformL}}$ and $\mathcal{D}_{\mathsf{Uniform},\Psi}$. Note that their definitions only differ in the second step:

1. Sample $\mathbf{R} \leftarrow \mathbb{Z}_2^n$ uniformly. Sample $E_{\mathsf{LPL}} \leftarrow \Phi_s$.
2. In the probability space Ω, set $U := \langle \mathbf{R}, \mathbf{k} \rangle \bmod 2$.
   In the probability space Ω', sample $U \in \mathbb{Z}_2$ uniform.
3. Set $L_{\mathsf{LPL}} := U + E_{\mathsf{LPL}}$.
4. Set $P_i := \frac{\Phi_s(L_{\mathsf{LPL}}-i)}{\Phi_s(L_{\mathsf{LPL}})+\Phi_s(L_{\mathsf{LPL}}-1)} = \Pr_{\widetilde{L}_{\mathsf{LPL}},\widetilde{U}}[\widetilde{U} = i \mid \widetilde{L}_{\mathsf{LPL}} = L_{\mathsf{LPL}}]$ for $i = 0, 1$, where $(\widetilde{L}_{\mathsf{LPL}}, \widetilde{U})$ follow the same distribution as[10] (L_{LPL}, U) do in Ω'.
5. Set $R_{\mathrm{Bayes}} := \frac{P_0}{P_1}$.
6. If $R_{\mathrm{Bayes}} > 1$, set $R_{\mathrm{Bayes},>1} = R_{\mathrm{Bayes}}$, otherwise $R_{\mathrm{Bayes},>1} = R_{\mathrm{Bayes}}^{-1}$.
7. Set $\tau = \min(P_0, P_1)$.
8. If $P_0 > P_1$, set $L_{\mathsf{LPN}} = 0$.
   If $P_0 < P_1$, set $L_{\mathsf{LPN}} = 1$.
   If $P_0 = P_1$, choose $L_{\mathsf{LPN}} \in \mathbb{Z}_2$ uniform. (only happens with probability 0)
9. Set $E_{\mathsf{LPN}} = L_{\mathsf{LPN}} - U \bmod 2$.

We now make some observations about these probability spaces: first, note that $R_{\mathrm{Bayes}} = \exp\big(\frac{E_{\mathsf{LPL}}+U-\frac{1}{2}}{s^2}\big)$. Hence, $R_{\mathrm{Bayes},>1}$ is distributed independently from U, due to symmetry of Φ_s. In particular, $R_{\mathrm{Bayes},>1}$ is distributed as $\widetilde{R}_{>1}$. Further, as $P_1 = 1 - P_0$, we have $\tau = (R_{\mathrm{Bayes},>1}+1)^{-1}$. In particular, τ is independent from U and follows Ψ.

Next, note that the conditional distribution of $P_i \mid U = j$ is the same as that of $P_{1-i} \mid U = 1-j$. As a consequence, E_{LPN} is independent from $(U, \mathbf{R})$. In particular, its distribution is the same in both Ω and Ω'.

[10] We keep L_{LPL} fixed and use an independent copy of the probability space Ω' for this definition. We always use Ω', not Ω.

We now compute its distribution, conditioned on L_{LPL}, i.e. we compute $\Pr[E_{\mathsf{LPN}} = 1 \mid L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}]$ as a function of ℓ_{LPL}. This does not depend on whether we use Ω or Ω', so we use Ω' in our computation:

$$
\begin{aligned}
&\Pr[E_{\mathsf{LPN}} = 1 \mid L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = && \text{(independence from } U\text{)}\\
&= \Pr[E_{\mathsf{LPN}} = 1 \mid U = 0, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = && \text{(Def. of } E_{\mathsf{LPN}}\text{)}\\
&= \Pr[L_{\mathsf{LPN}} \neq U \mid U = 0, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = && \text{(Def. of } L_{\mathsf{LPN}}\text{)}\\
&= \Pr[P_0 < P_1 \mid U = 0, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = && \text{(Def. of } \tau\text{)}\\
&= \Pr[P_0 = \tau \mid U = 0, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = && \text{(Bayes' rule)}\\
&= \Pr[U = 0 \mid P_0 = \tau, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] \cdot \frac{\Pr[P_0 = \tau \mid L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}]}{\Pr[U = 0 \mid L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}]} = &&\\
&\quad (U, L_{\mathsf{LPL}} \text{ are independent}) &&\\
&= \Pr[U = 0 \mid P_0 = \tau, L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] \cdot \frac{1/2}{1/2} = && \text{(Def. of } P_0\text{)}\\
&= \Pr[U = 0 \mid L_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}, \Pr_{\widetilde{L}_{\mathsf{LPL}}, \widetilde{U}}[\widetilde{U} = 0 \mid \widetilde{L}_{\mathsf{LPL}} = \ell_{\mathsf{LPL}}] = \tau] \cdot \frac{1/2}{1/2} = \tau &&
\end{aligned}
$$

