Simple Chosen-Ciphertext Security from Low-Noise LPN

Eike Kiltz[1⋆], Daniel Masny[1⋆⋆], Krzysztof Pietrzak[2⋆⋆⋆]

[1] Horst-Görtz Institute for IT Security and Faculty of Mathematics, Ruhr-Universität Bochum
[2] IST Austria

Abstract. Recently, Döttling et al. (ASIACRYPT 2012) proposed the first chosen-ciphertext (IND-CCA) secure public-key encryption scheme from the learning parity with noise (LPN) assumption. In this work we give an alternative scheme which is conceptually simpler and more efficient. At the core of our construction is a trapdoor technique originally proposed for lattices by Micciancio and Peikert (EUROCRYPT 2012), which we adapt to the LPN setting. The main technical tool is a new double-trapdoor mechanism, together with a trapdoor switching lemma based on a computational variant of the leftover hash lemma.

1 Introduction

The Learning Parity with Noise (LPN) problem has found a wide range of applications in symmetric cryptography, including encryption [1] and authentication [2,3,4]. Public-key primitives seem considerably harder to achieve. In particular, it is still an open problem to construct a public-key encryption scheme from the (standard) LPN problem. The LPN problem is very attractive, because of its similarity to the well-studied syndrome decoding problem and its assumed hardness in a post-quantum world. Further, many LPN based schemes are very efficient, such that they can be used even in low-cost RFID devices.

The first step towards a public-key encryption (PKE) scheme from LPN was made by Alekhnovich [5] who proposed a chosen-plaintext (IND-CPA) secure PKE based on a low-noise variant of LPN (low-noise LPN). A straightforward variant of Alekhnovich's PKE scheme can be seen as the LPN analog of Regev's encryption scheme from the learning with errors (LWE) problem. The LPN problem states that the distribution $\mathcal{D}_{\mathrm{LPN}_{n,m,p}} = \{(\mathbf{A}, \mathbf{As}+\mathbf{e}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_p^m, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n\}$ is indistinguishable from uniform, where $\mathcal{B}_p$ is the the Bernoulli distribution with parameter p, i.e., $\Pr[x = 1 \ : \ x \leftarrow \mathcal{B}_p] = p$. Whereas in standard LPN the Bernoulli parameter p is constant ($p = 0.1$ is a typical choice), in low-noise LPN we have $p = \Theta(1/\sqrt{n})$, where n is the dimension of the LPN

⋆ Supported by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.
⋆⋆ Supported by the DFG Research Training Group GRK 1817/1.
⋆⋆⋆ Supported by the European Research Council, ERC Starting Grant (259668-PSPC).

secret. When we decrease the Bernoulli parameter p, the LPN problem can only become easier. Indeed, while the best known algorithm for solving standard LPN runs in time $2^{O(n/\log n)}$ [6], low-noise LPN can be solved in time $2^{O(\sqrt{n})}$. Hence, for low-noise LPN, the dimension of the LPN secret has to be increased accordingly, which results in less efficient schemes. See also [7] for concrete efficiency considerations.

CCA-secure encryption from low-noise LPN. Recently, Döttling, Müller-Quade and Nascimento [8] showed how to extend Alekhnovich's IND-CPA secure scheme in order to get a chosen-ciphertext (IND-CCA) secure scheme. Like for Alekhnovich's scheme, their security proof is in the standard model (in particular, no random oracles), and relies on the low noise LPN assumption. During decryption only a certain part of the secret key is known and a q-ary erasure code is used to reconstruct the missing parts. Due to the additional overhead of the erasure code, the scheme has to add matrices $\mathbf{B}_1, \ldots, \mathbf{B}_q$ to the public-key, where the parameter q is estimated in [8] to be at least 400.[3] Hence the complexity of the scheme is estimated to be a couple of hundred times worse than Alekhnovich's scheme.

CCA-secure encryption from LWE. In a work predating [8], Micciancio and Peikert [9] extended Regev's LWE-based encryption scheme into a simple and efficient IND-CCA secure encryption scheme. Both schemes are randomness-recovering, but unlike [8], the scheme from [9] does not use erasure codes which results in considerably more compact (public and secret) keys.

This raises the question, whether it's possible to shorten the keys in the LPN setting by using the techniques from [9]. Unfortunately, it turns out that a straight forward application of their techniques will not work. Informally, using the leftover hash lemma as in [9] in the (binary) LPN setting results in an error that cannot be corrected using error correcting codes.

1.1 Our Contributions

In this work, we propose a simple and efficient IND-CCA secure PKE scheme from low-noise LPN. Compared to the IND-CPA secure scheme by Alekhnovich, we only loose roughly a factor two in efficiency.[4]

At a technical level, we design a new (tag-based) *double trapdoor function* which has two independent trapdoors. Each of these trapdoors depends on a hidden tag. If the function is evaluated with respect to a hidden tag, the corresponding trapdoor disappears and the function is hard to invert. For all other tags, the function can be inverted efficiently using one of the trapdoors. Our *switching lemma* (Lemma 4) shows that, under the LPN assumption, the hidden tags contained in the trapdoors can be switched without being noticed by any efficient adversary. The main difference to [9] is that we replace the leftover hash

[3] Stating a more exact value for q is difficult as in [8] no upper bound is given and the analysis is only sketched.

[4] Interestingly, the factor two between IND-CPA and IND-CCA security is also observed in the Diffie-Hellman world [10,11] and in the lattice world [12].

lemma by a computational variant based on the LPN problem with low noise. We use this double-trapdoor function to construct a tag-based encryption (TBE) scheme. (The latter can be efficiently transformed into a CCA-secure encryption scheme [13].) During the security reduction from low-noise LPN, we replace the first hidden tag with the challenge tag. Since this step is only *computationally indistinguishable* we have to give a security reduction in which the simulator has no access to the trapdoor being switched to the challenge tag. Here the second trapdoor is used to answer decryption queries correctly. Once both hidden tags are switched to the challenge tag, the simulator is not able to decrypt a message related to this challenge tag which allows us to argue about indistinguishability of the PKE scheme. We remark that previous LPN or LWE-based switching techniques (e.g., [9]) relied on purely statistical arguments such that a second trapdoor was not needed for simulating the decryption queries.

Efficiency. This tag-based encryption scheme directly implies a CCA-secure PKE scheme [13]. Compared to [8], this results in much smaller key sizes and comparable ciphertext size. Concretely, our scheme only has to add two matrices $\mathbf{B}_0, \mathbf{B}_1$ to the public-key and hence we expect the keys of our scheme be to a couple of hundred times smaller than that of [8]. We remark that our techniques can be extended to the case of LWE, but the resulting scheme is worse than the one from [9]. While for LPN replacing the leftover hash lemma is *necessary* to decrease the weight of the error, replacing the leftover hash lemma in the LWE setting actually has the opposite effect.

1.2 Open Problems

Designing an IND-CPA secure PKE from LPN with constant noise remains an open problem. Already a construction with any noise level $\omega(1/\sqrt{n})$ would be interesting to achieve.

2 Preliminaries

We use bold lower-case letters like $\mathbf{a} \in \mathbb{Z}_2^n$ to denote vectors and bold upper-case letters like $\mathbf{A} \in \mathbb{Z}_2^{n \times n}$ for matrices. With $|\mathbf{a}|$ we denote the Hamming weight (i.e., the number of 1's) of $\mathbf{a}$. We denote by $x \xleftarrow{\$} X$ that x is sampled according to the distribution X. If X is a set, then this denotes that x is sampled uniformly at random from X. Instead of using $\oplus$ for addition modulo 2, we use $+$ and $-$ to get a more generic construction, which adapts more easily to larger fields (for which addition and subtraction is not the same) as used in the LWE assumption.

2.1 The Bernoulli Distribution

$\mathcal{B}_p$ denotes the Bernoulli distribution with parameter $0 \leq p \leq 1/2$, i.e., $x \xleftarrow{\$} \mathcal{B}_p$ is the random variable over $\{0, 1\}$ with $\Pr[x = 1] = p$. To bound the tail of the sum of independent Bernoulli random variabels, we will use the following Chernoff bounds

Chernoff bound: For $\mathbf{d} \xleftarrow{\$} \mathcal{B}_p^m$ and $\delta > 0$:

$$\Pr_{\mathbf{d}}[|\mathbf{d}| > (1+\delta)pm] < e^{-\frac{\min(\delta,\delta^2)}{3}pm} \tag{1}$$

$$\text{in particular, for } \delta = 1 \qquad \Pr_{\mathbf{d}}[|\mathbf{d}| > 2pm] < e^{-pm/3} \tag{2}$$

2.2 Learning Parity with Noise

Let $n \in \mathbb{N}$ be the size of the secret solution vector, $m > n$ the number of the given samples and $0 \leq p \leq 1/2$ the Bernoulli parameter of the noise distribution.

