Lattice Klepto

Turning Post-Quantum Crypto Against Itself

Robin Kwant, Tanja Lange, and Kimberley Thissen

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, NL
r.j.h.kwant@student.tue.nl, tanja@hyperelliptic.org,
k.k.a.thissen@student.tue.nl

Abstract. This paper studies ways to backdoor lattice-based systems following Young and Yung's work on backdooring RSA and discrete-log based systems. For the NTRU encryption scheme we show how to build a backdoor and to change the system so that each ciphertext leaks information about the plaintext to the owner of the backdoor. For signature schemes the backdoor leaks information about the signing key to the backdoor owner.
As in Young and Yung's work the backdoor uses the freedom that random selections offer in the protocol to hide a secret message encrypted to the backdoor owner. The most interesting and very different part though is how to hide and retrieve the hidden messages.

Keywords: Post-quantum cryptography, kleptography, lattice-based encryption, NTRU, signatures.

1 Introduction

The attacks studied in cryptanalysis can typically be classified into mathematical, algorithmic attacks and side-channel attacks. The former tries to tackle the hard problem the system is based on or to find ways to circumvent the hard problem altogether; the latter uses information gathered during execution of algorithms (possibly after introducing faults or cache flushes) to learn secret information. Typically the analysis assumes that the attacker has full knowledge of the algorithm implemented and typically also of the implementation itself.

In the mid 90's, Young and Yung invented [15,16,17] the concept of *Cryptovirology* or *Kleptography* and studied how easily systems lend themselves to being backdoored. Their setups typically include a black-box implementation whose output should be indistinguishable from the output of a legitimate implementation for anybody but the owner of the backdoor key. The klepto implementation of a regular algorithm leaks (parts of) the secret message, a private key, or the

This work was supported by the European Communities through the Horizon 2020 program under project number 645622 (PQCRYPTO) and project number 645421 (ECRYPT-CSA). Permanent ID of this document: e14bc1779799664cf160742e72d7fa50. Date: 2017.08.11.

state of a random-number generator to the attacker. In a secure klepto scheme this is done in such a way that the attacker holds a secret key which gives him the unique power to decrypt that leaked information. If anybody inspects or reverse engineers the black-box implementation they may observe a difference in how the values are generated but must not be able to decrypt their own past leaks or those of others. Ideally they should not be able to even decrypt future leaks.

The properties of a secure klepto scheme are

- exclusivity,
- indistinguishability, and
- forward secrecy.

This implies that the backdoor encryption must use a public-key system and that only the public part of the backdoor key is stored on the device.

The study of kleptopgraphy has gained topicality in the wake of the Snowden revelations which mention "subversion of standards" as one of the targets of NSA and news articles [13] strongly indicating that the elliptic-curve based random-number generator DualEC [10] was designed with a backdoor. This backdoor is closely related to the "repeated DH Setup" by Young and Yung. Subsequent research has shown that this backdoor can be exploited in the wild [2] in TLS implementations and turned up more evidence about the origin [1] of DualEC and how it got incorporated into standards.

While the overall lesson is clear: do not accept black-box implementations of cryptographic algorithms and request justification for all choices made, the power of klepto schemes differs noticeably between RSA, finite field DH, and elliptic-curve cryptography (ECC). The most powerful backdoor against RSA produces keys that are indistinguishable from random keys but include an ECC-based encryption to a backdoor key of the same cryptographic security as the RSA key that allows instant factorization [18].

This raises the question how other public-key schemes can be turned into kleptographic schemes. Post-quantum cryptography has received a lot of interest in recent years and NIST calls for submissions of post-quantum algorithms by the end of 2017. So far schemes have been evaluated purely for security, functionality, speed, and at best for implementation security (side-channel countermeasures). We are not aware of any study of kleptographic attacks against these schemes.

This paper studies lattice-based encryption, in particular the NTRU [7] family of encryption schemes and signature schemes and shows how to turn them into klepto schemes with an ECC-based backdoor.

2 Background

This section briefly describes the NTRU encryption system and fixes parameters for our klepto scheme. For the NTRU encryption scheme we follow the original NTRU paper [7].

2.1. Background on Kleptography. Young and Yung call their the core of their klepto schemes a SETUP. SETUP is an abbreviation of "Secretly Embedded Trapdoor with Universal Protection".

Definition 2.1 (SETUP). *Let S be a publicly known cryptosystem. A SETUP mechanism is an algorithmic modification made to S to get S' such that:*

- *The input of S' agrees with the public specifications of the input of S.*
- *S' computes using the attacker's public encryption function E (and possibly other functions as well), contained within S'.*
- *The attacker's private decryption function D is not contained within S' and is known only by the attacker.*
- *The output of S' agrees with the public specifications of the output of S. At the same time, it contains published bits which are easily derivable by the attacker but are otherwise hidden.*
- *Furthermore, the outputs of S and S' are polynomially indistinguishable to everyone except the attacker.*

The definition of a weak SETUP mechanism is a relaxation of a regular SETUP mechanism. A weak SETUP is the same as a regular SETUP with the exception that it does not require the polynomial indistinguishability between the output of S and S' [16]. This may seem very easily detectable, but in practice this still works well because an end user does not know that the implementation contains a SETUP. Furthermore, an end user often does not know what the output of S should be.

2.2. Subliminal Channel. A subliminal channel is a secondary channel of communication hidden inside a communications channel that is presumed to be compromised. The concept of a subliminal channel was introduced as a solution to the *prisoners problem* by Simmons in 1984 [14]. In the prisoners problem two people Alice and Bob are incarcerated and wish to plan a breakout. Their only way of communicating is by passing over messages via Eve who is one of the guards. They are allowed to use encryption, but Eve will only pass along the messages if she is allowed to read the messages, so she needs access to the keys and the decryption function. As Eve will report any breakout plan, Alice and Bob have to hide their communications about breaking out within their communication.

