How to Encrypt with the LPN Problem

Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin

Orange Labs, 38–40 rue du General Leclerc, Issy les Moulineaux, France
`{henri.gilbert,matt.robshaw,yannick.seurin}@orange-ftgroup.com`

Abstract. We present a probabilistic private-key encryption scheme named LPN-C whose security can be reduced to the hardness of the Learning from Parity with Noise (LPN) problem. The proposed protocol involves only basic operations in GF(2) and an error-correcting code. We show that it achieves indistinguishability under adaptive chosen plaintext attacks (IND-P2-C0). Appending a secure MAC renders the scheme secure under adaptive chosen ciphertext attacks. This scheme enriches the range of available cryptographic primitives whose security relies on the hardness of the LPN problem.

Key words: symmetric encryption, LPN, error-correcting code.

1 Introduction

The connections between cryptography and learning theory are well known since the celebrated paper by Impagliazzo and Levin [18]. They showed that these two areas are in a sense complementary since the possibility of cryptography rules out the possibility of efficient learning and *vice-versa*. Since then, a lot of work has dealt with building cryptographic primitives based on presumably hard learning problems. Perhaps the most well-known of these problems among the cryptographic community is the so called *Learning from Parity with Noise* (LPN) problem, which can be described as learning an unknown k-bit vector $\boldsymbol{x}$ given noisy versions of its scalar product $\boldsymbol{a} \cdot \boldsymbol{x}$ with random vectors $\boldsymbol{a}$. The prominent lightweight authentication protocol HB$^+$ recently proposed by Juels and Weis [19], and its variants [7,9,12,26], are based on this problem.

Our work is concerned with encryption schemes in the symmetric setting, where a sender and a receiver share a secret key. Up to now, most of the work in this field has concentrated on studying various operating modes to use with a secure block cipher [2]. Departing from this approach, we will construct a symmetric encryption scheme that does not appeal to any assumption regarding the pseudorandomness of a block cipher, and whose security can *directly* be reduced to some hard problem, namely here the LPN problem. In a nutshell, our scheme, named LPN-C, uses a shared secret matrix M and random vectors $\boldsymbol{a}$ to compute "noisy" masking vectors $\boldsymbol{b} = \boldsymbol{a} \cdot M \oplus \boldsymbol{\nu}$. The vector $\boldsymbol{b}$ is then used to mask the plaintext, preliminary encoded with an error-correcting code. The receiver, knowing M, can remove the mask $\boldsymbol{a} \cdot M$, and then the noise with the error-correcting code. At the same time the noise $\boldsymbol{\nu}$ prevents an attacker from "learning" the secret matrix M.

Appeared in L. Aceto et al. (Eds.): ICALP (2) 2008, LNCS 5126, pp. 679–690, 2008.

Related work. We briefly review the related work building cryptographic primitives based on hard learning problems. We have already cited the authentication protocol HB^{+} [19], which was itself derived from a simpler protocol named HB by Hopper and Blum [17]. Both protocols possess a proof of security in a certain attack model relying on the LPN problem [19,20,21]. Gilbert, Robshaw, and Sibert [13] then showed a simple man-in-the-middle attack against HB^{+}. This triggered many trials to modify and protect HB^{+} against man-in-the-middle attacks [7,9,26] but these three proposals were recently broken [11]. The subsequent proposal $\text{HB}^{\#}$ [12] is the only one to be provably secure against (some) man-in-the-middle attacks.

Former proposals were made by Blum *et al.* [5], who described a pseudorandom number generator (PRNG), a one-way function, and a private-key cryptosystem (encrypting only one bit at a time, thus much less efficient than the proposal in this paper) based on very general hard-to-learn class of functions. They also proposed a PRNG explicitly based on the LPN problem (rather than on general class of functions) derived from an older proposal of one-way function based on the hardness of decoding a random linear code [14]. More recently, Regev [28] proposed a public-key cryptosystem based on the so-called LWE (Learning with Error) problem, a generalization of the LPN problem to fields $\text{GF}(p)$, $p > 2$ (and proved that an efficient algorithm for the LWE problem would imply an efficient *quantum* algorithm for worst-case lattice problems).

LPN-C carries some similarity with a scheme by Rao and Nam [27], which may be seen as a secret-key variant of the McEliece cryptosystem, and with the trapdoor cipher *TCHo* [1], by Aumasson *et al.* In the later, the additional noise added to $C(\boldsymbol{x}) \oplus \boldsymbol{\nu}$ is introduced via an LFSR whose feedback polynomial has a low-weight multiple used as the trapdoor.

Organisation. Our paper is organised as follows. First we give some basic definitions and facts about the LPN problem and private-key encryption. Then we describe the encryption scheme LPN-C. In Section 4 we analyse the security of the scheme, in particular we establish that it is secure in the sense IND-P2-C0. In Section 5 we give some practical parameter values and explore some possible variants of the scheme. Finally, we draw our conclusions and suggest some potential future work.

2 Preliminaries

Basic Notation. In the sequel, the security parameter will be denoted by k, and we will say that a function of k (from positive integers to positive real numbers) is *negligible* if it approaches zero faster than any inverse polynomial, and *noticeable* if it is larger than some inverse polynomial (for infinitely many values of k). An algorithm will be *efficient* if it runs in time polynomial in k and possibly the size of its inputs. PPT will stand for *Probabilistic Polynomial-Time* Turing machine.

