TCHo: a Hardware-Oriented Trapdoor Cipher

Jean-Philippe Aumasson[1⋆], Matthieu Finiasz[2], Willi Meier[1⋆⋆], and Serge Vaudenay[3]

[1] FHNW, Windisch, Switzerland
[2] ENSTA, Paris, France
[3] EPFL, Lausanne, Switzerland, http://lasecwww.epfl.ch/

Abstract. This paper improves the Finiasz-Vaudenay construction of TCHo, a hardware-oriented public-key cryptosystem, whose security relies on the hardness of finding a low-weight multiple of a given polynomial, and on the decoding of certain noisy cyclic linear codes. Our improvement makes it possible to decrypt in polynomial time (instead of exponential time), to directly prove semantic security (instead of one-wayness), and to achieve pretty good asymptotic performances. We further build IND-CCA secure schemes using the KEM/DEM and Fujisaki-Okamoto hybrid encryption frameworks in the random oracle model. This can encrypt an arbitrary message with an overhead of about 5 Kb in less than 15 ms, on an ASIC of about 10 000 gates at 4 MHz.

Keywords: public-key cryptosystem, post-quantum cryptography, hardware, linear feedback shift register, polynomial multiples.

1 Introduction

Since the introduction of public-key cryptography [12, 13], dozens of cryptosystems appeared, based on hard problems like integer factorization, discrete logarithms, lattice reduction, knapsacks, *etc.*, in various algebraic structures. But their non-trivial constructions made their use somewhat difficult in constrained environments (PDAs, RFID tags, *etc.*), where stream ciphers used to rule. In that sense, a secure public-key cryptosystem with stream cipher-like design would be a breakthrough. Furthermore, studying alternate designs for public-key encryption not based on factoring or discrete logarithm is an important duty for the academic research community to prepare a post-quantum era [25].

In [14], Finiasz and Vaudenay introduced a new public-key cryptosystem called TCHo, where the public key is a high-degree binary polynomial, and the private key a sparse multiple of the latter. Security relies on the *ad-hoc* problem

⋆ Supported by the Swiss National Science Foundation under project number 113329.
⋆⋆ Supported by Hasler Foundation http://www.haslerfoundation.ch under project number 2005.

of finding a low-weight multiple of a certain degree. This problem, or its variants, has been important in LFSR cryptanalysis since some attacks are possible only when the feedback polynomial or one of its multiples is sparse [24]. A few works [16, 18, 22] study the distribution of multiples of a given weight.

In this article, TCHo1 designates the original cryptosystem from [14] whereas TCHo2 designates our variant. By default, TCHo refers to TCHo2.

TCHo1 encryption is probabilistic, and can be roughly described as the transmission of a codeword over a noisy channel: one small LFSR encodes the message, while a large one randomly initialized, along with a source of biased random bits, produces the noise. A ciphertext is a XOR of the three bitstreams. The private key is used to "delete" the bitstream of the large LFSR by a kind of convolution product, thereby reducing the noise over the coded message, so as to be able to decode the cyclic linear code spanned by the first LFSR. Although the design of TCHo1 is very simple and well fitted for hardware, some major disadvantages are its prohibitive decryption time complexity, of exponential cost, the absence in [14] of an estimate of incorrect decryption probability, and the lack of asymptotic complexities. In this paper, we

- propose a variant leading us to polynomial decryption time,
- estimate the error probability in decryption,
- study asymptotic parameters,
- prove semantic security of this new scheme, under certain assumptions,
- and suggest two hybrid constructions to reach IND-CCA security.

Finally we present performances of TCHo in a software implementation.

2 Preliminaries

2.1 Notations

The logarithm in base 2 is denoted $\log_2$, and log is the natural logarithm.

A *bitstring* x is a sequence of bits. Its *length* $|x|$ is its number of bits, and may be finite or infinite. Its *Hamming weight*, or simply *weight*, is its number of ones. The concatenation of x and y is $x||y$. The sum over $\mathbb{F}_2$ is denoted $+$, and the product $\times$. A bitstring x can be written $(x_1, x_2, \ldots, x_n)$, and $(0, \ldots, 0)$ can simply be denoted 0. The sum (also denoted $+$) of two bitstrings of equal length produces a bitstring of same length, and is defined as a bitwise sum. A *bitstream* is a bitstring of unspecified (possibly infinite) length produced by some device or bit source, and shall be denoted by the symbol $\mathcal{S}$ with contextual subscript. The symbol $\mathcal{S}^\ell$ refers to the bitstream $\mathcal{S}$ truncated to its first ℓ bits.

The degree of a polynomial P in $\mathbb{F}_2[X]$ is denoted $\deg(P)$, and its *weight* is its number of non-zero coefficients.

If we speak about *random* bits, or random sequence, *etc.*, it is either uniform or biased randomness, and the distribution is specified only where the meaning can be ambiguous. A random source of independent bits with bias γ produces a zero with probability $\frac{1}{2}(1+\gamma)$ (and a one with probability $\frac{1}{2}(1-\gamma)$). The

produced bitstream is denoted $\mathcal{S}_\gamma$, and $\mathcal{S}_\gamma(r)$ if we specify the seed r of the generator. $\mathcal{S}_0$ is a uniform random bitstream.

When no probability distribution or space is explicitly set, *randomly chosen* means randomly chosen among all the objects of this kind, with respect to a uniform probability law.

A linear feedback shift register (LFSR) is entirely characterized by its feedback function, defined by a *feedback polynomial* $P = \sum_{i=0}^{\infty} p_i X^i$, the size of the LFSR being the degree of this polynomial. We use the notation $\mathcal{L}_P$ for the LFSR with feedback polynomial P. The bitstream determined by the initial state $s = (s_0, \dots, s_{\deg(P)-1})$ is denoted $\mathcal{S}_{\mathcal{L}_P(s)} = (s_0, \dots, s_i, \dots)$, such that $s_{i+\deg(P)} = \sum_{k=0}^{\deg(P)-1} p_k s_{i+k}$.

