When Stream Cipher Analysis Meets Public-Key Cryptography

Matthieu Finiasz and Serge Vaudenay

EPFL
CH-1015 Lausanne – Switzerland
http://lasecwww.epfl.ch/

Abstract. Inspired by fast correlation attacks on stream ciphers, we present a stream cipher-like construction for a public-key cryptosystem whose security relies on two problems: finding a low-weight multiple of a given polynomial and a Hidden Correlation problem. We obtain a weakly secure public-key cryptosystem we call TCHo (as for Trapdoor Cipher, Hardware Oriented). Using the Fujisaki-Okamoto construction, we can build an hybrid cryptosystem, TCHo^n-FO, resistant against adaptive chosen ciphertext attacks.

1 Introduction

Many of nowadays cryptosystems rely on the problem of factoring numbers. With the RSA [24] cryptosystem, the key recovery problem is equivalent to factoring a modulus into prime numbers. The message recovery problem is an ad-hoc problem, assumed to be hard, but potentially easier than factoring. To setup the cryptosystem we first generate secret prime numbers and then multiply them together to get the public key. Although the hardness of the factoring problem is an important open problem, it is known to be easy by using quantum computers [26].

With polynomials, the factoring problem is essentially easy. However, the problem of finding a multiple with low degree and weight is presumably hard. This problem occurs in correlation attacks on stream ciphers, and no polynomial time algorithm exists to solve it. Therefore, we can setup a new trapdoor system by first generating a secret low-weight polynomial and then looking for a suitable factor to produce a public key. We thus derive a new cryptosystem consisting simply of the XOR of two LFSRs with a random noise source. One LFSR is used to encode the data, the other contains the trapdoor. It is only used to hide the data, and the noise source provides non-linearity. The ciphertext is the result of this XOR. The key recovery problem is equivalent to finding a low-weight multiple of a given polynomial. The message recovery problem is an ad-hoc problem, potentially easier but still (assumed to be) hard: the Hidden Correlation (HC) problem. It further may remain a hard problem with quantum computers.

We think that analyzing this cryptosystem would be important in *any* case because this would either provide us with a secure post-quantum cryptosystem

(if not an efficient pre-quantum one), or with new improvements for existing correlation attacks against stream ciphers. We aim at making the first step in this direction.

In this paper we review existing algorithms for finding low-weight multiples of a given polynomial and for the decoding of a noisy LFSR. This provides heuristics to select parameters for TCHo, a one-way public-key encryption scheme. It can encrypt small blocks of up to 30 bits into ciphertexts of a few thousand bits. TCHo^n encrypts n such blocks independently. The stream-cipher-like structure of our construction makes it most suitable for hardware implementation: an ASIC of a few thousand gates (equipped with a randomness generator) at 4MHz should be able to encrypt a block in less than 4ms.

We believe TCHo could also fit on a passive RFID tag. Providing public-key encryption to RFID tags leads to new opportunities to solve privacy/security issues in RFID protocols. Current protocols with readers connected to a single server [1, 23] are based on symmetric cryptography and either compromise privacy or induce an important overhead complexity on the reader side. Public-key cryptography is the simplest (and most reliable) solution to solve these issues, however, no public-key encryption primitive can fit on an RFID tag. TCHo could possibly be the first construction to achieve this.

We have implemented the TCHo construction in C++ using `NTL` [27]. Key generation is pretty slow (about 4 minutes) as it requires to factorize polynomials with huge degrees. Encryption takes a fraction of a second in software, but this time is not significant for hardware implementation: LFSRs are not well suited for use with a CPU. Finally, our decryption implementation takes a few seconds.

Previous work. Quite a lot of work has been done trying to attack various stream ciphers like constructions. The first attacks were the (fast) correlation attack, invented by Siegenthaler [28] and improved by Meier and Staffelbach [20]. The main idea is to isolate one linear feedback shift-registers (LFSR) inside a stream cipher and approximate the rest of the construction by another large LFSR with some noise, then try to recover the initialization of the first LFSR and get a part of the key. Many improvements and applications to specific constructions have been made [4, 5, 13–15, 18, 21]. This work was inspired by a preliminary attack on Bluetooth E0 [17, 19]. Some of the most efficient techniques are compared in Section 2.2.

A key problem in most of these techniques is the possibility to find low-weight multiples of a given polynomial. The best techniques rely either on exhaustive search or on decoding techniques [3, 4, 22, 30]. Section 2.1 of this article is devoted to this problem.

More recently, algebraic attacks have been applied to break some constructions [6–8, 10, 11]. These attack can be very efficient but, even if some bounds have been proven [7, 9], predicting their efficiency is quite a difficult problem. Throughout this article we will neglect this category of attacks as they are not well suited for large randomized constructions like ours.

The concept of trapdoor in a stream cipher was already brought up by Camion, Mihaljevic and Imai [2]. They showed that using specific retroaction

polynomials in LFSRs, some bits of the output stream could depend on only some bits of the initialization, making it possible to speed up the recovery process for constructions which otherwise seemed perfectly sound. This can be well suited to kleptographic attack but it seems hard to build a public-key cryptosystem this way.

In Section 2, we define new computational problems and provide heuristics to solve them. This suggests some parameters ranges in which they can be reasonably assumed to be hard on average. Then we define a raw encryption scheme in Section 3 and prove that it is one-way under chosen plaintext attacks.

2 New Computational Problems

In this section, we formalize some computational problems which will be used and review existing algorithms to solve them. We focus on *exact complexities* as opposed to asymptotic ones. For this, we use two constants c_{easy} and c_{hard} to indicate that a complexity below $2^{c_{\mathsf{easy}}}$ operations is *essentially easy in practice* and efficient, and that a complexity over $2^{c_{\mathsf{hard}}}$ is *intractable in practice*. In practice, we can take $c_{\mathsf{easy}} = 30$ and $c_{\mathsf{hard}} = 80$.

Throughout this paper, we will define the bias of a source as $\mathrm{bias}(\mathcal{S}) = P(\mathcal{S} = 0) - P(\mathcal{S} = 1)$. This bias is hence taken between -1 and 1 and is null when the source is unbiased. In our system, we will only consider positive biases, that is, sources which produce more 0's than 1's.

2.1 The Low-Weight Polynomial Multiple Problem

As opposed to integers, factoring polynomials is essentially easy. But when it comes to finding low-weight polynomial multiples, the problem becomes harder (at least, it is not known to be easy). Here, we will only focus on polynomials over $\mathbf{F}_2$.

Problem 1 (Low-Weight Polynomial Multiple). We consider the following problem on average over the random selection of an instance P, issued by a given instance generator Gen. We say that the problem is hard for Gen if solving it requires more than $2^{c_{\mathsf{hard}}}$ complexity on average. We consider two kinds of generator. Let Gen_1 be a generator which selects a random primitive polynomial P of degree d_P. Let Gen_2 be a generator which selects a random polynomial K of degree d_K and weight w until it has a primitive factor P, produced as the output, whose degree d_P is in a given interval $[d_{\min}, d_{\max}]$.

Parameters: a weight w, two degrees $d_P < d_K$
Instance: a binary polynomial P of degree d_P
Problem: find a multiple of P with degree at most d_K and weight at most w

The Gen_1 generator simulates which P we can meet when we do stream cipher cryptanalysis. The Gen_2 generator simulates which P we can meet when we construct a public-key cryptosystem.