The $\mathrm{LPN}_{n,m,p}$ Problem. The $\mathrm{LPN}_{n,m,p}$ problem is the problem of solving a set of linear equations perturbed by some noise. To define the decision version of LPN we consider the distribution

$$\mathcal{D}_{\mathrm{LPN}_{n,m,p}} = ((\mathbf{A}, \mathbf{As}+\mathbf{e}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times n}, \mathbf{e} \xleftarrow{\$} \mathcal{B}_p^m, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n).$$

The challenge is to distinguish $\mathcal{D}_{\mathrm{LPN}_{n,m,p}}$ from uniform $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_2^{m\times n} \times \mathbb{Z}_2^m$. The advantage of an algorithm A in breaking the $\mathrm{LPN}_{n,m,p}$ assumption is

$$\mathbf{Adv}_{\mathrm{LPN}_{n,m,p}}(\mathsf{A}) = |\Pr[\mathsf{A}(\mathbf{A}, \mathbf{b}) = 1] - \Pr[\mathsf{A}(\mathbf{A}', \mathbf{b}') = 1]|,$$

where $(\mathbf{A}, \mathbf{b}) \xleftarrow{\$} \mathcal{D}_{\mathrm{LPN}_{n,m,p}}$ and $(\mathbf{A}', \mathbf{b}') \xleftarrow{\$} \mathbb{Z}_2^{m\times n} \times \mathbb{Z}_2^m$.

The hardness of $\mathrm{LPN}_{n,m,p}$ depends on the choice of the secret size n, the amount of samples m and the error distribution $\mathcal{B}_p$. Whereas in the standard LPN assumption the Bernoulli parameter p is constant, we use the "low-noise" version with $p \approx 1/\sqrt{n}$.

Below we introduce two variants of $\mathrm{LPN}_{n,m,p}$ which we'll use in our construction, both variants are basically equivalent to the standard $\mathrm{LPN}_{n,m,p}$ assumption.

Knapsack LPN. The knapsack LPN distribution [14] is

$$\mathcal{D}_{\mathrm{KLPN}^m_{n,m,p}} = ((\mathbf{A}, \mathbf{EA}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times(m-n)}, \mathbf{E} \xleftarrow{\$} \mathcal{B}_p^{m\times m}).$$

and the advantage of an A is defined as

$$\mathbf{Adv}_{\mathrm{KLPN}^m_{n,m,p}}(\mathsf{A}) = |\Pr[\mathrm{A}(\mathbf{A}, \mathbf{EA}) = 1] - \Pr[\mathrm{A}(\mathbf{A}, \mathbf{B}') = 1]|,$$

where $(\mathbf{A}, \mathbf{EA}) \xleftarrow{\$} \mathcal{D}_{\mathrm{KLPN}^m_{n,m,p}}$ and $\mathbf{B}' \xleftarrow{\$} \mathbb{Z}_2^{m\times(m-n)}$.

Knapsack LPN is as hard as LPN, the reduction stated below loses a factor of m due to a standard hybrid argument because we directly defined the m-fold knapsack LPN distribution (i.e., $\mathbf{E}$ contains m vectors, not just one).

Lemma 1. *For all algorithms* B *there exists an algorithm* A *that runs in roughly the same time as* A *and* $\mathbf{Adv}_{\mathrm{LPN}_{n,m,p}}(\mathsf{A}) \geq \frac{1}{m}\mathbf{Adv}_{\mathrm{KLPN}^m_{n,m,p}}(\mathsf{B})$.

Extended Knapsack LPN. The Knapsack LPN problem remains hard in the presence of additional leakage $\mathbf{Ez}$ about $\mathbf{E}$. The extended knapsack EKLPN distribution is defined as

$$\mathcal{D}_{\mathrm{EKLPN}^m_{n,m,p}} = ((\mathbf{A}, \mathbf{EA}, \mathbf{z}, \mathbf{Ez}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m\times(m-n)}, \mathbf{E} \xleftarrow{\$} \mathcal{B}_p^{m\times m}, \mathbf{z} \xleftarrow{\$} \mathcal{B}_p^m)$$

and A's advantage is

$$\mathbf{Adv}_{\mathrm{EKLPN}^m_{n,m,p}}(\mathsf{A}) = |\Pr[\mathrm{A}(\mathbf{A}, \mathbf{EA}, \mathbf{z}, \mathbf{Ez}) = 1] - \Pr[\mathrm{A}(\mathbf{A}, \mathbf{B}, \mathbf{z}, \mathbf{Ez}) = 1]|,$$

where $(\mathbf{A}, \mathbf{EA}, \mathbf{z}, \mathbf{Ez}) \xleftarrow{\$} \mathcal{D}_{\mathrm{EKLPN}^m_{n,m,p}}$ and $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_2^{m\times(m-n)}$.

The following lemma is a special case of [15, Theorem 3.1], who use a more general notion of the ELWE problem based on LWE and the leakage vector $\mathbf{z}$ is sampled from an arbitrary distribution (not just $\mathcal{B}_p^m$).

Lemma 2. *For all algorithms* B *there exists an algorithm* A *that runs in roughly the same time as* A *and* $\mathbf{Adv}_{\mathrm{LPN}_{n,m,p}}(\mathsf{A}) \geq \frac{1}{2m}\mathbf{Adv}_{\mathrm{EKLPN}^m_{n,m,p}}(\mathsf{B})$.

2.3 Asymptotically Good Codes

In order to state the security of our scheme in asymptotic terms we need asymptotically good linear codes, these are $[m, Rm, \delta m]$ codes with a constant rate R, constant relative distance δ and arbitrary large block length m. Moreover, we want the code to be efficiently constructible in order to get a uniform construction, and of course encoding and decoding need to be efficient to get an efficient scheme. Such codes exist:

Lemma 3 ([16]). *For any rate* $0 < R < 1$*, there exists a binary linear error-correcting code family which is polynomial time constructible, encodable and decodable and can decode from up to* $\lfloor \frac{\delta n}{2} \rfloor$ *errors where* $\delta \approx \frac{1}{2}(1 - R)$.

We emphasis, that for concrete instantiations of the scheme, an arbitrary, suitable error correction code can be used and the asymptotic behavior is not important.

2.4 Game-Based Proofs

We use game-based proofs [17]. A game G consists of an Initialize and a Finalize procedure, and possibly other procedures. An adversary A is executed in the game by first calling Initialize. Next, he can make arbitrary calls to the other procedures, some multiple times, some only once, depending on the specification of G. Finally, A makes one single call to Finalize which ends the game. The output of the game, denoted as G^{A}, is defined as the output of Finalize.

2.5 Tag-Based Encryption

A tag-based encryption scheme with tag-space $\mathcal{T}$ and message-space $\mathcal{M}$ consist of the following three PPT algorithms TBE = (Gen, Enc, Dec).

- $\mathsf{Gen}(1^k)$ outputs a secret key sk and a public key pk.
- $\mathsf{Enc}(pk, \tau, M)$ outputs a ciphertext c of $M \in \mathcal{M}$ with respect to tag $\tau \in \mathcal{T}$.
- $\mathsf{Dec}(sk, \tau, C)$ outputs the decrypted message M of ciphertext C with respect to tag $\tau \in \mathcal{T}$, or $\perp$.

We require the standard correctness condition $\mathsf{Dec}(sk, \tau, \mathsf{Enc}(pk, \tau, M)) = M$ for all τ, M and all (sk, pk) in the range of $\mathsf{Gen}(\cdot)$. To define security, let the advantage of an adversary A in the selective-tag weak CCA game [18] be

$$\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) = \left| \Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{TBE}} = 1] - \frac{1}{2} \right|,$$

where the games defining $\mathrm{G}_{\mathrm{TBE}}$ are defined in Figure 1. Here the term *selective* models the fact that A has to commit to the challenge tag τ^* in the beginning, before seeing the public-key.

