Analysis and Implementation of An Efficient Ring-LPN Based Commitment Scheme

Helger Lipmaa and Kateryna Pavlyk

University of Tartu, Estonia

Abstract. We analyze an efficient parallelizable commitment scheme that is statistically binding and computationally hiding under a variant of the decisional Ring-LPN assumption, conjectured to be secure against quantum computers. It works over medium-size binary finite fields, with both commitment and verification being dominated by 38 finite field multiplications. Such efficiency is achieved due to a precise analysis (that takes into account recent attacks against LPN) of underlying parameters. We report an initial parallel implementation by using the standard OpenCL library on three different platforms. On the AMD Radeon HD 7950 GPU, one can commit to 1024-bit messages in 1 bit per 104.7 cycles. We consider the analysis (which results in concrete parameters that subsequent work can try to falsify) together with the implementation the two most important aspects of the current work.

Keywords: commitment schemes, GPU implementation, learning parity with noise, postquantum

1 Introduction

A commitment scheme allows Alice to send a hidden value to Bob, so that she can later open the commitment only to the original value (the binding property), while before the opening the committed value stays hidden from Bob (the hiding property). Being one of the most basic public-key primitives, commitment schemes play an important role in the design of various cryptographic protocols. E.g., to achieve security against malicious participants, a participant can first commit to his inputs, and then present a zero-knowledge proof of correct behavior on the committed data. To not hinder real-life use, apart from being secure, a commitment scheme should also be highly efficient.

Probably the best known commitment scheme is the Pedersen scheme [15], computationally binding under the discrete logarithm assumption. However, the discrete logarithm assumption can be broken by using quantum computers. It is desirable to design postquantum commitment schemes, i.e., commitment schemes, secure against quantum computers. Such schemes can be used to design postquantum cryptographic protocols.

Moreover, one is interested in the design of *lightweight* postquantum commitment schemes. Here, by lightweight we mean real efficiency on (readily) available laptop or desktop computers. Existing commitment schemes are usually

not lightweight. For example, the Pedersen commitment scheme requires a committer to execute two exponentiations, and is thus not very efficient. Moreover, to achieve reasonable efficiency, Pedersen commitment scheme has to be implemented over well-chosen elliptic curves, by using far from mundane implementation techniques. (See, e.g., [7].) This makes implementation (and its verification) itself a burdensome process.

Currently, more and more computationally intensive tasks are done by GPU-s (graphics processors) that can solve many parallelizable computational problems much faster than the CPU-s. Most of the conventional (public-key) cryptographic primitives have not been designed with such architectures in mind. Thus, it is becoming necessary to design cryptographic primitives that are fast on SIMD (Single Instruction Multiple Data) architectures, implemented by modern GPU-s. At the current moment, even relatively cheap laptops have GPU-s that offer computational power vastly superior to the CPU-s. We expect this trend to continue in the future. Given quick advances in technology, in a few years such GPU-s will become available on tablets and smartphones.

Two most promising directions to achieve security against quantum computers seem to be (closely intertwined) lattice-based and code-based cryptography. Based on lattices, it is known how to implement very many functionalities (including, say, fully-homomorphic encryption), though often not efficiently enough for practice. Using code-based cryptography, it is known how to implement a somewhat smaller number of functionalities, but often very efficiently.

One of the most interesting code-based assumptions is *learning parity with noise* (LPN, [2]). Based on LPN and its variants, it is known how to design efficiently various symmetric primitives and protocols. Especially promising are ring-based variants of such protocols, first considered in [10]. However, design of similarly efficient public-key primitives is seriously lagging, and even recent approaches, like public-key cryptosystems based on the decisional transposed ring-LPN (TRLPN) assumption [8], are not yet sufficiently efficient.

We propose a new lightweight statistically binding and quantum-secure computationally hiding commitment scheme. While the new commitment scheme is a variant of some previous commitment schemes [11], we provide a much more precise analysis of the security parameters than given in previous work. In particular, we take into account recent attacks [9]. We also provide a partially optimized implementation on contemporary GPU-s.

Let $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$ be binary finite field, where $f(X)$ is a degree-n irreducible polynomial. We commit to $m \in \mathfrak{R}$, by using a randomizer $(r, e) \in \mathfrak{R}^2$ and a public key $(\boldsymbol{M}, \boldsymbol{R}) \in \mathfrak{R}^{19\times 2}$. The commitment is equal to $\boldsymbol{M}m \oplus \boldsymbol{R}r \oplus \boldsymbol{e}$, where the noise $\boldsymbol{e}$ comes from a capped Bernoulli distribution (see Sect. 2) with parameter $\tau \approx 0.128118$. (See the analysis in Sect. 4.)

We show that the new commitment scheme is binding if and only if the additive noise $\boldsymbol{e}$ is sufficiently small (in the sense of the ℓ_1-norm). This follows from a generalization and a careful analysis of the Gilbert-Varshamov bound to the finite field, where one of the steps in the generalization crucially depends on the fact that $\mathfrak{R}$ is a field. On the other hand, if the added noise is too small, then

the decisional TRLPN problem of Damgård and Park [8] becomes easy and thus the commitment scheme is not hiding. (This follows from the existing attacks against the LPN problem, see [9] and the references therein.) Hence, we have to choose the underlying parameters carefully, so that the commitment scheme be both binding and hiding yet efficient.

We recommend parameters (e.g., $n = 1024$ and $\tau = 0.128118$, that offers 140-bit security against known attacks, see Sect. 4) under which the new commitment scheme is statistically binding and computationally hiding. The proposed value of τ depends on several factors, including the loss of security due to the use of capped Bernoulli lemma, the preciseness of the Gilbert-Varshamov bound, and intricate details on the best known attacks against LPN [9].

Interestingly, the new commitment scheme is significantly more efficient than the most efficient known public-key cryptosystems [8] based on the same assumption. In the case of known public-key cryptosystems [8] the degree n (at least 10 000 bits[1] compared to 1 024 bits at the same security level) of the polynomial f is significantly higher than in our case. This can be compared to the case of discrete logarithm-based schemes, where there is only minimal difference in the efficiency between commitment schemes (e.g., Pedersen) and public-key cryptosystems (e.g., Elgamal). We leave it as an open question whether there is some intrinsic reason behind this.

The new commitment scheme is a ring-LPN based variant of previous LPN-based commitment schemes [13,11]. However, using a different assumption (ring-LPN) requires establishing concrete security parameters. As we will see in this paper, the choice of parameters under which this scheme is both secure under existing attacks yet efficient is far from being trivial. In comparison, [11] does not analyze concrete parameters at all and thus can be seen as being rather theoretical in its approach. Some of their choices (e.g., $N = K$ where N and K are the two main parameters of the LPN assumption) are applicable also in our case but they would make the new scheme unnecessarily inefficient. Finally, in the case where the best known attacks will be improved, one can use our methodology to increase some of the parameters of the new commitment scheme.

Given our security analysis, both commitment and verification are dominated by 38 binary finite field multiplications in a medium-size field, with $n = 1024$. This can be compared to say Pedersen commitment, where one has to execute exponentiations in an elliptic curve group defined over a smaller finite field group (e.g., over $\mathbb{F}_p$, with $\log_2 p \approx 256$). Apart from the obvious efficiency benefits, implementing of the new scheme from scratch is also much easier, partially since one can avoid learning the intricacies of elliptic curves.