We define the product of a binary polynomial $K = \sum_{i=0}^{\infty} k_i X^i$ of degree d and a bitstream $\mathcal{S}^{d+N} = (s_0, \dots, s_{d+N-1})$ as

$$K \otimes \mathcal{S}^{d+N} = (s'_0, \dots, s'_{N-1})$$

with $s'_i = s_i k_0 + s_{i+1} k_1 + \cdots + s_{i+d} k_d$. The operator thus defined is distributive over the bitstring sum, it verifies $(PQ) \otimes \mathcal{S} = P \otimes (Q \otimes \mathcal{S})$ and $P \otimes \mathcal{S}_{\mathcal{L}_P(x)} = 0$, for all $P, Q \in \mathbb{F}_2[X]$, $x \in \mathbb{F}_2^{\deg(P)}$, and $\mathcal{S}$. As a consequence, if P divides K, for any ℓ, x, s we have $K \otimes (\mathcal{S}^{\ell}_{\mathcal{L}_P(x)} + s) = K \otimes s$.

We shall use the acronyms CCA, CPA, IND, OW, respectively standing for the usual notions of Adaptive Chosen Ciphertext Attack, Chosen Plaintext Attack, Indistinguishability, and One-Wayness.

2.2 Computational Problem

Like TCHo1, the main problem on which TCHo relies can be stated as follows:

Low Weight Polynomial Multiple (lwpm)
Parameters: Three naturals w, d and d_P, such that $0 < d_P < d$ and $w < d$.
Instance: $P \in \mathbb{F}_2[X]$ of degree d_P.
Question: Find a multiple K of P of degree at most d and weight at most w.

In [14] the authors suggest several strategies to solve this problem (namely birthday paradox [28], syndrome decoding [6, 21], and exhaustive search). Inspired from this, we make the following average-case assumption:[4]

Assumption 1. *Let* Gen *be a random generator which generates a random polynomial K of degree d_K and weight w_K until it has an irreducible factor P whose degree d_P is in a given interval $[d_{\min}, d_{\max}]$. The output of* Gen *is P. We assume that $w_K \log_2 \frac{d_K}{d_{\max}} \geq \lambda$. For any d and w such that $\binom{d}{w-1} \leq 2^{d_{\min}}$ and $w \log_2 \frac{d}{d_{\max}} \geq \lambda$, the* lwpm *problem for an instance generated by* Gen *needs at least 2^λ operations to solve.*

[4] In [14], P is assumed to be primitive. Here, we only assume that it is irreducible as discussed later.

More concretely, the best algorithm to find one solution has a complexity within the order of $(d/d_P)^{w-1}$ when the existence of a solution is unexpected and $2^{d_P}(d/d_P)^{w-1}/\binom{d}{w-1}$ when many solutions exist.

As a nominal example, we will use $w_K = \Theta(\lambda)$, $d_{\min} = \Theta(\lambda^2)$, $d_{\max} = \Theta(\lambda^2)$, and $d_K = \Theta(\lambda^3)$. The assumption seemingly suggests that the problem needs exponential time (in λ) to solve LWPM with w and d asymptotically equivalent to the parameters of K. Hence, K can be used as a hidden trapdoor.

3 Description of the TCHo Scheme

3.1 Presentation

Just like TCHo1, TCHo uses a polynomial K of degree d_K and weight w_K as a secret key; a polynomial P of degree $d_P \in [d_{\min}, d_{\max}]$ as a public key; it produces ciphertexts of ℓ bits and uses a random source of bias γ. We use in TCHo a new parameter k (which replaces the old d_Q from TCHo1 because it is no longer the degree of a polynomial). It is the length of the plaintext.

TCHo differs from TCHo1 in the coding applied to the plaintext. In TCHo1, a code spanned by an LFSR with an arbitrary primitive polynomial Q was used, leading to an expensive decryption procedure. We can generalize TCHo1 and use an arbitrary code C of dimension k and length ℓ for which an efficient decoding procedure exists, and denote $C(x)$ the codeword of x in C. This code is subject to many constraints and cannot be chosen at random. In the decryption process of TCHo1, the ciphertext is multiplied by K to suppress $\mathcal{S}^{\ell}_{\mathcal{L}_P}$. In this process, the noise source $\mathcal{S}^{\ell}_{\gamma}$ becomes like $\mathcal{S}^{\ell-d_K}_{\gamma^{w_K}}$. In the general case, the multiplication by K being a linear operation, we will have $K \otimes C(x) = \tilde{C}(x)$, where $\tilde{C}$ is a new linear code of dimension k and length $\ell - d_K$. This means that when decrypting a ciphertext, one will have to decode in the modified code $\tilde{C}$. The only case where decoding in $\tilde{C}$ can be efficient for an arbitrary K is when C is a truncated cyclic linear code, that is, C is the output of an LFSR.[5] In that case, as for TCHo1, $K \otimes C(x)$ is equal to $C(x')$ truncated to $\ell - d_K$ bits, where x' is obtained from x exactly as with TCHo1. TCHo is a particular instance of this generalized TCHo1 construction with a repetition code. These codes offer straightforward encoding and decoding algorithms.

Another innovation of TCHo is that the need for P to be primitive is obviated; let n be the order of the polynomial P. In [14] primitivity is required so as not to have $X^n + 1$ as a trivial solution of LWPM, when $n \leq \ell$. However, for randomly chosen P, the order n is smaller than ℓ with probability about $\ell/2^{d_P}$, which is close to zero. Hence LWPM may remain as hard when P is a random irreducible polynomial, not necessarily primitive.

Parameters. A security parameter λ defines a parameter vector

$$(k, d_{\min}, d_{\max}, d_K, w_K, \gamma, \ell).$$

[5] Appendix A provides more discussion on the code selection.