$\underline{\mathsf{Initialize}(\tau^*)}$
$(sk, pk) \leftarrow \mathsf{Gen}(1^k)$
Return pk

$\underline{\mathsf{Challenge}(M_0, M_1)}$ //one time
$b_M \xleftarrow{\$} \{0,1\}$;
Return $\mathsf{Enc}(pk, \tau^*, M_{b_M})$

$\underline{\mathsf{Finalize}(d)}$
Return $(b_M = d)$

$\underline{\mathsf{queryDec}(\tau, C)}$//many times
If $\tau \neq \tau^*$ return $\mathsf{Dec}(sk, \tau, C)$
Else return $\perp$

Fig. 1. Games constituting $\mathrm{G}_{\mathrm{TBE}}$.

To construct an IND-CCA secure PKE, it is sufficient to construct a secure TBE (in the above sense) with tag-space $\mathcal{T}$ exponential in n [18]. The overhead of this transformation is small. It essentially consists of a one-time signature or a message-authentication code plus a commitment.

3 Tag-Based Encryption

3.1 Double Trapdoor Generator

We use a matrix representation $\mathbf{H}_\tau \in \mathbb{Z}_2^{n \times n}$ for finite field elements $\tau \in \mathbb{F}_{2^n}$ [19,12]. The structure of a finite field implies certain properties: $\mathbf{H}_\tau + \mathbf{H}_{\tau'} = \mathbf{H}_{\tau+\tau'}$ and $\mathbf{0} = \mathbf{H}_0$ for the zero element of the field. In particular all matrices $\mathbf{H}_\tau - \mathbf{H}_{\tau'} = \mathbf{H}_{\tau-\tau'} \neq \mathbf{H}_0$ for $\tau \neq \tau'$ are invertible.

Let n and m be two parameters and let $\mathbf{G} \in \mathbb{Z}_2^{m \times n}$ be a generator matrix for an efficiently decodable code. (This was called gadget matrix in [9].) The trapdoor generator is the following PPT algorithm which takes as input two tags $\tau_0, \tau_1 \in \mathbb{F}_{2^n}$:

$\underline{\mathsf{Gen}_{\mathrm{td}}(1^n, \tau_0, \tau_1) \to (\mathbf{T}_0, \mathbf{T}_1, ek)}$. Sample $\mathbf{T}_0, \mathbf{T}_1 \xleftarrow{\$} \mathcal{B}_p^{m \times m}$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$

$$\text{Let} \quad \mathbf{B}_0 := \mathbf{T}_0\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_0} \ , \ \mathbf{B}_1 := \mathbf{T}_1\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_1} \quad \text{and} \quad ek = (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1)$$

$\underline{\mathsf{Initialize}(t, \tau_0, \tau_1, \tau')}//\mathrm{G}_{\mathrm{real}}$
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, \tau_0, \tau_1)$
$\mathbf{z} \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{T} \xleftarrow{\$} \mathcal{B}_p^{m \times m}$;
Return $(\mathbf{T}_t, ek, \mathbf{z}, \mathbf{T}\mathbf{z})$

$\underline{\mathsf{Initialize}(t, \tau_0, \tau_1, \tau')}//\mathrm{G}_{\mathrm{uniform}}$
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, \tau_0, \tau_1)$;
Parse $ek = (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1)$
$\mathbf{B}'_t := \mathbf{B}_t$; $\mathbf{B}'_{\bar{t}} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$;
$ek' := (\mathbf{A}, \mathbf{B}'_0, \mathbf{B}'_1)$;
$\mathbf{z} \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{T} := \mathbf{T}_{\bar{t}}$;
Return $(\mathbf{T}_t, ek', \mathbf{z}, \mathbf{T}\mathbf{z})$

$\underline{\mathsf{Initialize}(t, \tau_0, \tau_1, \tau')}//\mathrm{G}_{\mathrm{corr}}$
$\tau'_t := \tau_t$; $\tau'_{\bar{t}} := \tau'$;
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, \tau'_0, \tau'_1)$;
$\mathbf{z} \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{T} := \mathbf{T}_{\bar{t}}$;
Return $(\mathbf{T}_t, ek, \mathbf{z}, \mathbf{T}\mathbf{z})$

$\underline{\mathsf{Finalize}(d)}$ $//\mathrm{G}_{\mathrm{real,uniform,corr}}$
Return d

Fig. 2. Procedures defining games $\mathrm{G}_{\mathrm{real}}$, $\mathrm{G}_{\mathrm{uniform}}$, $\mathrm{G}_{\mathrm{corr}}$. Here $ek := (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1) = (\mathbf{A}, \mathbf{T}_0\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_0}, \mathbf{T}_1\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_1})$ as defined in Section 3.1.

Looking ahead, a trapdoor $\mathbf{T}_i$ ($i \in \{0,1\}$) output by $\mathsf{Gen}_{\mathrm{td}}(1^n, \tau_0, \tau_1)$ can be used to invert the tag-based trapdoor function

$$f^\tau(\mathbf{s}, \mathbf{e}, \mathbf{e}'_0, \mathbf{e}'_1) = (\mathbf{A}\mathbf{s} + \mathbf{e}, (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_0)\mathbf{s} + \mathbf{e}'_0, (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_1)\mathbf{s} + \mathbf{e}'_1).$$

whenever $\tau \neq \tau_i$ (for one $i \in \{0,1\}$) and the error $\mathbf{T}_i\mathbf{e} + \mathbf{e}'_i$ is small enough, so it can be corrected (using the code given by $\mathbf{G}$) as follows: given $(\mathbf{c}, \mathbf{c}_0, \mathbf{c}_1)$ $(= f^\tau(\mathbf{s}, \mathbf{e}, \mathbf{e}'_0, \mathbf{e}'_1))$, compute $\left(\mathbf{T}_i \ \mathbf{I}\right) \cdot \left(-\mathbf{c} \ \mathbf{c}_i\right)^\intercal = \mathbf{G}\mathbf{H}_{\tau - \tau_i}\mathbf{s} - \mathbf{T}_i\mathbf{e} + \mathbf{e}'_i$, use error correction to decode $\mathbf{H}_{\tau - \tau_i}\mathbf{s}$, and then the invertability of $\mathbf{H}_{\tau - \tau_i}$ (recall that we assume $\tau \neq \tau_i$) to reconstruct $\mathbf{s}$ The remaining inputs $\mathbf{e}, \mathbf{e}'_0, \mathbf{e}'_1$ can now easily be computed.

The "switching lemma" below states that under the LPN assumption, the output of the trapdoor generator computationally hides the tags τ_0, τ_1, even if there is some additional information about the trapdoor leaked. This lemma will allow us to switch either τ_0 or τ_1 to an arbitrary tag $\tau' \in \mathbb{Z}_2^n$. During the switching procedure, we still have access to the other trapdoor. This allows us to answer decryption queries during a CCA security proof.

Lemma 4. *For every PPT algorithm* A *there exists a PPT algorithm* B *such that:*

$$|\Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{real}} = 1] - \Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{corr}} = 1]| \leq 3m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}).$$

where games $\mathrm{G}_{\mathrm{real}}$ *and* $\mathrm{G}_{\mathrm{corr}}$ *are defined in Figure 2.*

Proof. The proof follows by the following two equations combined with Lemma 1 and 2

$$|\Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{real}} = 1] - \Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{uniform}} = 1]| \leq \mathbf{Adv}_{\mathrm{KLPN}^m_{n,m,p}}(\mathsf{B}) \tag{3}$$

$$|\Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{corr}} = 1] - \Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{uniform}} = 1]| \leq \mathbf{Adv}_{\mathrm{EKLPN}^m_{n,m,p}}(\mathsf{B}), \tag{4}$$

where game $\mathrm{G}_{\mathrm{uniform}}$ is also defined in Figure 2.

To prove (3) we construct an algorithm B which on input a $\mathcal{D}_{\mathrm{KLPN}^m_{n,m,p}}$ or a random sample, simulates $\mathrm{G}^{\mathsf{A}}_{\mathrm{real}}$ or $\mathrm{G}^{\mathsf{A}}_{\mathrm{uniform}}$, respectively. $\mathsf{B}(\mathbf{A}, \mathbf{B})$ simulates A's view as follows.