To emphasize both on the parallelizability and the conceptual simplicity of the new commitment scheme, we finish the paper with a description of an initial, very simple, implementation. This implementation is not of industrial strength, but is mostly just provided to give a rough idea of achievable speed. More precisely, we implemented both a matrix-to-vector multiplication and a finite field

[1] This estimate, given in [8], does not account for the recent attacks [9,4]. According to [4], the key length of the public-key cryptosystem of [8] should be even larger.

multiplication over the finite field $\mathbb{F}_{2^{1024}}$. In addition, one can reuse extensive literature on the fast implementations of finite field multiplications: while we focused on the simplicity of the implementation, we are sure that up-to-date algorithms achieve better efficiency. The timings, reported in Tbl. 1, should then be multiplied by 38 (plus a small epsilon to account for finite field additions) to obtain the timings of both commitment and verification.

Our implementation uses the OpenCL standard for parallel programming. We tested this implementation on a rather mediocre[2] NVIDIA Quadro 2000M GPU (available in medium-class laptops), on a modern AMD Radeon HD 7950 GPU, and on the Intel i7-2860QM CPU, the results are summarized in Tbl. 1.

As shown in Sect. 5, since the HD 7950 implementation only utilizes 272 cores out of 1792, one can implement on average $1792/272 \approx 6.59$ binary finite field multiplications in parallel. Since one commitment requires 38 multiplications, in average one can schedule $1792/272/38 \approx 0.173$ commitments of 1024-bit messages per execution. Thus, one can commit to 1 bit per $18{,}488.6/(0.173 \cdot 1024) \approx 104.7$ cycles on the AMD GPU. (See Tbl. 1 for the origin of the number 18,488.6.) We emphasize that this is the peak throughput number, given full pipelines and optimal scheduling and that at this moment, this is only an estimation. We provide an efficiency comparison with the Pedersen commitment scheme in Sect. 5.

2 Preliminaries

By default, all vectors are column vectors. For $\boldsymbol{a} \in \mathbb{Z}_2^n$, let $\boldsymbol{a}[i]$ be its ith coordinate. Thus, $\boldsymbol{a} = (\boldsymbol{a}[1], \ldots, \boldsymbol{a}[n])$. For a vector $\boldsymbol{a} \in \mathbb{Z}_2^n$, let $||\boldsymbol{a}||_1 = \sum \boldsymbol{a}[i] = \sharp\{i : \boldsymbol{a}[i] \neq 0\}$ be its Hamming weight. For a set A, $a \xleftarrow{r} A$ means that a is uniformly picked from A, and for a randomized algorithm $\mathcal{A}$, $a \xleftarrow{r} \mathcal{A}$ means that a is uniformly picked by $\mathcal{A}$. Let κ be the computational security parameter, and let λ be the information-theoretical security parameter. In practice, one can assume that $\kappa = 128$ and $\lambda = 40$. We give κ and λ as unary inputs (denoted by 1^κ and 1^λ) to some of the algorithms.

A *commitment scheme* enables a party to commit to a message, and open it later to the same value. On the one hand, the commitment must hide the message. On the other hand, the committer should not be able to open the commitment to anything else but the original message. More formally, a commitment scheme (in the public parameters model) is a tuple of three efficient algorithms, gen, com and ver. The algorithm $\mathsf{gen}(1^\kappa, 1^\lambda)$ generates public parameters gk for the commitment scheme. After that, the randomized algorithm $\mathsf{com}_{\mathsf{gk}}(m; \cdot)$ commits to a message m by picking a uniformly random randomizer $r \xleftarrow{r} \mathcal{R}$ (here, $\mathcal{R}$ is a randomizer set specified by the commitment scheme and κ) and outputting $(y, z) \leftarrow \mathsf{com}_{\mathsf{gk}}(m; r)$. Here, y is the commitment to m, while z is the decommitment value. It is usually required that one can efficiently reconstruct m from z.

[2] This GPU is more than 35 times slower than the fastest GPU-s in the market, according to `https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison` (accessed in June 2015).

The verification algorithm ver verifies that z is the correct decommitment of y, i.e., $\mathsf{ver}_{\mathsf{gk}}(y, z)$ outputs either 1 or 0. It is required that $\mathsf{ver}_{\mathsf{gk}}(\mathsf{com}_{\mathsf{gk}}(m; r)) = 1$ for all valid gk, m and r.

A commitment scheme is *statistically binding* if with probability $1 - \kappa^{-\omega(1)}$ over the choice of gk, for any $m_1 \neq m_2$, r_1 and r_2, if $(y_i, z_i) = \mathsf{com}_{\mathsf{gk}}(m_i; r_i)$ then $y_1 \neq y_2$. A commitment scheme is computationally hiding if, given y, it is computationally difficult to infer any information about m. More precisely, the commitment scheme $(\mathsf{gen}, \mathsf{com}, \mathsf{ver})$ is *computationally hiding*, if for any non-uniform probabilistic polynomial time stateful adversary $\mathcal{A}$, the following value is negligible in κ:

$$\left| \Pr \left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{gen}(1^\kappa, 1^\lambda), (m_1, m_2) \leftarrow \mathcal{A}(\mathsf{gk}), \alpha \xleftarrow{r} \{1, 2\}, r \xleftarrow{r} \mathcal{R} : \\ \mathcal{A}(\mathsf{com}_{\mathsf{gk}}(\mathsf{gk}, m_\alpha; r)) = \alpha \end{array} \right] - \frac{1}{2} \right| .$$

An $[N, K]$ *code* C over a finite alphabet Σ is a subset of Σ^N. The elements of C are the codewords of C. If $|\Sigma| = q$, C is called a q-ary code. Associated with a code is an encoding map E that maps the message set Σ^K to Σ^N. The code is then the image of the encoding map, and it is said to be of length N and rank K. An $[N, K]$ code is called *linear* if any linear combination of its codewords is also a codeword. A *generator matrix* $\boldsymbol{R}$ of a linear $[N, K]$ code is a $N \times K$ matrix whose rows form a basis of the code. The (minimum) *distance* D of code C is the minimal Hamming distance between two distinct codewords of C. An $[N, K]$ code with a minimal distance D is known as an $[N, K, D]$ code. For arbitrary linear code C, its minimum distance equals the minimum Hamming weight of a nonzero codeword of C (see Proposition 2.1, [16]). According to the Singleton bound, $N - K + 1 \geq D$. According to the Gilbert-Varshamov bound, a random $[N, K]$ code has distance that matches the Singleton bound.

The code C has covering radius [6] d_C when the distance between C and any element of Σ^N is not more than d_C. An $[N, K, D]d_C$ *covering code* is a linear code with parameters $[N, K, D]$ and covering radius d_C. Denote $\varrho = d_C/N$. The best possible covering radius d_C satisfies the sphere-covering bound $\sum_{i=0}^{d_C} \binom{N}{i} \geq 2^{N-K}$ [6], or alternatively $2^K \geq 2^N / |B_N(d_C)| \geq 2^{(1-H_2(\varrho))N}$, where $H_2(p) = -p \log_2 p - (1-p) \log_2 (1-p)$, $p \in [0, 1]$, is the binary entropy function.

For $N \in \mathbb{N}^+$, let χ_N be a distribution over $\mathbb{Z}_2^N$. Let $K < N$ be another positive integer. The *decisional* (χ_N, N, K)*-LPN problem* [2] is (t, ε)-hard if for every distinguisher $\mathcal{D}$ of size t:

$$\left| \Pr_{\boldsymbol{R}, \boldsymbol{s}, \boldsymbol{e}} [\mathcal{D}(\boldsymbol{R}, \boldsymbol{R}\boldsymbol{s} \oplus \boldsymbol{e}) = 1] - \Pr_{\boldsymbol{R}, \boldsymbol{r}} [\mathcal{D}(\boldsymbol{R}, \boldsymbol{r}) = 1] \right| \leq \varepsilon \ ,$$

where $\boldsymbol{R} \xleftarrow{r} \mathbb{Z}_2^{N \times K}$, $\boldsymbol{s} \xleftarrow{r} \mathbb{Z}_2^K$, $\boldsymbol{e} \xleftarrow{r} \chi_N$, and $\boldsymbol{r} \xleftarrow{r} \mathbb{Z}_2^N$. $\mathcal{D}$ also has access to the description of χ_N.