This subliminal channel seems to solve a very specific problem, yet in times of surveillance this problem is and will be more frequently seen in practice. More and more countries propose laws which oblige citizens to give up their private keys if requested. If they want to continue having secure communications, this creates a situation directly analogous to the prisoners problem.

2.3. Concrete choices. For concreteness we consider ECC to exfiltrate secrets. The benefits of using ECC are small ciphertext size, needing just 256 bits at 128-bit security level in addition to the symmetric-key encryption of the message. Let $E/\mathbb{F}_p$ be an elliptic curve over the prime field $\mathbb{F}_p$, e.g. let $E/\mathbb{F}_p$ be P256 from [11] with base point P, and let $P_B = BP$ be the public key for the backdoor.

For symmetric encryption and authentication we use AES-GCM, this means that to exfiltrate $M \in \{0,1\}^\ell$ we need $256 + 128\ell + 128$ bits by sending $C = (AP, \text{AES-GCM}_K(M))$, where K is the key for AES-GCM derived from the DH key AP_B. Upon receipt of C the backdoor owner uses its secret backdoor key B to compute the same K from $B(AP)$.

Obviously the security level of the backdoor key is significantly reduced once a quantum computer exists and the schemes will no longer satisfy the property of exclusivity if the backdoor key is found by somebody having a quantum computer. However, there are no agreed upon post-quantum encryption schemes, yet, and, in showing how to exfiltrate these > 512 random bits, we provide a mechanism of exfiltrating any data, possibly split over multiple NTRU encryption messages.

Furthermore, NTRU has been proposed independently of post-quantum cryptography as a very efficient encryption system and was included into standards, such as IEEE P1363.1 and ASC X9 X9.98, on its own merits.

2.4. NTRU parameters. NTRU is an asymmetric cryptosystem commonly used in a hybrid cryptosystem to share keys for a symmetric encryption algorithm. NTRU is specified by six public parameters, the integers (N, p, q, d_f, d_g, d_r), in which $\gcd(p, q) = 1$ and q is much larger than p. In practice p is usually chosen as 3 and q a power of 2. NTRU works with operations on elements of the ring $R = \mathbb{Z}[X]/(X^N - 1)$. In the following we assume q is even and p is odd. An element can be represented as either a polynomial of degree at most $N - 1$ or a vector of length N containing the coefficients of that polynomial. The operation denoted as $\circledast$ is the cyclic convolution product, that is multiplication in R. Using the property $X^N \equiv 1 \bmod (X^N - 1)$ it is defined as $F \circledast G = H$ with

$$H_k = \sum_{i=0}^{k} F_i \cdot G_{k-i} + \sum_{i=k+1}^{N-1} F_i \cdot G_{N+k-i} = \sum_{i+j \equiv k \mod N} F_i \cdot G_j.$$

The parameters (d_f, d_g, d_r) specify the sets $(\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m)$, which are sets of polynomials of degree at most $N - 1$ with a fixed number of (small) nonzero coefficients. Concrete parameter choices are included in Table 5.1.

Definition 2.2 (Message space). *The message space $\mathcal{L}_m$ is defined as*

$$\mathcal{L}_m = \{m \in R | m \text{ has coefficients in } [-(p-1)/2, (p-1)/2]\}.$$

Messages are assumed to be integers in a radix p representation, with every digit a coefficient of the polynomial. The rest of this section follows definitions from [7].

Definition 2.3 (The set $\mathcal{L}(d_1, d_2)$). *The set of ternary polynomials $\mathcal{L}(d_1, d_2)$ is defined as:*

$$\mathcal{L}(d_1, d_2) = \left\{ F \in R \,\middle|\, F \text{ has exactly } \begin{array}{r} d_1 \text{ coefficients equal to } 1, \\ d_2 \text{ coefficients equal to } -1 \\ \text{the rest of the coefficients equal } 0 \end{array} \right\}$$

The key and randomness spaces ($\mathcal{L}_f, \mathcal{L}_g$, and$\mathcal{L}_r$) are defined as:

$$\begin{aligned}\mathcal{L}_f &= \mathcal{L}(d_f, d_f - 1)\\ \mathcal{L}_g &= \mathcal{L}(d_g, d_g)\\ \mathcal{L}_r &= \mathcal{L}(d_r, d_r)\end{aligned}$$

$\mathcal{L}_f$ is not set as $\mathcal{L}(d_f, d_f)$ because a polynomial $f \in \mathcal{L}(d_f, d_f)$ would have $f(1) = 0$ which is not invertible, while f needs to be invertible for key creation explained now.

2.5. NTRU key generation. To create a key, two random polynomials $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$ are chosen such that inverses F_q and F_p of f exist in R modulo q and p respectively.

The public key

$$h \equiv F_q \circledast g \bmod q, \tag{2}$$

is computed.

The private key is the pair (f, F_p), in which F_p is derivable from f and p but is precomputed for practical purposes. The reduction modulo q of the polynomial means a reduction of the coefficients to equivalent representatives in the interval $(-q/2, q/2]$.

2.6. NTRU encryption. A message $m \in \mathcal{L}_m$ is chosen and a random $r \in \mathcal{L}_r$ is selected. Now ciphertext

$$c \equiv p \cdot r \circledast h + m \bmod q, \tag{3}$$

is computed.

2.7. NTRU decryption. To obtain message m, first the quantity $a \equiv f \circledast c \bmod q$ is computed. Because

$$a \equiv f \circledast (p \cdot r \circledast h + m) \equiv f \circledast (p \cdot r \circledast F_q \circledast g + m) \equiv p \cdot r \circledast g + f \circledast m \bmod q, \tag{4}$$

reducing modulo p yields $f \circledast m$ if the polynomials are sparse enough.

In that case,

$$m \equiv a \circledast F_p \bmod p.$$

We now consider possible exceptions to this equivalence.