$\underline{\mathsf{Initialize}(t, \tau_0, \tau_1, \tau')}$
$\mathbf{z} \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{T}, \mathbf{T}_t \xleftarrow{\$} \mathcal{B}_p^{m \times m}$;
$\mathbf{B}_t := \mathbf{T}_t\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_t}$; $\mathbf{B}_{\overline{t}} := \mathbf{B} - \mathbf{G}\mathbf{H}_{\tau_{\overline{t}}}$;
$ek := (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1)$;
Return $(\mathbf{T}_t, ek, \mathbf{z}, \mathbf{T}\mathbf{z})$

$\underline{\mathsf{Finalize}(d)}$
Return d

We now analyse $\mathsf{B}(\mathbf{A}, \mathbf{B})$'s simulation. $\mathbf{A}$ is always uniform, $\mathbf{z}$ and $\mathbf{T}\mathbf{z}$ are distributed as in the real and random game. B generates $\tau_{\overline{t}}$, $\mathbf{T}_t$ and $\mathbf{B}_t$ exactly as $\mathsf{Gen}_{\mathrm{td}}$.

KLPN **Case:** $\mathbf{B} = \mathbf{T}_{\overline{t}}\mathbf{A}$ implies that $\mathbf{B}_{\overline{t}} = \mathbf{T}_{\overline{t}}\mathbf{A} - \mathbf{G}\mathbf{H}_{\tau_{\overline{t}}}$ has the same distribution as in $\mathrm{G}_{\mathrm{real}}$. Hence B simulates $\mathrm{G}_{\mathrm{real}}$ and $\Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{real}} = 1] = \Pr[\mathsf{B}(\mathbf{A}, \mathbf{B}) = 1 \mid (\mathbf{A}, \mathbf{B}) \xleftarrow{\$} \mathcal{D}_{\mathrm{KLPN}^m_{n,m,p}}]$.

Uniform Case: $\mathbf{B}$ is uniform and this implies that $\mathbf{B}_{\overline{t}}$ is uniform, too. Since $\mathbf{B}_{\overline{t}}$ is independent of $\mathbf{T}_{\overline{t}}$, $\mathbf{T}\mathbf{z}$ has the correct distribution. Hence B simulates $\mathrm{G}_{\mathrm{uniform}}$ and $\Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{uniform}} = 1] = \Pr[\mathsf{B}(\mathbf{A}, \mathbf{B}) = 1 \mid (\mathbf{A}, \mathbf{B})$ uniform$]$.

This concludes the proof of (3).

To prove (4) we use the $\mathrm{EKLPN}^m_{m-n,m,p}$ assumption. We reuse B, now with input $\mathsf{B}(\mathbf{A}, \mathbf{z}, \mathbf{B}, \mathbf{b})$ and change it slightly by setting $\mathbf{B}_{\overline{t}} := \mathbf{B} - \mathbf{G}\mathbf{H}_{\tau'}$ and replacing $\mathbf{T}\mathbf{z}$ by $\mathbf{b} = \mathbf{T}_{\overline{t}}\mathbf{z}$ during the $\mathsf{Initialize}$ procedure. With almost the same argument B simulates in the uniform case $\mathrm{G}_{\mathrm{uniform}}$ and in the LPN case $\mathrm{G}_{\mathrm{corr}}$ correctly.

3.2 Description of the Scheme

Our scheme uses the following parameters whose concrete choices will be justified later.

- The dimension n of the LPN secret (with $n = \Theta(k^2)$) and $m \geq 2n$ controlling the security of the scheme. (See Theorem 2.)

– A constant $0 < c < 1/4$ defining:
  - The Bernoulli parameter $p = \sqrt{c/m}$.
  - The bounding parameter $\beta = 2\sqrt{cm}$ to check consistency during decryption.
  - A binary linear error-correcting code $\mathbf{G} : Z_2^n \to Z_2^m$ which corrects up to αm errors for some α with $4c < \alpha < 1$.
– Further, we use an efficient error correcting code with generator matrix $\mathbf{G}_2 : \mathcal{M} \to \mathbb{Z}_2^\ell$ where the parameter $\ell \geq m$ is chosen such that the encoding scheme is able to correct at least $2\ell\sqrt{c}/\sqrt{m} = 2\ell p$ errors (note that $\mathbf{G}$ must correct a constant fraction of errors, whereas $\mathbf{G}_2$ only needs to correct a square root fraction).

The following three algorithms describe our TBE $= (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ based on LPN with tag space $\mathcal{T} = \mathbb{F}_{2^n} \setminus \{0\}$:

$\underline{\mathsf{Gen}(1^k) \to (sk, pk)}$**.** The algorithm calls the trapdoor generator $\mathsf{Gen}_{\mathrm{td}}(1^n, 0, 0) \to (\mathbf{T}_0, \mathbf{T}_1, (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1))$ and picks $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$. The private and public key is defined as

$$sk := (0, \mathbf{T}_0) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^{m \times m}, \quad pk := (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1, \mathbf{C}) \in (\mathbb{Z}_2^{m \times n})^3 \times \mathbb{Z}_2^{\ell \times n}.$$

(Recall that $\mathbf{B}_i = \mathbf{T}_i \mathbf{A}$.)

$\underline{\mathsf{Enc}(pk, \tau, M) \to C = (\mathbf{c}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)}$**.** The algorithm picks

$$\mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_p^m; \;\; \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_p^\ell; \;\; \mathbf{T}'_0, \mathbf{T}'_1 \xleftarrow{\$} \mathcal{B}_p^{m \times m} \text{ and } \mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$$

and defines

$$\begin{aligned} \mathbf{c} &:= \mathbf{A}\mathbf{s} + \mathbf{e}_1 && \in \mathbb{Z}_2^m \\ \mathbf{c}_0 &:= (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_0)\mathbf{s} + \mathbf{T}'_0\mathbf{e}_1 && \in \mathbb{Z}_2^m \\ \mathbf{c}_1 &:= (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_1)\mathbf{s} + \mathbf{T}'_1\mathbf{e}_1 && \in \mathbb{Z}_2^m \\ \mathbf{c}_2 &:= \mathbf{C}\mathbf{s} + \mathbf{e}_2 + \mathbf{G}_2(M). && \in \mathbb{Z}_2^\ell \end{aligned}$$

$\underline{\mathsf{Dec}(sk, \tau, C) \to (M \;\; \textbf{or} \;\; \perp)}$**.** The algorithm parses $sk = (\tau_0, \mathbf{T}_0)$ and computes

$$\tilde{\mathbf{c}}_0 := \begin{pmatrix} \mathbf{T}_0 & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} -\mathbf{c} \\ \mathbf{c}_0 \end{pmatrix} \quad (= \mathbf{G}\mathbf{H}_{\tau-\tau_0}\mathbf{s} + (\mathbf{T}'_0 - \mathbf{T}_0)\mathbf{e}_1).$$

Then it uses the error correction property of $\mathbf{G}$ to reconstruct $\mathbf{H}_{\tau-\tau_0}\mathbf{s}$ (from the error $(\mathbf{T}'_0 - \mathbf{T}_0)\mathbf{e}_1$), and further computes $\mathbf{s} = \mathbf{H}_{\tau-\tau_0}^{-1}\mathbf{H}_{\tau-\tau_0}\mathbf{s}$. If

$$|\underbrace{\mathbf{c} - \mathbf{A}\mathbf{s}}_{\mathbf{e}_1}| \leq \beta \;\wedge\; |\underbrace{\mathbf{c}_0 - (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_0)\mathbf{s}}_{\mathbf{T}'_0\mathbf{e}_1}| \leq \frac{\alpha m}{2} \;\wedge\; |\underbrace{\mathbf{c}_1 - (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_1)\mathbf{s}}_{\mathbf{T}'_1\mathbf{e}_1}| \leq \frac{\alpha m}{2} \tag{5}$$

is true, compute $\mathbf{c}_2 - \mathbf{C}\mathbf{s} = \mathbf{G}_2(M) + \mathbf{e}_2$ and reconstruct (using the error correction property of $\mathbf{G}_2$) M and output it, otherwise output $\perp$.

The scheme has a couple of straightforward simplifications which we did not apply in order to facilitate the proof. First, $\tau_0 = 0$ can be omitted from sk and the description of the scheme. Second, the matrix $\mathbf{B}_1$ can be chosen uniformly. (The latter is shown implicitly in the proof.)

We also remark that that scheme is randomness-recovering and can therefore also be seen as an adaptive tag-based trapdoor function [13], where the domain consists of sampling $(\mathbf{s}, \mathbf{e}_1, \mathbf{T}'_0\mathbf{e}_1, \mathbf{T}'_1\mathbf{e}_1, \mathbf{e}_2)$ as in Enc.

A discussion how to transform the TBE scheme into a IND-CCA secure encryption scheme is done in Appendix A.