Note that if P is irreducible, if x has order $n \leq d_K$ in the group $(\mathbf{F}_2[x]/P(x))^*$, and if $w \geq 2$, then $K = x^n + 1$ solves the problem. Taking P primitive ensures that the order $n = 2^{d_P} - 1$ is maximal. Also, for values of w greater than the Hamming weight of P, then P itself solves the problem. However, for d_K and w small, the problem is believed to be hard. There are several strategies to solve it. Heuristically, if P is generated by Gen_1, the average number of solutions with nonzero constant term is

$$\mathcal{N}_{\mathsf{sol}} = 2^{-d_P} \sum_{i=0}^{w-1} \binom{d_K}{i} \approx 2^{-d_P} \binom{d_K}{w-1}. \tag{1}$$

When using Gen_2, the average number of solutions becomes $1 + \mathcal{N}_{\mathsf{sol}}$.

Strategy 1 – The birthday paradox. This strategy consists in building two lists of polynomials which are sums respectively of 0 or 1 and of a polynomial with weight $\frac{w-1}{2}$ and null constant term, all reduced modulo P. Once this is done, one simply looks for collisions. The lists have a size of $L = \binom{d_K}{(w-1)/2}$ and the complexity is $\mathcal{O}\left(L(\log(L) + d_P)\right)$. This strategy is always faster than exhaustive search, but requires a lot of memory.

Strategy 2 – Wagner's generalized birthday paradox. When the number of solutions becomes large enough (of order $\mathcal{O}\left(2^{d_P/3}\right)$), techniques based on Wagner's generalized birthday paradox [30] can become more efficient. This algorithm is not fit for finding all possible polynomials (or the hidden one from Gen_2) but can find one solution among many. If there exists $a \geq 2$ such that $\binom{d_K}{(w-1)/2^a} \geq 2^{d_P/(a+1)}$, then one solution can be found with a complexity $\mathcal{O}\left(2^a 2^{d_P/(a+1)}\right)$.

For instance, when $d_K \geq 2^{d_P/(1+\log_2(w-1))}$, we can use $a = \log_2(w-1)$ and find a multiple within $\mathcal{O}\left((w-1)d_K\right)$. Clearly, such a low complexity cannot be reached for any $w \leq d_P$ when $d_K \leq 2^{d_P/(1+\log_2(d_P-1))}$.

Strategy 3 – Syndrome decoding. Solving the problem can be done using a syndrome decoding algorithm. Compute the matrix of all the $x^i \mod P(x)$ for i from 1 to d_K and then find a low-weight word in the preimages of 1 of this matrix. When a single solution exists, this has a cost of:

$$\mathcal{O}\left(\mathcal{P}oly\left(d_K\right)\left(\tfrac{d_K}{d_P}\right)^{w-1}\right), \tag{2}$$

where $\mathcal{P}oly\left(d_K\right)$ is a polynomial of degree 2 or 3 in d_K (see [16] for example). We neglect this polynomial part as improved algorithm like [3] can compensate it. This complexity holds when there is a unique multiple polynomial of degree d_K and weight $\leq w$. When there are more solutions this cost is approximately divided by $\mathcal{N}_{\mathsf{sol}}$.

Strategy 4 – Exhaustive search. When looking for multiples of degree just above d_P, an exhaustive search on Q such that $K = P \times Q$ can be faster. The complexity of finding all multiples is $\mathcal{O}\left(\mathcal{P}oly\left(d_K\right)2^{d_K - d_P}\right)$.

The best algorithm for finding low-weight multiples depends on both the parameters and our objective (whether we want to find a single, many, or all solutions). When a single solution exists, the best choice is Strategy 3. This leads us to the following assumption

Assumption 2 (Low-Weight Polynomial Multiple). *When* $w \log_2 \frac{d_K}{d_P} \geq c_{\mathsf{hard}}$ *and* $\binom{d_K}{w-1} \leq 2^{d_P}$*, the low-weight polynomial multiple problem is hard on average for* Gen_1*. When* $w \log_2 \frac{d_K}{d_{\min}} \geq c_{\mathsf{hard}}$ *and* $\binom{d_K}{w-1} \leq 2^{d_{\min}}$*, the low-weight polynomial multiple problem is hard on average for* Gen_2*.*

2.2 The Noisy LFSR Decoding Problem

A binary linear code of length ℓ is a vector subspace of $\{0,1\}^\ell$. Elements are codewords. We consider the problem of decoding *noisy* strings, i.e. decoding the XOR of a codeword together with the output of a random source $\mathcal{S}_\gamma$ with bias γ. This source represents the error produced by a binary symmetric channel. In what follows we concentrate on codes which consist of all possible ℓ-bit strings which can be output from an LFSR $\mathcal{L}_P$ with a fixed retroaction polynomial P of degree d_P, i.e. sequences $Z = (z_1, \ldots, z_\ell)$ such that $(z_t, \ldots, z_{t+d_P}) \bullet P = 0$ for $t = 1, \ldots, \ell - d_P$ (where $\bullet$ denotes a scalar product, which means we consider P as a binary $(d_P + 1)$-tuple). We formalize the decoding problem of the noisy LFSR channel as follows.

Problem 3 (Noisy LFSR Decoding). We consider the following problem on average over the random selection of P, X, and the biased noise. We say that the problem is hard if getting a single bit of X (that is, decoding X with probability higher than 2^{1-d_P}) requires over $2^{c_{\mathsf{hard}}}$ complexity on average.

Parameters: a length ℓ, a polynomial P of degree d_P, a bias γ
Noisy LFSR channel: given a uniformly distributed random seed X of length d_P, generate Y, the XOR of the output of length ℓ of $\mathcal{L}_P$ initialized with X and a random noise generated by $\mathcal{S}_\gamma$
Problem: given Y, recover X

When γ is so close to 1 that errors are unlikely (e.g. $1-\gamma \ll \ell^{-1}$), the problem can easily be solved with high probability of success by Gaussian elimination. When γ is so small that we cannot even distinguish $\mathcal{S}_\gamma$ from an unbiased source (e.g. $\gamma \ll \ell^{-\frac{1}{2}}$), the problem is impossible to solve with relevant probability of success. In what follows we may assume that the channel transmits less data than its capacity[1] $C(\gamma) = 1 + \left(\frac{1}{2} + \frac{\gamma}{2}\right) \log_2 \left(\frac{1}{2} + \frac{\gamma}{2}\right) + \left(\frac{1}{2} - \frac{\gamma}{2}\right) \log_2 \left(\frac{1}{2} - \frac{\gamma}{2}\right)$, i.e. $\frac{d_P}{\ell} \leq C(\gamma)$. Thus, with unbounded computational power, decoding is possible. However, this is not always the case in only $\mathcal{O}\left(2^{c_{\mathsf{hard}}}\right)$ operations.

Three main classes of algorithms exist to solve this problem: those based on information set decoding, those trying to perform maximum likelihood (noted ML hereafter) decoding, and those based on iterative decoding techniques. Note

[1] When γ is small, we have $C(\gamma) \approx \frac{\gamma^2}{2 \log 2}$ (with less than 1% error for $\gamma \leq \frac{1}{4}$).

that by simply guessing the noise weight, the problem reduces to a permuted kernel problem [25].