In the standard definition of the decisional LPN problem, the error distribution χ_N is the Bernoulli distribution with a parameter $0 < \tau < 1/2$, i.e., every bit $\boldsymbol{e}[i]$ is chosen independently and identically distributed with $\Pr[\boldsymbol{e}[i] = 1] = \tau$. In this case, we write $\chi_N = \mathsf{Ber}_\tau^N$. The decisional $(\mathsf{Ber}_\tau^N, N, K)$-LPN problem is

closely related to the long-standing open problem of efficiently decoding random linear codes, and — assuming $N = \Theta(K)$ like in the current paper — is believed to be hard even in the presence of quantum computers. Moreover, the search (given $\boldsymbol{R}\boldsymbol{s} \oplus \boldsymbol{e}$, compute $\boldsymbol{s}$) and the decision version of the LPN assumption are known to be polynomially equivalent [12]. The first subexponential algorithm to solve the (search) LPN was proposed in [3]. The most efficient currently known attack is by Guo, Johansson, and Löndahl [9].[3]

Consider a coin that shows heads with probability τ and tails with probability $1-\tau$. The Hoeffding inequality states that the probability that N coin tosses yields heads at least $(\tau+\varepsilon)N$ times is at most $1-\exp(-2\varepsilon^2 N)$. I.e., if a random variable comes from the Bernoulli distribution with parameter τ, then by the Hoeffding inequality it is larger than $\tau^* N$ with probability $2^{-\lambda}$, where

$$\tau^* := \tau + \sqrt{\frac{\lambda}{2\log_2 e \cdot N}} \ . \tag{1}$$

Thus, when working with the decisional LPN assumption, one can always assume that $||\boldsymbol{e}||_1 \leq \tau^* N$. Following [11], we denote by $\overline{\mathsf{Ber}}_\tau^N$ the corresponding *capped Bernoulli distribution*, and by $(\overline{\mathsf{Ber}}_\tau^N, N, K)$-LPN the resulting assumption. I.e., $\boldsymbol{e} \xleftarrow{r} \overline{\mathsf{Ber}}_\tau^N$ means that $\boldsymbol{e}$ is first chosen according to Ber_τ^N. If $||\boldsymbol{e}||_1 > \tau^* N$, one resamples $\boldsymbol{e}$ again until its norm becomes not greater than $\tau^* N$. Clearly, $(\overline{\mathsf{Ber}}_\tau^N, N, K)$-LPN is difficult iff $(\mathsf{Ber}_\tau^N, N, K)$-LPN is difficult; see [11].

Heyse et alt. [10] proposed a ring variant of the decisional LPN assumption. A variant of it, TRLPN, was defined by Damgård and Park [8]. Consider the ring $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$, where f is some degree n irreducible polynomial over $\mathbb{Z}_2[X]$. (See [10] for a treatment in the case f is reducible.) The elements of $\mathfrak{R}$ are thus degree-n polynomials over $\mathbb{Z}_2[X]$, with their addition and multiplication defined modulo f. Define $||m||_1 := \sum_{i=0}^{n-1} m_i$, and for a vector $\boldsymbol{e} \in \mathfrak{R}^a$ for some a, let $||\boldsymbol{e}||_1 := \sum_{i=1}^{a} ||e_i||_1$ be its ℓ_1-norm.

For a polynomial ring $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$, let $\mathsf{Ber}_\tau^{\mathfrak{R}}$ denote the distribution over polynomials from $\mathfrak{R}$, where each of the coefficients of the polynomial is drawn independently from Ber_τ. For N with $n \mid N$, the capped Bernoulli distribution $\overline{\mathsf{Ber}}_\tau^{\mathfrak{R},N/n}$ is defined in a natural way: one first chooses $\boldsymbol{a} \xleftarrow{r} \overline{\mathsf{Ber}}_\tau^N$, and then outputs $\boldsymbol{e}$, where $e_i \leftarrow \sum_{j=0}^{n-1} a_{(i-1)n+j+1} X^j$ for $1 \leq i \leq N/n$.

Let $N > K$, and let ψ be a distribution on $\mathbb{Z}_2^{N \times K}$. The *decisional* $(\chi_N, N, K; \psi)$*-LPN problem* [8] is defined exactly as the decisional (χ_N, N, K)-LPN problem, except that $\boldsymbol{R}$ is drawn from ψ. Now, for a ring element $r = \sum_{i=0}^{n-1} r_i X^i \in \mathfrak{R}$, let $\mathsf{vec}(r)$ be the natural isomorphic mapping of its coefficient vector to $\mathbb{Z}_2^n$, that is, $\mathsf{vec}(r)[i] = r_{i-1}$. For $r \in \mathfrak{R}$, we define $\mathsf{mat}(r) \in \mathbb{Z}_2^{n \times n}$ to be the matrix for which $\mathsf{mat}(r)^\top \cdot \mathsf{vec}(r') = \mathsf{vec}(r \cdot r')$ for all $r' \in \mathfrak{R}$. The ith column vector of $\mathsf{mat}(r)$ is equal to $\mathsf{vec}(rX^{i-1})$, $\mathsf{mat}(r)^{(i)} = \mathsf{vec}(rX^{i-1})$.

[3] In a recent eprint, [4] made the complexity analysis of [9] somewhat more precise. However, since it is currently only an eprint, we will ignore its analysis.

Let $K = 2n$ and $N > K$ be such that $n \mid N$. Let $\Psi^{\mathfrak{R},N,K}$ denote the distribution over $\mathbb{Z}_2^{N\times K}$, whose samples consist of $(N/n)\times 2$ square matrices from $\mathbb{Z}_2^{n\times n}$, where each square matrix is individually sampled as $\mathsf{mat}(r)$ for uniformly random $r \xleftarrow{r} \mathfrak{R}$. As in [8], we call the decisional $(\chi_N, N, K; \Psi^{\mathfrak{R},N,K})$-LPN problem the *decisional transposed ring-LPN (TRLPN) problem*. More precisely, in TRLPN, the adversary has access to $(f, \tau, \boldsymbol{R}, \boldsymbol{y})$, and has to guess whether $\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x} \oplus \boldsymbol{e}$, for random $\boldsymbol{x}$ and a small-weight $\boldsymbol{e}$, or $\boldsymbol{y}$ is random.

The decisional TRLPN problem is motivated by the fact that if $\boldsymbol{A} = \mathsf{mat}(a)$ and $\boldsymbol{b} = \mathsf{vec}(b)$, then $\boldsymbol{A}^\top \boldsymbol{b} = \mathsf{mat}(a)^\top \mathsf{vec}(b) = \mathsf{vec}(a \cdot b)$ can be computed by using a single ring multiplication, that depending on the choice of f can be computationally more efficient than a general matrix-vector product. More importantly, instead of communicating $2N/n$ matrices $\boldsymbol{A} \in \mathbb{Z}_2^{n\times n}$ as the public key, it suffices to communicate $2N/n$ ring elements, and memory requirements of all relevant algorithms will be reduced by a factor of n. See [10] for more motivation.

The difficulty of the TRLPN problem is positively correlated with the parameters n and τ. Since the efficiency mainly depends on n, one should choose as small n as possible such that there still exists a $\tau > 0$ so that the constructed primitive or protocol is secure. For an efficient implementation, it is also desirable that n is a power of 2.

For a ring $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$, let $t_\times^{\mathfrak{R}}$ denote the computational complexity of one ring multiplication (as a function of f and thus also of n).