3.3 Correctness and Equivalence of the Trapdoors

Theorem 1 (Corectness). *Let $\mathbf{G}$, $\mathbf{G}_2$ be the codes given above. Then with overwhelming probability over the choice of the public and secret keys and for all $\tau \in \mathcal{T}$, $M \in \mathcal{M}$, $\mathsf{Dec}(sk, \tau, C)$ outputs M with overwhelming probability over $C \leftarrow \mathsf{Enc}(pk, \tau, M)$.*

Proof. We start with showing why the chosen $\beta = 2\sqrt{cm}$, $p = \sqrt{c/m}$ are suitable for our application. The Chernoff bound 2 yields:

$$\Pr_{\mathbf{e} \xleftarrow{\$} \mathcal{B}_p^m}[|\mathbf{e}| > \underbrace{\beta}_{=2pm}] < e^{-pm/3} = 2^{-\Theta(\sqrt{m})} \tag{6}$$

The analysis of our choice of the constants $4c < \alpha < 1$ is a bit more involved. We start by upper bounding the probability p' that the inner product $\mathbf{t}^T\mathbf{e}$ of $\mathbf{e}$ with a vector $\mathbf{t} \xleftarrow{\$} \mathcal{B}_p^m$ is 1, assuming the Hamming weight of $\mathbf{e}$ is at most β. Note that a necessary condition for $\mathbf{t}^T\mathbf{e} = 1$ is that $\mathbf{t}[i] = 1$ for at least one of the i's where $\mathbf{e}[i] = 1$. We use this in the second step below, the third step follows by the union bound

$$p' = \Pr_{\mathbf{t}}[\mathbf{t}^T\mathbf{e} = 1 \mid |\mathbf{e}| \leq \beta] \leq \Pr_{\mathbf{t}}[\exists i \ : (\mathbf{e}[i] = 1) \wedge (\mathbf{t}[i] = 1) \mid |\mathbf{e}| \leq \beta] \leq \beta p = 2c$$

Let $\mathbf{T} \xleftarrow{\$} \mathcal{B}_p^{m \times m}$. By the Chernoff bound (1) we have with $\delta = \alpha/(2p') - 1$ (note that $p' \leq 2c < \alpha/2$)

$$\Pr_{\mathbf{T}}\left[|\mathbf{Te}| > \frac{\alpha}{2}m \mid |\mathbf{e}| \leq \beta\right] = \Pr_{\mathbf{T}}\left[|\mathbf{Te}| > (1+\delta)p'm||\mathbf{e}| \leq \beta\right] < e^{-\frac{\min(\delta,\delta^2)}{3}p'm}. \tag{7}$$

Now $\delta p' = \alpha/2 - p' \geq \alpha/2 - 2c > 0$ and $\delta = \alpha/(2p') - 1 \geq \alpha/(4c) - 1 > 0$ are lower bounded by constants and therefore

$$\Pr_{\mathbf{T}}\left[|\mathbf{Te}| > \frac{\alpha}{2}m \mid |\mathbf{e}| \leq \beta\right] < e^{-\frac{\min(\delta,\delta^2)}{3}p'm} = 2^{-\Theta(m)}. \tag{8}$$

As C is a properly generated ciphertext

$$|\mathbf{e}_1| \leq \beta \ \ \wedge \ \ |\mathbf{T}_0\mathbf{e}_1| \leq \frac{\alpha m}{2} \ \ \wedge \ \ |\mathbf{T}_1\mathbf{e}_1| \leq \frac{\alpha m}{2}$$

holds with overwhelming probability $1 - 2^{-\Theta(\sqrt{m})}$ by (6) and (8), assume this is the case. Then by the error correction property of the code $\mathbf{G}$ we decode the correct $\mathbf{s}$ from $\tilde{\mathbf{c}}_0 := \mathbf{G}\mathbf{H}_{\tau-\tau_0}\mathbf{s} + (\mathbf{T}'_0 - \mathbf{T}_0)\mathbf{e}_1$ since the error term satisfies $|(\mathbf{T}'_0 - \mathbf{T}_0)\mathbf{e}_1| \leq \alpha m$. Moreover, the consistency check 5 will pass.

It remains to show that the correct message M is reconstructed. We use $\mathbf{G}_2$ to derive M from $\mathbf{c}_2 - \mathbf{C}\mathbf{s} = \mathbf{e}_2 + \mathbf{G}_2(M)$, which gives the correct M if the Hamming weight of $\mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_p^\ell$ lies within the $2\ell\sqrt{c}/\sqrt{m} = 2\ell p$ bits error correction capacity of $\mathbf{G}_2$. Using the Chernoff bound 2 we can upper bound the probability of this not being the case (the last step uses $\ell \geq m$, which we assumed is the case)

$$\Pr_{\mathbf{e}_2}[|\mathbf{e}_2| > 2\ell p] < e^{-\ell p/3} = e^{-\ell\sqrt{c}/3\sqrt{m}} = 2^{-\Omega(\sqrt{m})}$$

The next lemma will be central in our security proof. It states that the output distribution of a decryption oracle is basically independent of which of two possible secret keys the oracle uses to decrypt.

Lemma 5. *Let* $\mathsf{Dec0} = \mathsf{Dec}$ *and let* $\mathsf{Dec1}$ *be defined like* Dec*, except that* $\mathbf{c}_1$ *instead of* $\mathbf{c}_0$ *is used to reconstruct* $\mathbf{s}$*. Then, with overwhelming probability over the choice of the public and secret keys,* $\mathsf{Dec0}$ *and* $\mathsf{Dec1}$ *have the same output distribution: Let* $(\mathbf{T}_0, \mathbf{T}_1, (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1)) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, \tau_0, \tau_1)$*,* $sk_0 = (\tau_0, \mathbf{T}_0)$*,* $sk_1 = (\tau_1, \mathbf{T}_1)$ *and* $pk := (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1, \mathbf{C})$ *with* $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$*. Then*

$$\Pr_{pk, sk_0, sk_1}[\forall \tau_0, \tau_1, \tau \notin \{\tau_0, \tau_1\}, C : (\mathsf{Dec0}(sk_0, \tau, C) = \mathsf{Dec1}(sk_1, \tau, C)] \geq 1 - 2^{-\Theta(m)}$$

Proof. If $M = \mathsf{Dec0}(sk_0, \tau, C = (\mathbf{c}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2))$, then by the consistency check (5) of $\mathsf{Dec} = \mathsf{Dec0}$ we reconstruct some $\mathbf{s}$ where

$$\begin{aligned} & \mathbf{e} := \mathbf{c} - \mathbf{A}\mathbf{s} \text{ with } |\mathbf{e}| \leq \beta \\ \wedge \quad & \mathbf{t}_0 := \mathbf{c}_0 - (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_0)\mathbf{s} \text{ with } |\mathbf{t}_0| \leq \alpha m/2 \\ \wedge \quad & \mathbf{t}_1 := \mathbf{c}_1 - (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_1)\mathbf{s} \text{ with } |\mathbf{t}_1| \leq \alpha m/2 \end{aligned}$$

Using the above notation, the computation of $\mathsf{Dec1}(sk_1, \tau, C)$ can be expressed as

$$\tilde{\mathbf{c}}_1 := \mathbf{c}_1 - \mathbf{T}_1\mathbf{c} = (\mathbf{G}\mathbf{H}_\tau + \mathbf{B}_1)\mathbf{s} + \mathbf{t}_1 - \mathbf{T}_1\mathbf{A}\mathbf{s} - \mathbf{T}_1\mathbf{e} = \mathbf{G}\mathbf{H}_{\tau-\tau_1}\mathbf{s} + \mathbf{t}_1 - \mathbf{T}_1\mathbf{e}.$$

$\mathsf{Dec1}(sk_1, \tau, C)$ reconstructs the same $\mathbf{s}$ if the error term $|\mathbf{t}_1 - \mathbf{T}_1\mathbf{e}|$ is at most $\leq \alpha m$. We already know that $|\mathbf{t}_1| \leq \alpha m/2$. Thus, by the triangle inequality it is sufficient to show $|\mathbf{T}_1\mathbf{e}| \leq \alpha m/2$ to guarantee the correct decoding of $\mathbf{s}$. By (8), the probability that this is the case when we chose some $\mathbf{e}$ satisfying $|\mathbf{e}| = \beta'$ (for any $\beta' \leq \beta$) at *random*, is

$$\Pr_{\mathbf{e}, |\mathbf{e}| = \beta', \mathbf{T}_1}[|\mathbf{T}_1\mathbf{e}| \leq \alpha m/2] \geq 1 - 2^{-\Theta(m)},$$

We need the above to hold fore every small $\mathbf{e}$, not just a randomly chosen one. Taking the union bound over all $2^{\log(m)O(\sqrt{m})}$ possible $\mathbf{e} \in \mathbb{Z}_2^m$ satisfying $|\mathbf{e}| \leq \beta = 2\sqrt{cm} = \Theta(\sqrt{m})$ we further get

$$\Pr_{\mathbf{T}_1}[\forall \mathbf{e}, |\mathbf{e}| \leq \beta : \ |\mathbf{T}_1\mathbf{e}| \ \leq \ \alpha m/2] \geq 1 - 2^{-\Theta(m) + \log(m)O(\sqrt{m})} \ = \ 1 - 2^{-\Theta(m)}$$

This shows that with overwhelming probability over the choice of $\mathbf{T}_1$ the same $\mathbf{s}$, and thus also the same message M is computed by $\mathsf{Dec1}(sk_1, \tau, C)$. The proof that whenever $\mathsf{Dec1}$ outputs some $M \neq \perp$, then $\mathsf{Dec0}$ must output the same M (with overwhelming probability over the choice of $\mathbf{T}_0$) is symmetric.