Information Set Decoding consists in picking d_P bits at random among the ℓ output bits and perform a Gaussian elimination on the corresponding columns of the generator matrix of the LFSR. Decoding succeeds if there are no error among the selected bits, namely with probability $\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{d_P}$. By iterating this simple algorithm enough time to get the correct decoding we can decode within a complexity roughly $\left(\frac{1}{2}+\frac{\gamma}{2}\right)^{-d_P}$ (improved decoding algorithms like [3] make it possible to neglect the cost of the Gaussian elimination). When the bias γ is too small, namely for $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_P}-1$, this requires over $2^{c_{\mathsf{hard}}}$ iterations.

The converse approach consisting in trying to guess the error bits could seem interesting when the level of noise is low (γ close to 1), but it is never more efficient than Information Set Decoding.

Maximum Likelihood decoding consists in trying to find the most likely X, given the ℓ-bit output stream we have. As the source $\mathcal{S}_\gamma$ is memoryless with a positive bias, the maximum likelihood corresponds to the X generating the closest (in terms of Hamming distance) codeword to Y.

This gives us a basic ML decoding algorithm: try all the possible initializations for $\mathcal{L}_P$ and sort them according to their distance to Y. This is however very costly (about $\mathcal{O}\left(2^{d_P}\ell\right)$) and can be slightly improved to $\mathcal{O}\left(2^{d_P}d_P\right)$ by using a Walsh transform as done in [19]. This method is successful if the solution is on top of the list. This is the case only if we try to transmit less data than the channel capacity. For $d_P \leq c_{\mathsf{easy}}$ and $\frac{1}{4} \geq \gamma \geq \sqrt{\frac{d_P}{\ell}2\log 2}$ (the $\frac{1}{4}$ is here to ensure that the approximation is valid), we can thus efficiently solve the problem. On the contrary, decoding can be impossible for two different reasons: either the noise is too high (that is, the bias is too small) and we sent more data than the channel capacity (for instance, if $\gamma \leq 1.18\,\ell^{-\frac{1}{2}}$, the channel cannot transmit more than one bit of information, which means Y cannot contain more than one bit of information on X), or the cost of the maximum likelihood decoding algorithm is too high.

First case – If we send more data than the capacity of the channel we will not be able to recover X completely, but we can still get some information on it. This would not be sufficient to solve a decoding problem, but this is already enough to threaten a cryptographic construction.

The difference between the distance of Y to any incorrect codeword and the distance to the correct one can be approximated to a normal law of expected value $-\gamma\frac{\ell}{2}$ and variance $\frac{\ell}{2}$ so the expected rank of the correct X in the maximum likelihood list is approximately $1+(2^{d_P}-1)\varphi\left(\frac{-\gamma}{2}\sqrt{\ell}\right)$ where

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

The amount of information one can get on X depends directly on this rank.

Second case – If the complexity of the ML decoding algorithm is too high, we can try too improve it. More subtle algorithms exist to recover the initialization of $\mathcal{L}_P$: they cannot decode as much noise as the basic algorithm but can have a significantly lower complexity. The basic idea is that instead of decoding in the full code it is possible to decode only in a subcode and then extend the decoding to the rest of the code. For example, one could write the generator matrix of the code defined by P and only consider columns ending with $d_P - k$ zeros: this way one would have to decode in a code of dimension k (instead of d_P) but with less bits, that is a length $\ell.2^{-(d_P-k)}$ on average. When γ is small, the right X tops the ML list if $\frac{k}{\ell.2^{-(d_P-k)}} \leq \frac{\gamma^2}{2\log 2}$. In general, it takes time $2^k\varphi\left(-\frac{\gamma}{2}\sqrt{\ell 2^{-(d_P-k)}}\right)$ to find the right k bits of X.

This improvement makes it possible to recover the initialization of an LFSR at a lower cost, provided enough output bits are available. This approach can still be improved: if too few output bits are available, we can compute new ones from those we have. If we write all the output bits which are the XOR of d different bits, starting from ℓ bits we can obtain $\binom{\ell}{d} \simeq \frac{\ell^d}{d!}$ bits instead of ℓ, but with a noise of bias γ^d instead of γ. This means that, for any value of d (the size of the combinations we consider) and k (the dimension of the subcode we obtain) we can decode if:

$$\frac{kd!}{\ell^d \cdot 2^{-(d_P-k)}} \leq \frac{\gamma^{2d}}{2\log 2}. \tag{3}$$

In general, even if the previous bound is not reached, the correct X has rank:

$$2^k\varphi\left(-\frac{\gamma^d}{2}\sqrt{\tfrac{\ell^d}{d!}2^{-(d_P-k)}}\right).$$

The complexity of the decoding is then the sum of the three following steps:

- Computing the combinations: $\mathcal{O}\left(\frac{\ell^d}{d!} \times k\right)$
- Decoding (with a Walsh Transform): $\mathcal{O}\left(k2^k\right)$.
- Finding the right codeword in the ML list: $\mathcal{O}\left(2^k\varphi\left(-\frac{\gamma^d}{2}\sqrt{\frac{\ell^d}{d!}2^{-(d_P-k)}}\right)\right)$

If one cannot pay more than $\mathcal{O}\left(c_{\mathsf{hard}}2^{c_{\mathsf{hard}}}\right)$ complexity, we must have $k \leq c_{\mathsf{hard}}$ and d roughly less than $c_{\mathsf{hard}}\frac{\log 2}{\log \ell}$ so that $\frac{\ell^d}{d!} \leq 2^{c_{\mathsf{hard}}}$. The rank in the ML list is $2^k\varphi(-t)$ where $t \leq \frac{\gamma^d}{2}2^{c_{\mathsf{hard}}-\frac{d_P}{2}}$. For $d_P \geq 2c_{\mathsf{hard}}$ we have $t \leq \frac{1}{2}$. So the rank is higher than $2^k\varphi(-\frac{1}{2}) \geq 0.3 \cdot 2^k$. So, this algorithm yields less than one bit of information of X.

Iterative decoding techniques are less efficient in terms of error correction, but can be applied to longer LFSRs. The idea is to find low-weight multiples of P which form some parity check equations, and use them to decode as in a Low Density Parity Check (LDPC) code.