We use bold type $\boldsymbol{x}$ to indicate a row vector while scalars x are written in normal text. The i-th bit of $\boldsymbol{x}$ is denoted $\boldsymbol{x}[i]$. The bitwise addition of two vectors will be denoted $\oplus$ just as for scalars, the scalar product of $\boldsymbol{a}$ and $\boldsymbol{b}$ will be denoted $\boldsymbol{a} \cdot \boldsymbol{b}$, and their concatenation $\boldsymbol{a}\|\boldsymbol{b}$. We denote the *Hamming weight* of $\boldsymbol{x}$ by $\mathrm{Hwt}(\boldsymbol{x})$.

Given a finite set S and a probability distribution Δ on S, $s \leftarrow \Delta$ denotes the drawing of an element of S according to Δ and $s \overset{\$}{\leftarrow} S$ denotes the random drawing of an element of S endowed with the uniform probability distribution. Ber_η will denote the Bernoulli distribution of parameter $\eta \in]0, \frac{1}{2}[$, *i.e.* a bit $\nu \leftarrow \mathrm{Ber}_\eta$ is such that $\Pr[\nu = 1] = \eta$ and $\Pr[\nu = 0] = 1 - \eta$. We also define the corresponding vectorial distribution $\mathrm{Ber}_{n,\eta}$: an n-bit vector $\boldsymbol{\nu} \leftarrow \mathrm{Ber}_{n,\eta}$ is such that each bit of $\boldsymbol{\nu}$ is independently drawn according to Ber_η. Finally, we will need to define the two following oracles: we will let U_n denote the oracle returning independent uniformly random n-bit strings, and for a fixed k-bit string $\boldsymbol{s}$, $\Pi_{\boldsymbol{s},\eta}$ will be the oracle returning independent $(k+1)$-bit strings according to the distribution (to which we will informally refer to as *an LPN distribution*):

$$\{\boldsymbol{a} \overset{\$}{\leftarrow} \{0,1\}^k ; \nu \leftarrow \mathrm{Ber}_\eta \ : \ (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} \oplus \nu)\} \ .$$

The LPN problem. The LPN problem is the problem of retrieving $\boldsymbol{s}$ given access to the oracle $\Pi_{\boldsymbol{s},\eta}$. For a fixed value of k, we will say that an algorithm $\mathcal{A}$ (T, q, δ)-solves the LPN problem with noise parameter η if $\mathcal{A}$ runs in time at most T, makes at most q oracle queries, and

$$\Pr\left[\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^k \ : \ \mathcal{A}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = \boldsymbol{s}\right] \geq \delta \ .$$

By saying that the LPN problem is hard, we mean that any efficient adversary solves it with only negligible probability. There is a significant amount of literature dealing with the hardness of the LPN problem. It is closely related to the problem of decoding a random linear code [4] and is NP-Hard. It is NP-Hard to even find a vector $\boldsymbol{x}$ satisfying more than half of the equations outputted by $\Pi_{\boldsymbol{s},\eta}$ [16]. The average-case hardness has also been intensively investigated [5,6,17]. The current best known algorithms to solve it are the BKW algorithm due to Blum, Kalai, and Wasserman [6] and its improved variants by Fossorier *et al.* [10] and Levieil and Fouque [23]. They all require $2^{\Theta(k/\log k)}$ oracle queries and running time.

Private-key encryption. We briefly recall the basic definitions dealing with the semantics of probabilistic private-key encryption.

Definition 1 (Private-key cryptosystem). *A* probabilistic private-key encryption scheme *is a triple of algorithms $\Gamma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ such that:*

- *the key generation algorithm $\mathcal{G}$, on input the security parameter k, returns a random secret key $K \in \mathcal{K}(k)$: $K \overset{\$}{\leftarrow} \mathcal{G}(1^k)$;*

– *the encryption algorithm $\mathcal{E}$ is a PPT algorithm that takes as input a secret key K and a plaintext $\boldsymbol{X} \in \{0,1\}^*$ and returns a ciphertext $\boldsymbol{Y}$: $\boldsymbol{Y} \leftarrow \mathcal{E}_K(\boldsymbol{X})$;*
– *the decryption algorithm $\mathcal{D}$ is a deterministic, polynomial-time algorithm that takes as input a secret key K and a string $\boldsymbol{Y}$ and returns either the corresponding plaintext $\boldsymbol{X}$ or a special symbol $\perp$: $\mathcal{D}_K(\boldsymbol{Y}) \in \{0,1\}^* \cup \{\perp\}$.*

It is usually required that $\mathcal{D}_K(\mathcal{E}_K(\boldsymbol{X})) = \boldsymbol{X}$ for all $\boldsymbol{X} \in \{0,1\}^*$. One can slightly relax this condition, and only require that $\mathcal{D}_K(\mathcal{E}_K(\boldsymbol{X})) = \boldsymbol{X}$ except with negligible probability.