3.4 Proof of Security

Theorem 2 (CCA Security). *If the* LPN *assumption holds,* TBE *from Section 3.2 is secure against* selective-tag weak CCA *adversaries. In particular, for every PPT algorithm* A *there exist PPT algorithms* B *and* C *with roughly the same running time, such that:*

$$\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) \leq 6m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) + \mathbf{Adv}_{\mathrm{LPN}_{n,m+\ell,p}}(\mathsf{C}) + \mathrm{negl}(n).$$

Proof. Let A be an adversary attacking TBE. The games used in the proof are given in Figure 3, where G_1 is the same as the original TBE security game from Figure 1.

$\underline{\mathsf{Initialize}(\tau^*)}$ //G_1
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, 0, 0)$;
$\mathbf{e}^* \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_2^n$ $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$;
$\mathbf{c}^* := \mathbf{A}\mathbf{s}^* + \mathbf{e}^*$;
$\mathbf{T}_0^* \xleftarrow{\$} \mathcal{B}_p^{m \times m}$; $\mathbf{c}_0^* := (\mathbf{G}\mathbf{H}_{\tau^*} + \mathbf{B}_0)\mathbf{s}^* + \mathbf{T}_0^*\mathbf{e}^*$
$\mathbf{T}_1^* \xleftarrow{\$} \mathcal{B}_p^{m \times m}$; $\mathbf{c}_1^* := (\mathbf{G}\mathbf{H}_{\tau^*} + \mathbf{B}_1)\mathbf{s}^* + \mathbf{T}_1^*\mathbf{e}^*$
$sk = (0, \mathbf{T}_0)$;
Return $pk := (ek, \mathbf{C})$

$\underline{\mathsf{Initialize}(\tau^*)}$ //$\mathrm{G}_{2,3}$
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, 0, \tau^*)$;
$\mathbf{e}^* \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_2^n$ $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$
$\mathbf{c}^* := \mathbf{A}\mathbf{s}^* + \mathbf{e}^*$;
$\mathbf{T}_0^* \xleftarrow{\$} \mathcal{B}_p^{m \times m}$; $\mathbf{c}_0^* := (\mathbf{G}\mathbf{H}_{\tau^*} + \mathbf{B}_0)\mathbf{s}^* + \mathbf{T}_0^*\mathbf{e}^*$
$\mathbf{T}_1^* := \mathbf{T}_1$; $\mathbf{c}_1^* = \mathbf{T}_1^*\mathbf{c}^*$;
$sk = (0, \mathbf{T}_0)$; //G_2
$sk = (\tau^*, \mathbf{T}_1)$; //$\mathrm{G}_3$
Return $pk := (ek, \mathbf{C})$

$\underline{\mathsf{Initialize}(\tau^*)}$ //$\mathrm{G}_{4,5}$
$(\mathbf{T}_0, \mathbf{T}_1, ek) \leftarrow \mathsf{Gen}_{\mathrm{td}}(1^n, \tau^*, \tau^*)$
$\mathbf{e}^* \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_2^n$; $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$
$\mathbf{c}^* := \mathbf{A}\mathbf{s}^* + \mathbf{e}^*$; //$\mathrm{G}_4$
$\mathbf{c}^* \xleftarrow{\$} \mathbb{Z}_2^m$; //$\mathrm{G}_5$
$\mathbf{T}_0^* := \mathbf{T}_0$; $\mathbf{c}_0^* = \mathbf{T}_0^*\mathbf{c}^*$
$\mathbf{T}_1^* := \mathbf{T}_1$; $\mathbf{c}_1^* = \mathbf{T}_1^*\mathbf{c}^*$
$sk = (\tau^*, \mathbf{T}_1)$
Return $pk := (ek, \mathbf{C})$

$\underline{\mathsf{queryDec}(\tau, C)}$ //G_{1-5}
If $(\tau = \tau^*)$ Return $\perp$
Return $\mathsf{Dec}(sk, \tau, C)$

$\underline{\mathsf{Finalize}(d)}$ //G_{1-5}
Return $(b_M = d)$

$\underline{\mathsf{Challenge}(M_0, M_1)}$ //G_{1-4}
$b_M \xleftarrow{\$} \{0,1\}$;
$\mathbf{e}_2^* \xleftarrow{\$} \mathcal{B}_p^\ell$;
$\mathbf{c}_2^* := \mathbf{C}\mathbf{s}^* + \mathbf{e}_2^* + \mathbf{G}_2(M_{b_M})$
Return $C^* = (\mathbf{c}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*)$

$\underline{\mathsf{Challenge}(M_0, M_1)}$ //G_5
$\mathbf{c}_2^* \xleftarrow{\$} \mathbb{Z}_2^\ell$
Return C^*

Fig. 3. The different procedures of the games 1 to 5. G_1 is exactly the same as $\mathrm{G}_{\mathrm{TBE}}$, where the message-independent part of a $\mathsf{Challenge}$ query is already pre-computed in $\mathsf{Initialize}$.

From G_1 to G_2 we switch the hidden trapdoor tag of trapdoor $\mathbf{T}_1$ from 0 to τ^*.

Lemma 6. *There exists a PPT algorithm* B *such that*

$$\begin{aligned} &|\Pr[\mathrm{G}_1^{\mathsf{A}} = 1] - \Pr[\mathrm{G}_2^{\mathsf{A}} = 1]| \leq |\Pr[\mathrm{G}_{\mathrm{real}}^{\mathsf{B}} = 1] - \Pr[\mathrm{G}_{\mathrm{corr}}^{\mathsf{B}} = 1]| \\ &\leq 3m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}), \end{aligned}$$

where games $\mathrm{G}_{\mathrm{real}}$ *and* $\mathrm{G}_{\mathrm{corr}}$ *are defined in Figure 2..*

Proof. We describe algorithm B who simulates G_1 in $\mathrm{G}_{\mathrm{real}}$ or G_2 in $\mathrm{G}_{\mathrm{corr}}$.

$\underline{\mathsf{Initialize}(\tau^*)}$
$(ek, \mathbf{T}_0, \mathbf{e}^*, \mathbf{T}_1^*\mathbf{e}^*) \leftarrow \mathsf{Initialize}(0,0,0,\tau^*)$;
$\mathbf{C} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$; $\mathbf{s}^* \xleftarrow{\$} \mathbb{Z}_2^n$;
$\mathbf{c}^* := \mathbf{A}\mathbf{s}^* + \mathbf{e}^*$;
$\mathbf{T}_0^* \xleftarrow{\$} \mathcal{B}_p^{m \times m}$;
$\mathbf{c}_0^* := (\mathbf{G}\mathbf{H}_{\tau^*} + \mathbf{B}_0)\mathbf{s}^* + \mathbf{T}_0^*\mathbf{e}^*$;
$\mathbf{c}_1^* := (\mathbf{G}\mathbf{H}_{\tau^*} + \mathbf{B}_1)\mathbf{s}^* + \mathbf{T}_1^*\mathbf{e}^*$;
Return $pk := (ek, \mathbf{C})$

$\underline{\mathsf{queryDec}(\tau, C)}$
If $(\tau = \tau^*)$ Return $\perp$
Return $\mathsf{Dec}(sk = (0, \mathbf{T}_0), \tau, C)$

$\underline{\mathsf{Challenge}(M_0, M_1)}$
$b_M \xleftarrow{\$} \{0,1\}$;
$\mathbf{e}_2^* \xleftarrow{\$} \mathcal{B}_p^{\ell}$;
$\mathbf{c}_2^* := \mathbf{C}\mathbf{s}^* + \mathbf{e}_2^* + \mathbf{G}_2(M_{b_M})$
Return C^*

$\underline{\mathsf{Finalize}(d)}$
$\mathsf{Finalize}(b_M = d)$

The definition of $\mathrm{G}_{\mathrm{real}}$ and $\mathrm{G}_{\mathrm{corr}}$ imply the correctness of the output of $\mathsf{Initialize}$. $\mathbf{e}^*$ and $\mathbf{T}_1^*\mathbf{e}^*$ have the correct distribution, too. Hence B simulates G_1 in $\mathrm{G}_{\mathrm{real}}$ or G_2 in $\mathrm{G}_{\mathrm{corr}}$. The Lemma follows using Lemma 4.