3 Ring-LPN Based Commitment Scheme

In [11], the authors proposed an LPN-based commitment scheme. We follow a route that is common both in coding theory (in the context of cyclic codes) and cryptography (in the context of lattices but also LPN [10]), by embedding vectors from $\mathbb{Z}_2^n$ to the ring $\mathbb{Z}_2[X]/(f(X))$, where f is a well chosen (irreducible) degree-n polynomial. This enables to replace matrices with ring elements, and matrix-to-vector multiplications with ring multiplications. The most intricate part of the construction is its choice of parameters, coupled by a precise analysis of their correctness. The commitment scheme is given in Fig. 1.

We recall that the distribution $\Psi^{\mathfrak{R},N,K}$ is defined over $N \times K$ matrices. In particular, the expected value of $||\boldsymbol{e}||_1$ is equal to $\tau \cdot N$, and thus by the Hoeffding inequality, $||\boldsymbol{e}||_1 \leq D'$ with a high probability. For efficiency purposes, one should choose an irreducible $f(X)$ with a minimal number of non-zero monomials. Clearly, the commitment algorithm computes $(\boldsymbol{M}, \boldsymbol{R}) \cdot (m, r)^\top \oplus \boldsymbol{e}$.

Before stating Thm. 1 about the security of the commitment scheme, we first establish some technical lemmas. The first lemma motivates the stated lower bound $\beta \geq \lceil 2/(1-2\tau^*) \rceil$. Intuitively, we have an $[N, K, D]$ code for K as in the definition of the protocol, and $D = 2\tau^* N + 1$ as in Thm. 1. The Singleton bound gives us the following lower bound on N. This result is also necessary since there is a mutual dependency between τ^* (given in Eq. (1)) and N (defined in the commitment scheme). In an actual implementation, one should set N to be

Public Parameters Generation $\mathsf{gen}(1^\kappa, 1^\lambda)$**:** generate TRLPN parameters n and $\tau > 0$. Let τ^* be as in Eq. (1). Let $K = 2n$ and $N = \beta n$ for some integer $\beta \geq \lceil 2/(1-2\tau^*)\rceil$. Define $D' := \lfloor \tau^* \cdot N \rceil$. Choose a suitable degree-n irreducible polynomial f, and let $\mathfrak{R} := \mathbb{Z}_2[X]/(f(X))$. Choose uniformly random $\boldsymbol{M} \xleftarrow{r} \mathfrak{R}^{N/n}$ and $\boldsymbol{R} \xleftarrow{r} \mathfrak{R}^{N/n}$. Output $\mathsf{gk} := (f, \tau, \boldsymbol{M}, \boldsymbol{R})$.

Commitment $\mathsf{com}_{\mathsf{gk}}(m; \cdot, \cdot)$**:** If $m \notin \mathfrak{R}$, then reject. Choose random $r \xleftarrow{r} \mathfrak{R}$ and $\boldsymbol{e} \xleftarrow{r} \overline{\mathsf{Ber}}_\tau^{\mathfrak{R},N/n}$. Output $\mathsf{com}_{\mathsf{gk}}(m; r, \boldsymbol{e}) = (\boldsymbol{y}, z) \leftarrow (\boldsymbol{M}m \oplus \boldsymbol{R}r \oplus \boldsymbol{e}, (m, r))$ with $\boldsymbol{y} \in \mathfrak{R}^{N/n}$.

Verification $\mathsf{ver}_{\mathsf{gk}}(\boldsymbol{y}, z = (m, r))$**:** reject if $\boldsymbol{y} \notin \mathfrak{R}^{N/n}$, $m \notin \mathfrak{R}$, or $r \notin \mathfrak{R}$. Otherwise, compute $\boldsymbol{e} \leftarrow \boldsymbol{y} \oplus \boldsymbol{M}m \oplus \boldsymbol{R}r$, and accept iff $||\boldsymbol{e}||_1 \leq D'$.

Fig. 1. The commitment scheme

equal to the smallest multiplier of n greater or equal than the bound computed in Lem. 1, and only then compute τ^* from it according to Eq. (1).

Lemma 1. *Consider* $\tau < 1/2$, $\tau^* < 1/2$ *as in Eq.* (1), $K = 2n$, *and* $D = 2\tau^* N + 1$. *Then in any* $[N, K, D]$ *code,* $N \geq \frac{2}{1-2\tau^*} \cdot n = \frac{2}{1-2\tau} \cdot n - (\sqrt{4n(1-2\tau)\lambda \ln 2 + \lambda^2 \ln^2 2} - \lambda \ln 2)/(1-2\tau)^2$. *If* $n \mid N$, *then* $N \geq 3n$.

Proof. By the Singleton bound, $N \geq K + D - 1$. Due to the choice of K and D, this means that $N \geq 2n + 2\tau^* N$, and thus the minimum choice for N is $N = \frac{2}{1-2\tau^*} \cdot n$. Combining this value of N with τ^* as in Eq. (1), after solving a quadratic equation $(1-2\tau)N\sqrt{2\log_2 e} - 2\sqrt{\lambda N} - 2n\sqrt{2\log_2 e} = 0$, and taking the smaller of two solutions, $\sqrt{N} = (\sqrt{\lambda \ln 2} - \sqrt{\lambda \ln 2 + 4n(1-2\tau)})/\left(\sqrt{2}(1-2\tau)\right)$, we get the first claim of the current lemma. The second claim (i.e., that if $n \mid N$ then $N \geq 3n$) follows, since $\tau > 0$. □

One can take any value of N, $n \mid N$, that satisfies this lemma. To improve on efficiency, we recommend to choose N to be first integer larger than $K/(1-2\tau^*)$ that divides by n. In practice, since we have $\tau^* \leq 1/4$, we can always take $N = 2K$. However, when $\tau^* \leq 1/6$, we only need $N \geq \frac{3 \cdot 2n}{2} = 3n$ while $K = 2n$. Thus, in a paradoxical manner, a smaller τ^*, and thus a smaller τ, *may* help to improve the efficiency of this commitment scheme. However, as we will see later, such a small value of N would collapse the security for other reasons.

We show next that this commitment scheme is binding if the following probability $\mathsf{GV}^{\mathfrak{R}}$ (the *Gilbert-Varshamov bound for finite fields*; a variant of the well-known Gilbert-Varshamov bound) is (say) $1 - 2^{-\lambda}$.

Definition 1. *Let* f *be a degree-*n *polynomial that is irreducible over* $\mathbb{Z}_2$. *Assume that* $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$ *and* $0 < D \leq N - K + 1$. *Let* $(\mathfrak{R}^2)^*$ *denote* $\mathfrak{R}^2 \setminus \mathbf{0}_2$, *where* $0 \in \mathfrak{R}$. *Define*

$$\mathsf{GV}^{\mathfrak{R}}(N, K, D) := \min_{\boldsymbol{x} \in (\mathfrak{R}^2)^*} \Pr_{\boldsymbol{A} \xleftarrow{r} \mathfrak{R}^{(N/n) \times 2}} [||\boldsymbol{A} \cdot \boldsymbol{x}||_1 \geq D] \quad .$$

We bound $\mathsf{GV}^{\mathfrak{R}}$ under the assumption that f is irreducible. The following lemma basically states that a random linear code with a generator matrix, distributed according to $\Psi^{\mathfrak{R},N,K}$, meets the Singleton bound with a very high probability.