3 Description of LPN-C

Let $C : \{0,1\}^r \to \{0,1\}^m$ be an $[m, r, d]$ error-correcting code (*i.e.* of length m, dimension r, and minimal distance d) with correction capacity $t = \lfloor \frac{d-1}{2} \rfloor$. This error-correcting code is assumed to be publicly known. Let M be a secret $k \times m$ matrix (constituting the secret key of the cryptosystem). To encrypt an r-bit vector $\boldsymbol{x}$, the sender draws a k-bit random vector $\boldsymbol{a}$ and computes

$$\boldsymbol{y} = C(\boldsymbol{x}) \oplus \boldsymbol{a} \cdot M \oplus \boldsymbol{\nu} \ ,$$

where $\boldsymbol{\nu} \leftarrow \mathrm{Ber}_{m,\eta}$ is an m-bit noise vector such that each of its bits is (independently) 1 with probability η and 0 with probability $1 - \eta$. The ciphertext is the pair $(\boldsymbol{a}, \boldsymbol{y})$.

Upon reception of this pair, the receiver decrypts by computing $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M = C(\boldsymbol{x}) \oplus \boldsymbol{\nu}$, and decoding the resulting value. If decoding is not possible (which may happen when the code is not *perfect*), then the decryption algorithm returns $\perp$. When the message is not r-bit long, it is padded till its length is the next multiple of r and encrypted blockwise. The steps for LPN-C are given in Fig. 1.

Parameters	Security parameter k Polynomials (in k) m, r, d with $m > r$ Noise level $\eta \in]0, \frac{1}{2}[$
Public Components	An $[m, r, d]$ error-correcting code $C : \{0,1\}^r \to \{0,1\}^m$ and the corresponding decoding algorithm C^{-1}
Secret Key Generation	On input 1^k, output a random $k \times m$ binary matrix M
Encryption Algorithm	On input an r-bit vector $\boldsymbol{x}$, draw a random k-bit vector $\boldsymbol{a}$ and a noise vector $\boldsymbol{\nu}$, compute $\boldsymbol{y} = C(\boldsymbol{x}) \oplus \boldsymbol{a} \cdot M \oplus \boldsymbol{\nu}$, and output $(\boldsymbol{a}, \boldsymbol{y})$
Decryption Algorithm	On input $(\boldsymbol{a}, \boldsymbol{y})$, compute $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$, decode the resulting value by running C^{-1} and return the corresponding output or $\perp$ if unable to decode

Fig. 1. Description of LPN-C.

As can be seen from its description, LPN-C encryption involves only basic operations (at least when a simple linear code is used) reduced to scalar products and exclusive-or's. The decryption requires to implement the decoding procedure, which implies more work on the receiver side, though there are error-correcting codes with very efficient decoding algorithms [25].

Decryption failures. Decryption failures happen when the Hamming weight of the noise vector $\boldsymbol{\nu}$ is greater than the correction capacity t of the error-correcting code, $\mathrm{Hwt}(\boldsymbol{\nu}) > t$. When the noise vector is randomly drawn, the probability of decryption failure is given by

$$P_{\mathrm{DF}} = \sum_{i=t+1}^{m} \binom{m}{i} \eta^i (1-\eta)^{m-i} \ .$$

In order to eliminate such decryption failures, the Hamming weight of the noise vector can be tested before being used. If $\mathrm{Hwt}(\boldsymbol{\nu}) > t$, the sender draws a new noise vector according to $\mathrm{Ber}_{m,\eta}$. When the parameters are chosen such that $\eta m < t$, then this happens only with negligible probability and the encryption algorithm remains efficient.

4 Security Proofs

4.1 Security model

The security notions for probabilistic private-key encryption have been formalized by Bellare *et al.* [2] and thoroughly studied by Katz and Yung in [22]. The two main security goals for symmetric encryption are indistinguishability (IND) and non-malleability (NM). Indistinguishability deals with the secrecy afforded by the scheme: an adversary must be unable to distinguish the encryption of two (adversarially chosen) plaintexts. This definition was introduced in the context of public-key encryption as a more practical equivalent to semantic security [15]. Non-malleability was introduced (again in the context of public-key encryption) by Dolev, Dwork, and Naor [8] and deals with ciphertext modification: given a challenge ciphertext Y, an adversary must be unable to generate a different ciphertext Y' so that the respective plaintexts are meaningfully related.

Adversaries run in two phases (they are denoted as a pair of algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$) and are classified according to the oracles (encryption and/or decryption) they are allowed to access in each phase. At the end of the first phase, $\mathcal{A}_1$ outputs a distribution on the space of the plaintexts (*i.e.* a pair of plaintexts $(\boldsymbol{x_1}, \boldsymbol{x_2})$ of probability 1/2 each in the case of IND or a more complex distribution in the case of NM). Then, a ciphertext is selected at random according to the distribution and transmitted to $\mathcal{A}_2$ (this represents $\mathcal{A}$'s challenge) and the success of $\mathcal{A}$ is determined according to the security goal (*e.g.* in the case of IND, determine whether $\boldsymbol{x_1}$ or $\boldsymbol{x_2}$ was encrypted). The adversary is denoted PX-CY, where P stands for the encryption oracle and C for the decryption oracle, and where $X, Y \in \{0, 1, 2\}$ indicates when $\mathcal{A}$ is allowed to access the oracle:

– 0: $\mathcal{A}$ never accesses the oracle

– 1: $\mathcal{A}$ can only access the oracle during phase 1, hence before seeing the challenge (also termed *non-adaptive*)
– 2: $\mathcal{A}$ can access the oracle during phases 1 and 2 (also termed *adaptive*)

We only give the formal definition of indistinguishability since this is the security goal we will be primarily interested in. A formal definition of non-malleability can be found in [22].

