Toward an Asymmetric White-Box Proposal

Lucas Barthelemy[1,2]

Abstract. This article presents a proposal for an asymmetric white-box scheme. While symmetric white-box is a well studied topic (in particular for AES white-box) with a rich literature, there is almost no public article on the topic of asymmetric white-box. However, asymmetric white-box designs are used in practice by the industry and are a real challenge. Proprietary implementations can be found in the wild but are usually heavily obfuscated and their design is not public, which makes their study impractical. The lack of public research on that topic makes it hard to assess the security of those implementations and can cause serious security issues. Our main contribution is to bring a public proposal for an asymmetric white-box scheme. Our proposal is a lattice-based cryptographic scheme that combines classical white-box techniques and arithmetic techniques to offer resilience to the white-box context. In addition, thanks to some homomorphic properties of our scheme, we use homomorphic encoding techniques to increase the security of our proposal in a white-box setting. The resulting scheme successfully performs a decryption function without exposing its secret key while its weight remains under 20 MB. While some of our techniques are designed around specific characteristics of our proposal, some of them may be adapted to other asymmetric cryptosystems. Moreover, those techniques can be used and improved in a less restrictive model than the white-box one: the grey-box model. This proposal aims to raise awareness from the research community on the study of asymmetric white-box cryptography.

Keywords: white-box cryptography, asymmetric white-box cryptography, lattice based cryptography, software protection, homomorphic cryptography

1 Introduction

Unlike its most common black-box counterpart, white-box cryptography aims to protect the secret key of a cryptosystem not only against remote attackers, but also against the user of that cryptosystem itself. Mostly used in fields like D.R.M. (Digital Right Management), the white-box model assumes that the user of a software performing cryptographic operations may try to extract a secret key from the software. Note that in this model, the only focus is to protect the secret key used by our cryptosystem. In the case of decryption for example, the user still has a legitimate access to the resulting data. Many applications today, whether on our smartphones, computers or even cars, use and store sensitive information to function. Those sensitive information are usually protected by the use of a cryptographic function. The white-box model consider the case where the attacker has complete access to the device the cryptographic function is being performed on (whether because it was stolen or because the legitimate user itself is the attacker). He can access the code itself, read any data stored in memory and modify the behaviour of any software running on the device (create breakpoints or faults, remove the ability to generate randomness, etc. . .).

As an example, a user paying for a streaming platform must be able to decrypt the content he has subscribed to, but retrieving the secret key used in that decryption could yield information on how the system acknowledge his (and others) subscription and how content is managed by the application. This information could then lead to security issues (dumping content online for free, impersonating the provider, for example). Because the attacker in this case is the end user of the software, he has full control of the device performing the cryptographic operations that are used by the software to protect sensitive information.

In 2003, Chow, Eisen, Johnson and Oorschot proposed a white-box design based on AES [10]. From that moment forward, many proposition also based on AES were made (for example, [9] in 2006, [22] in 2009 and [15] in 2011), most of them broken after a few years (for example [6] in 2005, [16] in 2013 or [7] in 2015). Some white-box proposals cryptanalysis are specific to the white-box context, but some of them are derived from classic side-channel attacks (grey-box model) adapted to the white-box model. While AES is known to be vulnerable to side-channel attacks, it is still at the center of most white-box proposals.

More recently, a CTF challenge for the CHES conference was orchestrated to assess the state of white-box cryptography: the WhibOx challenge [1] [2]. This challenge offered anyone the chance to submit a proposal for an AES-128 white-box implementation. While the latest edition of that challenge still hold three unbroken proposals, all proposals to the previous edition were broken and most proposals were broken within days of their submission. One of the restrictions to submit a proposal was in the weight of the binary performing the cryptographic operation. Each submission had to uphold a limitation of 20 MB. In our work, we aimed at that same threshold to justify that our proposal could be as usable in practice as state of the art AES white-box schemes.

Although there is almost no article on the subject, asymmetric white-box is not something unheard of by the industry. There are already a collection of patents [21] [17] [14] regarding the subject and implementations can be found in the wild. However, those implementations are usually proprietary code and heavily obfuscated. Not only does this make the study of asymmetric white-box extremely challenging, but it is extremely dangerous for the products themselves. Indeed, the lack of standard to go from makes it very hard to assess the security of these proposals. The goal of this work is to start the discussion around asymmetric white-box. If we are already using them, we should find a way to make them as resistant as possible.

While there is a discussion that an AES white-box design behaves as an asymmetric scheme (the white-box design acting as a public key hiding the AES secret key), many applications using protocols involving an asymmetric cryptography scheme (for example, a signature in the case of a bank transaction) could benefit from a white-box design of that scheme.