Lemma 2. *Let $N = \beta n$ for an integer $\beta \geq 2$, $K = 2n$, and D be as in the commitment scheme of Sect. 3 such that $D < N/3 + 2$. Assume that $0 < D \leq N - K + 1$. Then*

$$\mathsf{GV}^{\mathfrak{R}}(N, K, D) \geq 1 - 2^{-n(\beta - \beta(H_2((D-2)/(\beta n))+2))+1} \ .$$

Proof. For some $\boldsymbol{x} \in (\mathfrak{R}^2)^*$, denote $S := \Pr_{\boldsymbol{A} \xleftarrow{r} \mathfrak{R}^{\beta \times 2}}[||\boldsymbol{A} \cdot \boldsymbol{x}||_1 \leq D-1]$. Since $\mathfrak{R}$ is a field, every non-zero element of it is irreducible. (Here we need $\mathfrak{R}$ to be a field.) Thus, for any non-zero $m \in \mathfrak{R}$ and any $y_i \in \mathfrak{R}$, $i \in \{1, \beta\}$, $j \in \{1, 2\}$, $\Pr_{\boldsymbol{A}_{ij} \xleftarrow{r} \mathfrak{R}}[\boldsymbol{A}_{ij} m = y_i] = \Pr_{\boldsymbol{A}_{ij} \xleftarrow{r} \mathfrak{R}}[\boldsymbol{A}_{ij} = m^{-1} y_i] = \frac{1}{2^n}$. Thus, since $\boldsymbol{A}$ is chosen uniformly, $\boldsymbol{x} \neq \boldsymbol{0}_2$ and the vector consisting of uniformly chosen coordinates is uniformly chosen, then also $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is uniformly random.

Now, we estimate S as the number of all $\boldsymbol{y} \in \mathfrak{R}^\beta$ with $||\boldsymbol{y}||_1 \leq D-1$ divided by the ring size $|\mathfrak{R}^\beta|$. For $\boldsymbol{y} \in \mathfrak{R}^\beta$, let $\boldsymbol{y^*} \in \mathbb{Z}_2^N$ be its canonical representation as a bit-vector, i.e., $y^*_{in+j} = y_i[j]$. Clearly, $||\boldsymbol{y}||_1 = ||\boldsymbol{y^*}||_1$. Thus, we need to find S, the number of all $\boldsymbol{y^*} \in \mathbb{Z}_2^N$ with $||\boldsymbol{y^*}||_1 \leq D-1$, which is equal to $\sum_{j=1}^{D-1} S_j$, where $S_j = |\{\boldsymbol{y^*} : ||\boldsymbol{y^*}||_1 = j\}| = \binom{N}{j}$ is the number of vectors from $\mathbb{Z}_2^N$ that have exactly j non-zero coefficients. In other words, $S = B_N(D-2)/2^N$, where $B_N(D-2)$ is the size of the Hamming ball of radius $D-2$. Then using the following well-known bound for the sum of the first k binomial coefficients for fixed t, $0 \leq k \leq t/2$, $B_t(k) = \sum_{i=0}^{k} \binom{t}{i} \leq \binom{t}{k}(1 + k/(t-2k+1))$, since $D < N/2 + 2$ (this follows from $D < N/3 + 2$ that we made) we obtain $S = B_N(D-2)/2^N \leq \frac{1}{2^N} \cdot \binom{N}{D-2} \cdot (1 + (D-2)/(N-2D+5))$. Using the Stirling approximation of the factorial, it is easy to see that for every $0 \leq \alpha \leq 1$ it holds that $\lim_{t\to\infty} \frac{1}{t} \log_2 \binom{t}{\alpha t} = H_2(\alpha)$ while $\log_2 \binom{t}{\alpha t} \leq t H_2(\alpha)$, where $H_2(p) = -p \log_2 p - (1-p)\log_2(1-p)$, $p \in [0,1]$, is the standard binary entropy function. Thus $\binom{t}{k} \leq 2^{t H_2(k/t)}$, and we obtain that

$$S \leq \frac{2^{N \cdot H_2((D-2)/N)}}{2^N} \cdot \left(1 + \frac{D-2}{N-2D+5}\right). \tag{2}$$

Since by assumption $D < N/3 + 2$, we may replace $1 + \frac{D-2}{N-D+5}$ with 2, thus obtaining $S \leq 2^{N\, H_2((D-2)/N) - N + 1}$.

Now a union bound over all non-zero $\boldsymbol{x}$ implies that for all messages $\boldsymbol{x}$, it holds that $\Pr_{\boldsymbol{A}}[||\boldsymbol{A} \cdot \boldsymbol{x}||_1 \leq D-1] \leq 2^{2n} 2^{N\, H_2((D-2)/N) - N + 1}$, and $\mathsf{GV}^{\mathfrak{R}}(N, K, D) \geq 1 - 2^{-N(1 - H_2((D-2)/N)) + 2n + 1}$. □

We comment that we made only three approximations: the first one to bound the sum of binomial coefficients, the second one to bound a binomial by using its Stirling approximation, and then bounding a fraction in Eq. (2) by 2. All three approximations are very tight in their regions.

Theorem 1. *Consider the ring $\mathfrak{R} = \mathbb{Z}_2[X]/(f(X))$ for an irreducible degree-n polynomial f. Let $D = 2D' + 1$, where D' is as in the description of the commitment scheme Γ of the current section. Γ is statistically binding with probability $\mathsf{GV}^{\mathfrak{R}}(N, K, D)$. If the decisional $(\overline{\mathsf{Ber}}_\tau^{\mathfrak{R}, N/n}, N, K; \Psi^{\mathfrak{R}, N, K})$-LPN problem (i.e., a TRLPN problem) is (t, ε)-hard, then Γ is $(t - \Theta(t_\times^{\mathfrak{R}}), 2\varepsilon)$-computationally hiding.*

Proof. STATISTICAL BINDING: assume that for $(\boldsymbol{y_j}, z_j) = \mathsf{com}_{\mathsf{gk}}(m_j; r_j, \boldsymbol{e_j})$ where $j \in \{1,2\}$, $\boldsymbol{y_1} = \boldsymbol{y_2}$. Thus, $(\boldsymbol{M}, \boldsymbol{R}) \cdot (m_1 \oplus m_2, r_1 \oplus r_2)^\top = \boldsymbol{e_1} \oplus \boldsymbol{e_2}$. Since $(m_1, r_1) \neq (m_2, r_2)$, with probability $\mathsf{GV}^{\mathfrak{R}}(N, K, D)$, $||\boldsymbol{e_1} \oplus \boldsymbol{e_2}||_1 \geq D$. However, $||\boldsymbol{e_1} \oplus \boldsymbol{e_2}||_1 \leq ||\boldsymbol{e_1}||_1 + ||\boldsymbol{e_2}||_1 \leq D' + D' < D$. Contradiction with the choice of N.

COMPUTATIONAL HIDING: Assume by contradiction that $\mathcal{A} = \mathcal{A}_{\mathsf{hiding}}$ is a time $t_{\mathcal{A}}$ adversary that can break the hiding property of the new commitment scheme with probability $1/2 + \varepsilon_{\mathcal{A}}$ for some $\varepsilon_{\mathcal{A}} > 0$. We construct the following adversary $\mathcal{B} = \mathcal{B}_{\mathsf{lpn}}$ that breaks the decisional LPN assumption with the help of $\mathcal{A}$ in related time, with probability $1/2 + \varepsilon_{\mathcal{A}}/2$. From this, the claim follows.