Key Generation. We generate a random polynomial K of degree d_K and weight w_K with constant term 1 until it has a primitive factor P of degree d_P belonging to the interval $[d_{\min}, d_{\max}]$. This works just like TCHo1, in time $\mathcal{O}\left(\frac{d_{\max}}{d_{\max}-d_{\min}} d_K^2 \log d_K \log\log d_K\right)$ using the Cantor-Zassenhaus algorithm [8] and the probabilistic primitivity test from [14].

Encryption. TCHo encrypts a plaintext x of length k in the following way:

$$\mathsf{TCHo}_{\mathsf{enc}}(x, r_1||r_2) = C(x) + \mathcal{S}^{\ell}_{\mathcal{L}_P(r_1)} + \mathcal{S}^{\ell}_{\gamma}(r_2).$$

The codeword $C(x)$ of a bitstring x of length k is formed of contiguous repetitions of x truncated to ℓ bits, and so the minimum distance of the code is $\lfloor \ell/k \rfloor$. It has length ℓ and the code has dimension k. Complexity is $\mathcal{O}(\ell \cdot d_P)$, provided that the random generator has no higher complexity. The ciphertext length is ℓ. Note that ℓ/k is the expansion factor of the message.

Decryption. Given $y = \mathsf{TCHo}_{\mathsf{enc}}(x, r_1||r_2)$, decryption works as follows:

1. K is used to delete $\mathcal{S}_{\mathcal{L}_P}$ in y:[6]
$$K \otimes y \approx \tilde{C}(x) + \mathcal{S}^{\ell-d_K}_{\gamma^{w_K}}$$
where $\tilde{C}(x)$ is equal to a truncated codeword $C(x')$, with $x' = f(x)$ for some linear map f. Complexity is $\mathcal{O}(w_K \cdot \ell)$ for this operation only.
2. $K \otimes y$ is decoded to find x'. Decoding is performed using majority logic decoding (MJD), which is equivalent to maximum likelihood decoding for these codes, but runs in time $\mathcal{O}(\ell - d_K)$, instead of $\mathcal{O}\left(k \cdot 2^k\right)$. It allows to encrypt larger blocks.
3. $\mathsf{TCHo}_{\mathsf{dec}}(y) = f^{-1}(x') = x$ is computed. This operation takes $\mathcal{O}\left(k^3\right)$ complexity. Note that the matrix of f^{-1} can be precomputed from K and C.

The overall decryption complexity thus becomes $\mathcal{O}\left(w_K \cdot \ell + k^3\right)$.

3.2 Reliability

Here $\tilde{C}$ has minimum distance $\delta = \lfloor (\ell - d_K)/k \rfloor$, but decoding more than $\lfloor (\delta - 1)/2 \rfloor$ errors will of course be possible. The probability of erroneous decoding is exactly the probability that at least one bit is more frequently erroneous than correct, that is (under the heuristic assumption that the correlation in $K \otimes \mathcal{S}^{\ell}_{\gamma}$ is similar to the correlation in $\mathcal{S}^{\ell-d_K}_{\gamma^{w_K}}$),

$$\rho \approx 1 - \left(\sum_{i=\lceil \delta/2 \rceil}^{\delta} 2^{-\delta} (1+\gamma^{w_K})^i (1-\gamma^{w_K})^{\delta-i} \binom{\delta}{i} \right)^k . \tag{1}$$

[6] Each bit of the word obtained after multiplying by K by $\mathcal{S}^{\ell}_{\gamma}$ is the sum of w_K bits with bias γ. Hence they have a bias of γ^{w_K}. However, the noisy bits are correlated, depending on the offsets of the non-zero coefficients of K, but experiment shows that $K \otimes \mathcal{S}^{\ell}_{\gamma}$ behaves mostly like $\mathcal{S}^{\ell-d_K}_{\gamma^{w_K}}$. So we write $K \otimes \mathcal{S}^{\ell}_{\gamma} \approx \mathcal{S}^{\ell-d_K}_{\gamma^{w_K}}$.

This probability can also be expressed using the central limit theorem (summing k times on the δ bits), and we get

$$\rho \approx k \cdot \varphi\left(-\sqrt{\frac{\gamma^{2w_K}}{1-\gamma^{2w_K}} \times \frac{\ell - d_K}{k}}\right). \tag{2}$$

where φ is the cumulative distribution function of a normal distribution:

$$\varphi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \mathrm{e}^{-t^2/2} dt.$$

Table 1. Examples of TCHo parameters vectors.

	k	$d_{\min}$–$d_{\max}$	d_K	w_K	γ	$\frac{1}{2}(1-\gamma^{w_K})$	ℓ	ρ
I_{65}	128	5 800–7 000	25 820	45	0.981	0.289	50 000	$2^{-26.5}$
II_{65}	128	8 500–12 470	24 730	67	0.987	0.292	68 000	$2^{-48.5}$
III	128	3 010–4 433	44 677	25	$1-\frac{3}{64}$	0.349	90 000	$2^{-22.4}$
IV	128	7 150–8 000	24 500	51	0.98	0.322	56 000	$2^{-22.9}$
V	128	6 000–8 795	17 600	81	$1-\frac{3}{128}$	0.427	150 000	$2^{-13.0}$
VI	128	9 000–13 200	31 500	65	$1-\frac{1}{64}$	0.320	100 000	$2^{-54.7}$

3.3 Selecting the Parameters

Table 1 shows some parameters suiting the security constraints for $\lambda = 80$.