Definition 2 (IND-PX-CY). *Let $\Gamma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For $X, Y \in \{0, 1, 2\}$ and a security parameter k, the advantage of $\mathcal{A}$ in breaking the indistinguishability of Γ is defined as:*

$$\mathrm{Adv}_{\mathcal{A},\Gamma}^{\text{IND-PX-CY}}(k) \overset{\text{def}}{=} \Bigg| \Pr\Big[K \overset{\$}{\leftarrow} \mathcal{G}(1^k); (\boldsymbol{x_0}, \boldsymbol{x_1}, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}'_1}(1^k); \\ b \overset{\$}{\leftarrow} \{0,1\}; \boldsymbol{y} \leftarrow \mathcal{E}_K(\boldsymbol{x_b}) \,:\, \mathcal{A}_2^{\mathcal{O}_2,\mathcal{O}'_2}(1^k, s, \boldsymbol{y}) = b\Big] - \frac{1}{2}\Bigg|$$

where $(\mathcal{O}_1, \mathcal{O}_2)$ is $(\emptyset, \emptyset)$, $(\mathcal{E}_K(\cdot), \emptyset)$, $(\mathcal{E}_K(\cdot), \mathcal{E}_K(\cdot))$ when X is resp. 0, 1, 2 and $(\mathcal{O}'_1, \mathcal{O}'_2)$ is $(\emptyset, \emptyset)$, $(\mathcal{D}_K(\cdot), \emptyset)$, $(\mathcal{D}_K(\cdot), \mathcal{D}_K(\cdot))$ when Y is resp. 0, 1, 2, and s is some state information. Note that the plaintexts returned by $\mathcal{A}_1$ must respect $|\boldsymbol{x_0}| = |\boldsymbol{x_1}|$ and that when $Y = 2$, $\mathcal{A}_2$ is not allowed to query $\mathcal{D}_K(\boldsymbol{y})$.
We say that Γ is secure in the sense IND-PX-CY if $\mathrm{Adv}_{\mathcal{A},\Gamma}^{\text{IND-PX-CY}}(k)$ is negligible for any PPT adversary $\mathcal{A}$.

Important relationships between the different security properties have been proved by Katz and Yung [22]. The most meaningful for us are:

– non-adaptive CPA-security implies adaptive CPA-security:

$$\text{IND-P1-C}Y \Rightarrow \text{IND-P2-C}Y \quad \text{and} \quad \text{NM-P1-C}Y \Rightarrow \text{NM-P2-C}Y$$

– IND and NM are equivalent in the case of P2-C2 attacks (but unrelated for other attacks): IND-P2-C2 $\Leftrightarrow$ NM-P2-C2.

4.2 Proof of indistinguishability under chosen plaintext attacks

We now prove that LPN-C is secure in the sense IND-P2-C0, by reducing its security to the LPN problem. First, we will recall the following useful lemma which was proved in [20] following [28], and which states that the hardness of the LPN problem implies that the two oracles U_{k+1} and $\Pi_{\boldsymbol{s},\eta}$ are indistinguishable.

Lemma 1 ([20], Lemma 1). *Assume there exists an algorithm $\mathcal{M}$ making q oracle queries, running in time T, and such that*

$$\left|\Pr\left[\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^k \,:\, \mathcal{M}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{M}^{U_{k+1}}(1^k) = 1\right]\right| \geq \delta \ .$$

Then there is an algorithm $\mathcal{A}$ making $q' = \mathcal{O}(q \cdot \delta^{-2} \log k)$ oracle queries, running in time $T' = \mathcal{O}(T \cdot k\delta^{-2} \log k)$, and such that

$$\Pr\left[\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^k \,:\, \mathcal{A}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = \boldsymbol{s}\right] \geq \frac{\delta}{4} \ .$$

A full proof of this result can be found in [20]. We will reduce the security of LPN-C to the problem of distinguishing U_{k+1} and $\Pi_{\boldsymbol{s},\eta}$ rather than directly to the LPN problem.

Theorem 1. *Assume there is an adversary $\mathcal{A}$, running in time T, and attacking LPN-C with parameters (k, m, r, d, η) in the sense of IND-P2-C0 with advantage δ by making at most q queries to the encryption oracle. Then there is an algorithm $\mathcal{M}$ making $\mathcal{O}(q)$ oracle queries, running in time $\mathcal{O}(T)$, and such that*

$$\left|\Pr\left[\boldsymbol{s} \xleftarrow{\$} \{0,1\}^k : \mathcal{M}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{M}^{U_{k+1}}(1^k) = 1\right]\right| \geq \frac{\delta}{m} \ .$$

Proof. As already pointed out, non-adaptive CPA-security (P1) implies adaptive CPA-security (P2), hence we may restrict ourselves to adversaries accessing the encryption oracle only during the first phase of the attack (before seeing the challenge ciphertext).