Note that in the last step, we condition on the event that the very value we want to compute is τ. Since τ is a function of L_{LPL}, it follows that $\Pr[E_{\mathsf{LPL}} = 1 \mid \tau] = \tau$. This shows that in Ω, the triple $(\mathbf{R}, L_{\mathsf{LPN}}, \tau)$ follows $\mathsf{LPN}_{n,\Psi}$. In Ω', the triple follows $\mathcal{D}_{\mathsf{Uniform},\Psi}$. Clearly, the pair $(\mathbf{R}, L_{\mathsf{LPL}})$ follows $\mathsf{LPL}_{n,s}$ in Ω and $\mathcal{D}_{\mathsf{UniformL}}$ in Ω'.

This means that we can transform $\mathsf{LPL}_{n,s}$ resp. $\mathcal{D}_{\mathsf{UniformL}}$-samples to $\mathsf{LPN}_{n,\Psi}$- resp. $\mathcal{D}_{\mathsf{Uniform},\Psi}$-samples just as we compute $(\mathbf{R}, L_{\mathsf{LPN}}, \tau)$ from $(\mathbf{R}, L_{\mathsf{LPL}})$ in the definition of Ω resp. Ω'.

Conversely, note that in Ω resp. Ω', we can compute P_0, P_1 from L_{LPL} and τ. From this, we get R_{Bayes}. As $R_{\mathrm{Bayes}} = \exp\left(\frac{L_{\mathsf{LPL}} + U - \frac{1}{2}}{s^2}\right)$, we can compute $L_{\mathsf{LPL}} = s^2 \ln(R_{\mathrm{Bayes}}) + \frac{1}{2}$. This allows to transform $\mathsf{LPN}_{n,\Psi}$- resp. $\mathcal{D}_{\mathsf{Uniform},\Psi}$-samples into $\mathsf{LPL}_{n,s}$- resp. $\mathcal{D}_{\mathsf{UniformL}}$-samples. The transformation is clearly efficient, which finally proves the claim. □

Next, we consider the reduction from LPN with varying noise to LPN with varying dimension. Again, let us repeat the theorem for readability.

Theorem 6. *Consider $s > 0$ and $0 < \tau' < \frac{1}{2}$. Then, provided s is sufficiently large, the $\mathsf{LPL}_{n,s}$ problem is at least as hard as the $\mathsf{LPN}_{n,\tau'}$ problem.*
More precisely, if $\mathsf{LPN}_{n-X,\tau'}$ is (t, ε, Q)-secure[11]*, then $\mathsf{LPL}_{n,s}$ is (t', ε', Q') secure with $Q = Q' - X, t \approx t', \varepsilon \approx \varepsilon'$. Here, X is a random variable measuring the loss of dimension.*
X follows a Bernoulli distribution on Q tries with success probability p, where $p = \Pr_{\tau \leftarrow \Psi}[\tau < \tau']$ and where Ψ is defined as in Lemma 2.

[11] Here, the (known) dimension of the LPN secret is random as well. If $X > n$, the problem is considered trivial.

Let $0 < \Delta n$ be a (small) real number measuring the acceptable loss of dimension. Then by setting s such that

$$s \ln\Big(\frac{1-\tau'}{\tau'}\Big) > \mathrm{Fc}^{-1}\Big(\frac{\Delta n}{2Q}\Big) + \frac{1}{2s},$$

where $\mathrm{Fc}(x) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-\frac{t^2}{2})dt$ is the complementary cdf of the normal distribution, we can ensure $\mathbb{E}[X] = pQ < \Delta n$.