2.8. NTRU decryption failures. When Equation (4) is not an exact equation in R due to the modular reductions in the decryption step, then m might be not or only partially recovered. In Equation 4 first the term $a \equiv p \cdot r \circledast g + f \circledast m$ is reduced modulo q after which it is reduced modulo p. Since $\gcd(p, q) = 1$, this resulting a reduced modulo q and p is not well defined, as reducing a different representative of a modulo p could give a different result. In practice this problem is avoided by choosing the uniquely defined representative of a with coefficients in the interval $(-q/2, q/2]$. The resulting a in equation (4) equals $p \cdot r \circledast g + f \circledast m$ in R if the maximum absolute value of any coefficient is not too big. This property is captured by the *width*:

Definition 2.4 (Width). *Let* $l = \sum_{i=0}^{N-1} l_i X^i \in R$. *The width of* l *is defined as*

$$|l|_\infty = \max_{0 \le i \le N-1} l_i - \min_{0 \le i \le N-1} l_i.$$

If the width of the term $S = p \cdot r \circledast g + f \circledast m$ exceeds q, some coefficients in the recovered polynomial will differ from the coefficients of m. If m is used as the key for symmetric authenticated encryption the user will quickly notice that the authenticator does not verify. This is called a decryption failure and has to be taken into account in parameter selection.

3 The NTRU Backdoor

In this section an example of a modified NTRU encryption with a backdoor using a weak SETUP is described and analyzed, after which countermeasures are given. The backdoor has the purpose of leaking secrets of the encrypting party to a third party. This information is made available exclusively to the third party by encrypting it to the party's ECC key.

3.1. Description. This version differs from regular NTRU in the sense that a secondary encrypted message along with the regular message is included in the ciphertext. This secondary message is available to a third party. As in Section 2.3, the public key encryption of this secondary message will be denoted as C, encrypting plaintext M. The key setup on the receiving end stays exactly the same. We take C to be the ECC encrypted and authenticated message to be exfiltrated; in the typical hybrid setting of NTRU, M is the symmetric session key of the legitimate user, so M typically has 256 bits (for encryption and MAC part) and C has 640 bits.

3.2. Encryption. Let $\rho < q$ be an integer coprime to p. Consider C as a polynomial in R with coefficients modulo ρ,i.e., $C \in \mathbb{Z}_\rho[X]/(X^N - 1)$, $\rho < q$ and $\gcd(\rho, p) = 1$. To obtain this representation, first take the bitstring C and interpret it as a large integer, then take its coefficients in base ρ as polynomial coefficients.

On the sending end, a slight adaptation of Equation (3) is used. First ciphertext c is computed as in Equation (3). Now the new ciphertext c' including the secondary message, is computed as

$$c' = c + k \cdot p, \tag{5}$$

with k a polynomial in R with coefficients in $\mathbb{Z}_\rho$ such that $c' \equiv C \bmod \rho$. This polynomial k can be obtained by solving the integer equation $C_i \equiv c_i + k_i \cdot p \bmod \rho$ for every coefficient of k. Having the $\gcd(\rho, p) = 1$ by definition, ensures the existence of these solutions.

3.3. Decryption by the attacker. The attacker reduces $c' \bmod \rho$ and recovers the polynomial C, since $C \equiv c' \bmod \rho$. The attacker interprets C as a bitstring and decrypts it with his private key (as in Section 2.3) to obtains the leaked information.

3.4. Decryption by the intended receiver. Decryption at the receiver end stays exactly the same. First the quantity $a' = f \circledast c' \bmod q$ is computed as in Equation (4). Because

$$\begin{aligned} a' &\equiv f \circledast (p \cdot k + p \cdot r \circledast h + m) \\ &\equiv f \circledast (p \cdot k + p \cdot r \circledast F_q \circledast g + m) \\ &\equiv p \cdot k \circledast f + p \cdot r \circledast g + f \circledast m \bmod q, \end{aligned} \tag{6}$$

reducing a' modulo p still yields $f \circledast m$ if the coefficients are not too large (see the comment on decryption failures above). Thus $m \equiv a' \circledast F_p \bmod p$.

4 Analysis of the Backdoor Quality

In this section we analyze how much more likely a decryption failure gets depending on the size of the backdoor parameter ρ. A large ρ value is convenient for the attacker to send more data but obviously makes failures more likely. **4.1.**

Decryption failures. As pointed out in Section 2.8, a decryption failure occurs when the polynomial

$$S = p \cdot r \circledast g + f \circledast m,$$

has a width larger than q. Adding the $k \cdot p$ term to the ciphertext c' in Equation (5) makes decryption failures more likely because now a decryption failure occurs when the polynomial

$$T = p \cdot k \circledast f + p \cdot r \circledast g + f \circledast m,$$

has a width larger than q, as generally $|T|_\infty > |S|_\infty$. Because for a single coefficient of T it applies that

$$T_l = S_l + \sum_{i+j \equiv l \bmod N} p \cdot k_i \cdot f_j,$$

the contribution of this extra convolution product $p \cdot k \circledast f$ to a single coefficient is at most

$$p \cdot (\lceil (\rho - 1)/2 \rceil) \cdot (2 \cdot d_f - 1). \tag{7}$$

Let $\alpha = \min(d_g, d_r)$, the maximum width of S is given by

$$\max |S|_\infty = 2 \cdot p \cdot \alpha + (2 \cdot d_f - 1) \cdot (p - 1)/2, \tag{8}$$

so the maximum width of T would be

$$\max |T|_\infty = 2 \cdot p \cdot \alpha + (2 \cdot d_f - 1) \cdot ((p - 1)/2 + p \cdot \lceil (\rho - 1)/2 \rceil). \tag{9}$$

4.2. Parameter choices. Because of the possible decryption failures it is important to pick parameters that minimize this phenomenon while maintaining global security. It is recommended to keep ρ as small as possible. In the case

where $p = 3$ the value $\rho = 2$ is quite suitable. Other options would be $\rho = 4$ and $\rho = 5$ as this would give space to leak more information, but as noted above, decryption failures will be much more likely because the extra contribution of the term in Equation (7) can become much larger. For most parameter sets $\rho = 2$ will most likely be the only option that works without increasing the probability of decryption failures too much.