Asymptotically, we choose the parameters in terms of λ and k as follows.

$$\begin{array}{lll} w_K = \Theta(\lambda) & d_K = \Theta(\lambda^2 \cdot k) & \ell = \Theta(\lambda^2 \cdot k) \\ d_{\min} = \Theta(\lambda^2) & d_{\max} = \Theta(\lambda^2) & \gamma = 1 - \Theta(\frac{1}{\lambda}) \end{array}$$

In addition to this, the plaintext length k must satisfy $k = \mathcal{O}(\lambda)$. We do not provide any fixed relation between k and λ because, depending on the application, we may either want to encrypt a constant-size plaintext (*i.e.* $k = \mathcal{O}(1)$) or a plaintext as long as possible (*i.e.* $k = \Theta(\lambda)$). With those parameters

- key generation takes $\mathcal{O}\left(\lambda^4 \cdot k^2 \cdot \log\lambda \cdot \log\log\lambda\right)$,
- encryption takes $\mathcal{O}\left(\lambda^4 \cdot k\right)$,
- decryption takes $\mathcal{O}\left(\lambda^3 \cdot k\right)$,
- the unreliability is $\rho = \mathcal{O}\left(\frac{k}{\lambda} \cdot 2^{-\lambda^2}\right)$ (heuristically),
- the private key length is $w_K \log_2 d_K = \mathcal{O}(\lambda \log\lambda)$,
- the public key length is $d_P = \mathcal{O}\left(\lambda^2\right)$,
- the plaintext length is k,
- the ciphertext length is $\ell = \mathcal{O}\left(\lambda^2 \cdot k\right)$.

4 Security

Clearly, TCHo is not OW-CCA secure: given a valid ciphertext, it suffices to modify one bit and ask an oracle to decrypt it to get with high probability the plaintext corresponding to the original ciphertext. Thus it is not IND-CCA secure either. Like RSA, TCHo is malleable, given a single ciphertext: if y is a ciphertext of x, then $y + C(\tilde{x})$ is a valid ciphertext of $x + \tilde{x}$, for any $\tilde{x} \in \{0,1\}^k$. In what follows we study semantic security.

Lemma 2. *There exists a constant ν such that for any λ, t, ε and* TCHo *parameters, if, for a random P generated by* TCHo *key generation, $\mathcal{S}^\ell_{\mathcal{L}_P} + \mathcal{S}^\ell_\gamma$ cannot be distinguished from $\mathcal{S}^\ell_0$ in time t with an advantage larger than ε, then* TCHo *encryption is $(t - \nu \cdot \ell, \varepsilon)$-IND-CPA secure.*

On the asymptotic side, letting t be polynomial and ε be exponentially small in terms of λ, we obtain that TCHo is IND-CPA secure.

Proof. We proceed by reduction: let $\mathcal{A}^{\mathsf{ror}} = (\mathcal{A}^{\mathsf{ror}}_1, \mathcal{A}^{\mathsf{ror}}_2)$ be an adversary in a real-or-random game, which, given a chosen plaintext $x = \mathcal{A}^{\mathsf{ror}}_1(1^\lambda)$ and a bitstring z of length ℓ, decides whether z is a ciphertext of x or of an unknown randomly chosen plaintext x'; this adversary returns $\mathcal{A}^{\mathsf{ror}}_2(z) \in \{0,1\}$, and succeeds with an advantage ε, in time t. Since a ciphertext of TCHo consists of some bitstring noised with a random source, the ciphertexts space is equal to $\{0,1\}^\ell$, so there are no trivial instances of the problem, and every element of $\{0,1\}^\ell$ can be a ciphertext of one or several messages.

We build a distinguisher between $\mathcal{S}^\ell_{\mathcal{L}_P} + \mathcal{S}^\ell_\gamma$ and $\mathcal{S}^\ell_0$ in the following way: given an unknown instance $\mathcal{S}^\ell_\star$, choose a plaintext $x = \mathcal{A}^{\mathsf{ror}}_1(1^\lambda)$ independently of $\mathcal{S}^\ell_\star$, and compute $z = C(x) + \mathcal{S}^\ell_\star$, then return $\mathcal{A}^{\mathsf{ror}}_2(z)$. If $\mathcal{S}^\ell_\star$ is random, then so is z, otherwise z is a valid ciphertext of x, therefore we got an adversary distinguishing a noised LFSR stream from random with exactly the same advantage than a real-or-random one, in time greater than t. As real-or-random security implies [5] semantic security, TCHo is IND-CPA secure.

The cost of simulation is $\mathcal{O}(\ell)$ so if $\mathcal{A}^{\mathsf{ror}}$ has complexity $t - \nu \cdot \ell$, for ν large enough, the distinguisher has complexity bounded by t. □

Let P be a random polynomial of degree $d_P \in [d_{\min}, d_{\max}]$ and weight w_P. In order to determine whether a bitstring is $\mathcal{S}^\ell_{\mathcal{L}_P} + \mathcal{S}^\ell_\gamma$ or $\mathcal{S}^\ell_0$, one strategy consists in multiplying the stream by P, and deciding whether the obtained stream has bias γ^{w_P} or not. It is infeasible to distinguish a random source with bias γ^{w_P} from a uniform one as soon as $\gamma^{w_P} < 2^{-\lambda/2}$. Instead of multiplying by P, one can multiply by multiples of P of lower weight and degree less than ℓ and use the obtained bits. For a random P there are on average $\binom{d-1}{w-2} 2^{-d_P}$ multiples of weight w and degree d with non-zero constant term. Hence the total number of bits of bias γ^w one can obtain using all the multiples of weight w is approximately

$$N_w \approx 2^{-d_P} \sum_{d=w-1}^{\ell-1} (\ell - d) \binom{d-1}{w-2} = 2^{-d_P} \binom{\ell}{w}.$$

When there are too many such bits, we must reduce this number. Let N be the number of used bits. We have $N \leq N_w$. If γ^w is small, the advantage of the best distinguisher using N bits is [4] $\mathsf{Adv} \approx \gamma^w \sqrt{N/(2\pi)}$. The complexity of the distinguisher using these N bits can be lower-bounded by the sum of

- wN (we have to calculate all bits),
- the cost of finding at least one multiple of P with degree up to ℓ and weight w, which can be lower bounded by $(\ell/d_P)^{w-1} \times 2^{d_P}/\binom{\ell}{w-1}$ (we use here the lower bound for syndrome decoding from [14]).