Proof.
Computing a bound for p. For the statement about $\mathbb{E}[X]$, recall that $\tau \leftarrow \Psi$ outputs samples as follows:
$\widetilde{U} \leftarrow \{0,1\}, \widetilde{E} \leftarrow \Phi_s$, $\widetilde{R}_{>1} = \exp\Big(\frac{|\widetilde{E}+\widetilde{U}-\frac{1}{2}|}{s^2}\Big)$ and $\tau = \big(\widetilde{R}_{>1}+1\big)^{-1}$. Note that, due to symmetry of Φ_s, this distribution does not depend on $\widetilde{U}$, so we set $\widetilde{U} = 0$. We have

$$\begin{aligned}\tau < \tau' &\iff \frac{1}{\widetilde{R}_{>1}+1} < \tau' \iff \widetilde{R}_{>1} = \exp\Big(\frac{|\widetilde{E}-\frac{1}{2}|}{s^2}\Big) > \frac{1-\tau'}{\tau'} \\ &\iff \Big|\frac{\widetilde{E}}{s} - \frac{1}{2s}\Big| > s\ln\big(\frac{1-\tau'}{\tau'}\big)\end{aligned}$$

Since $\frac{\widetilde{E}}{s}$ follows Φ_1, the probability $p = \Pr[\tau < \tau']$ satisfies

$$p = \mathrm{Fc}\Big(s\ln\big(\frac{1-\tau'}{\tau'}\big)+\frac{1}{2s}\Big)+\mathrm{Fc}\Big(s\ln\big(\frac{1-\tau'}{\tau'}\big)-\frac{1}{2s}\Big) < 2\,\mathrm{Fc}\Big(s\ln\big(\frac{1-\tau'}{\tau'}\big)-\frac{1}{2s}\Big), \quad (3)$$

which readily implies the first claim.

Actual reduction. For the actual reduction, we use Lemma 2. So it suffices to transform $\mathsf{LPN}_{n-X,\tau'}$-samples with secret $\mathbf{k}' \in \mathbb{Z}_2^{n-X}$ into $\mathsf{LPN}_{n,\Psi}$-samples for secret $\mathbf{k} \in \mathbb{Z}_2^n$ in such way that we can compute $\mathbf{k}'$ from $\mathbf{k}$. It is easy to see that our transformation will map the uniform distribution on $\mathbb{Z}_2^{n-X} \times \mathbb{Z}_2$ to $\mathcal{D}_{\mathsf{Uniform},\Psi}$.

Our transformation works as follows:

If $X \geq n$, there is nothing to do. Otherwise, uniformly pick X distinct query indices $q_1, \ldots, q_X \in \{1, \ldots, Q\}$. These will be the queries for which $\tau < \tau'$. We furthermore pick $\mathbf{r}_1, \ldots, \mathbf{r}_X$ uniformly from $\mathbb{Z}_2^n$. These will be the $\mathbf{r}$'s used in those queries. Use Gaussian elimination to pick some linear bijection $T\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^{n-X} \times \mathbb{Z}_2^X$ such that $T \cdot \mathbf{r}_i \in \{0\} \times \mathbb{Z}_2^X$. Store its inverse T^{-1}. Pick $\mathbf{k}'' \in \mathbb{Z}_2^X$ unformly. We implicitly set $\mathbf{k} = T^{\mathsf{t}} \cdot (\mathbf{k}', \mathbf{k}'')$.

We now answer the i-th query for a $\mathsf{LPL}_{n,\Psi}$-sample as follows:

Case 1: $i = q_j$
We choose $\mathbf{r} := \mathbf{r}_j$. Choose $\tau \leftarrow \Psi$, conditioned on $\tau < \tau'$. Note that this can be done efficiently to arbitrary precision by virtue of the known cdfs of the distributions involved. Set $T \cdot \mathbf{r} = (\mathbf{r}', \mathbf{r}'')$ with $\mathbf{r}' = 0$ and $\mathbf{r}'' \in \mathbb{Z}_2^X$. Set $u = \langle \mathbf{r}'', \mathbf{k}'' \rangle \bmod 2$, $e \leftarrow \mathcal{B}_\tau$ and set $\ell = u + e \bmod 2$. Output $(\mathbf{r}, \ell, \tau)$.