In a next lemma we show that the adversary isn't able to distinguish whether the simulator uses the trapdoor $\mathbf{T}_0$ or $\mathbf{T}_1$ to answer decryption queries. To show this lemma, we use equivalence of the trapdoors shown in Lemma 5.

Lemma 7. $|\Pr[\mathrm{G}_2^{\mathsf{A}} = 1] - \Pr[\mathrm{G}_3^{\mathsf{A}} = 1]| \leq \mathrm{negl}(n)$.

Proof. We have to prove that an adversary can't figure out which trapdoor is used to answer decryption queries. Otherwise he is able to distinguish G_2 from G_3. Lemma 5 already shows that Dec has the same output for two different trapdoors with overwhelming probability, if the tags related to the trapdoors are not queried. In our case, τ_0 and τ_1 are either 0 or τ^*. The adversary is not allowed to query $0 \notin \mathcal{T}$ and τ^*. Hence he has only a negligible chance to distinguish G_2 and G_3

From G_3 to G_4 we switch the hidden trapdoor tag of trapdoor $\mathbf{T}_0$ from 0 to τ^*. Its proof is analogue to the one of Lemma 6 and therefore omitted.

Lemma 8. *There exists a PPT algorithm* B *such that*

$$|\Pr[\mathrm{G}_3^{\mathsf{A}} = 1] - \Pr[\mathrm{G}_4^{\mathsf{A}} = 1]| \leq |\Pr[\mathrm{G}_{\mathrm{real}}^{\mathsf{B}} = 1] - \Pr[\mathrm{G}_{\mathrm{corr}}^{\mathsf{B}} = 1]|$$
$$\leq 3m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}),$$

where games $\mathrm{G}_{\mathrm{real}}$ *and* $\mathrm{G}_{\mathrm{corr}}$ *are defined in Figure 2.*

In game G_5, the last game, we make the challenge ciphertext independent of the plaintexts M_0 and M_1.

Lemma 9. *There exists a PPT algorithm* C *such that*

$$|\Pr[\mathrm{G}_4^{\mathsf{A}} = 1] - \Pr[\mathrm{G}_5^{\mathsf{A}} = 1]| \leq \mathbf{Adv}_{\mathrm{LPN}_{n,m+\ell,p}}(\mathsf{C}).$$

Proof. We give a description of C and show, that he simulates G_4 and G_5 correctly. C receives a LPN challenge $(\mathbf{A}, \mathbf{C}), (\mathbf{b_A}, \mathbf{b_C})$ where $(\mathbf{b_A}, \mathbf{b_C}) = (\mathbf{As} + \mathbf{e}_1, \mathbf{Cs} + \mathbf{e}_2)$ or uniform.

$\underline{\mathsf{Initialize}(\tau^*)}$
$\mathbf{T}_0, \mathbf{T}_1 \xleftarrow{\$} \mathcal{B}_p^{m \times m}$;
$\mathbf{B}_0 := \mathbf{T}_0\mathbf{A} - \mathbf{GH}_{\tau^*}$;
$\mathbf{B}_1 := \mathbf{T}_1\mathbf{A} - \mathbf{GH}_{\tau^*}$;
$\mathbf{c}^* := \mathbf{b_A}$;
$\mathbf{T}_0^* := \mathbf{T}_0$; $\mathbf{c}_0^* := \mathbf{T}_0^*\mathbf{b_A}$;
$\mathbf{T}_1^* := \mathbf{T}_1$; $\mathbf{c}_1^* := \mathbf{T}_1^*\mathbf{b_A}$;
Return $pk := (\mathbf{A}, \mathbf{B}_0, \mathbf{B}_1, \mathbf{C})$

$\underline{\mathsf{queryDec}(\tau, C)}$
If $(\tau = \tau^*)$ Return $\perp$
Return $\mathsf{Dec}((\tau^*, \mathbf{T}_1), \tau, C)$

$\underline{\mathsf{Challenge}(M_0, M_1)}$
$b_M \xleftarrow{\$} \{0,1\}$;
$\mathbf{c}_2 := \mathbf{b_C} + \mathbf{G}_2(M_{b_M})$
Return C^*

$\underline{\mathsf{Finalize}(d)}$
$\mathsf{Finalize}(b_M = d)$

Now we analyse if C simulates correctly. First note that $\mathbf{A}$ and $\mathbf{C}$ are uniformly distributed, as required.

LPN **Case:** $\mathbf{b_A} = \mathbf{As} + \mathbf{e}_1$ and $\mathbf{b_C} = \mathbf{Cs} + \mathbf{e}_2$. To show that G_4 is simulated correctly, we have to show, that the distribution of c^* is correct. We implicitly set $\mathbf{s}^* = \mathbf{s}$, $\mathbf{e}_1^* = \mathbf{e}_1$ and $\mathbf{e}_2^* = \mathbf{e}_2$.

$\mathbf{c}^* := \mathbf{b_A} = \mathbf{As}^* + \mathbf{e}_1^*$
$\mathbf{c}_0^* := \mathbf{T}_0^*\mathbf{b_A} = (\mathbf{GH}_{\tau^*} - \mathbf{GH}_{\tau^*} + \mathbf{T}_0^*\mathbf{A})\mathbf{s}^* + \mathbf{T}_0^*\mathbf{e}_1^* = (\mathbf{GH}_{\tau^*} + \mathbf{B}_0)\mathbf{s}^* + \mathbf{T}_0^*\mathbf{e}_1^*$
$\mathbf{c}_1^* := \mathbf{T}_1^*\mathbf{b_A} = (\mathbf{GH}_{\tau^*} + \mathbf{B}_1)\mathbf{s}^* + \mathbf{T}_1^*\mathbf{e}_1^*$
$\mathbf{c}_2^* := \mathbf{b_C} + \mathbf{G}_2(M_{b_M}) = \mathbf{Cs}^* + \mathbf{e}_2^* + \mathbf{G}_2(M_{b_M})$.

Uniform Case: $\mathbf{c}_2^*$ is independent of M_{b_M} since $\mathbf{b_C}$ is uniformly distributed and hence $\mathbf{b_C} + \mathbf{G}_2(M_{b_M})$ is uniform, too. In this case, C simulates G_5.

In G_5, the challenge ciphertext is independent of the message and hence from the challenge bit b_M. The best, an adversary can do now, is to guess b_M and output the guess.

Lemma 10. $\Pr[G_5^{\mathsf{A}} = 1] = \Pr[G_5^{\mathsf{A}} = 0] = \frac{1}{2}$.