By optimizing over the choice of w and N, the best advantage-over-complexity ratio for this strategy is

$$R = \max_{\substack{w \in [0, d_P] \\ N \geq 1}} \frac{\gamma^w/\sqrt{2\pi}}{w\sqrt{N} + \frac{1}{\sqrt{N}} \left(\frac{\ell}{d_P}\right)^{w-1} \times \frac{2^{d_P}}{\binom{\ell}{w}}}.$$

Given the optimal w, the maximum in dependence of N is reached when

$$N = \max\left(1, \left(\frac{\ell}{d_P}\right)^{w-1} \times \frac{2^{d_P}}{w\binom{\ell}{w}}\right).$$

By using the approximation $\binom{\ell}{w} \approx \ell^w/w!$ and the Stirling approximation we can show that for $w \in [0.33d_P, 1.88d_P]$ this N is equal to 1. But then, R is bounded by $\gamma^w/w\sqrt{2\pi}$ which is maximal for the smallest w. On the other hand, for $w < 0.33d_P$ we can show that the R ratio increases with w so the best ratio is for the threshold w such that N decreases to 1. We deduce that $R = \mathcal{O}\left(\gamma^{\Omega(d_{\min})}/d_{\min}\right)$. With our asymptotic parameters, we obtain $R = \mathsf{exp}(-\Omega(\lambda))/\lambda^2$.

For a more precise bound we shall use

$$R = \max_{\substack{w \in [0, d_{\max}] \\ N \geq 1}} \frac{\gamma^w/\sqrt{2\pi}}{w\sqrt{N} + \frac{1}{\sqrt{N}} \left(\frac{\ell}{d_{\min}}\right)^{w-1} \times \frac{2^{d_{\min}}}{\binom{\ell}{w}}}. \tag{3}$$

Experience shows this is reached for $N = 1$. Intuitively, this means that using a single multiple polynomial which is essentially easy to get is the best strategy because the advantage benefit is not worth working hard on lowering w.

As an example, the parameter vector I_{65} (as well as II_{65}) in Table 1 gives $R \leq 2^{-65}$ for the optimal $w = 1936$ and $N = 1$. (Actually, all other parameter vectors satisfy $R \leq 2^{-80}$.) Note that in the worst case where $d_P = d_{\min}$, "random multiples" of P with degree close to $d_{\min}$ have random weights with expected value $d_{\min}/2 = 2900$ and standard deviation $\sqrt{d_{\min}}/2 = 38$. So, a weight of 1936 is within 25 standard deviations, which is pretty large. With higher degrees, the distance is more important. As our computation assumes that getting a bit of bias γ^w is easy, our analysis may still be pessimistic. So, those parameters may be more secure than what this $R \leq 2^{-65}$ bound suggests.

Assumption 3. *Suppose* $d_{\min} \geq 2\lambda$ *and* $\gamma \leq 2^{1-\lambda/d_{\min}} - 1$ *and the conditions of Assumption 1 are met. Then, for any* ℓ*, on average over* P *generated by* Gen *as defined in Assumption 1, a distinguisher between* $\mathcal{S}^{\ell}_{\mathcal{L}_P} + \mathcal{S}^{\ell}_{\gamma}$ *and* $\mathcal{S}^{\ell}_{0}$ *has an advantage/complexity ratio lower than* R *as defined by Eq.* (3).

This leads to the following result.

Theorem 4. *Under Assumptions 1 and 3, there exists* ν *such that for any* λ *and* t *and any* TCHo *parameters satisfying the conditions in Assumptions 1 and 3,* TCHo *is* $(t - \nu \cdot \ell, R \cdot t)$*-IND-CPA secure.*

Security Level Assessment. The above parameters provide semantic security against adversaries with an advantage/complexity ratio upper bounded by R as given by Eq. (3). More precisely, to compare this with a security level of an exhaustive key search for an s-bit key, we should set $R = 2^{-s}$ in Eq. (3). Asymptotically, we have $s = \Theta(\lambda)$. For the parameter vectors I_{65} and II_{65} we have $s \geq 65$. For all others we have $s \geq 80$.

5 Construction of an IND-CCA Secure Scheme

We propose a generic hybrid construction by using the (revisited) Fujisaki-Okamoto paradigm based on tag-KEM [1, 2, 15]. The encryption scheme obtained offers IND-CCA security when the public encryption scheme is OW-CPA and Γ-uniform, and the symmetric cipher one-time secure. For instance, one can simply choose $\mathsf{Sym}_{\mathsf{enc}(\psi)}(x) = x + F(\psi)$ for some random oracle F. The construction requires two random oracles H and G. The IND-CPA security of TCHo implies OW-CPA security, and the proof of Γ-uniformity of TCHo1 [14] applies to TCHo as well. So the following hybrid encryption scheme is IND-CCA secure.

Encryption. Given a message x:

1. Choose a random σ uniformly in $\{0,1\}^k$
2. Compute the symmetric key: $\psi \leftarrow G(\sigma)$
3. Encrypt the message x: $y \leftarrow \mathsf{Sym}_{\mathsf{enc}(\psi)}(x)$
4. Encapsulate the key: $\chi \leftarrow \mathsf{TCHo}_{\mathsf{enc}}(\sigma, H(\sigma||y))$
5. Output the ciphertext (χ, y).

Decryption. Given a ciphertext (χ, y):

1. Compute the encapsulated key: $\psi \leftarrow G(\mathsf{TCHo}_{\mathsf{dec}}(\chi))$
2. Decrypt the message: $x \leftarrow \mathsf{Sym}_{\mathsf{dec}(\psi)}(y)$
3. Output the plaintext x.