For the typical 128-bit security level the klepto ciphertext C has only 640 bits, which is less than N for typical parameter choices, meaning that $\rho = 2$ is sufficient to exfiltrate ciphertexts as described in Section 2.3.

4.3. Optimization. Decryption failures will be less likely if the vector $k \cdot p$ added in Equation (5) is sparse. This is the case when k is sparse. A way to keep k sparse is to minimize the number of bits needed to store C and pad it with zeros. Depending on what information will be leaked it might even be possible to split up C over several messages. In that case C will only be partially leaked, but can be recovered if multiple messages containing all the parts are recorded.

Another optimization that works in the case where $\rho = 2$ is to append a one bit shorter message C' with an indicator bit i such that instead of C, $[i|C']$ is leaked. The polynomial k is now computed regularly. If this k contains more ones then it contains zeros the term $\bar{k} \cdot p$ is added instead of $k \cdot p$, with $\bar{k}$ the bitwise complement of k. The attacker now recovers either $[0|C']$ or $[1|\bar{C}']$ and is able to recover C' by taking the complement if $i = 1$. Note that this indicator costs one bit in space, so C' has at most $N - 1$ bits where C would have N.

Another trick for $\rho = 1$ is to randomly pick between ± 1 for nonzero k_i in order to halve the average width of $p \cdot k \circledast f$.

5 Practical implementation

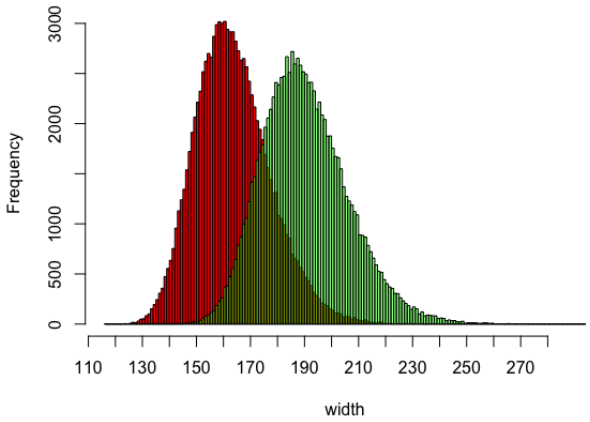
We wrote an implementation of NTRU in Sage [3] and added the backdoor as described in Section 3, using parameter $\rho = 2$. We ran experiments to look at the impact of the backdoor with respect to decryption failures. In every experiment a pseudorandom ternary message m is generated along with a pseudorandom N bit binary message C. This way C is as long as N, which for most parameter sets is longer than necessary, but we were interested in seeing the overall impact and using a shorter C will make the system more likely to function correctly. None of the optimizations discussed in Section 4.3 were applied in the implementation. The first set of experiments counts the number of decryption failures caused by the backdoor in 2 different ways using NTRU parameters from [4]. First a subset of experiments was conducted in which a new key is generated with every trial, in this case all trials are independent. Secondly a subset of experiments was conducted in which the same key is used more than once, these trials are not independent but they do represent a real world situation in which keys are generated once and then reused often. Doing multiple trials with the same keys also allows for more experiments as generating a new key is relatively expensive computationally.

Table 5.1. Decryption failure check results

Parameters						# keys	# trials per key	# failures
N	p	q	d_f	d_g	d_r			
613	3	2048	55	204	55	20000	1	0
						100	10000	0
887	3	2048	81	295	81	10000	1	0
						100	10000	0
1171	3	2048	106	390	106	5000	1	0
						100	10000	0

Since no decryption failures occurred in these experiments, the increased probability of a decryption failure caused by the backdoor will probably go unnoticed in practice. With Equation (7) the maximum contribution to the width of T with the parameters used can be computed, with respect to q this difference is relatively small enough. Looking at Equations (8) and (9) the maximum width T is only slightly larger than the maximum width of S with the parameters used in the experiments. This could possibly explain the lack of decryption failures. Note that the maximum width was not expected to be obtained. These extreme widths are in general very rare, as f and r are chosen to be sparse. They are intentionally centered around 0 to let a lot of cancellations occur. This behavior is not unique to the parameters used in the experiments, in most parameter sets used in practice $d_f = d_r$. When $\rho = 2$ is chosen, the most significant term in Equation (9) is generally the first one, so the contribution of the backdoor to the maximum width is generally small enough. In the implementation any message can be leaked as long as its encryption does not exceed N bits. The trialled version with parameter $N = 613$ is 27 bits too short for the hidden ciphertext C described in Section 2.3 but also has slightly lower security. The versions with $N = 887$ and $N = 1171$ have ample space, even for longer messages C. For instance, in $N = 1171$ there could be an 256-bit ECC key and a 128-bit authentication tag, which leaves $(1171-256-128 = 787)$ bits for a message. The 787 bits fit 6 blocks of 128 bit ciphertext and the remaining 19 bits could be used for the optimizations discussed in Section 4.3. To get an idea of how much the probability of decryption failures increases on average instead of just the worst case, a second set of experiments was run. In the second set of experiments, the width of the terms S and T were stored. A decryption failure occurs when $|S|_\infty > q$ or $|T|_\infty > q$, without and with the backdoor respectively. For these experiments the parameters from Table 5.1 were used. The results are presented in histograms where red corresponds to $|S|_\infty$ and green corresponds to $|T|_\infty$. Note that all observed widths are significantly smaller even than $q/2$.