The proof proceeds by a hybrid argument. We will first define the following hybrid distributions on $\{0,1\}^{k+m}$. For $j \in [0..m]$, let M' denote a $k \times (m-j)$ binary matrix. We define the probability distribution $\mathcal{P}_{j,M',\eta}$ as

$$\{\boldsymbol{a} \xleftarrow{\$} \{0,1\}^k; \boldsymbol{r} \xleftarrow{\$} \{0,1\}^j; \boldsymbol{\nu} \leftarrow \mathrm{Ber}_{(m-j),\eta} : \boldsymbol{a}\|\boldsymbol{r}\|(\boldsymbol{a} \cdot M' \oplus \boldsymbol{\nu})\} \ .$$

Hence the returned vector $\boldsymbol{a}\|\boldsymbol{b}$ is such that the first j bits of $\boldsymbol{b}$ are uniformly random, whereas the last $(m-j)$ bits are distributed according to $(m-j)$ independent LPN distributions related to the respective columns of M'. Note that $\mathcal{P}_{m,M',\eta} = U_{k+m}$.

We will also define the following hybrid encryption oracles $\mathcal{E}'_{j,M',\eta}$ associated with the secret matrix M' and noise parameter η: on input the r-bit plaintext $\boldsymbol{x}$, the encryption oracle encodes it to $C(\boldsymbol{x})$, draws a random $(k+m)$-bit vector $\boldsymbol{a}\|\boldsymbol{b}$ distributed according to $\mathcal{P}_{j,M',\eta}$, and returns $(\boldsymbol{a}, C(\boldsymbol{x}) \oplus \boldsymbol{b})$.

We now describe how the distinguisher $\mathcal{M}$ proceeds. Recall that $\mathcal{M}$ has access to an oracle and wants to distinguish whether this is U_{k+1} or $\Pi_{\boldsymbol{s},\eta}$. On input the security parameter 1^k, $\mathcal{M}$ draws a random $j \in [1..m]$. If $j < m$, it also draws a random $k \times (m-j)$ binary matrix M'. It then launches the first phase $\mathcal{A}_1$ of the adversary $\mathcal{A}$. Each time $\mathcal{A}_1$ asks for the encryption of some $\boldsymbol{x}$, $\mathcal{M}$ obtains a sample $(\boldsymbol{a}, z)$ from its oracle, draws a random $(j-1)$-bit vector $\boldsymbol{r} \xleftarrow{\$} \{0,1\}^{j-1}$, and draws a $(m-j)$-bit noise vector $\boldsymbol{\nu}$ distributed according to $\mathrm{Ber}_{(m-j),\eta}$. It then forms the masking vector $\boldsymbol{b} = \boldsymbol{r}\|z\|(\boldsymbol{a} \cdot M' \oplus \boldsymbol{\nu})$ and returns $(\boldsymbol{a}, C(\boldsymbol{x}) \oplus \boldsymbol{b})$.

The adversary $\mathcal{A}_1$ then returns two plaintexts $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$. The distinguisher $\mathcal{M}$ selects a uniformly random $\alpha \in \{1,2\}$ and returns to $\mathcal{A}_2$ the ciphertext corresponding to $\boldsymbol{x_\alpha}$ encrypted exactly as described just before. If the answer of $\mathcal{A}_2$ is correct, then $\mathcal{M}$ returns 1, otherwise it returns 0.

It is straightforward to verify that when $\mathcal{M}$'s oracle is U_{k+1}, $\mathcal{M}$ simulates the encryption oracle $\mathcal{E}'_{j,M',\eta}$, whereas when $\mathcal{M}$'s oracle is $\Pi_{\boldsymbol{s},\eta}$, then $\mathcal{M}$ simulates the encryption oracle $\mathcal{E}'_{j-1,M'',\eta}$ where $M'' = \boldsymbol{s}\|M'$ is the matrix obtained as the concatenation of $\boldsymbol{s}$ and M'. Hence the advantage of the distinguisher can be expressed as

$$\begin{aligned}
\mathrm{Adv} &= \left|\Pr\left[\boldsymbol{s} \xleftarrow{\$} \{0,1\}^k \,:\, \mathcal{M}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{M}^{U_{k+1}}(1^k) = 1\right]\right| \\
&= \frac{1}{m}\left|\sum_{j=0}^{m-1} \Pr\left[\mathcal{A}^{\mathcal{E}'_{j,M',\eta}} \text{ succeeds}\right] - \sum_{j=1}^{m} \Pr\left[\mathcal{A}^{\mathcal{E}'_{j,M',\eta}} \text{ succeeds}\right]\right| \\
&= \frac{1}{m}\left|\Pr\left[\mathcal{A}^{\mathcal{E}'_{0,M',\eta}} \text{ succeeds}\right] - \Pr\left[\mathcal{A}^{\mathcal{E}'_{m,M',\eta}} \text{ succeeds}\right]\right| \; .
\end{aligned}$$