1. The challenger first generates the parameters (f, τ). She sets $\beta \xleftarrow{r} \{1,2\}$. If $\beta = 1$, then she sets $\boldsymbol{y} \leftarrow \boldsymbol{R}r \oplus \boldsymbol{e}$ for $\boldsymbol{R} \xleftarrow{r} \mathfrak{R}^{N/n}$, $r \xleftarrow{r} \mathfrak{R}$ and $\boldsymbol{e} \xleftarrow{r} \overline{\mathsf{Ber}}_{\tau}^{\mathfrak{R},N/n}$. Otherwise, she sets $\boldsymbol{R} \xleftarrow{r} \mathfrak{R}^{N/n}$ and $\boldsymbol{y} \xleftarrow{r} \mathfrak{R}^{N/n}$. She sends $(f, \tau, \boldsymbol{R}, \boldsymbol{y})$ to $\mathcal{B}$.
2. $\mathcal{B}$ creates $\boldsymbol{M} \xleftarrow{r} \mathfrak{R}^{N/n}$. He sends $\mathsf{gk} \leftarrow (f, \tau, \boldsymbol{M}, \boldsymbol{R})$ to $\mathcal{A}$.
3. Given input gk, $\mathcal{A}$ sends to $\mathcal{B}$ a challenge pair (m_1, m_2).
4. $\mathcal{B}$ picks $\alpha \xleftarrow{r} \{1,2\}$. He sends $\boldsymbol{M}m_\alpha \oplus \boldsymbol{y}$ to $\mathcal{A}$. $\mathcal{A}$ answers with α'.
5. If $\alpha = \alpha'$ ($\mathcal{A}$ guessed correctly), then $\mathcal{B}$ outputs $\beta' \leftarrow 1$ (guesses that $\beta = 1$), otherwise $\mathcal{B}$ outputs $\beta' \leftarrow 2$ (guesses that $\beta = 2$).

Clearly, the computation of $\mathcal{B}$ is dominated by $t_{\mathcal{A}} + N/n \cdot t_{\times}^{\mathfrak{R}}$, where $t_{\times}^{\mathfrak{R}}$ denotes the computational complexity of one ring multiplication. Here, $N/n \cdot t_{\times}^{\mathfrak{R}}$ enters from the computation of the vector-to-scalar-product $\boldsymbol{M} \cdot m_\alpha$.

If $\beta = 1$, then $\boldsymbol{M}m_\alpha \oplus \boldsymbol{y} = \boldsymbol{M}m_\alpha \oplus \boldsymbol{R}r \oplus \boldsymbol{e}$, which is a valid output of $\mathsf{com}_{\mathsf{gk}}(m_\alpha; r, \boldsymbol{e})$, and by assumption on $\mathcal{A}$, $\mathcal{A}$ can guess α from this with probability $\frac{1}{2} + \varepsilon_{\mathcal{A}}$. If $\beta = 2$, then $\boldsymbol{M} \cdot m_\alpha \oplus \boldsymbol{y}$ is uniformly random (and thus does not depend on α), and thus $\mathcal{A}$ can guess α from this with probability $\frac{1}{2}$. By a standard argument, $\Pr[\beta' = \beta] = \Pr[\beta' = \beta | \beta = 1] \Pr[\beta = 1] + \Pr[\beta' = \beta | \beta = 2] \Pr[\beta = 2] = \Pr[\beta' = 1 | \beta = 1] \cdot \frac{1}{2} + \Pr[\beta' = 2 | \beta = 2] \cdot \frac{1}{2} = \Pr[\alpha = \alpha' | \beta = 1] \cdot \frac{1}{2} + \Pr[\alpha \neq \alpha' | \beta = 2] \cdot \frac{1}{2} = \left(\frac{1}{2} + \varepsilon_{\mathcal{A}}\right) \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon_{\mathcal{A}}}{2}$. Thus, $\mathcal{B}$ breaks the decisional LPN assumption with probability $1/2 + \varepsilon_{\mathcal{A}}/2$ in time that is dominated by $t_{\mathcal{A}} + N/n \cdot t_{\times}^{\mathfrak{R}}$ operations. □

4 Recommended Parameter Choices

To achieve binding, we must assume that the parameters n and τ (and thus also D) are chosen so that $\mathsf{GV}^{\mathfrak{R}}(N, K, D) \geq 1 - 2^{-\lambda}$. On the other hand, to achieve computational hiding, n and τ are such that the decisional TRLPN problem is $(\kappa^{\omega(1)}, \kappa^{-\omega(1)})$-hard. Since the complexity of the best known attacks [9,4] depends intimately on the choice of several internal variables, we used the following strategy. We computed for every β from 10 to 25 (where $N = \beta n$ as before), the value of D such that Lem. 2 returns an upper bound $2^{-\lambda} \leq 2^{-40}$. We then found, for this D, the minimum value of β for which the attack of [9] has computational complexity of at least 2^{130} bit operations; this fixes also the maximum value of $\tau^* = D/(2\beta n)$ and thus of τ.

We now give more details. The most efficient known attack against LPN was recently published in [9]. This attack uses covering codes. Assume that we have

an $[n'', \ell]d_C$ covering code, for certain parameters n'' and ℓ. As in [9], assume that d_C is the smallest integer, such that $\sum_{i=0}^{d_C} \binom{n''}{i} > 2^{n''-\ell}$. The latter optimistic estimate comes from the sphere-covering bound (i.e., assuming that there is a perfect $[n'', \ell]$ code with covering radius d_C). In reality, for most of the values n'' it is not known how to construct such codes; thus, in practice, the attack from [9] has worse complexity than we estimate in what follows.

In bit operations, the computational complexity of the attack from [9] is $2^{f_{n,\tau}(q,a,t,b,w_0,w_1,\ell,n'')}$, where

$$f_{n,\tau}(q,a,t,b,w_0,w_1,\ell,n'') := T_{pre} + \frac{aqn + (n+1)tq + m\sum_{i=0}^{w_0}\binom{n'-n''}{i}i + (n''-\ell)(2m+2^\ell) + \ell 2^\ell \sum_{i=0}^{w_0}\binom{n'-n''}{i}}{\Pr(w_0, n'-n'')\cdot \Pr(w_1, n'')}. \tag{3}$$

Here T_{pre} is the precomputation time of the Four Russian Matrix Inversion algorithm [1], the rest is the complexity of five-step LPN solving algorithm using covering codes [9]. Here (n, τ) are parameters of the LPN instance, q is the number of queries, $m = q - n - t2^b$, and q satisfies $q - t2^b > 1/(\varepsilon^{2^{t+1}} \cdot (\varepsilon')^{2w_1})$, where $\varepsilon = 1 - 2\tau$, $\varepsilon' = 1 - \frac{2d_C}{n''}$. The lower bound for q is due to the fast Walsh-Hadamard transform used in the solving phase of the algorithm from [9]. The probability $\Pr(w, j) = \sum_{i=0}^{w}(1-\tau)^{j-i}\tau^i\binom{j}{i}$ expresses the possibility of having at most w errors in j positions, therefore the denominator of Eq. (3) is the success probability in one iteration. See [9,4] for detailed explanation of the parameters. The only value that directly depends on d_C is ε', and the latter only gives a lower bound on $q - t2^b$.

We note that [9,4] did not add the term T_{pre} to the computational complexity. According to [1] (page 145), $T_{pre} = \sum_{i=0}^{a-1}(3\cdot 2^s - 4)(n - is - s) = -\frac{1}{2}a(3\cdot 2^s - 4)(as - 2n + s)$ in the case of a $n \times n$ matrix, where s is a parameter such that $a = \lceil n/s \rceil$. Assuming $a = n/s$, $T_{pre} \approx 1.5 \cdot 2^{n/a}(a-1)n$, and thus for any given n we can find numerically a value a that results in $T_{pre} \approx 2^{128}$.

In particular, in the most interesting case when $n = 1024$ (choosing n to be a power of 2 makes it possible to use a number of optimizations), we are forced to take $\beta = 19$, resulting in $\tau = 0.128118$. In this case, the best parameters for the attack from [9] that we found are (here we use the notation from [9]; see [9] for a definition of each parameter) $q \approx 2^{104.9}$, $a = 9$, $t = 6$, $b = 101$, $w_0 = 2$, $c = 30$, $\ell = 101$, and $n'' = 390$. With those parameters, the attack from [9] (when using Eq. (3) for computational complexity) takes approximate time $2^{131.1}$. However, the given complexity formula of this attack assumes the existence of a perfect $[390, 101]$ covering code. Since there is no such perfect code, the actual attack will be presumably less efficient.