These results confirm that on average the probability of a decryption failure increases, but this increase is small enough to go unnoticed in a practical situation because large widths are rare. An interesting side effect is that the standard deviation also increases when the backdoor is added. The green spike is generally lower and less steep, which means that the $|T|_\infty$ values are less predictable



	$\|S\|_\infty$	$\|T\|_\infty$
μ	164.0471	190.7234
σ	13.81889	15.95371
min	116	139
max	251	294

Fig. 5.1. $(N, p, q, d_f, d_g, d_r) = (613, 3, 2048, 55, 204, 55)$, 10 keys, 10000 trials per key.

than the $|S|_\infty$ values. This phenomena gives rise to some questions explained in Section 10.2.

6 Countermeasures

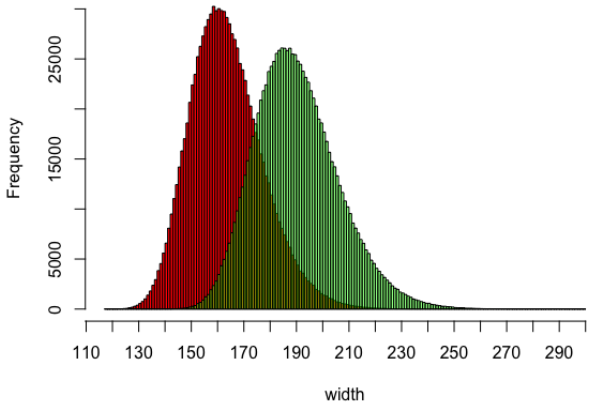
There are ways to find out that the ciphertext was tampered with. One of those being the recovery of the randomness. From Equation (3) we obtain

$$c - m = r \circledast h \bmod q,$$

meaning r can be recovered by

$$r = (c - m) \circledast h^{-1} \bmod q$$

if inverse h^{-1} exists in R modulo q. In the case of a ciphertext with an extra term added, doing the same computation will result in $r + k \cdot p \circledast h^{-1}$ instead of r, which with high probability will not be an element of $\mathcal{L}_r$. Since by specification $r \in \mathcal{L}_r$, the receiver can check whether $(c - m) \circledast h^{-1} \in \mathcal{L}_r$. If this is not the case and h is invertible in R modulo q, the ciphertext has been tampered with and a warning can be sent back to the sender. To make sure that this is possible, it is important that h is always invertible in R modulo q. Remember that h depends on the choice of f and g so a change has to be made to the selection of those. Public key h is defined as $h = F_q \circledast g \bmod q$. By definition F_q is invertible so the only extra requirement is that g must also be invertible, this can be done



	$\|S\|_\infty$	$\|T\|_\infty$
μ	163.9682	190.6541
σ	13.79992	15.95437
min	117	140
max	257	300

Fig. 5.2. $(N, p, q, d_f, d_g, d_r) = (613, 3, 2048, 55, 204, 55)$, 100 keys, 10000 trials per key

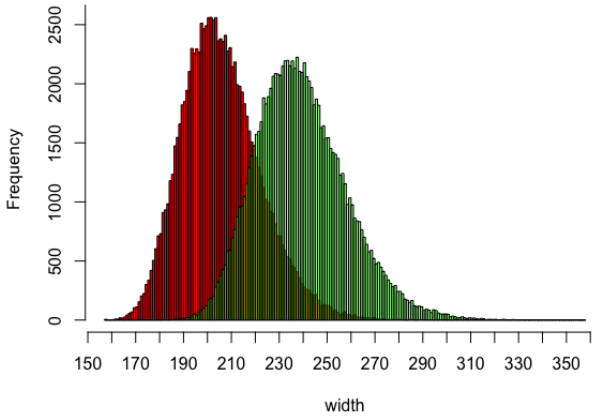
by choosing $g \in \mathcal{L}_g$ in a similar manner as f. Since invertibility is required, $\mathcal{L}_g$ can no longer be defined as $\mathcal{L}_g = \mathcal{L}(d_g, d_g)$ and would need to be defined as $\mathcal{L}_g = \mathcal{L}(d_g, d_g - 1)$.

7 Subliminal Channel in NTRU

In this section a modification of NTRU with a SETUP is shown in which an extra possibly secret channel for information is added. This channel differs from the backdoor discussed in Section 3 as it is intended for the receiver of the message instead of a third party.

7.1. Description. In this adaptation, Bob sends a regular message m and a subliminal encrypted message C to receiver Alice. We use the same ECC-AES-GCM-based setup as described in Section 2.3 to construct C. To include this C, a technique inspired by the countermeasures described in Section 6 is used. In addition to the regular setup, both Alice and Bob agree upon an injective map ϕ which maps C to an element of $\mathcal{L}_r$ and a pair of ECC keys to generate and decrypt C.

7.2. Key setup. Alice chooses $f \in \mathcal{L}_f$ and computes F_q and F_p as in Section 2.5. Now $g \in \mathcal{L}(d_g, d_g - 1)$ is chosen such that inverse g^{-1} exists in R_q and public key h is computed normally as in Equation (2). Choosing $g \in \mathcal{L}(d_g, d_g - 1)$ is justifiable as a protection against the specific backdoor mentioned earlier. Alice publishes her public key h so that others including Bob, can send her messages.



	$\lvert S\rvert_\infty$	$\lvert T\rvert_\infty$
μ	206.3269	239.786
σ	16.43092	18.94655
min	157	182
max	298	358

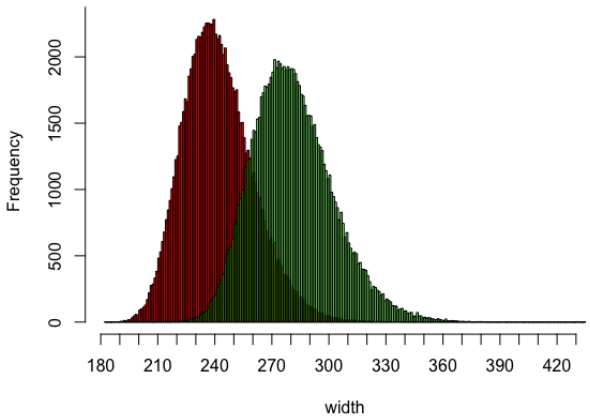
Fig. 5.3. $(N, p, q, d_f, d_g, d_r) = (887, 3, 2048, 81, 295, 81)$, 10 keys, 10000 trials per key.