Note that the encryption oracle $\mathcal{E}'_{0,M',\eta}$ is exactly the real LPN-C encryption oracle. On the other hand the encryption oracle $\mathcal{E}'_{m,M',\eta}$ encrypts all plaintexts by blinding them with uniformly random vectors $\boldsymbol{b}$ so that in this case the adversary $\mathcal{A}$ cannot do better (or worse) than guessing α at random and has a success probability of $1/2$. Hence

$$\left|\Pr\left[\mathcal{A}^{\mathcal{E}'_{0,M',\eta}} \text{ succeeds}\right] - \Pr\left[\mathcal{A}^{\mathcal{E}'_{m,M',\eta}} \text{ succeeds}\right]\right|$$

is exactly the advantage of the adversary which is greater than δ by hypothesis. The theorem follows. □

Remark 1. Note that when the error-correcting code is linear, the scheme is clearly malleable, even when the adversary has no access at all to the encryption nor the decryption oracle (the scheme is not NM-P0-C0). Indeed, an adversary receiving a ciphertext $(\boldsymbol{a}, \boldsymbol{y})$ corresponding to some plaintext $\boldsymbol{x}$, can forge a new ciphertext corresponding to some other plaintext $\boldsymbol{x} \oplus \boldsymbol{x'}$ simply by modifying the ciphertext to $(\boldsymbol{a}, \boldsymbol{y} \oplus C(\boldsymbol{x'}))$. The same kind of attacks, though more elaborate, would probably apply for non-linear error-correcting codes. Since NM-P2-C2 is equivalent to IND-P2-C2, the scheme cannot be IND-P2-C2 either. We investigate the security with respect to IND-P2-C1 attacks in the next subsection.

4.3 An IND-P0-C1 attack.

Here we show that the scheme is insecure (*i.e.* distinguishable) when the attacker has (non-adaptive) access to the decryption oracle. The idea is to query the decryption oracle many times with the same vector $\boldsymbol{a}$ in order to get many approximate equations on $\boldsymbol{a} \cdot M$. Consider an adversary querying the decryption oracle with ciphertexts $(\boldsymbol{a}, \boldsymbol{y_i})$ for a fixed $\boldsymbol{a}$ and random $\boldsymbol{y_i}$'s. Each time $\boldsymbol{y_i} \oplus \boldsymbol{a} \cdot M$ is at Hamming distance less than t from a codeword, the decryption oracle will return $\boldsymbol{x_i}$ such that $\mathrm{Hwt}(C(\boldsymbol{x_i}) \oplus \boldsymbol{y_i} \oplus \boldsymbol{a} \cdot M) \le t$. This will give an approximation for each bit of $\boldsymbol{a} \cdot M$ with noise parameter less than $\frac{t}{m}$.

Indeed, let us fix some bit position j, and evaluate the probability p that, given that the decryption oracle returned the plaintext $\boldsymbol{x_i}$, the j-th bit of $\boldsymbol{a} \cdot M$ is *not* equal to the j-th bit of $C(\boldsymbol{x_i}) \oplus \boldsymbol{y_i}$:

$$p = \Pr_{\boldsymbol{y_i} \xleftarrow{\$} \{0,1\}^m} \left[(\boldsymbol{a} \cdot M)[j] \neq (C(\boldsymbol{x_i}) \oplus \boldsymbol{y_i})[j] \,\middle|\, \mathcal{D}_K(\boldsymbol{a}, \boldsymbol{y_i}) = \boldsymbol{x_i}\right] \; .$$

Obviously, the sum over j of this quantity is equal to the expected value of the number of errors, hence is less than t. Consequently the error probability is less than t/m. Assume the vector $\boldsymbol{a}$ was chosen to have only one non-null coordinate (say, the l-th one). Then this will enable to retrieve with high confidence the bit in position (l, j) of the secret matrix M with a few attempts (according to the Chernoff bound, since the repeated experiments use independent $\boldsymbol{y_i}$'s). Repeating the procedure $k \cdot m$ times will enable the adversary to retrieve the matrix M, which completely compromises the security of the scheme.

Note that for this reasoning to be correct, the probability that the decryption oracle does not return $\perp$ must be noticeable. Otherwise the adversary will have to make an exponential number of attempts to get enough equations. Clearly

$$\Pr_{\boldsymbol{y_i} \xleftarrow{\$} \{0,1\}^m} \left[\mathcal{D}_K(\boldsymbol{a}, \boldsymbol{y_i}) \neq \perp \right] = 2^r \sum_{i=0}^{t} \frac{\binom{m}{i}}{2^m} \simeq 2^{-(1-\frac{r}{m}-H(\frac{t}{m}))m} \quad ,$$

where H is the entropy function $H(x) = -x \log_2(x) - (1-x) \log_2(1-x)$. The concrete value of this probability will depend on the error-correcting code which is used. If it is good enough this value will not be too small.

At the same time this suggests a method to thwart the attack. Assume that LPN-C is modified in the following way: an additional parameter t' such that $\eta m < t' < t$ is chosen. When the number of errors in $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$ is greater than t' (*i.e.* $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$ is at Hamming distance greater than t' from any codeword), the decryption algorithm returns $\perp$. If t' is such that $2^{-(1-\frac{r}{m}-H(\frac{t'}{m}))m}$ is negligible, then the previous attack is not possible anymore. At the same time, this implies to drastically reduce the noise parameter η and the LPN problem becomes easier. The scheme also remains malleable, as the attack in Remark 1 remains applicable (hence the scheme cannot be IND-P2-C2 either). However, it could be that such a modified scheme is IND-P2-C1. This remains an open problem.