Case 2: $i \neq q_j$ **for all** q_j
Call the $\mathsf{LPN}_{n-X,\tau'}$-oracle to receive $(\mathbf{r}', \ell')$ with $\ell' = \langle \mathbf{r}', \mathbf{k}' \rangle + e'$. Sample $\mathbf{r}'' \in \mathbb{Z}_2^X$ uniform. Set $\mathbf{r} = T^{-1} \cdot (\mathbf{r}', \mathbf{r}'')$. Set $u'' = \langle \mathbf{r}'', \mathbf{k}'' \rangle$. Sample $\tau \leftarrow \Psi$, conditioned on $\tau \geq \tau'$. Again, this can be done efficiently.
Set $\tau_{\text{Extra}} = \frac{\tau - \tau'}{1 - 2\tau'}$. This is done to ensure that $E_1 + E_2 \bmod 2$ follows $\mathcal{B}_\tau$ if $E_1 \leftarrow \mathcal{B}_{\tau'}, E_2 \leftarrow \mathcal{B}_{\tau_{\text{Extra}}}$. Sample $e_{\text{Extra}} \leftarrow \mathcal{B}_{\tau_{\text{Extra}}}$. Set $\ell = \ell' + u'' + e_{\text{Extra}} \bmod 2$.
Output $(\mathbf{r}, \ell, \tau)$.

Analysis. Note that the distribution of τ follows Ψ. Indeed, rather than sampling Q times from Ψ as $\tau_1, \ldots, \tau_Q \leftarrow \Psi$ and then deciding how many of the τ's are smaller than τ', we sample that number first, then their position among the Q positions and then the actual τ_i's. Further, we have that the $\mathbf{r}$'s are uniformly random and independent from the τ's. This is clear in both cases 1 and 2. Next, note that with $\mathbf{k} = T^{\mathsf{t}} \cdot (\mathbf{k}, \mathbf{k}'')$, we have that $\langle \mathbf{r}, \mathbf{k} \rangle = \langle T\mathbf{r}, T^{-\mathsf{t}}\mathbf{k} \rangle = \langle \mathbf{r}', \mathbf{k}' \rangle + \langle \mathbf{r}'', \mathbf{k}'' \rangle$.

In case 1, we thereby have $\ell = \langle \mathbf{r}, \mathbf{k} \rangle + e$ where $e \leftarrow \mathcal{B}_\tau$.
In case 2, we have $\ell = \ell' + u'' + e_{\text{Extra}} = \langle \mathbf{r}', \mathbf{k}' \rangle + e' + \langle \mathbf{r}'', \mathbf{k}'' \rangle + e_{\text{Extra}} = \langle \mathbf{r}, \mathbf{k} \rangle + (e' + e_{\text{Extra}})$. Now, we constructed the noise rate τ_{Extra} such that

$$\Pr[e' + e_{\text{Extra}} = 1 \bmod 2] = (1 - \tau')(\tau_{\text{Extra}}) + (1 - \tau_{\text{Extra}})\tau' = \tau,$$

as one can easily verify. Hence, our samples follow the correct distribution, finishing the proof. □

We finally give the detail for Lemma 3, showing that giving the adversary access to intermediate values only loses a factor $\sqrt{2}$ for the noise. Again, we repeat the statement of the lemma:

Lemma 6. *Let $d \geq 2$. Then the $\mathsf{LPL}_{n,\sqrt{2}s,d}$-search problem is at least as hard as the search-$\mathsf{LPL}_{n,s}$ problem. More precisely, if search-$\mathsf{LPL}_{n,s}$ is (t, ε)-hard with q samples, then $\mathsf{LPL}_{n,\sqrt{2}s,d}$ is (t', ε')-hard with q samples, where $t \approx t', \varepsilon \approx \varepsilon'$.*

Proof. We transform $\mathsf{LPL}_{n,s}$-samples for secret $\mathbf{k}$ into $\mathsf{LPL}_{n,\sqrt{2}s,d}$-samples for the same $\mathbf{k}$. To this end, consider a $\mathsf{LPL}_{n,s}$-sample $(\mathbf{r}, \ell)$ with $\ell = \langle \mathbf{r}, \mathbf{k} \rangle + e$. We denote the (unknown) $\langle \mathbf{r}, \mathbf{k} \rangle$ by u.

We sample $u_1, \ldots, u_{d-1} \in \mathbb{Z}_2$ uniform and independent. Let the unknown $u_d \in \mathbb{Z}_2$ be such that $\sum_i u_i \bmod 2 = u$. Set $u'_k = \sum_{i=1}^{k} u_i \bmod 2$ for $1 < k \leq d$. We known all u'_k except for $u'_d = u$.
Sample $e_k \leftarrow \Phi_{\sqrt{2}s}$ and set $\ell_k = u_k + e_k$ for $1 \leq k < d$.
Sample $e'_k \leftarrow \Phi_{\sqrt{2}s}$ and set $\ell'_k = u'_k + e'_k$ for $2 < k < d$.