Combining the Lemmas 6–10 concludes the theorem:

$$
\begin{aligned}
&\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) \\
&= \left| \Pr[\mathrm{G}^{\mathsf{A}}_{\mathrm{TBE}} = 1] - \frac{1}{2} \right| \\
&\leq \left| \Pr[\mathrm{G}^{\mathsf{A}}_{2} = 1] + 3m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) - \frac{1}{2} \right| \\
&\leq \left| \Pr[\mathrm{G}^{\mathsf{A}}_{3} = 1] + \mathrm{negl}(n) + 3m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) - \frac{1}{2} \right| \\
&\leq \left| \Pr[\mathrm{G}^{\mathsf{A}}_{4} = 1] + \mathrm{negl}(n) + 6m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) - \frac{1}{2} \right| \\
&\leq \left| \Pr[\mathrm{G}^{\mathsf{A}}_{5} = 1] + \mathbf{Adv}_{\mathrm{LPN}_{n,m+\ell,p}}(\mathsf{C}) + \mathrm{negl}(n) + 6m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) - \frac{1}{2} \right| \\
&\leq 6m \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) + \mathbf{Adv}_{\mathrm{LPN}_{n,m+\ell,p}}(\mathsf{C}) + \mathrm{negl}(n).
\end{aligned}
$$

References

1. Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to encrypt with the LPN problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690. Springer, July 2008. 1
2. Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, December 2001. 1
3. Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and concurrent security of the HB and HB+ protocols. *Journal of Cryptology*, 23(3):402–421, July 2010. 1
4. Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 7–26. Springer, May 2011. 1
5. Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307. IEEE Computer Society Press, October 2003. 1
6. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd Annual ACM Symposium on Theory of Computing*, pages 435–440. ACM Press, May 2000. 2
7. Ivan Damgård and Sunoo Park. Is public-key encryption based on lpn practical? Cryptology ePrint Archive, Report 2012/699, 2012. `http://eprint.iacr.org/`. 2
8. Nico Döttling, Jörn Müller-Quade, and Anderson CA Nascimento. Ind-cca secure cryptography based on a variant of the lpn problem. In *Advances in Cryptology–ASIACRYPT 2012*, volume 7658, pages 485–503. Springer, 2012. 2, 3, 17
9. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, April 2012. 2, 3, 7

10. Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, August 2004. 2
11. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, August 2007. 2
12. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, May 2010. 2, 6
13. Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 673–692. Springer, May 2010. 3, 10
14. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, August 2011. 4
15. Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 334–352. Springer, May 2012. 5
16. Jørn Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652–656, 1972. 5
17. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006. 5
18. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, March 2006. 6
19. Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 177–191. Springer, August 2009. 6
20. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007. 17
21. Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, February 2005. 17
22. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 355–374. Springer, April 2012. 17

23. Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *Advances in Cryptology–ASIACRYPT 2012*, pages 663–680. Springer, 2012. 17

A IND-CCA Secure Encryption

There are generic constructions to transform a TBE to an IND-CCA PKE. [20] is based on one-time signatures (OTS). The other one is based on a message authentication code (MAC) and a commitment scheme [21].

To transform a TBE to an IND-CCA secure encryption, we do not use a OTS based on LPN like in [8], since this transformation would be too expensive. This would cause a large tag, since the the verification key of the OTS is the tag. Further, the size of the ciphertext grows with a bigger tag and the ciphertext will be signed with the OTS. In order to use this approach a collision resistant hash function is necessary to shrink the ciphertext to the message size signed by the signature.

More efficient is the technique based on a commitment scheme and a MAC [21]. LPN-based euf MACs have a large secret to which we have to commit [22]. This commitment is used as the tag for the TBE. A large secret key will cause again a large commitment, large tag and even larger ciphertext. The MAC is used to create a tag for the ciphertext of a TBE. The advantage of this transformation is, that we do not need a collision resistant hash function.

In the commitment and MAC-based transformation, the MAC has to be existential unforgeable given one tag query for an arbitrary message. A pairwise independent function fulfils this task in a less complex way and with a smaller secret compared to LPN-based MACs. But now we have to shrink the size of the ciphertext to the domain of the pairwise independent function. A collision resistant hash function leads to an efficient transformation of a TBE to an IND-CCA PKE. As alternative to a collision resistant hash function, we could also use an almost pairwise independent hash function instead of the pairwise independent hash function. As commitment scheme, we use the simple and efficient construction of [23]. Their commitment scheme is perfectly binding and computationally hiding.

B An IND-CPA Secure Public Key Encryption Scheme

The following three algorithms describe an IND-CPA-PKE = $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The scheme is a simplified version of the TBE to achieve just IND-CPA security. An IND-CPA adversary plays the $\mathrm{G}_{\mathrm{TBE}}$ without having access to $\mathsf{qeryDec}$. This makes the proof and hence the scheme much easier, since there is no need for having access to a trapdoor to answer decryption queries. Further an efficient error correction code $\mathbf{G}$ is required to reconstruct the message. This code corrects up to αm errors with $4c < \alpha < 1$ for Bernoulli parameter $p = \sqrt{c/m}$, and maps the message space $\mathcal{M}$ into $\mathbb{Z}_2^\ell$. The dimensions n, $m-n$ of the LPN secrets (with $n = \Theta(k^2)$) controll the security of the scheme.

$\underline{\mathsf{Gen}(1^k) \to (sk, pk)}$**.** The algorithm picks $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$, $\mathbf{T} \xleftarrow{\$} \mathcal{B}_p^{\ell \times m}$ and sets $\mathbf{C} = \mathbf{TA}$. The private key is $\mathbf{T}$ and the public key $pk := (\mathbf{A}, \mathbf{C})$.

$\underline{\mathsf{Enc}(pk, M) \to C = (\mathbf{c}, \mathbf{c}_2)}$**.** Sample $\mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_p^m$; $\mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_p^\ell$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$ and set

$$\mathbf{c} := \mathbf{As} + \mathbf{e}_1 \quad \text{and} \quad \mathbf{c}_2 := \mathbf{Cs} + \mathbf{e}_2 + \mathbf{G}(M).$$

$\underline{\mathsf{Dec}(sk, C) \to (M \textbf{ or } \perp)}$**.** The algorithm computes

$$\tilde{\mathbf{c}} := \begin{pmatrix} \mathbf{T} & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} -\mathbf{c} \\ \mathbf{c}_2 \end{pmatrix} \quad (= \mathbf{G}(M) - \mathbf{Te}_1 + \mathbf{e}_2).$$

Output M, which is reconstructed from $\tilde{\mathbf{c}}$ by using $\mathbf{G}$.

A Simple Trapdoor Function. By changing the construction a little bit, one obtains a simple trapdoor function. A trapdoor $\mathbf{T} \in \mathbb{Z}_2^{m \times n}$ output by $\mathsf{Gen}(1^n)$ can be used to invert the trapdoor function

$$f^\tau(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) = (\mathbf{As} + \mathbf{e}_1, (\mathbf{C} + \mathbf{G})\mathbf{s} + \mathbf{e}_2).$$

$\mathbf{s}$ is reconstructed by using the error correction of code $\mathbf{G}$. Details can be seen in the correctness of the PKE. When $\mathbf{s}$ is reconstructed, $\mathbf{e}_1$ and $\mathbf{e}_2$ are easily obtained by subtracting $\mathbf{As}$ and $(\mathbf{C} + \mathbf{G})\mathbf{s}$ from the output of the function.

Correctness. The encoding scheme has to correct an error $\mathbf{e}_2 - \mathbf{Te}_1$, for $\mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_p^m$ with overwhelming probability. The correctness follows from the correctness of the proposed TBE. To give an example, for message space $\mathbb{Z}_2^n$ and $\ell = m$, the generator matrix $\mathbf{G}$ of the TBE can be used. This encoding scheme is stronger than necessary, since it corrects even an error $\mathbf{T}_2\mathbf{e}_2 - \mathbf{T}_1\mathbf{e}_1$ for $\mathbf{e}_1, \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_p^m$, $\mathbf{T}_1, \mathbf{T}_2 \xleftarrow{\$} \mathcal{B}_p^{m \times m}$ with overwhelming probability.

Security.

Theorem 3. *If the* LPN *assumption holds,* PKE *is secure against* IND-CPA *adversaries. In particular, for every PPT algorithm* A *there exist PPT algorithms* B *and* C *with roughly the same running time, such that:*

$$\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) \leq \ell \cdot \mathbf{Adv}_{\mathrm{LPN}_{m-n,m,p}}(\mathsf{B}) + \mathbf{Adv}_{\mathrm{LPN}_{n,m+\ell,p}}(\mathsf{C}) + \mathrm{negl}(n).$$

Proof Sketch. First we switch $\mathbf{C} = \mathbf{TA}$ to a uniform $\mathbf{C}$. If an adversary has less advantage in the uniform setting, we will break the $\mathrm{KLPN}^\ell_{m-n,m,p}$ assumption. Then we switch $\mathbf{c}^*, \mathbf{c}_2^*$ to uniform by using a $\mathrm{LPN}_{n,m+\ell,p}$ instance $\mathbf{A}, \mathbf{C}, \mathbf{b}_\mathbf{A}, \mathbf{b}_\mathbf{C}$ and setting $\mathbf{c}^* = \mathbf{b}_\mathbf{A}$, $\mathbf{c}_2^* = \mathbf{b}_\mathbf{B} + \mathbf{G}(M)$. Now the ciphertext is uniform and the advantage of an adversary is negligible.