As stated in [4], iterative decoding using parity check equations of weight $d \geq 4$ (that is multiples of P of weight d), succeeds if it is possible to find enough of these parity check equations for the iterative process to converge. Decoding is thus possible if:

$$\ell \geq 2^{\alpha_d(\gamma)+\frac{d_P}{d-1}} \quad \text{with} \quad \alpha_d(\gamma) = \tfrac{1}{d-1}\log_2\left[(d-1)!\tfrac{1}{C(\gamma^{d-2})}\right]. \tag{4}$$

From this, we can see that whatever the parameters of the system, there exists a d which makes iterative decoding possible. This means that for the smallest d satisfying this equation, there should be about just enough parity check equations: finding them requires to find nearly all multiples of P of weight d and degree less than ℓ. Among the techniques described in Section 2.1, the Strategy 1 based on the birthday paradox is the most efficient. It has a cost:

$$\mathcal{C}_{\mathsf{parity}} = \binom{\ell-1}{\lceil\frac{d-1}{2}\rceil} \quad \Leftrightarrow \quad \log_2\left(\mathcal{C}_{\mathsf{parity}}\right) \simeq \tfrac{d-1}{2}\log_2\ell - \log_2\left(\tfrac{d-1}{2}!\right). \tag{5}$$

The cost for the decoding is then negligible compared to this cost. This means that if one cannot pay more than $2^{c_{\mathsf{hard}}}$ to decode, one can only decode if $2^{c_{\mathsf{hard}}} \geq \mathcal{C}_{\mathsf{parity}}$. Mixing Equations (4) and (5) we see that one can decode only if:

$$c_{\mathsf{hard}} \geq \tfrac{1}{2}\left[\log_2(d-1)! - \log_2 C(\gamma^{d-2}) + d_P\right] - \log_2\left(\tfrac{d-1}{2}!\right).$$

Roughly, we get the same constraint: if $d_P \geq 2c_{\mathsf{hard}}$, decoding is not possible.

Property 4 (Noisy LFSR Decoding). *The noisy LFSR decoding problem can efficiently be solved when $d_P \leq c_{\mathsf{easy}}$ and $\frac{1}{4} \geq \gamma \geq \sqrt{\frac{d_P}{\ell}2\log 2}$. For $\gamma \leq 1.18\,\ell^{-\frac{1}{2}}$, Y contains less than 1 bit of information of X, that is, the mutual information between X and Y is smaller than 1.*

Assumption 5 (Noisy LFSR Decoding). *The noisy LFSR decoding is hard on average when P is generated by Gen_1 (as specified in Problem 1) and when $d_P \geq 2c_{\mathsf{hard}}$ and $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_P} - 1$. When $d_{\min} \geq 2c_{\mathsf{hard}}$, $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_{\min}} - 1$, P is generated by Gen_2, and w and d_K are such that the Low-Weight Polynomial Multiple is hard, the noisy LFSR decoding is hard as well.*

2.3 The Hidden Correlation Problem

We combine the previous problems into Hidden Correlation (HC) problem.

Problem 6 (Hidden Correlation).

Parameters: a length ℓ, two relatively prime[2] polynomials P and Q of degree d_P and d_Q respectively, a bias γ

[2] When P and Q have a common factor, the decoding problem is ambiguous so we exclude those cases. As we will see later, we will always choose distinct primitive polynomials P and Q so they never have a common factor.

HC channel: given a uniformly distributed random seed X of length d_Q, generate Y, the XOR of the output of length ℓ of $\mathcal{L}_Q$ initialized with X, the output of $\mathcal{L}_P$ initialized at random, and a random noise generated by $\mathcal{S}_\gamma$ (both random sources being independent)
Problem: given Y, recover X

As for the noisy LFSR decoding problem, we say that the problem is hard if no algorithm that outputs the correct X with probability higher than $2^{1-c_{\mathsf{hard}}}$ and average complexity less than $2^{c_{\mathsf{hard}}}$ exists.

This problem is meant to be used given an oracle that solves the noisy LFSR decoding problem with parameters (ℓ', Q, γ') where $\ell' = \ell - d_K$ for some degree d_K and $\gamma' = \gamma^w$ for some weight w. There are three main strategies to solve it:

1. consider $\mathcal{L}_P \oplus \mathcal{L}_Q$ as a single LFSR $\mathcal{L}_{P\times Q}$, recover its initializations and deduce the initializations of both $\mathcal{L}_P$ and $\mathcal{L}_Q$ from it, thus recovering X. Achieving this requires to be able to solve the noisy LFSR decoding problem with parameters $(\ell, P \times Q, \gamma)$.
2. suppress the output of $\mathcal{L}_P$ in Y and decode X with a shorter output and a higher noise level. One should first find a polynomial K, multiple of P of degree d_K and weight w (note that choosing $K = P$ is also possible). Then multiply Y by K to suppress the influence of $\mathcal{L}_P$ and try to solve the noisy LFSR decoding problem with parameters $(\ell - d_K, Q, \gamma^w)$.
   As discussed in Section 2.1, the problem of finding K can be hard. The key idea of our construction is that this K can be a trapdoor.
3. suppress the output of $\mathcal{L}_Q$ in Y and recover the initialization of $\mathcal{L}_P$ (with a shorter output and a higher noise level). Once this is done, recovering X consists in decoding in $\mathcal{L}_Q \oplus \mathcal{S}_\gamma$ only, with the full output ℓ and the same bias γ: the oracle can do this. We can use the same method as above and find a multiple K of Q. In the end, we need to solve the noisy LFSR problem with parameters $(\ell - d_K, P, \gamma^w)$.

By taking P random of degree $d_P \geq 2c_{\mathsf{hard}}$ and $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_P} - 1$ we know from Assumption 5 that the decoding problem for $\mathcal{L}_P \oplus \mathcal{S}_\gamma$ is intractable. This renders strategies 1 and 3 impossible.

To build a public-key cryptosystem on this problem, we need to find parameters which render strategy 2 computationally infeasible, and at the same time let us have a trapdoor polynomial K making it possible to solve it. We do this by proving an upper bound on the information one can get on X using strategy 2 in the favorable case where we can compute all multiples of low weight (except the hidden one), and bounded by the maximum number of iterations ($2^{c_{\mathsf{hard}}}$) we can manage. The number of possibles multiples of weight w and degree $d_K \leq \ell$ (except the hidden K) is given by equation (1).

For each multiple K' of P of weight i it is possible to suppress the influence of $\mathcal{L}_P$ and recover a little bit of information on X. This information I_i is bounded by $I_i \leq \ell C(\gamma^i)$.

By ignoring the cost of finding a multiple of weight i we upper bound the cost of recovering the I_i bits of information by $\mathcal{O}(\ell i)$, the cost of computing

$Y' = Y \bullet K'$. Then an adversary can use all possible multiples of a given weight, as long as there are less than $\frac{1}{\ell i} 2^{c_{\mathsf{hard}}}$ such multiples. For large values of i, when there are more multiples of P, he is limited to exactly $\frac{1}{\ell i} 2^{c_{\mathsf{hard}}}$ multiples. The total information one can get is at most:

$$\mathcal{I} = \sum_{i=2}^{\infty} \ell C(\gamma^i) \min\left(\frac{\binom{\ell}{i}}{2^{d_P}}, \frac{2^{c_{\mathsf{hard}}}}{\ell i}\right).$$

We can consider that this attack is not a threat if $\mathcal{I} \leq 1$ (which means that after using about $2^{c_{\mathsf{hard}}}$ polynomials, an adversary gets less than 1 bit of information on X). This bound is not tight since we neglected the cost of finding the multiples.