Next, for the remaining ℓ_d, ℓ'_d sample $\widetilde{e} \leftarrow \Phi_s$.
If $u'_{d-1} = 0$, we set $\ell_d = \ell - \widetilde{e}$ and set $\ell'_d = \ell + \widetilde{e}$.
If $u'_{d-1} = 1$, we set $\ell_d = 1 - \ell + \widetilde{e}$ and set $\ell'_d = \ell + \widetilde{e}$.
In both cases, we output $\mathbf{r}$ and all ℓ_k, ℓ'_k.

Let us analyse the distribution of the output. Clearly, $u_1, \ldots, u_d$ are distributed as in $\mathsf{LPL}_{n,\sqrt{2}s,d}$. The ℓ_k's and ℓ'_k's are distributed as desired for $k \neq d$. Now consider $k = d$: By definition, $u'_d = u'_{d-1} + u_d \bmod 2$. So if $u'_{d-1} = 0$, we

have $u'_d = u_d = u$, hence

$$\begin{pmatrix}\ell_d\\ \ell'_d\end{pmatrix} = \begin{pmatrix}u+e-\widetilde{e}\\ u+e+\widetilde{e}\end{pmatrix} = \begin{pmatrix}u_d\\ u'_d\end{pmatrix} + e\begin{pmatrix}1\\ 1\end{pmatrix} + \widetilde{e}\begin{pmatrix}-1\\ 1\end{pmatrix}.$$

If $u'_{d-1} = 0$, we have $u_d = 1 - u'_d$, which actually holds over $\mathbb{Z}$ and not just mod 2. It follows that

$$\begin{pmatrix}\ell_d\\ \ell'_d\end{pmatrix} = \begin{pmatrix}1-(u+e)+\widetilde{e}\\ u+e+\widetilde{e}\end{pmatrix} = \begin{pmatrix}u_d\\ u'_d\end{pmatrix} + e\begin{pmatrix}-1\\ 1\end{pmatrix} + \widetilde{e}\begin{pmatrix}1\\ 1\end{pmatrix}.$$

Since $e, \widetilde{e}$ are independent Gaussians of width s, and $(1,1)^\mathsf{t}$ and $(-1,1)^\mathsf{t}$ are orthogonal of length $\sqrt{2}$, it follows that ℓ_d and ℓ'_d are distributed as desired. □

In the proof of Lemma 4, we needed that 1 probe with Gaussian noise s is equivalent to c probes with Gaussian noise $\sqrt{c}s$. Note that we implicitly used that at the end of the proof above with $c = 2$.

Lemma 7. *For an unknown value $x \in \mathbb{R}$, consider the distributions*

$$\Psi := x + e \quad \text{for } e \leftarrow \Phi_s$$

on $\mathbb{R}$ and

$$\Psi' := (x + e_1, x + e_2, \ldots, x + e_c) \quad \text{for } e_i \leftarrow \Phi_{\sqrt{c}s}$$

Then these distributions are computationally equivalent, i.e. individual Ψ-samples can be transformed into individual Ψ' samples and vice versa for the same x.

Proof. Write $\mathbb{R}^n = V \oplus V'$ as an orthogonal direct sum, where V is the span of $(1, \ldots, 1)^\mathsf{t}$ and V' is its orthogonal complement. Then Ψ' can be transformed into Ψ by orthogonal projection onto V. Conversely, we can transform $y \leftarrow \Psi$ into a Ψ'-sample via $y \cdot (1, \ldots, 1)^\mathsf{t} + e'$, where e' is an $n-1$-dimensional Gaussian in V' whose coordinate (wrt. an orthonormal basis) have standard deviation $\sqrt{c}gp$. □

Finally, we give the proof for our main security statement, Thm. 2. Again, we repeat the statement first:

Theorem 7. *Consider our re-keying scheme Π_{noisy} with parameters n, d, m. Assume $n - m - d > \lambda, 2t < d$ and $\frac{n}{m} = \Theta(1)$. Model H as a random oracle. Then Π_{noisy} is secure in the continuous t-noisy probing model with Gaussian noise $\Phi_{\sqrt{t+1}s}$ under the Search-$\mathsf{LPL}_{n,s,d}$-Assumption.*

Clearly, using Lemma 3 together with Thm. 1, Thm. 2 proves our re-keying scheme Π_{noisy} secure under LPN.