Table 1 shows examples of parameters for a symmetric encryption key of typical length 128 bits. So the construction encrypts a message with an overhead of ℓ bits (the length of a ciphertext in TCHo).

6 Implementation of TCHo

TCHo was implemented in C++, using the NTL library [26] for arithmetic over $\mathbb{F}_2[X]$, including GCD and factorization algorithms. All performances were measured on 1.5 GHz Pentium 4 computer.

6.1 Choice of Parameters

Here we summarize the inequalities that must hold to get IND-CPA and $2^{\Theta(\lambda)}$ security, deduced from Assumptions 1 and 3, when using block repetition codes.

- To correctly decrypt, ρ, given by Eq. (2), must be small.
- K must be impossible to recover from P:

$$\binom{d_K}{w_K-1} \leq 2^{d_{\min}} \text{ and } w_K \log_2 \frac{d_K}{d_{\max}} \geq \lambda.$$

- Semantic security is assumed to hold when

$$d_{\min} \geq 2\lambda, \gamma \leq 2^{1-\lambda/d_{\min}} - 1 \text{ and } R \leq 2^{-\lambda},$$

  where R is given by Eq. (3).

In practice, one may fix a block size k, a security level λ, and a ciphertext length ℓ, then deduce the degree and weight of K, an interval for the degree of a public key P, and a bias γ for the pseudo-random bits. But there is no strict rule to choose parameters $(k, d_{\min}, d_{\max}, d_K, w_K, \gamma, \ell)$, indeed TCHo is very flexible, and one may adapt them to its requirements, *e.g.* by allowing an average failure probability so as to reduce the expansion, or by setting a high degree d for the private key K and a high expansion in order to get a negligible error probability ρ, at the price of a very long key generation. Experiments in Section 6.3 will give concrete examples of these trade-offs.

6.2 Chosen Algorithms

Our LFSR implementation uses a variant of the block-oriented algorithm introduced in [9, 10]. In software, LFSR's are slower than in hardware; for a random polynomial of degree 6 000, our implementation could only reach a rate of 150 Kb/s. The number of bitwise operations required to compute a bitstream of length ℓ is roughly $\frac{1}{16}\ell d_P$. Our generator for $\mathcal{S}_\gamma$ uses a source of uniform pseudo-random bits to produce blocks of n bits in two steps:

1. pick a weight $q \in [0, n]$ (with suitable probability distribution),
2. pick a word of weight q (uniformly).

The first step is accomplished by partitioning the interval $[0, 1] \subset \mathbb{Q}$ into n intervals with respect to the weight distribution induced by the bias, and then picking a random, uniform rational number in this interval with high enough precision. For blocks of 32 bits and precision 2^{-64}, the statistical distance to

the ideal generator is negligible. The pseudo-random generator ISAAC [19] is used as a source of random bits[7]. Compared to the LFSR, our generator is quite efficient: more than 28 Mb of biased random bits are produced per second.

6.3 Software Implementation Results

Table 2 shows performances for the repetition codes scenarios described in Table 1. Encryption time is roughly equal to the time needed to compute $\mathcal{S}^{\ell}_{\mathcal{L}_P(r_1)}$ (in all scenarios $\mathcal{S}^{\ell}_{\gamma}$ is computed in less than 1 ms), while for decryption the most expensive operation is the multiplication by K (majority decoding and product by the precomputed matrix require less than 1 ms).

Table 2. Performances of TCHo with repetition codes.

	enc. (ms)	dec. (ms)	kgen. (s)	unreliability	sec. key (bit)	pub. key (bit)	plaintext (bit)	ciphertext (bit)
I_{65}	38.7	47.4	1 180	$2^{-26.5}$	455	7 000	128	50 000
II_{65}	148.0	115.4	361	$2^{-48.5}$	507	12 470	128	68 000
III	75.5	49.0	2 290	$2^{-22.4}$	281	4 433	128	90 000
IV	90.1	65.1	1 970	$2^{-22.9}$	506	8 000	128	56 000
V	228.4	423.7	200	$2^{-13.0}$	726	8 795	128	150 000
VI	232.5	178.7	870	$2^{-54.7}$	652	13 200	128	100 000

Using precomputed look-up tables could speed up encryption: given a P, we can compute a table of $d_P \times \ell$ bits, containing the bitstreams produced by each initial state of $\mathcal{L}_P$ of weight 1. Computing such a table takes less than a second using optimized algorithms, then the generation of a bitstream requires roughly $\frac{\ell}{32} \times \frac{d_P}{2}$ XOR operations (in our implementation, with a 32 bits processor). Experimentally the time gain is not significant, since memory access takes a non-negligible time (about 70 megabytes are precomputed for common parameters).

Results in Table 2 show that a trade-off must be made between key generation time, encryption and decryption time, ciphertext expansion, and reliability. The parameter sets proposed all tend to optimize one of these points while keeping the others at a reasonable level. Depending on the application, users should choose one set or an other.

- $\mathbf{I_{65}}$: Fast encryption/decryption for low security requirements of 2^{65}.
- $\mathbf{II_{65}}$: Well balanced parameters for low security requirements.
- **III**: Fast encryption/decryption. This also implies smaller key sizes.
- **IV**: Smaller message expansion and reasonably fast encryption/decryption.
- **V**: "Fast" key generation.
- **VI**: Negligible unreliability is reached.

[7] Some weaknesses on ISAAC were reported in [3]. So the question whether ISAAC is still appropriate for our design is left open.

One can note that, even though it is possible to improve them a little, the ciphertext expansion and the key generation time will always remain very high. Concerning ciphertext expansion it is possible to improve it significantly by encrypting larger blocks. For a standard 128 bit key exchange, it seems impossible to go below blocks of 50 000 bits (for a security of 2^{80} operations), but if more data needs to be exchanged, using larger blocks (while adjusting ℓ so as to keep the same unreliability) can decrease expansion to a factor of about 100.