Assumption 7 (Hidden Correlation Problem). *When $d_P \geq 2c_{\mathsf{hard}}$, $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_P} - 1$, and $\mathcal{I} \leq 1$, the Hidden Correlation problem is hard on average when P is generated by Gen_1 as specified in Problem 1. When $d_{\min} \geq 2c_{\mathsf{hard}}$, $\gamma \leq 2^{1-c_{\mathsf{hard}}/d_{\min}} - 1$, $\mathcal{I} \leq 1$ (with d_P replaced by $d_{\min}$), P is generated by Gen_2, and w and d_K are such that the Low-Weight Polynomial Multiple is hard, the hidden correlation problem is hard as well.*

3 TCHo Encryption

3.1 Specifications

This construction depends on a number of domain parameters. These are:

- a bias $\gamma \in [0, 1]$ (we only consider positive biases here),
- two integers $d_{\min}$ and $d_{\max}$ bounding the degree of P: $d_P \in [d_{\min}, d_{\max}]$.
- two integers d_Q and d_K: the degrees of polynomials Q and K,
- a primitive polynomial Q of degree d_Q
- an integer w corresponding to the weight of the polynomial K,
- an integer ℓ which is the length of the produced stream.

As we will see later in Table 1, suitable parameters might look like:

$$\gamma = 0.98, d_P \in [6\,000, 6\,600], d_Q = 20, d_K = 11\,560, w = 99, \ell = 13\,080.$$

Key generation. The public key is a primitive polynomial P. The private key is a low-weight multiple K of P.

To generate such a key pair one will first pick a random binary polynomial K coprime with Q of given weight w and degree d_K. Then one factors this polynomial and checks if it has a primitive factor of suitable degree $d_P \in [d_{\min}, d_{\max}]$. If such a factor exists one uses it as the public key P and a key pair was found, otherwise one picks another random polynomial K, factors it, etc. The complexity of this step is detailed in Appendix A. With the above parameters, this takes a little more than 4 minutes using `NTL` [27] on a 1.5GHz Pentium 4.

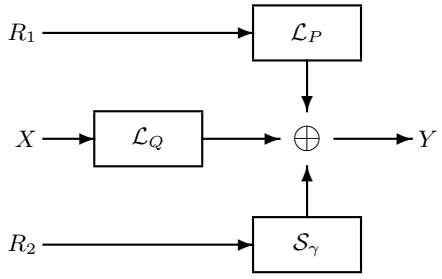

Fig. 1. Encryption.

Encryption. The core of the system is a stream-cipher-like construction, built as the XOR of two LFSRs $\mathcal{L}_P$ and $\mathcal{L}_Q$ (with respective retroaction polynomials P and Q) and a biased random source $\mathcal{S}_\gamma$ where $0 \leq \gamma \leq 1$ represents the bias of this source.

The word X of d_Q bits to be encrypted is used to initialize $\mathcal{L}_Q$. $\mathcal{L}_P$ is randomly initialized with a string R_1. R_2 denotes the random coins for $\mathcal{S}_\gamma$. We assume that R_1 and R_2 are independent. The ciphertext is the binary stream Y of length ℓ equal to the XOR of the streams generated by $\mathcal{L}_P$, $\mathcal{L}_Q$ and $\mathcal{S}_\gamma$ (see Fig. 1). We denote $Y = \mathsf{TCHo}_P(X; R)$ for $R = (R_1, R_2)$. The cost of an encryption is $\mathcal{O}\left((d_P + d_Q + \varepsilon_\gamma) \times \ell\right)$, where ε_γ represents the cost of generating one random bit with bias γ. A dedicated hardware of $\mathcal{O}\left(d_P + d_Q + \varepsilon_\gamma\right)$ gates (typically, a few thousand gates) runs in time $\mathcal{O}\left(\ell\right)$ (typically, a few thousand clock cycles).

Decryption. To decrypt, one first uses the private key K to suppress the influence of $\mathcal{L}_P$. For $i = 1, \ldots, \ell - d_K$, compute $Y'_i = (Y_i, Y_{i+1}, \ldots, Y_{i+d_K}) \bullet K$. This results in a new cipher Y' of length $\ell - d_K$ equal to the XOR of a stream generated by $\mathcal{L}_Q$ and a source of bias γ^w. The complexity is $\mathcal{O}\left(d_K \cdot \ell\right)$.

From Property 4 we know that if $\ell - d_K \geq \frac{2 d_Q \log 2}{\gamma^{2w}}$ (and $\gamma^w \leq \frac{1}{4}$) we can recover the initialization of $\mathcal{L}_Q$ (thus X) using an ML decoding algorithm. Using a Walsh transform, this decryption costs $\mathcal{O}\left(d_Q\, 2^{d_Q}\right)$ operations. Finally, X is recovered by solving a linear equation in time $\mathcal{O}\left(d_K^3\right)$. Overall decryption complexity is thus $\mathcal{O}\left(d_K \ell + d_Q 2^{d_Q} + d_K^3\right)$. Decoding is feasible for $d_Q \leq c_{\mathsf{easy}}$. However, that decryption is non-deterministic: there is a probability that S_γ generates too much noise and that decryption returns an incorrect result. Bounds on this failure rate are hard to estimate as the linear code corresponding to a truncated LFSR is not as easy to study as for a full length LFSR. However, during all our tests, no decodings ever failed or returned the wrong plaintext. The bound used in Section 3.2 for parameter selection, taken from information theory, seems to be sufficient to have a negligible decryption failure rate.

Encrypting larger blocks. To encrypt n blocks $X_1, \ldots, X_n$ we define TCHo^n, the concatenation of n independent instances of TCHo (thus able to encrypt $n \cdot c_{\mathsf{easy}}$

Table 1. Some suitable parameters for TCHo.

c_{hard}	d_Q	d_P	γ	w	d_K	ℓ	$\mathcal{I}$	key gen. cost[3]
80	20	$500-600$	0.4	6	6 200 000	8 000 000	$2^{-1.7}$	$\sim 2^{56.5}$
		$1\,000-1\,100$	0.74	13	78 350	148 000	$2^{-1.9}$	$\sim 2^{42.5}$
		$3\,000-3\,500$	0.93	35	17 100	21 600	2^{-17}	$\sim 2^{36.5}$
		$6\,000-6\,600$	0.98	99	11 560	13 080	$2^{-1.1}$	$\sim 2^{36}$
		$6\,000-6\,060$	0.98	99	10 620	12 140	2^{-4}	$\sim 2^{39}$
	30	$1\,000-1\,500$	0.7	11	232 000	339 000	2^{-7}	$\sim 2^{44.5}$
		$6\,000-6\,600$	0.979	93	12 150	14 150	$2^{-2.1}$	$\sim 2^{36}$
90	20	$7\,000-7\,700$	0.98	105	13 950	15 900	$2^{-2.1}$	$\sim 2^{36.5}$
	30	$7\,000-7\,700$	0.98	99	14 460	16 750	$2^{-0.1}$	$\sim 2^{36.5}$
128	20	$10\,000-11\,000$	0.977	108	25 050	29 300	$2^{-5.8}$	$\sim 2^{38.5}$
	30	$10\,000-11\,000$	0.977	102	26 300	31 100	$2^{-2.9}$	$\sim 2^{38.5}$

bits of data), by using independent random coins $R_1, \ldots, R_n$:

$$\mathsf{TCHo}_P^n(X_1, \ldots, X_n; R_1, \ldots, R_n) = \big[Y_1, \ldots, Y_n\big], \quad \text{where} \quad Y_i = \mathsf{TCHo}_P(X_i; R_i).$$

3.2 Parameters Selection

We review here all the constraints on the system parameters in order for a legitimate decryption to be possible and for an attack by a computationally bounded adversary (bounded by $\mathcal{O}\left(2^{c_{\mathsf{hard}}}\right)$ operations) to be impossible. As constraints need to be satisfied for any $d_P \in [d_{\min}, d_{\max}]$, in the following equations, d_P was replaced either by $d_{\min}$ or by $d_{\max}$ depending on the type of inequality.