Our contribution : In this article, we consider techniques used in AES white-box proposals, and apply them to a different cryptosystem. Different cryptosystems may have different properties regarding, for example, vulnerability to side-channel attacks, memory overhead compared to a classical scheme or support for different encoding techniques that prevent key extraction. In particular, we make a proposal for an asymmetric lattice based scheme that uses both classical white-box techniques and small homomorphic properties combined with arithmetic techniques to resist the white-box model. The initial proposal presented in [10] transform the classical AES in a set of lookup tables protected by various encoding techniques. In our proposal, we use both the Number Theoretic Transform (NTT) [18] and the Residue Number System (RNS) [3] representation to reduce all our computations to a set of very small integer products. This allows us to change our scheme into a fully table-based implementation where all computations involving the secret key are replaced by a series of lookup tables. A series of encodings that benefit from small homomorphic properties of our scheme are then applied to those tables to prevent key extraction.

Section 2 introduces our lattice based scheme, its notations and basic operations, before its transformation into a table-based scheme. This scheme is based on cryptosystem similar to the BGV [8] or the FV [12] cryptosystems. It is described in the following article [13].

Section 3 covers how we transform our lattice based scheme to answer the white-box model challenges. it discusses the use of the Number Theoretic Transform [18] and Residue Number System [3] representation that allows for a fully table-based implementation. Then, it introduces a new type of encodings: homomorphic encodings. Those encodings exploit the latent small homomorphic properties of our lattice based scheme to prevent key-extraction from our lookup tables. We also discuss the security of this initial design and what could be done to improve it.

Section 4 discusses two additional countermeasures that can be used to vastly improve the security of our scheme. Each of those countermeasures have specific drawbacks that should be considered before practical use.

This proposal was fully implemented in python and the resulting application matched our weight objective of 20 MB. While this is still quite the overhead compared to a regular scheme for some applications (embedded cryptography for example), we also mention that our techniques can be adjusted to reduce the memory consumption overhead at the cost of computation time overhead.

Being the first academic proposal for an asymmetric white-box scheme, we hope this work encourages the research community to study the topic of asymmetric white-box cryptography. We hope this will yield new ways to both break and improve such proposals from the community.

2 Our Initial Lattice Based Scheme

This section will introduce the lattice based scheme chosen for our proposal. It is a variant of the BGV [8] cryptosystem influenced by the design presented in [13]. This cryptosystem was chosen for its simplicity, while maintaining some key properties of lattice based cryptography. Some changes were made to the parameters as well for security concerns. The first two techniques detailed in this section are quite generic and can be adapted to many other asymmetric schemes. However, the last technique involving homomorphic properties will be harder to apply to another asymmetric cryptosystem. For example, the RSA cryptosystem presents some multiplicative homomorphic properties, but their use for a white-box design would be limited compared to a lattice based cryptosystem.

2.1 Notations

Our cryptosystem involves elements of the cyclotomic ring:

$$R_q = \frac{(\mathbf{Z}/q\mathbf{Z})[X]}{X^n + 1}$$

with parameters q, a big prime integer (15 bits in our case), and n a power of 2. In our case, we set our security parameter $n = 1024$.

Elements of R_q can be seen as polynomials of degree $n-1$, with coefficients modulo q, modulo $X^n + 1$. The product modulo R_q of two elements a and s of R_q will be denoted $a\cdot s$ while the addition of two elements, denoted $a + s$, will simply consist of a coefficient-wise addition of those elements. The notation $a[i]$ denotes the coefficient of degree i of the element a.

The underlying problem is the following:

Problem 1. Given $a, s, e \in R_q$, it is hard to distinguish $\{a, a\cdot s + e\}$ from a uniform distribution over $R_q \times R_q$.

This problem can be reduced to the *Closest Vector Problem* (CVP) on a lattice of dimension n with elements of R_q [11]. This problem consists in finding the closest lattice point from a vector in a space of dimension n.

For simplicity, a plaintext represented by an element of R_q for which all coefficients are set to 0 will simply be denoted 0. In a similar way, 1 will refer to a plaintext represented by an element of R_q for which all coefficients are set to 0 except for the coefficient of degree 0 set to 1, and X^i will denote an element for which all coefficient are set to 0 except the coefficient of degree i set to 1. Those notations will be used in the section regarding homomorphic encodings.

2.2 Keygen, Encryption, Decryption

This section briefly presents the three main functions of our scheme. Those are once again more detailed in [13]. However, this scheme is extremely generic and white-box techniques discussed in the rest of this article should be applicable to other cryptosystems like BGV [8] or FV [12]. The two functions *GaussianPoly* and *UniformPoly* generate random elements of R_q with, respectively, Gaussian and uniform distribution. the public and secret keys are, respectively, denoted by *pk* and *sk*. While α and m are, respectively, a ciphertext and a plaintext.