4.4 Achieving P2-C2 security

The most straightforward way to get an encryption scheme secure against chosen-ciphertext attacks from an encryption scheme secure against chosen-plaintext attacks is to add message authenticity, *e.g.* by using a Message Authentication Code (MAC). This idea was suggested in [8,22] and was carefully studied by Bellare and Namprempre [3]. They explored the three paradigms *Encrypt-and-MAC*, *MAC-then-Encrypt* and *Encrypt-then-MAC* and showed that the later one was the most secure way to proceed. More precisely, assume that the sender and the receiver share an additional secret key K_m for the goal of message authentication, and let $\mathrm{MAC}_{K_m}(\cdot)$ be a secure[1] MAC. LPN-C is modified as follows: let $\boldsymbol{A} = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_n})$ be the vectors used to encrypt in LPN-C, and $\boldsymbol{Y} = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_n})$ be the ciphertexts to transmit. A MAC of the ciphertext is added

[1] that is, strongly unforgeable under chosen plaintext attacks; see [3] for a precise definition.

to the transmission and computed as $\boldsymbol{\tau} = \mathrm{MAC}_{K_m}(\boldsymbol{A} \| \boldsymbol{Y})$. The decryption algorithm is modified to return $\perp$ each time the MAC is not valid.

Given that the original scheme is IND-P2-C0, generic results of [3] imply that the enhanced scheme is IND/NM-P2-C2. This generic method has the drawback to rely on an additional assumption, namely the unforgeability of the MAC. We go one step further and propose a way to build a MAC only relying on the LPN problem and a one-way function.

Let M_2 be a secret $l \times l'$ binary matrix, where l and l' are polynomials in k. Let $H : \{0,1\}^* \to \{0,1\}^l$ be a one-way function. For $\boldsymbol{X} \in \{0,1\}^*$ define

$$\mathrm{MAC}_{M_2}(\boldsymbol{X}) = H(\boldsymbol{X}) \cdot M_2 \oplus \boldsymbol{\nu'}$$

where $\boldsymbol{\nu'} \leftarrow \mathrm{Ber}_{l',\eta}$. We sketch the proof of the security of this MAC in the Random Oracle model in the full version of this paper.

5 Concrete Parameters for LPN-C

We now discuss some example parameters for LPN-C as well as some possible practical variants. We will define the expansion factor of the scheme as $\sigma = \frac{|\text{ciphertext}|}{|\text{plaintext}|} = \frac{m+k}{r}$, and the secret key size $|K| = k \cdot m$. There are various trade-offs possible when fixing the values of the parameters (k, η, m, r, d). First, the hardness of the LPN problem depends on k and η (it increases with k and η). However an increase to k implies a higher expansion factor and a bigger key size, whereas an increase to η implies to use a code with a bigger correction capacity and minimal distance, hence a bigger factor $\frac{m}{r}$. Depending on how the noise vectors $\boldsymbol{\nu}$ are generated, decryption failures may also be an issue.

Example values for k and η were given by Levieil and Fouque [23]. If one is seeking 80-bit security, suitable parameters are $(k = 512, \eta = 0.125)$, or $(k = 768, \eta = 0.05)$. Example parameters for LPN-C are given below, where we used the list of Best Known Linear Codes available in MAGMA 2.13 [24].

LPN-C					expansion	storage	storage	decryption
k	η	m	r	d	factor σ	$\|K\|$ (bits)	(Toeplitz)	failure P_{DF}
512	0.125	80	27	21	21.9	40,960	591	0.42
512	0.125	160	42	42	16	81,920	671	0.44
768	0.05	80	53	9	16	61,440	847	0.37
768	0.05	160	99	17	9.4	122,880	927	0.41
768	0.05	160	75	25	12.4	122,880	927	0.06

Possible variants. A first possibility is to increase the size of the secret matrix M in order to decrease the expansion factor σ. Indeed, assume that M is now a $k \times (n \cdot m)$ binary matrix for some integer $n > 1$. Then it becomes possible to encrypt n blocks of r bits with the same random vector $\boldsymbol{a}$. The expansion factor becomes $\sigma = \frac{n \cdot m + k}{n \cdot r}$. Asymptotically when n increases, the expansion factor of the scheme tends to the one of the error-correcting code $\frac{m}{r}$.

Another possibility would be to pre-share the vectors $\boldsymbol{a_i}$'s, or to generate them from a small seed an a pseudorandom number generator. The expansion factor would then fall to $\sigma = \frac{m}{r}$, but synchronization issues could arise.

Finally, we mention the possibility (already used in HB$^{\#}$ [12]) to use Toeplitz matrices in order to decrease the size of the secret key. A $(k \times m)$-binary *Toeplitz* matrix M is a matrix for which the entries on every upper-left to lower-right diagonal have the same value. The entire matrix is specified by the top row and the first column. Thus a Toeplitz matrix can be stored in $k + m - 1$ bits rather than the km bits required for a truly random matrix. However, the security implications of such a design choice remain to be studied.