7.3. Encryption. Bob takes the secret message M, generates C, uses the function ϕ to map C to an element $r \in \mathcal{L}_r$ and encrypts m by computing c using Equation (3) with this choice of r. Bob now sends c to Alice.

7.4. Decryption. Alice receives c and recovers m using Equation (4). She now computes h^{-1} and uses this to recover $r \equiv (c-m) \circledast h^{-1} \bmod q$. She now recovers C as the preimage of r using ϕ^{-1}. For efficiency it is possible to precompute h^{-1}.

7.5. Encoding messages. In this section an example for the injective map ϕ mentioned earlier is described. It is somewhat similar to Algorithm 2.2 in [12]. Let C,the encryption of a message M, be represented as a unique number chosen in the discrete interval $\left[0, \binom{N}{d_r} \cdot \binom{N-d_r}{d_r} - 1\right]$. Then ϕ is an injective map $\left[0, \binom{N}{d_r} \cdot \binom{N-d_r}{d_r} - 1\right] \to \mathcal{L}_r$ that encodes an encrypted message C to an $r \in \mathcal{L}_r$. The inverse ϕ^{-1} gives preimage C from the image r.

The set $\mathcal{L}_r$ can be represented as a tree, with every level representing one coefficient. We now describe how this tree is constructed, see Figure 7.1 for a visualization. The root is defined as representing r_0, the level of the leaves r_n. Every leaf corresponds to a unique element of $\mathcal{L}_r$, and is defined by the unique path from the root to the leaf. Every node has at most 3 branches depending on whether it can still be completed, because left and right branches are limited: The leftmost branch corresponds to choosing a -1, the middle branch a 0 and the right branch a 1 on that level. Now the set $\mathcal{L}_r$ can be indexed by counting the leaves from left to right, where the leftmost leaf has index 0.



	$\lvert S\rvert_\infty$	$\lvert T\rvert_\infty$
μ	242.3299	281.6776
σ	18.47012	21.31923
min	182	218
max	365	435

Fig. 5.4. $(N, p, q, d_f, d_g, d_r) = (1171, 3, 2048, 106, 390, 106)$, 10 keys, 10000 trials per key

The tree itself does not have to be stored in memory, at every node the number of leaves can be computed by $\binom{n}{k} \cdot \binom{n-k}{l}$, with n being the number of levels from the node to a leaf, k the number of -1s and l the number of 1s that are not used yet at that node. The left, middle, and right subbranches of a node have $\binom{n-1}{k-1} \cdot \binom{n-k}{l}$,$\binom{n-1}{k} \cdot \binom{n-k-1}{l}$ and $\binom{n-1}{k} \cdot \binom{n-k-1}{l-1}$ leaves respectively.

To convert an index C into an $r \in \mathcal{L}_r$ the tree is traversed starting from the root, and a running index j is kept, so at the root $i = 0$ and $j = C$. At every level i the number of leaves in the left subbranch L_i is computed. If $j \leq L_i$, the left branch is taken and $r_i = -1$. If this is not the case, the number of leaves in the middle subbranch is computed and added to L_i to obtain L'_i which is the number of leaves in the left and middle subbranch combined. Now if $L_i < j \leq L'_i$ the middle branch is taken, $r_i = 0$ and we set $j = j - L_i$. If $j > L'_i$ the right branch is taken, $r_i = 1$ and we set $j = j - L'_i$. This process repeats until a leaf is reached.

The inverse ϕ^{-1} works in a similar matter, the tree is traversed starting from the root according to the path specified in r. A running index j is kept for which $j = 0$ at the root. Now at every level i the number of leaves that are "skipped" by not choosing the left or middle branch respectively, are added to j. So if $r_i = 0$, the middle branch is taken, L_i is computed and we set $j = j + L_i$. Else if $r_i = 1$, the right branch is taken, L'_i is computed and we set $j = j + L'_i$. This process repeats until all the bits of r are evaluated. Now finally $C = j$.

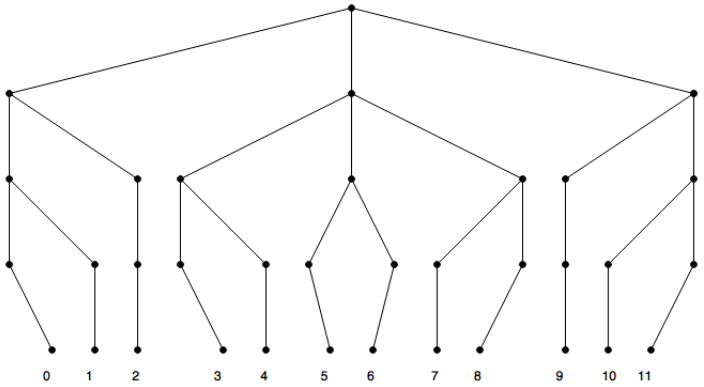


Fig. 7.1. Example of a tree for $d_r = 1$ and $N = 4$

7.6. Why does it work?. As pointed out in Section 6 the randomness r can be recovered by the receiver if g is chosen to be invertible. This phenomenon is exploited by putting a message in r rather than choosing r randomly.

This subliminal channel changes the choice of r to being deterministic in C but does not change the range for r, hence it does not introduce any extra decryption errors and is completely undetectable from the observable distribution, even to Eve who obtained the NTRU key. For properly chosen elliptic curves, C is indistinguishable from random bitstrings and thus r is indistinguishable from a randomly chosen element from $\mathcal{L}_r$.