Since the number of multiplications the new commitment scheme uses is 2β, and each multiplications takes $\Theta(n^2)$ bit-operations, we can estimate the computational complexity by measuring the parameter $2\beta n^2$. We emphasize that the actual computational complexity can only be measured by an optimized implementation, see Sect. 5.

5 Efficiency Issues And Implementation

Recall that the length of the public key is $\Theta(n)$ bits (with the recommended parameters, $N/n = 38$ ring elements, and up to n bits to describe f.) The commitment and the verification time are both dominated by $38t_{\times}^{\mathfrak{R}}$ bit-operations. In the case where $\mathfrak{R}$ supports Fast Fourier Transform (e.g., when $f(X) = X^n - 1$), then $t_{\times}^{\mathfrak{R}} = \Theta(n \log n)$. Then, both time complexities are $\Theta(n \log n)$. In the case $f(X)$ is irreducible, one cannot implement the usual Fast Fourier Transform. While Cyclotomic Fast Fourier Transform [17] has additive complexity $O(n^2/(\log n)^{\log_2(8/3)})$, we leave implementing that algorithm as a further work and — for the sake of simplicity — concentrate instead on quadratic-time algorithms. One reason for that is to emphasize that this commitment scheme is extremely competitive even in the case of suboptimal implementations.

Importantly, the new commitment scheme is parallelizable. First, all 38 field multiplications, needed in one commitment or verification, can be performed in parallel. On top of it, every field multiplication can be parallelized by itself. In particular, in the field multiplication $a(X) = b(X)c(X)$ every coefficient a_i can be computed in parallel. Since all values a_i are independent, this means that parallelization of factor of n can be achieved. In practice, however it may be faster to compute some w coefficients at once, where w is either the machine word length or some other related constant.

Based on such considerations, we implemented a single finite field $\mathbb{F}_{2^{1024}}$ multiplication on several modern data-parallel computational architectures. More precisely, we used the OpenCL environment that is an open standard for the general-purpose computation for GPU-s. In addition to GPU-s, one can use OpenCL to develop parallel implementations on modern multicore CPU-s.

We report implementation results on three different platforms. First, NVIDIA's rather old mobile GPU Quadro 2000M[4]. Second, on AMD's gaming GPU Radeon HD 7950, and third, on Intel's Core i7-2860QM CPU. (See Tbl. 1). We remark that the used CPU supports 256-bit integer operations via AVX (Advanced Vector Extensions). According to information on Bitcoin mining (see footnote 2), some of the cutting edge GPU-s perform 35 times faster than the Quadro 2000M (not even talking about CPU-s).

In what follows, we describe a partial implementation of the LPN-based commitment scheme and of the ring-LPN based commitment scheme.

LPN-Based Commitment Scheme. First, we implemented a partial version of the ring-LPN based commitment scheme, by first precomputing (once) the matrices $\mathsf{mat}(\boldsymbol{M}_i)$ and $\mathsf{mat}(\boldsymbol{R}_i)$, and then implementing only the matrix-to-vector multiplication. This means that the CPU has to store the whole matrix $\mathsf{mat}(x)$ ($n^2/8$ bytes, i.e., 128 KiB when $n = 1024$). (In the ring-based implementation, described later, memory consumption will be obviously smaller.) Here, we used L-bit (for a parameter L that depends on the concrete GPU/CPU) operations,

[4] `http://www.nvidia.com/content/PDF/product-comparison/Product-Comparison-Quadro-mobile-series.pdf`, accessed in June 2015

this means that an L-bit entry-wise product can be implemented as a single word-wide AND operation.

Moreover, we implemented both uncoalesced and coalesced field multiplication. In the coalesced implementation, we parallelized the work so that every byte of $\boldsymbol{y}$ in the multiplication $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is computed by a different GPU core. This means that we utilize $n/8$ cores. (I.e., 128 cores, when $n = 1024$. If there are less — say c — cores available, then every core has to execute $n/(8c)$ threads.) Since each core computes 8 coefficients of $a(X)$, its computation is dominated asymptotically by $8n/L$ (word-wide) AND and XOR operations, on top of which one has to add $8 \log_2 L$ bit-operations that are required to compute Hamming weight, together with some additional operations.

In the uncoalesced implementation, every core computes a single coefficient of $\boldsymbol{y}$. This is followed by a short epilogue where the outputs of eight consequent cores are combined into one output byte. Here, we need n cores (i.e., given c cores, every core has to execute n/c threads). Every core's computation is dominated asymptotically by n/L (word-wide) AND and XOR operations, followed by $\log_2 L$ bit-operations to compute Hamming weight. However, our uncoalesced implementation requires 3 synchronized rounds to combine the results of consequent cores into one output byte. Every such round has to start with a synchronization (**barrier** in OpenCL). Since synchronization is somewhat costly (and we also need more cores), in some of the cases an coalesced implementation (that theoretically requires 8 times more computation) is actually faster.

To optimize the throughput, we had to include some hand optimizations. First, we had to find out the optimal unroll count: if the loops are not unrolled at all, then the computation time is dominated by the costly branch instructions. However, if there are too many unrolls, then due to the way OpenCL operates, there is going to be a large usage of hardware registers, which makes computation lower. The latter specifically affects the GPU-s of NVIDIA due to the worse optimization by the compiler. To take this into account, in the case of the Quadro 2000M, we also used the NVIDIA's extension (via compiler flag `-cl-nv-maxrregcount`) to OpenCL that allows to limit the number of used registers to some value `reg`. We again chose the value `reg` carefully so as to increase the throughput. (No such extension exists for AMD's GPU or Intel's CPU.)

To summarize, in the case $n = 1024$ we obtain the results given in Tbl. 1. We expect that a carefully parallelized implementation of the new commitment scheme will be much faster on HD 7950 and other top-of-the-line GPU-s. Moreover, we used the OpenCL library for the compatibility with both NVIDIA's and AMD's GPU-s (and with multicore CPU-s). If one is interested in the top performance on the NVIDIA's GPU-s only, one could use the CUDA library (or even program in the PTX virtual assembly language). Due to the larger dependency on the hardware, a well-optimized CUDA program is usually significantly faster than an OpenCL program on the same hardware platform. The same comments hold also for the implementation ring-LPN based commitment scheme that we describe in the next subsubsection.

Table 1. Some values about the used GPU-s and CPU-s as returned by the OpenCL's `clGetDeviceInfo` command, together with our implementation data

	GPU 1	GPU 2	CPU 1
`clGetDeviceInfo` string	Return value	Return value	Return value
`DEVICE_NAME`	`Tahiti`	`Quadro 2000M`	`Intel(R) Core(TM) i7-2860QM CPU @ 2.50GHz`
`DEVICE_VENDOR`	`Advanced Micro Devices, Inc.`	`NVIDIA Corporation`	`Intel(R) Corporation`
`DEVICE_VERSION`	`OpenCL 1.2 AMD-APP (1642.5)`	`OpenCL 1.1 CUDA`	`OpenCL 1.2 (Build 57)`
`DRIVER_VERSION`	`1642.5 (VM)`	`347.52`	`5.0.0.57`
`DEVICE_MAX_COMPUTE_UNITS`	28	4	8
`DEVICE_MAX_CLOCK_FREQUENCY`	960	1100	2500
`DEVICE_GLOBAL_MEM_SIZE`	3221225472	2147483648	2147352576
Cores	1792	192	8
Optimal parameters and timing (matrix-to-vector)			
Implementation	Uncoalesced	Uncoalesced	Coalesced
L	`ulong2` (128 bits)	`ulong` (64 bits)	`ulong2` (128 bits)
max reg count	N/A	180	N/A
unroll	4	5	11
mult per second (per core)	70 313.60	30 547.41	46 019.33
cycles per core	13 653.1	36 009.6	54 325.0
threads	128	128	1024
cycles x ♯threads / ♯cores	975.2	24006.4	6953600
Timing (finite field multiplication)			
mult per second (per core)	51923.78	70861.68	35348.18
cycles per core	18488.640	15523.20	70725
Threads	272	272	272
cycles x ♯threads / ♯cores	2821.5	21991	2404650