- legitimate decryption is possible if: $\ell - d_K \geq \frac{2d_Q \log 2}{\gamma^{2w}}$, $\gamma^w \leq \frac{1}{4}$, and $d_Q \leq c_{\mathsf{easy}}$ (see Section 2.3).
- recovering K is impossible if the conditions of Assumption 2 are verified, that is: $w \log_2 \frac{d_K}{d_{\max}} > c_{\mathsf{hard}}$ and $\binom{d_K}{w-1} \leq 2^{d_{\min}}$.
- without K, decryption is equivalent to solving an instance of the HC problem. From Assumption 7 we know this is impossible if:
  $d_{\min} \geq 2c_{\mathsf{hard}}$, $\gamma \leq 2^{1-\frac{c_{\mathsf{hard}}}{d_{\min}}} - 1$, and $\mathcal{I} = \displaystyle\sum_{i=2}^{\infty} \ell C(\gamma^i) \min\left(\frac{\binom{\ell}{i}}{2^{d_{\min}}}, \frac{2^{c_{\mathsf{hard}}}}{\ell i}\right) \leq 1.$

We are looking for parameters which satisfy all the inequalities stated above. One should first choose $d_Q \leq c_{\mathsf{easy}}$ and $d_{\min} \geq 2c_{\mathsf{hard}}$, and deduce the largest possible $\gamma = 2^{1-c_{\mathsf{hard}}/d_{\min}} - 1$. Then find a w for which $\ell = (\ell - d_K) + d_K \geq \frac{2d_Q \log 2}{\gamma^{2w}} + d_{\max} 2^{\frac{c_{\mathsf{hard}}}{w}}$ is small enough and deduce the minimum d_K and ℓ. Once parameters are obtained one can just check whether $\mathcal{I} \leq 1$ or not and maybe reduce γ accordingly. Table 1 gives a few sets of parameters satisfying these inequalities.

[3] This is the cost of generating a key when using an algorithm based on linear algebra. See Appendix A for details.

As one can notice, no small value of d_P appear in the tables. This is mainly because parameters which would otherwise be suitable for all the constraints always lead to a value of $\mathcal{I}$ too large. This however is not a problem as the optimal values (in terms of encryption/decryption complexity and transmission rate) seem to appear for larger values of d_P.

3.3 Security

Key recovery. The key generation algorithm we use corresponds exactly to the Gen_2 generator of Assumption 2. If the conditions of this assumption are respected, the key recovery problem is exactly the problem of solving a hard instance of the low-weight polynomial multiple problem. For suitable parameters, key recovery should thus be computationally impossible.

Message recovery requires to solve the hidden correlation problem. Solving this problem is hard in general, thus with suitable parameters the encryption can be seen as a one-way function. In order to use the Fujisaki-Okamoto construction we need to prove that this encryption is (t, ε)-secure in the sense of One-Way Encryption (OWE) and Γ-uniform (see Appendix B for definitions).

Theorem 8. *Under Assumptions 2, 5, and 7,* TCHo *is* $(2^{c_{\mathsf{hard}}}, \frac{2}{2^{d_Q}})$*-OWE-CPA-secure.*

Proof. We have chosen parameters such that, for any value of P and K, an adversary spending less than $\mathcal{O}\left(2^{c_{\mathsf{hard}}}\right)$ time always gets less than 1 bit of information on the plaintext X using a chosen plaintext attack (CPA). Therefore, he always has a probability lower than $\frac{2}{2^{d_Q}}$ of guessing the correct X. □

Theorem 9. *The* TCHo *encryption scheme is* $\left(\frac{1}{2} + \frac{\gamma}{2}\right)^{\ell}$*-uniform.*

Proof. We need to upper bound the probability (on the random coins R_1 and R_2 introduced in Section 3.1) that a given plaintext is mapped to a given ciphertext. As we only consider positive biases, for a given initialization of $\mathcal{L}_P$ the most likely ciphertexts correspond to the random coins R_2 giving $\mathcal{S}_\gamma = 0$. This happens with probability $\left(\frac{1}{2} + \frac{\gamma}{2}\right)^{\ell}$. When taking the average on the possible initializations of $\mathcal{L}_P$ (on all the possible random coins R_1) this probability can only decrease. □

Semantic Security. Deciding whether Y encrypts X under TCHo_P or is random is equivalent to deciding whether a binary string is an output form a noisy LFSR channel with parameters ℓ, P, γ or is random. This may be a hard problem as well. This question may deserve further work.

Malleability. Obviously, from $Y = \mathsf{TCHo}_P\big(X; (R_1, R_2)\big)$, an adversary can forge a new ciphertext $Y' = \mathsf{TCHo}_P\big(X \oplus \delta; (R_1 \oplus \rho, R_2)\big)$ without even knowing X, R_1 or R_2. Hence, raw TCHo is clearly not OWE-CCA-secure, just like the raw RSA cryptosystem.

Extension to TCHo^n. Let $\mathsf{TCHo}_P^n(X_1, \ldots, X_n; R_1, \ldots, R_n) = [Y_1, \ldots, Y_n]$. An adversary A knows $Y_1, \ldots, Y_n$ and P and wants to recover $X_1, \ldots, X_n$. We know that spending less than $2^{c_{\mathsf{hard}}}$ time, Y_i cannot give A more than 1 bit of information on X_i. The other Y_j cannot help as A can generate as many couples (X_j, Y_j) as he wants by himself (he knows P and nothing else is required for encryption) and this will not give him any information on X_i. As the X_i are independent, if A tries to guess the values of $X_1, \ldots X_n$ simultaneously he will not have a better probability of success than if he tries to guess them one after the other: his probability of success will be less or equal to $\left(\frac{2}{2^{d_Q}}\right)^n$. The concatenation TCHo^n is thus $(2^{c_{\mathsf{hard}}}, \frac{2^n}{2^{d_Q n}})$-OWE-CPA-secure.

Concerning Γ-uniformity, the results also extend well for TCHo^n. For the concatenation we have:

$$\Gamma(X_1, \ldots, X_n, Y_1, \ldots, Y_n) = \Pr\left[h \leftarrow_R \mathtt{COINS} : \forall i, Y_i = \mathcal{E}_{pk}(X_i; h)\right].$$

As before, the best probability will be obtained if $\mathcal{S}_\gamma = 0$ for each of the n independent encryptions. This happens with probability $\left(\frac{1}{2} + \frac{\gamma}{2}\right)^{\ell n}$ and thus TCHo^n is $\left(\frac{1}{2} + \frac{\gamma}{2}\right)^{\ell n}$-uniform. We deduce that we can build, in the random oracle model, an IND-CCA secure hybrid cryptosystem **TCHo^n-FO** based on TCHo^n and an FG-secure symmetric encryption, by using the Fujisaki-Okamoto construction [12].