In contrast, not much can be done concerning the prohibitive key generation time. Given the values of $d_{\min}$ and w_K, while keeping the security constant, it is possible to choose optimal values for $d_{\max}$ and d_K. These values will always correspond to $d_{\max} \approx 1.5\, d_{\min}$ (it would be an equality if factorization was done in quadratic time). However, factoring a polynomial of degree over 20 000 is a costly operation which is difficult to speed-up.

6.4 Hardware Implementation

Encryption requires the computation of ℓ bits from a large LFSR, as many bits with bias γ, and the repetition of the plaintext $\lfloor \ell/k \rfloor$ times. Let's examine those three operations.

- LFSR's can be very fast in integrated circuits: the number of gates required is roughly equal to the length of the register, and it outputs one bit per clock cycle. We assume that the over-cost induced by our large registers does not dramatically slow down the computation, and remains feasible in spite of the unusual size.
- To compute the non-uniform random bitstream, one may use a specially tuned generator fed with physical entropy; otherwise, a solution is to use an algorithm producing non-uniform random sequence from a uniform one. For instance, to generate words of given length, one may use a binary search tree (precomputed) where each leaf is labeled with a word, and go through the tree by successive coin flips in order to simulate the bias. Such a construction roughly requires as many uniform bits as biased bits produced (in comparison, our software generator needs about three uniform bits to compute a biased one).
- Repetition of a word is straightforward.

Note that, since the operations are independent, parallelization is possible.

Decryption looks more complicated to implement, but it only consists of linear operations over $\mathbb{F}_2$, usually easily implemented. For instance, there exists [29] a library for FPGA devices performing matrix-vector product and dot product efficiently (note that the product $K \otimes \mathcal{S}$ is simply a sequence of dot products). It also requires a small amount of additional memory to perform the majority decoding (namely $k \log_2 \frac{\ell - d_K}{k}$ bits to count the number of occurrences of each bit of $\tilde{m}$).

It thus appears that TCHo's encryption and decryption only need hardware-friendly operations (no integer multiplication or addition, no modular arithmetic). However, the implementation should be flexible, so as to be adaptable

to any public key – that is, tune the LFSR taps. Unfortunately, we could not implement TCHo in a hardware environment, but we can estimate requirements and performances: looking at the parameters in Table 1, a 128-bit key can be encrypted with a circuit of about 10 000 gates (for the LFSR and the repetition), with an external source of randomness. With an ASIC running at 4 MHz (0.25 μs cycle time), we roughly estimate encryption time to 15 ms. The power consumption is estimated to be of at most 20-100μW, which is suitable for RFID.

7 Comparison with Other Cryptosystems

The security of TCHo relies mostly on results from coding theory and it is thus tempting to compare it to the famous code based cryptosystem of McEliece [23]. The two cryptosystems function in a similar way: first the message is encoded using a public code, then some random noise is added to it. However, the two constructions are quite different in the way noise is added: in McEliece's cryptosystem, a small amount of completely random noise is added to the codeword, whereas in TCHo a huge amount of structured noise is added. In TCHo, this noise should even be indistinguishable from an unbiased random binary sequence: decoding is only possible because this noise has a hidden structure. In McEliece, it is the code which contains a hidden structure which make decoding possible.

To measure the efficiency of TCHo, we need to compare both the timing we obtained for practical parameters and the asymptotic complexities of TCHo with those of other ciphers. For practical comparisons we used the benchmark feature of the `Crypto++` library [11] running on the same 1.5 GHz Pentium 4 as our tests. We then use RSA 1024/2048 as a reference for comparison with other systems. Results are presented in Table 3. The key generation time of TCHo is of course way higher than for any other public key cryptosystem, however, encryption and decryption speed are close to those of RSA or elliptic curve cryptosystems [20]. NTRU [17] is however much faster. Anyway, we believe that for a hardware oriented cryptosystem these performances are not bad.

From an asymptotic point of view, things are a little different. We need to compare parameters yielding an equivalent asymptotic security of 2^λ. For RSA this means that we use a modulus of size $\mathcal{O}\left(\lambda^3\right)$ and for EC a group of order

Table 3. Comparison of TCHo with other public-key cryptosystems.

	security	enc. (ms)	dec. (ms)	kgen. (s)	sk/pk (bit)	pt (bit)	ct (bit)
TCHo I_{65}	2^{65}	38.7	47.4	1 180	455/7 000	128	50 000
TCHo IV	2^{80}	90.1	65.1	1 970	506/8 000	128	56 000
RSA 1024	2^{72}	0.4	12.8	0.3	2 048/1 024	1 024	1 024
RSA 2048	2^{102}	1.0	75.0	1.8	4 096/2 048	2 048	2 048
EC on $GF(2^{163})$	2^{78}	16.9	10.2	–	160/326	160	326
NTRU `ees251ep4`	2^{80}	~ 0.1	~ 0.2	~ 0.003	502/2 008	251	2 008

Table 4. Asymptotic comparison of TCHo with other cryptosystems (the $\mathcal{O}()$'s have been omitted).

	security	enc.	dec.	kgen.	sk/pk	pt	ct
TCHo	2^λ	λ^5	λ^4	$\lambda^6 \cdot \log\lambda \cdot \log\log\lambda$	$\lambda \cdot \log\lambda/\lambda^2$	λ	λ^3
RSA	2^λ	λ^6	λ^9	λ^{12}	λ^3/λ^3	λ^3	λ^3
EC	2^λ	λ^3	λ^3	λ^3	λ/λ	λ	λ
NTRU	2^λ	λ^2	λ^2	λ^2	λ/λ	λ	λ
McEliece	2^λ	λ^2	$\lambda^2 \cdot \log\lambda$	λ^3	λ^2/λ^2	λ	λ

$2^{\mathcal{O}(\lambda)}$. For NTRU, the asymptotic complexity is not explicitly known, but it is assumed that a length of $\mathcal{O}(\lambda)$ can achieve a security of 2^λ. The results obtained are reported in Table 4, where we also added the McEliece cryptosystem[8] [23]. It appears that TCHo is better than RSA on all points, including the key generation complexity. However, some alternate public-key cryptosystems remain better asymptotically.