Ring-LPN Based Commitment Scheme. We also implemented a binary finite field $\mathbb{F}_{2^{1024}}$ multiplication. Here, we used a version of the Brauer's exponentiation algorithm (first used in the context of finite field multiplication by Lopéz and Dahab [14]). That is, in the computation of a $\mathbb{F}_{2^{1024}}$ multiplication $c(X) = a(X)b(X)$, we first precompute $a(X)b'(X)$ for all degree-($\leq W$) polynomials $b'(X)$. After that, we use a parallel variant of the school book multiplication method, where each thread uses W-bit precomputed values to compute an L-bit intermediate result in an $(n/L) \times (2n/L)$ matrix. (Note that there are $(n/L) \cdot (n/L+1)$ threads, since the rest of the entries of this matrix are equal to 0.) We then sum up in parallel the entries in every column of the intermediate matrix, obtaining a degree $2n$ polynomial $c'(X)$, and then reduce $c'(X)$ modulo $f(X)$. Since we chose $f(X)$ with a small Hamming weight (namely, 5), the reduction step is almost negligible. In our implementation, $L = 64$ and $W = 4$; those constants were chosen to minimize the execution time. This means that the maximal number of threads is 272. Similar strategy was outlined in say [5], but our implementation is independent.

Since the CPU has 8 cores and the HD 7950 GPU has 1792 cores, the 7950 GPU is approximately 850 times faster, see the last row of Tbl. 1. Thus, on the HD 7950 GPU, one can implement $1792/272 \approx 6.59$ finite field multiplications

in parallel. Since one commitment (and verification) requires 38 multiplications, one can schedule on average $1792/(272 \cdot 38) = 0.173$ commitments of 1024-bit numbers in parallel.

Interestingly, our implementation of the finite field multiplication on the HD 7950 GPU and the Intel CPU is somewhat slower than the matrix-to-vector multiplication (in cycles per core), while on the Quadro 2000M the opposite is true. The relative slow-down on the first two processing units is due to the fact that in our implementation of matrix-to-vector multiplication, we use the `ulong2` data type to perform 128-bit operations in parallel, while in our implementation of finite field multiplications, we did only use the 64-bit `ulong` data type. The implementation of finite-field implementations also uses more threads than the the matrix-to-vector multiplication. This can mean that if the communication and storage of the public key is not a bottleneck, one might actually want to implement the (non-ring) LPN based commitment scheme.

Comparison with Pedersen Commitment. In the case of the simplest discrete logarithm-based commitment scheme, the Pedersen commitment, the committer has (m, r) and then computes $y = g^m h^r$. The verifier just recomputes y, given the same m and r. Assuming that one uses elliptic curves, the committer's and the verifier's computation is dominated by two exponentiations.

Efficiency-wise, the main difference between the described commitment scheme and the Pedersen scheme is that in the former, one has to execute a small number of multiplications (over a medium-sized finite field) while in the Pedersen scheme one has to execute a small number of *exponentiations* (in elliptic curves defined over a small finite field). Every exponentiation requires at least κ finite field multiplications in a field of the size $\approx 2^{2\kappa}$. Thus, Pedersen with $\kappa = 128$ uses at least $128/38 \approx 3.4$ times more multiplications than the new commitment scheme. Moreover, Pedersen has additional overhead due to the use of much more complicated elliptic-curve group multiplications instead of simpler finite-field multiplications. Finally, Pedersen is not as readily parallelizable as the new commitment scheme, and it only allows to commit to 256-bit strings instead of 1024-bit strings. For concrete numbers, we refer to say [7] for a recent highly optimized implementation (that uses a 254-bit base field) of an elliptic curve exponentiation with $\geq 100\,000$ cycles. Thus in their implementation of Pedersen, it takes at least 787 cycles to commit to a bit as compared with ≈ 104.7 cycles to implement the new commitment scheme on the AMD GPU.

Finally, recall that discrete logarithm is not secure against quantum computers, while LPN is assumed to be. Hence, the described commitment scheme is not only more efficient, but also presumably more (quantum-)secure. Moreover, recent attacks have indicated that discrete logarithm might not be as secure against conventional (non-quantum) computers as thought up to now.

Acknowledgments. The first author was supported by Estonian Research Council and European Union through the European Regional Development

Fund. The second author was supported by institutional research funding IUT20-57 of the Estonian Ministry of Education and Research.

References

1. Bard, G.V.: Algebraic Cryptanalysis. Springer (2009)
2. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic Primitives Based on Hard Learning Problems. In: CRYPTO 1993. LNCS, vol. 773, pp. 278–291
3. Blum, A., Kalai, A., Wasserman, H.: Noise-Tolerant Learning, the Parity Problem, And the Statistical Query Model. In: STOC 2000, pp. 435–440
4. Bogos, S., Tramèr, F., Vaudenay, S.: On Solving LPN using BKW and Variants. Technical Report 2015/049, International Association for Cryptologic Research (2015) Available at `http://eprint.iacr.org/2015/049`, last retrieved version from 2015.01.30.
5. Bose, U., Bhattacharya, A.K., Das, A.: GPU-Based Implementation of 128-Bit Secure Eta Pairing over a Binary Field. In: AFRICACRYPT 2013. LNCS, vol. 7918, pp. 26–42
6. Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: Covering Codes. North-Holland Mathematical Library, vol. 54. North Holland (2005)
7. Costello, C., Hisil, H., Smith, B.: Faster Compact Diffie-Hellman: Endomorphisms on the x-line. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 183–200
8. Damgård, I., Park, S.: Is Public-Key Encryption Based on LPN Practical? Technical Report 2012/699, International Association for Cryptologic Research (2012) Available at http://eprint.iacr.org/2012/699, last checked revision from 8 Oct 2013.
9. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN Using Covering Codes. In: ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 1–20
10. Heyse, S., Kiltz, E., Lyubashevsky, V., Paar, C., Pietrzak, K.: Lapin: An Efficient Authentication Protocol Based on Ring-LPN. In: FSE 2012. LNCS, vol. 7549, pp. 346–365
11. Jain, A., Krenn, S., Pietrzak, K., Tentes, A.: Commitments and Efficient Zero-Knowledge Proofs from Learning Parity with Noise. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 663–680
12. Katz, J., Shin, J.S.: Parallel and Concurrent Security of the HB and HB$^+$ Protocols. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87
13. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In: ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389
14. López, J., Dahab, R.: High-Speed Software Multiplication in F_{2^m}. In: INDOCRYPT 2000. LNCS, vol. 1977, pp. 203–212
15. Pedersen, T.P.: Non-Interactive And Information-Theoretic Secure Verifiable Secret Sharing. In: CRYPTO 1991. LNCS, vol. 576, pp. 129–140
16. Roth, R.: Introduction to Coding Theory. Cambridge University Press (2006)
17. Wu, X., Wang, Y., Yan, Z.: On Algorithms and Complexities of Cyclotomic Fast Fourier Transforms Over Arbitrary Finite Fields. IEEE Transactions on Signal Processing **60**(3) (2012) pp. 1149–1158