Proof. **Construction of Simulator of the Real World.** By Lemma 4, we may assume that the adversary only probes bits of $\mathbf{u}_i = \mathbf{R} \cdot \mathsf{msk}_i$ or to partial sums $\mathbf{u}'_k = \sum_{i=1}^{k} \mathbf{u}_i$. We construct our simulator *Sim* as follows: *Sim* chooses

its own master key msk' and uses that to simulate probes just as the real game (but with msk' as master key).

Indistinguishability. To argue about indistinguishability, our proof proceeds as follows:
First, we show that any adversary A that distinguishes the real world from ideal world + simulator can be used to obtain a noiseless version of $\mathbf{Rk}$, which gives linear equations $\mathbf{Rk} = \mathbf{z}$ about the $\mathsf{LPL}_{n,s,d}$-secret $\mathbf{k}$.
Second, we analyze how we actually simulate for A and how we take several such linear equations.
Third, we need to analyze the probability that the linear equations uniquely determine $\mathbf{k}$.

Obtaining linear relations. We now argue that we can obtain noiseless versions of $\mathbf{Rk}$, hence linear equations about the $\mathsf{LPL}_{n,s,d}$ secret:
Assume that we have some adversary A that distinguishes our simulation in the ideal world from the real world with advantage $\varepsilon = \frac{1}{\mathrm{poly}(\lambda)}$, running in time $T = \mathrm{poly}(\lambda)$ and using $Q_H = \mathrm{poly}(\lambda)$ oracle queries and Q sessions. In the real world, A receives session keys sk^i with $\mathsf{sk}^i = H(\mathbf{R}^i \cdot \mathsf{msk})$, whereas in the simulated situation, the session keys sk^i are unrelated to msk' and this is indeed the only difference. Consequently, A queries the random oracle H with some $\mathbf{R}^i \cdot \mathsf{msk}$ resp. with $\mathbf{R}^i \cdot \mathsf{msk}'$ for $i \in \{1, \ldots, Q\}$ with probability at least ε. We call such a query (and the corresponding $i, \mathbf{R}^i$) "good". Clearly, if there exists a good query, we can guess[12] with probability at least $\frac{1}{QQ_H}$ which which oracle query was the good one and which $\mathbf{R}^i$ it corresponds to. If we are successful, this clearly gives an linear relation $\mathbf{R}^i\mathbf{k} = \mathbf{z}$.

Simulating with $\mathsf{LPL}_{n,s,d}$ and Combining the Equations from Non-Independent Runs. We use A to solve the search-$\mathsf{LPL}_{n,s,d}$-problem for unknown LPL-key $\mathbf{k}$ as follows: Choose $\mathrm{rep} = \lceil \frac{n}{m} \rceil + 1 = \Theta(1)$ for the number of runs of A. Perform rep many runs of A, using the $\mathsf{LPL}_{n,s,d}$ oracle to simulate queries. Note that when simulating the probing queries by the $\mathsf{LPL}_{n,s,d}$-oracle, we patch up the matrices $\mathbf{R}$ from the rows $\mathbf{r}$ received from the $\mathsf{LPL}_{n,s,d}$ oracle. If $\mathbf{R}$ has full rank, which happens with overwhelming probability for $n - m > \lambda$, then splitting $\mathbf{u}$ into shares (as done by $\mathsf{LPL}_{n,s,d}$) is equivalent to splitting $\mathbf{k}$ (as required to simulate Π_{noisy}), so the simulation is correct.

In our simulation, the master key corresponds to the unknown $\mathsf{LPL}_{n,s,d}$ key $\mathbf{k}$. The output of $\mathsf{LPL}_{n,s,d}$ allows to answer probing queries in the obvious way. Recall that by Lemma 4, we may assume that there are no probes on shares of the master secret key or on internal wires of $\mathsf{Refresh}$.