4 Conclusion

We presented the first public-key cryptosystem which resembles a stream cipher. Stream ciphers are usually very efficient when it comes to low-cost hardware implementation, so implementation may be quite competitive on these platforms. Software implementations are pretty fast as well. One issue with our construction might be the need for a huge amount of random coins, but hardware entropy accumulators may supply them efficiently. The main drawback is currently the overhead size and the decryption complexity. Although quite reasonable, future work will decrease it. One option would consist in replacing $\mathcal{L}_Q$ by a better binary code. As suggested by Willi Meier, repetition codes seem to offer a very good decryption complexity and can be available for higher dimensions.

TCHo is currently a prototype. Our concept might still have to be improved before becoming a real product, but TCHo is definitely a first step in a new direction for public-key cryptography. We encourage analysis of TCHo since it will either demonstrate its security or lead to improvements on stream cipher attacks.

References

1. Gildas Avoine and Philippe Oechslin. A scalable and provably secure hash-based RFID protocol. In *PerSec 2005*, 2005.

2. Paul Camion, Miodrag J. Mihaljević, and Hideki Imai. Two alerts for design of certain stream ciphers: Trapped LFSR and weak resilient function over GF(q). In K. Nyberg and H. Heys, editors, *SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 196–213. Springer, 2003.
3. 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, January 1998.
4. Anne Canteaut and Michaël Trabbia. Improved fast correlation attacks using parity check equations of weight 4 and 5. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer, 2000.
5. Vladimor V. Chepyshov, Thomas Johansson, and Ben Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In Bruce Schneier, editor, *FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2001.
6. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407, Bruges, Belgium, May 2000. Springer.
7. Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359, Warsaw, Poland, May 2003. Springer.
8. Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287, Queenstown, New Zealand, December 2002. Springer.
9. Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Results on algebraic immunity for cryptographically significant boolean functions. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 92–106, Chennai, India, December 2004. Springer.
10. Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139:61–88, June 1999.
11. Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (F5). In *ISSAC 2002*, pages 75–83, Lille, France, July 2002. ACM.
12. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *CRYPO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
13. Thomas Johansson and Fredrik Jönsson. Fast correlation attacks based on turbo code techniques. In Michael J. Wiener, editor, *CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197. Springer, August 1999.
14. Thomas Johansson and Fredrik Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In Jacques Stern, editor, *EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362, Prague, Czech Republic, May 1999. Springer.
15. Thomas Johansson and Fredrik Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315. Springer, August 2000.
16. 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, Davos, Switzerland, May 1988. Springer.
17. Yi Lu. *Applied Stream Ciphers in Mobile Communications*. Phd thesis num. 3491, EPFL, 2006. `http://library.epfl.ch/theses/?nr=3491`.
18. Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack: A practical attack on bluetooth encryption. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2005.
19. Yi Lu and Serge Vaudenay. Faster correlation attack on bluetooth keystream generator E0. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 407–425. Springer, August 2004.
20. Willi Meier and Othmar Staffelbach. Fast correltaion attacks on stream ciphers. In C. G. Günther, editor, *EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314, Davos, Switzerland, May 1988. Springer.
21. Miodrag J. Mihaljevic, Marc P. C. Fossorier, and Hideki Imai. A low-complexity and high-performance algorithm for the fast correlation attack. In Bruce Schneier, editor, *FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 196–212, New York, USA, April 2001. Springer.
22. Håvard Molland, John Erik Mathiassen, and Tor Helleseth. Improved fast correlation attack using low rate codes. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference*, volume 2898 of *Lecture Notes in Computer Science*, pages 67–81, Cirencester, UK, December 2003. Springer.
23. David Molnar and David Wagner. Privacy and security in library RFID: issues, practices, and architectures. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *CCS 2004*, pages 210–219. ACM, 2004.
24. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
25. Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In G. Brassard, editor, *CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 606–609. Springer, 1989.
26. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
27. Victor Shoup. NTL: A library for doing number theory. Available online from `http://www.shoup.net/ntl/`.
28. Thomas Siegenthaler. Cryptanalysts representation of nonlinearly filtered ML-sequences. In Franz Pichler, editor, *EUROCRYPT 85*, volume 219 of *Lecture Notes in Computer Science*, pages 103–110, Linz, Austria, April 1986. Springer.
29. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition edition, 2003.
30. David Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 2002.

A Key Generation

Average number of trials. The cost of a key generation depends directly on the number of trials required before finding a suitable key pair. Here we try to evaluate the probability that a random low-weight binary polynomial has a primitive factor of degree d_P.

It is known that, asymptotically, the number of irreducible binary polynomials of degree d is equivalent to $\frac{2^d}{d}$. If we call P one of these irreducible polynomials, a random binary polynomial K of degree $d_K > d$ is divisible by P (and has P in his factorization) if $K \equiv 0 \mod P$. The probability this happens is 2^{-d}. Taking into account this probability and the number of irreducible polynomials, we obtain the probability that K has an irreducible factor of degree d is approximately $\frac{1}{d}$. Given that P is irreducible, the probability that it is primitive is the probability that the logarithm of x in $(\mathbf{F}_2[x]/P(x))^*$ is coprime with 2^d-1. The probability that two random numbers are coprime relates to Buffon's needle problem and is $\frac{6}{\pi^2}$. Given that 2^d-1 is odd for sure, the probability that P is primitive given that it is irreducible can be heuristically approximated to $\frac{8}{\pi^2} \approx 81\%$.

This is true asymptotically for random polynomials. In our case we only consider polynomials K of low weight. For this particular case we have no proof that the same result still holds[4]. However, experimental statistics tend to prove that this approximation is still very accurate for low-weight polynomials. The average number of trials is thus approximately $\frac{\pi^2}{8} d_P$.

Factoring algorithm. Concerning the algorithm to be used for the factorization, the choice is very wide. The straightforward method would be to use a generic factoring algorithm like Berlekamp's or Cantor-Zassenhaus (see Chapter 14 of [29] for more details on polynomial factorization). However, when dealing with high degree polynomials, methods based on linear algebra tend to use a lot of memory and degrees above 2^{16} are too high for standard computers. Moreover, in our case, a complete factorization is not required: it is enough to check whether our polynomial has an irreducible factor of degree d_P, and this can be done quite efficiently.

The polynomial $X^{2^d} + X$ is the product of all binary irreducible polynomials whose degrees divide d. Checking if K has a factor of degree d_P reduces to the computation of $\gcd(X^{2^{d_P}} + X, K)$. If this gcd has a degree less than d_P one can be sure that K has no suitable factors, otherwise one simply needs to factor this "low degree" polynomial using whatever algorithm.

Computing this gcd is much faster than a complete factorization for high degree K and a small d_P. The computation of $X^{2^{d_P}} + X \mod K$ using successive squarings can however be quite long when d_P increases, thus, the best solution depends on the system parameters. Using Berlekamp's factoring algorithm the complexity of the factoring is $\mathcal{O}\left({d_K}^{2.4}\right)$ (where 2.4 is the cost for linear algebra) and using the gcd-based algorithm it is $\mathcal{O}\left(d_P\,{d_K}^2\right)$. Taking into account the number of iterations for a key generation we obtain $\mathcal{O}\left(d_P\,{d_K}^{2.4}\right)$ and $\mathcal{O}\left({d_P}^2\,{d_K}^2\right)$. Cantor-Zassenhaus gives $\mathcal{O}\left(d_P d_K^2 \log d_K \log\log d_K\right)$, which is the best complexity, but with a very large constant term.