6 Conclusions

We have presented LPN-C, a novel symmetric encryption scheme whose security can be reduced to the LPN problem. Due to the low-cost computations (essentially of bitwise nature) required on the sender side, this encryption scheme could be suitable for environments with restricted computation power, typically RFIDs. Moreover, due to some similarities it could be possible to combine it with one of the authentication protocols HB^{+} or HB$^{\#}$.

Among open problems we highlight the design of an efficient MAC directly from the LPN problem without any other assumption, as well as an understanding of the impact of the use of Toeplitz matrices in LPN-C (and HB$^{\#}$).

References

1. J.-P. Aumasson, M. Finiasz, W. Meier, and S. Vaudenay. *TCHo*: A Hardware-Oriented Trapdoor Cipher. In *Proceedings of ACISP 2007*, LNCS 4586, pp. 184–199, Springer, 2007.
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. In *Proceedings of FOCS '97*, pp. 394–403, 1997.
3. M. Bellare and C. Namprempre. Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. In *Proceedings of Asiacrypt 2000*, LNCS 1976, pp. 531–545, Springer, 2000.
4. E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Trans. Info. Theory*, volume 24, pp. 384–386, 1978.
5. A. Blum, M. Furst, M. Kearns, and R. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In *Proceedings of Crypto '93*, LNCS 773, pp. 278–291, Springer, 1993.
6. A. Blum, A. Kalai, and H. Wasserman. Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model. *J. ACM*, volume 50, number 4, pp. 506–519, 2003. Preliminary version in *Proceedings of STOC 2000.*
7. J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: A Lightweight Authentication Protocol Secure Against Some Attacks. In *Proceedings of SecPerU 2006*, pp. 28–33, IEEE Computer Society Press, 2006.

8. D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. *SIAM Journal of Computing*, volume 30, number 2, pp. 391-437, 2000.
9. D.N. Duc and K. Kim. Securing HB$^+$ Against GRS Man-in-the-Middle Attack. In *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, Jan. 23–26, 2007.
10. M.P.C. Fossorier, M.J. Mihaljevic, H. Imai, Y. Cui, and K. Matsuura. A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication. Available from `http://eprint.iacr.org/2006/197.pdf`.
11. H. Gilbert, M.J.B. Robshaw, and Y. Seurin. Good Variants of HB$^+$ are Hard to Find. In *Proceedings of Financial Crypto 2008*, to appear.
12. H. Gilbert, M.J.B. Robshaw, and Y. Seurin. HB$^\#$: Increasing the Security and Efficiency of HB$^+$. In *Proceedings of Eurocrypt 2008*, LNCS 4965, pp. 361–378, Springer, 2008.
13. H. Gilbert, M.J.B. Robshaw, and H. Sibert. An Active Attack Against HB$^+$: A Provably Secure Lightweight Authentication Protocol. *IEE Electronics Letters*, volume 41, number 21, pp. 1169–1170, 2005.
14. O. Goldreich, H. Krawczyk, and M. Luby. On the Existence of Pseudorandom Generators. In *Proceedings of FOCS '88*, pp. 12–21, 1988.
15. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, volume 28, number 2, pp. 270–299, 1984.
16. J. Håstad. Some Optimal Inapproximability Results. *J. ACM*, volume 48, number 4, pp. 798-859, 2001.
17. N. Hopper and M. Blum. Secure Human Identification Protocols. In *Proceedings of Asiacrypt 2001*, LNCS 2248, pp. 52–66, Springer, 2001.
18. R. Impagliazzo and L.A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Proceedings of FOCS '90*, pp. 812–821, 1990.
19. A. Juels and S.A. Weis. Authenticating Pervasive Devices With Human Protocols. In *Proceedings of Crypto 2005*, LNCS 3126, pp. 293–308, Springer, 2005.
20. J. Katz and J. Shin. Parallel and Concurrent Security of the HB and HB$^+$ Protocols. In *Proceedings of Eurocrypt 2006*, LNCS 4004, pp. 73–87, Springer, 2006.
21. J. Katz and A. Smith. Analysing the HB and HB$^+$ Protocols in the "Large Error" Case. Available from `http://eprint.iacr.org/2006/326.pdf`.
22. J. Katz and M. Yung. Complete Characterization of Security Notions for Probabilistic Private-Key Encryption. *Journal of Cryptology*, volume 19, number 1, pp. 67–95, 2006. Preliminary version in *Proceedings of STOC 2000.*
23. E. Levieil and P.-A. Fouque. An Improved LPN Algorithm. In *Proceedings of SCN 2006*, LNCS 4116, pp. 348–359, Springer, 2006.
24. MAGMA Computational Algebra System.
Homepage `http://magma.maths.usyd.edu.au/magma`
25. F.J. MacWilliams and N.J.A. Sloane. The Theory of Error-Correcting Codes. North-Holland Mathematical Library, 1983.
26. J. Munilla and A. Peinado. HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols. *Computer Networks*, volume 51, pp. 2262–2267, 2007.
27. T.R.N. Rao and K.H. Nam. Private-Key Algebraic-Code Encryptions. *IEEE Transactions on Information Theory*, volume 35, number 4, pp. 829–833, 1989.
28. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of STOC 2005*, pp. 84–93, 2005.