We need that A makes a good query in each of the rep runs, which are dependent, as they share the same key $\mathbf{k}$. For this, we have

$$\mathop{\mathbb{E}}_{\mathbf{k}}[\Pr[A \text{ makes good query} \mid \mathbf{k}]^{\mathrm{rep}}] \geq \Pr[A \text{ makes good query}]^{\mathrm{rep}} \geq \varepsilon^{\mathrm{rep}}$$

[12] As $\mathbf{R}^i \cdot \mathbf{k} = \mathbf{u}^i$ and we know an approximation for $\mathbf{u}^i$, for realistic parameters there is usually at most one fitting i, given z. So the factor of $\frac{1}{Q}$ can actually be removed.

by Jensen's inequality and $\varepsilon^{\text{rep}} = \frac{1}{\text{poly}(\lambda)}$. The probability to correctly guess for each run which of the H-queries $\mathbf{z}$ is good and to which good session i it corresponds to, is at least $\left(\frac{1}{Q_H Q}\right)^{\text{rep}} = \frac{1}{\text{poly}(\lambda)}$, provided there are good queries to start with. Collecting the equations of the form $\mathbf{z} = \mathbf{R} \cdot \mathbf{k}$ from the rep runs allows to easily compute $\mathbf{k}$ by linear algebra, provided that the rows of the rep collected good $\mathbf{R}^i$ together span $\mathbb{Z}_2^n$.

Obtained Equations are Full Rank. We now show that the system of linear equations that we obtained has full rank and allows to determine $\mathbf{k}$ with probability $1 - o(1)$. Note that the good $\mathbf{R}$'s are not uniform since we condition of A making a good query and A's selection from the possible sessions. Note that if the obtained linear relations are not full rank, then there exists a subspace $V \subset \mathbb{Z}_2^n$ of dimension $n-1$ such that all rows of all obtained $\mathbf{R}^i$ are contained in V.

Fix one possible subspace $V \subset \mathbb{Z}_2^n$ of dimension $n-1$. Consider one run of A (without imposing that there be a good query) and let $\mathbf{R}^1, \ldots, \mathbf{R}^Q$ be the uniform, independent, session-specific $\mathbf{R}$-matrices in that run. Let $\mathbf{R} \prec V$ denote the event that the m rows of $\mathbf{R}$ are contained in V, which happens with probability (over $\mathbf{R}$ uniform) 2^{-m}. We have

$$
\begin{aligned}
2^{-m}Q &\geq \Pr[\mathbf{R}^i \prec V \text{ for some } i] \geq \Pr[\mathbf{R}^i \prec V \text{ for some } i, A \text{ makes good query}] \\
&\geq \Pr[\mathbf{R}^i \prec V \text{ for some } i | A \text{ makes good query}] \cdot \Pr[A \text{ makes good query}] \\
&\geq \Pr[\mathbf{R}^i \prec V \text{ for good } i | A \text{ makes good query}] \cdot \varepsilon
\end{aligned}
$$

It follows that $\Pr[\mathbf{R}_{\text{good}} \prec V] \leq 2^{-m}\varepsilon^{-1}Q$ for a single good $\mathbf{R}_{\text{good}}$ that is obtained by our reduction. The probability that $\mathbf{R} \prec V$ for all rep many good $\mathbf{R}$'s is bounded by $(2^{-m}\varepsilon^{-1}Q)^{\text{rep}}$. Taking a union bound over all $2^n - 1$ possible subspaces V, we bound the probability that the rows of the good $\mathbf{R}$'s together do not span all of $\mathbb{Z}_2^n$ by $2^n \cdot ((2^{-m}\varepsilon^{-1}Q)^{\text{rep}}) = \frac{Q^{\text{rep}}\varepsilon^{-\text{rep}}}{2^{m \cdot \text{rep} - n}} = \frac{\text{poly}(\lambda)}{2^{\Theta(m)}} = o(1)$, since $n - m \cdot \text{rep} > m$.

This implies that we can solve the search-$\mathsf{LPL}_{n,s,d}$ problem with probability $(1 - o(1))(\frac{\varepsilon}{Q_H \cdot Q})^{\lceil \frac{n}{m} \rceil + 1} = \frac{1}{\text{poly}(\lambda)}$ using $Qm \cdot \text{rep} < Q(n + 2m) = \text{poly}(\lambda)$ many $\mathsf{LPL}_{n,s,d}$ samples in time approximately $(\lceil \frac{n}{m} \rceil + 1) \cdot T_A = \Theta(T_A)$, where T_A is the time needed to run A. □