[4] In our construction we only considers polynomials K with a non null constant term, which are therefore not divisible by X. If one chooses them with an odd weight they will never be divisible by $X+1$ and will never have an irreducible factor of degree 1. Some similar properties might also hold for other degrees!

Using degree ranges. The number of attempts required for the key generation to find a polynomial with degree $d_P \in [d_{\min}, d_{\max}]$ is divided by $\delta = d_{\max} - d_{\min} + 1$. If we use a general factoring algorithm, this directly divides the key generation complexity by δ but if we use the gcd method, the time used to compute $X^{2^{d_P}} + X \mod K$ does not change and we have to compute δ gcds. In both cases this improves the key generation complexity, but for large values of δ, the gcd method becomes very slow. Using Berlekamp's algorithm the complexity becomes $\mathcal{O}\left(\frac{1}{\delta}\, d_P\, {d_K}^{2.4}\right)$, with Cantor-Zassenhaus it is $\mathcal{O}\left(\frac{1}{\delta}\, d_P\, {d_K}^2 \log d_K \log\log d_K\right)$, and for the gcd-based technique it becomes $\mathcal{O}\left(\frac{d_P}{\delta}(d_P+\delta){d_K}^2\right)$.

For small values of δ the best choice is the gcd-based technique, but for larger δ generic factoring algorithms are faster. If d_K is large, Cantor-Zassenhaus is the best algorithm.

Primitivity testing. In general, checking whether an arbitrary polynomial is primitive is hard. However, we only need here to probabilistically filter out random polynomials which are not primitive. This is essentially easy. A probabilistic method for finding a generator of $\mathbf{Z}_p^*$ can be adapted to checking whether a polynomial is primitive. One simply need to find all divisors p_i of $2^{d_P} - 1$ smaller than a given λ, and check if the order of X in $\left(GF(2)[X]/P(X)\right)^*$ divides $\frac{2^{d_P}-1}{p_i}$. If this is the case for none of the p_i, P is primitive with probability $1 - \lambda^{-1}$.

So, with complexity $\mathcal{O}\left(\mathcal{P}oly\left(d_P\right)\sqrt{\lambda}\right)$ it is possible to reduce the risk that his algorithm accepts a non-primitive polynomial to $\mathcal{O}\left(\lambda^{-1}\right)$. It is thus possible to reach error rates as low as 2^{-40} with a cost negligible compared to the factorization step.

B Definitions

We report here definitions taken from the article by Fujisaki and Okamoto [12].

Definition 10 (Asymmetric Encryption). *An asymmetric encryption scheme Π is a triple of algorithms $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$, associated with two finite sets,* $\texttt{COINS}(k)$ *(a set of random coins) and* $\texttt{MSPC}(k)$ *(a message space), for $k \in \mathbb{N}$.*

- *$\mathcal{K}$, the key generation algorithm, is a probabilistic algorithm which on input 1^k ($k \in \mathbb{N}$) outputs a pair of keys, $(pk, sk) \leftarrow \mathcal{K}(1^k)$.*
- *$\mathcal{E}$, the encryption algorithm, is a probabilistic algorithm that takes a key pk, an message $x \in \texttt{MSPC}$ and a string $r \leftarrow_R \texttt{COINS}(k)$, where $\leftarrow_R$ represents a random affectation, and produces a ciphertext $y = \mathcal{E}_{pk}(x; r)$.*
- *$\mathcal{D}$, the decryption algorithm, is a deterministic algorithm that takes a key sk and a ciphertext y and returns a message $x \leftarrow \mathcal{D}_{sk}(y)$.*

It is required that, for any $k \in \mathbb{N}$, if $(pk, sk) \leftarrow \mathcal{K}(1^k)$, $x \in \texttt{MSPC}$, and $y \leftarrow \mathcal{E}_{pk}(x)$, then $\mathcal{D}_{sk}(y) = x$.

Definition 11 (OWE). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \texttt{COINS}, \texttt{MSPC})$ be an asymmetric encryption scheme. Let A be an adversary knowing y and pk and trying to recover x. We say that Π is (t, ε)-OWE-secure if, for any A running in at most time t and for any $x \in \texttt{MSPC}$:*

$$\Pr\left[(pk, sk) \leftarrow \mathcal{K}(1^k); y \leftarrow \mathcal{E}_{pk}(x) : A(pk, y) = \mathcal{D}_{sk}(y)\right] \leq \varepsilon.$$

In other words, the probability among all the possible key pairs for A of recovering x from its encrypted value is bounded by ε.

Definition 12 (Γ-uniformity). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \texttt{COINS}, \texttt{MSPC})$ be an asymmetric encryption scheme. For given $(pk, sk) \leftarrow \mathcal{K}(1^k)$, $x \in \texttt{MSPC}$ and $y \in \{0,1\}^*$, define*

$$\Gamma(x, y) = \Pr\left[h \leftarrow_R \texttt{COINS} : y = \mathcal{E}_{pk}(x; h)\right].$$

We say that Π is Γ-uniform, if, for any $(pk, sk) \leftarrow \mathcal{K}(1^k)$, any $x \in \texttt{MSPC}$ and any $y \in \{0,1\}^*$, $\Gamma(x, y) \leq \Gamma$.

Definition 13 (Find-Guess-security). *Let $\Pi = (\mathcal{E}, \mathcal{D}, \texttt{KSPC}, \texttt{MSPC})$ be a symmetric encryption scheme where $\texttt{KSPC}$ is the key space and $\texttt{MSPC}$ is the message space. Let A be an adversary. We say Π is (t, ε)-FG-secure if, for any A running in at most time t:*

$$\Big|2 \cdot \Pr\Big[a \leftarrow_R \texttt{KSPC}; (x_0, x_1) \leftarrow A; b \leftarrow_R \{0,1\}; \\ y = \mathcal{E}_a(x_b) : A(x_0, x_1, y) = b\Big] - 1\Big| \leq \varepsilon.$$

This means that an adversary choosing x_0 and x_1 cannot decide which of them corresponds to a given ciphertext $y = \mathcal{E}_a(x_b)$ with probability more than $\frac{1}{2} + \frac{\varepsilon}{2}$.

Definition 14 (IND-CCA-security). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \texttt{COINS}, \texttt{MSPC})$ be an asymmetric encryption scheme and let A be an adversary having access to a decryption oracle $\mathcal{D}_{sk}$. We say that Π is (t, q_D, ε)-IND-CCA-secure if for any A running in at most time t and asking at most q_D queries to the decryption oracle:*

$$\Big|2 \cdot \Pr\Big[(pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1) \leftarrow A^{\mathcal{D}_{sk}}; b \leftarrow_R \{0,1\}; \\ y = \mathcal{E}_{pk}(x_b) : A^{\mathcal{D}_{sk}}(x_0, x_1, y) = b\Big] - 1\Big| \leq \varepsilon.$$

This means that an adversary choosing x_0 and x_1 cannot decide which of them corresponds to a given ciphertext $y = \mathcal{E}_{pk}(x_b)$ with probability more than $\frac{1}{2} + \frac{\varepsilon}{2}$. In the random oracle model, $\mathcal{E}_{sk}$ might also use some random oracles for encryption. We then get a similar definition of IND-CCA security by bounding the number of queries to these random oracles.