8 pqNTRUSign

This section briefly describes pqNTRUSign, also known as NTRU-MLS, which is short for NTRU Modular Lattice Signature. For this we follow the original paper [6] from PQCrypto 2014. Though other NTRU signature schemes, such as NSS [8] and NTRUSign [5], have been broken, this scheme has no known attacks against the currently proposed parameters.

8.1. pqNTRUSign parameters. The signature scheme works in NTRU lattices, so the set up is very similar to NTRU (Section 2.4). pqNTRUSign is specified by five parameters, the integers (N, p, q, B_s, B_t), where $\gcd(p, q) = 1$, q is much larger than p and B_s and B_t are some bounds on the norms of some elements; typically $p = 3$ and q has 15 or more bits. Similar to NTRU, all computations take place in the ring $R = \mathbb{Z}[X]/(X^N - 1)$ and polynomials are often reduced modulo q or p. Unlike NTRU, only the size of the polynomial coefficients is limited but there is no limit on the number of non-zero coefficients. We write R_p to denote elements of R with coefficients in $\mathbb{Z}_p$; we consider elements automatically lifted to $\mathbb{Z}$ using integers in $(-p/2, p/2]$; all integer modular reductions are made explicit.

8.2. pqNTRUSign key generation. To generate a key pick $F \in R_3, g \in R_p$ such that both are invertible modulo p and q. Let $f = pF$. The private key is the pair (f, g).

The public key is $h \equiv f^{-1} \circledast g \bmod q$.

Similar to NTRU, polynomials in $L_h = \{(s,t) \in R^2 | t \equiv h \circledast s \bmod q\}$ will be considered, this is the NTRU lattice which is emphasized in the naming of the signature scheme.

8.3. pqNTRUSign signature. To sign message $m \in \{0,1\}^*$ compute $(s_p, t_p) = H(h\|m)$, where $H : R_q \times \{0,1\}^* \to R_p \times R_p$ is a hash function.

The next step picks a random polynomial r from a certain distribution. For NTRU-MLS this is from R_ℓ for some integer $\ell \approx q/p$ and for pqNTRUSign (as presented at the PQCrypto 2017 rump session [9]) this is from a bimodal Gaussian distribution. For our klepto scheme the details do not matter; we note that both distributions are sufficiently wide.

Let $s_0 = s_p + pr$ and $t_0 \equiv s_0 \circledast h \bmod q$. Now compute $a \equiv (t_p - t_0) \circledast g^{-1} \bmod p$. Then the candidate signature is $(s,t) = (s_0, t_0) + (a \circledast f, a \circledast g)$. Note that this last computation takes place in R, i.e., there is no reduction on the coefficients, while $a \in R_p$ and $t_0 \in R_q$. The latter ensures that all coefficients are small. Note further that by construction $s \equiv s_p \bmod p$ and $t \equiv t_p \bmod p$ because $f = pF$.

NTRU-MLS outputs (s,t) if no coefficient in $a \circledast f$ is larger than B_s, no coefficient in $a \circledast g$ is larger than B_t and the coefficients of s and t are bounded by $\|s\| \leq \frac{q}{2} - B_s$ and $\|t\| \leq \frac{q}{2} - B_t$. Else the procedure restarts with a different choice of r.

The details for the bounds in the latest version of pqNTRUSign are less clear but a similar rejection sampling on (s,t) is performed.

The signature is on m is (s,t); to save space the pqNTRUSign authors also suggest a version in which the signature is s and $t \equiv s \circledast h \bmod q$ is recomputed.

8.4. pqNTRUSign Verification. In order to verify the signature, either first recompute $t \equiv s \circledast h \bmod q$ or check that t in the signature verifies this equivalence. Also check the bounds on the coefficients of s and t. If any of the checks fails, reject the signature.

Then compute $(s_p, t_p) = H(h\|m)$ and accept the signature if $s \equiv s_p \bmod p$ and $t \equiv t_p \bmod p$, else reject it.

9 The pqNTRUSign backdoor

In this section we show how to backdoor pqNTRUSign using a weak SETUP. Signatures are easier to backdoor than NTRU because the signer can check for verification failures himself and restart with a new random choice. Since the regular algorithm uses rejection sampling on the outputs these restarts will not raise suspicion if they do not get significantly more frequent. The backdoor is based on the same idea as that in NTRU: taking the signature modulo 2 reveals a secondary ciphertext C encrypted to the public key of the klepto scheme (for

details see Section 2.3). As in the NTRU backdoor we choose reduction modulo 2 because the typical choice of p is 3 which is coprime to 2 and larger moduli increase the chance of resampling.

The most obvious target to leak in a signature scheme is the signing key. In pqNTRUSign this would be $F \in R_3$, needing $\lfloor N \log_2 3 \rfloor + 1$ bits in optimal packing. Alternatively, an evil implementer could point to the importance of short secret keys and generate F deterministically from a short random seed that can be leaked in a shorter message.

Unlike in NTRU we will not be able to transmit N bits at once but only a small number (in order to keep resampling rates acceptable). This means that C needs to be transmitted over multiple signatures and then concatenated at the receiver end. The GCM part of the encryption then also serves as a check for correctness. In the following, C will be a ciphertext to be leaked, encoded as a binary polynomial of degree less than $d \leq N$.

In line with the paper topic we chose to exploit the flexibility in random choices for a klepto scheme but would like to point out that it could as well be used as a subliminal channel to hide encrypted messages. Because the signer can validate the signature himself there no distinction between the capacity of the klepto/covert channel and the subliminal channel.

There are no modification to the key generation or verification algorithm and the owner of the klepto backdoor obtains and deciphers the ciphertext as for NTRU (apart from sorting and arranging partial ciphertexts).

9.1. Trivial backdoor. We want to achieve that $s \equiv C \bmod 2$, up to the degree of C, i.e., that this equivalence holds for the coefficients of $1, X, X^2, \ldots, X^{d-1}$ for some d.

In the trivial backdoor we check whether s satisfies this equation or else reject the signature in the rejection step. This means that the change to the signature algorithm is minimal but increases signature generation time by a factor of 2^d on average.