8 Conclusion

Our TCHo cryptosystem is much more efficient than TCHo1: encryption and decryption algorithms are faster, larger blocks can be encrypted, a precise estimate of the decryption failure probability can be given, and experimental results are much better than for TCHo1. Meanwhile, TCHo performs pretty well asymptotically. It is semantically secure, which makes it possible to use it to build an IND-CCA secure hybrid encryption scheme using the KEM/DEM framework. However, it inherits some undesirable properties of the original scheme: first the key generation is still heavy and the expansion rate remains huge.

As TCHo seems well suited for tiny hardware we may consider using it for ensuring strong privacy in RFID as suggested in [27].

Finally, as TCHo security only relies on heuristic assumptions, further work could be devoted to giving concrete elements of proof or attack.

References

1. Masayuki Abe, Rosario Gennaro, and Kaoru Kurosawa. Tag-KEM/DEM: A new framework for hybrid encryption. IACR ePrint archive 2005/027, 2005. Available at `http://eprint.iacr.org/2005/027`. Newer version in [2].
2. Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *EUROCRYPT'05*, pages 128–146, 2005. Older version in [1].

[8] We could not find any practical timings to include in Table 3. For asymptotic behavior we use a code of length 2^m correcting t errors, with $t = \mathcal{O}\left(\frac{\lambda}{\log\lambda}\right)$ and $m = \mathcal{O}(\log t + \log\log t)$.

3. Jean-Philippe Aumasson. On the pseudo-random generator ISAAC. IACR ePrint archive 2006/438, 2006. Available at `http://eprint.iacr.org/2006/438`.
4. Thomas Baignères, Pascal Junod, and Serge Vaudenay. How far can we go beyond linear cryptanalysis? In P.J. Lee, editor, *ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2004.
5. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, page 394. IEEE Computer Society, 1997.
6. Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.
7. Anne Canteaut and Michaël Trabbia. Improved fast correlation attacks using parity check equations of weight 4 and 5. In B. Preneel, editor, *EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer, 2000.
8. David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.
9. Sandeepan Chowdhury and Subhamoy Maitra. Efficient software implementation of linear feedback shift registers. In C. Pandu Rangan and C. Ding, editors, *INDOCRYPT'01*, volume 2247 of *Lecture Notes in Computer Science*, pages 297–307. Springer, 2001.
10. Sandeepan Chowdhury and Subhamoy Maitra. Efficient software implementation of LFSR and boolean function and its application in nonlinear combiner model. In J. Zhou, M. Yung, and Y. Han, editors, *ACNS 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2003.
11. Wei Dai. Crypto++ library. `http://www.eskimo.com/~weidai/`.
12. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
13. James H. Ellis. The possibility of secure non-secret digital encryption. GCHQ-CESG publication, 1970.
14. Matthieu Finiasz and Serge Vaudenay. When stream cipher analysis meets public-key cryptography (invited talk). To appear in the Proceedings of SAC 2006, *Lecture Notes in Computer Science*.
15. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
16. Kishan Chand Gupta and Subhamoy Maitra. Multiples of primitive polynomials over GF(2). In C. Pandu Rangan and C. Ding, editors, *INDOCRYPT'01*, volume 2247 of *Lecture Notes in Computer Science*, pages 62–72. Springer, 2001.
17. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In J. Buhler, editor, *ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
18. K Jambunathan. On choice of connection-polynominals for LFSR-based stream ciphers. In B. K. Roy and E. Okamoto, editors, *INDOCRYPT'00*, volume 1977 of *Lecture Notes in Computer Science*, pages 9–18. Springer, 2000.
19. Robert J. Jenkins, Jr. ISAAC. In D. Gollmann, editor, *FSE'96*, volume 1039 of *Lecture Notes in Computer Science*, pages 41–49. Springer, 1996.
20. Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

21. Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.
22. Subhamoy Maitra, Kishan Chand Gupta, and Ayineedi Venkateswarlu. Results on multiples of primitive polynomials and their products over GF(2). *Theoretical Computer Science*, 341(1-3):311–343, 2005.
23. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Prog. Rep.,* Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114–116, January 1978.
24. Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer, 1988.
25. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
26. Victor Shoup. NTL: A library for doing number theory. `http://shoup.net/ntl/`.
27. Serge Vaudenay. RFID privacy based on public-key cryptography (invited talk). In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC'06*, volume 4296 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2006.
28. David Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 2002.
29. Ling Zhuo and Viktor K. Prasanna. High performance linear algebra operations on reconfigurable systems. In *SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 2. IEEE Computer Society, 2005.

A On the Choice of the Code

TCHo1 uses a code C generated by $\mathcal{L}_Q^\ell$ with a primitive polynomial Q of degree k. The drawback of this code is that decoding requires $\mathcal{O}\left(k2^k\right)$.

Note that if Q is a trinomial, decoding algorithms more efficient than MLD exist; the Algorithm B in [24] or Gallager decoding as used, *e.g.*, in [7] for fast correlation attacks can be applied. The success probability of these algorithms depends on the weight of the feedback polynomial of the LFSR, the bias γ^{w_K}, and the ratio between the length of known output and the size of the LFSR for which the initial state is searched for. Again, concerning the reliability of these iterative algorithms, only experimental results seem to be available. For trinomials it can be seen from Table 3 in [24] that, for example, correct decoding is expected if the known output has length 100 times the LFSR-length, and $\frac{1}{2}(1+\gamma^{w_K})$ is 0.6 or larger.

We rather use block repetition codes which is equivalent to setting $Q = X^k+1$ in TCHo1 although this would be illegal in TCHo1 since X^k+1 is not primitive.