9.2. Modified signature. To avoid too many rejections we will now modify the signature generation. As a warm up put $d = 1$, i.e. we will leak 1 bit.

Changing s to $s' = s + p$, i.e., adding p to the constant will change the parity of the constant but not affect $s \equiv s_p \bmod p$. This change implies choosing $s'_0 = s_0 + p$ instead of s_0 and $r' = r + 1$ instead of r which only minimally affects the distribution of the randomness. There is a minimal chance that s will violate B_s if s was valid.

However, $t \equiv h \circledast s \bmod q$ may no longer hold. If $t'_0 \equiv s'_0 \circledast h \equiv s_0 \circledast h + ph \bmod q$ equals t_0 modulo p, i.e., t_0 had small enough coefficients that adding ph did not cause a reduction in it, then $a' = a$ and verification will work for $t' = t'_0 + a \circledast g$ and s' (provided that they also satisfy B_s and B_t). Note that h is a full-size polynomial, i.e. its coefficients can range over the full interval $(-q/2, q/2]$, and the equivalence has to hold in all N coefficients. If either of these checks fails, a possible fix is to use $s' = s - p$ instead, otherwise a new r needs to be sampled.

Now let $c(X) = \sum_{i=0}^{d-1} c_i X^i \in R_2$ for some larger d and let $k(X) = \sum_{i=0}^{d-1} k_i X^i$ with $k_i \in \{0, \pm 1\}$ such that $s' = s + pk \equiv c \bmod 2$ on the bottom d coefficients. As for NTRU this is possible because $\gcd(2, p) = 1$. Then $r' = r + k$ and $s'_0 = s_0 + pk$, which still likely pass the size test for s since p is much smaller than q.

However, for increasing d, $t'_0 \equiv t_0 + ph \circledast k \bmod q$ will increasingly likely invoke a reduction modulo q when adding $ph \circledast k$.

Again we can vary the sign on the k_i to reduce the size of $h \circledast k \bmod q$. For small d this can be done exhautively to find the minimum and for larger d randomizing signs to reach roughly as many $+1$ as -1 seems beneficial.

A final optimization is to skip validity tests on (s, t) before including the backdoor and choosing signs in k such that (s', t') is smaller.

We plan on providing experimental results in the very near future to determine acceptable rejection rates and good sizes for d.

10 Final Remarks

As shown in Sections 3, 7, and 9 it is feasible and practical to modify NTRU and pqNTRUSign in such a way that they contains a backdoor or subliminal channel. Countermeasures against the NTRU backdoor have been described in Section 6.

10.1. Minimization of decryption failures. In Section 4 some optimizations have been given in order to reduce the increased probability of decryption failures with the backdoor added. In Section 5 some experimental results are given. By doing more experiments and with more parameter sets, the increased probability of decryption failures might be estimated and parameters can be selected which allow for more information to be leaked without increasing the failure probability too much. Research can also be done to find the theoretical probability instead of an estimation. With this estimation parameters can be computed that preserve global security, but at the same time minimize the probability of decryption failures.

10.2. Statistical countermeasures. In Section 5 experimental results were given on the width of the polynomial T with respect to the width of S. These results showed that the width of T is less predictable but still small. The standard deviation was larger for the values of T. This occurs because adding an extra message to the ciphertext means adding some randomness. This yields the question, whether a receiver of messages could distinguish the ones that were tampered with from the ones that were not and alert the sender? How many messages would it need to be able to do so? These are questions that might be worthwhile looking into.

10.3. Potential biases in pqNTRUSign klepto signatures. The result of the modified signatures of the pqNTRUSign scheme in Section 9 could potentially be biased as the random generation is influenced. If the user would collect a set of signatures generated by this black box algorithm, it will likely show that

the signatures are not as random as the user would expect. This behavior could be analyzed.

10.4. Further research. For backdoors in NTRUSign [5] and NSS [8] see the thesis by Kimberley Thissen http://repository.tue.nl/854465. For full details and further considerations on NTRU see the thesis by Robin Kwant http://repository.tue.nl/854433.

References

1. Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. Dual EC: A standardized back door. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, volume 9100 of *Lecture Notes in Computer Science*, pages 256–281. Springer, 2016.
2. Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 319–335. USENIX Association, 2014.
3. The Sage Developers. *SageMath, the Sage Mathematics Software System*, 2017. http://www.sagemath.org.
4. Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 437–455, 2009.
5. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: digital signatures using the NTRU lattice. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track the RSA Conference 2003, San Francisco, CA, USA, April 13-17, Proceedings*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2003.
6. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2014.
7. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
8. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: an NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2001.

9. Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. pqNTRUSign: update and recent results, 2017. http://2017.pqcrypto.org/conference/slides/recent-results/zhang.pdf.
10. National Institute of Standards and Technology. Special Publication 800-90: Recommendation for random number generation using deterministic random bit generators, 2012. First version June 2006, second version March 2007, http://csrc.nist.gov/publications/PubsSPs.html#800-90A.
11. National Security Agency. Suite B cryptography / cryptographic interoperability, 2005. https://web.archive.org/web/20150724150910/https://www.nsa.gov/ia/programs/suiteb_cryptography/.
12. Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 95–145. Springer, Berlin, 2008.
13. Nicole Perlroth, Jeff Larson, and Scott Shane. N.S.A. able to foil basic safeguards of privacy on web. *International New York Times*, September 2013. http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html.
14. Gustavus J. Simmons. Subliminal channels; past and present. *European Transactions on Telecommunications*, 5(4):459–474, 1994.
15. Adam L. Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *1996 IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, CA, USA*, pages 129–140. IEEE Computer Society, 1996.
16. Adam L. Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997.
17. Adam L. Young and Moti Yung. *Malicious cryptography - exposing cryptovirology*. Wiley, 2004.
18. Adam L. Young and Moti Yung. Kleptography from standard assumptions and applications. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2010.