Algorithm 1: KeyGen

```
input  : k == security_level
output: ($pk$, $sk$)

$sk$ = GaussianPoly($n$, $q$, $k$)
$pka$ = UniformPoly($n$, $q$, $k$)
$pkb = pka \cdot sk + 2\cdot$ GaussianNoise($n$, $q$, $k$)
$pk$ = ($pka$, $pkb$)
```

Algorithm 2: Encrypt

```
input  : $pk$, $m$
output: $\alpha = (\alpha_1, \alpha_2)$
$u$ = GaussianPoly($n$, $q$, $k$)
$e_1$ = 2·GaussianPoly($n$, $q$, $k$)
$e_2$ = 2·GaussianPoly($n$, $q$, $k$)
$\alpha = (pka \cdot u + e_1, pkb \cdot u + e_2 + m)$
```

Algorithm 3: Decrypt

```
input  : $\alpha = (\alpha_1, \alpha_2)$, $sk$
output: $m$
$m = \alpha_2 - \alpha_1 \cdot sk$
for each coefficient of $m$: do
    $m[i] = (m[i] > q/2)?\ m[i]\%2\text{: } (1 - m[i])\%2$
end
```

Ciphertexts are pairs of elements of R_q, while a plaintext m is a binary element of R_q, that is to say a polynomial with coefficients in $\{0, 1\}$. During decryption, the computation:

$$m = \alpha_2 - \alpha_1 \cdot sk$$

involves the secret key s_k. It is that computation that we will need to hide in a white-box context as it involves the secret key.

3 Shaping our Scheme into a White-Box

White-Box Setting : In a white-box context, the attacker is supposed to have full access to the device the cryptographic function, in our case the decryption function, is being performed on. He has full access to any software running on the device and can consult and modify memory at will. For simplicity we assume that the attacker has access to the source code of our scheme and to any data it uses and is trying to extract the secret key sk used in our scheme.

In this section, we will explain how we modified our scheme to be more resistant in a white-box context. During decryption, we need to compute the following polynomial product:

$$m = \alpha_2 - \alpha_1 \cdot sk \in R_q$$

In the current state of our scheme, a white-box attacker could simply read the source code implementing our scheme and consult its data to retrieve the value of the secret key sk. Our goal is to protect this critical computation so that, in a white-box model, sk cannot be simply extracted from our implementation.

The proposal presented in [10] can be broken down in two points: replace any operation involving the secret key by a set of lookup tables and prevent key extraction from those tables through the use of encodings. There are some white-box cryptanalysis techniques that can bypass those protections, we will not discuss them in this article as it is not the main topic [6] [16]. However, we will mention that those techniques rely on the fact that encodings are "chained" together between lookup tables. Because our scheme uses small homomorphic properties to ensure encodings are never decoded in their encrypted forms, we believe such techniques will not be easily transformed to fit our scheme.

This section will first discuss the use of both the Number Theoretic Transform (NTT) and a Residue Number System (RNS) representation to achieve a fully table-based implementation. The use of both the NTT and the RNS representation will transform our critical computation in a set of computations involving very small integers, those computations will then be replaced by a set of lookup tables.Then, we will present encoding techniques that benefit from the small homomorphic properties of our scheme to prevent key extraction from the resulting lookup tables.

In the end, a user should dispose of a series of lookup tables performing the decryption operation of our scheme that do not leak the secret key embedded within them.

3.1 The NTT and the RNS representation

To transform our scheme so that it resists an attacker in a white-box model, we will transform its critical computation $\alpha_2 - \alpha_1 \cdot sk$ into a set of lookup tables. We seek to achieve such a transformation without reducing the values of the parameters of our ring n and q as they are both tied to the overall security of our scheme.

NTT Similar to a Fast Fourier Transform (FFT), the Number Theoretic Transform (NTT) [18] switches from a coefficient representation of polynomials into an evaluated representation in n^{th} roots of unity. Once in such a representation, computations can be performed on each evaluated value value-wise. Let ω be an n^{th} root of unity in R_q, the NTT procedure maps a polynomial α_1 in R_q with coefficients $\alpha_1[0] \ldots \alpha_1[n-1]$ to a set of n linear combinations of n distinct powers of ω as follows:

$$\beta_1[k] = \alpha_1[0] \times \omega^{0k} + \alpha_1[1] \times \omega^{1k} + \alpha_1[2] \times \omega^{2k} + \cdots + \alpha_1[n-1] \times \omega^{(n-1)k}$$

$$\forall k, 0 \leq k < n$$

Let β_1, β_2 and γ be, respectively, the NTT representation of α_1, α_2 and sk. We can then perform our key computation value-wise:

$$\delta[k] = \beta_2[k] - \beta_1[k] \times \gamma[k]$$

and finally compute the inverse NTT transformation to recover m.

This technique allows us to perform n addition/product of integers in $\mathbb{Z}/q\mathbb{Z}$ instead of one addition/product of polynomials in R_q. This is a first step toward a table-based implementation but is not sufficient on its own as n lookup tables the size of q, while feasible in theory, would be unusable in practice.

RNS Our computation $m = \alpha_2 - \alpha_1 \cdot sk$ is now, with the use of the NTT, a series of products of big integers. Another technique allows for further decomposition of those computations into operations involving smaller integers: Montgomery's multiplication algorithm in a Residue Number System (RNS) [3].

Let $\beta = \{p_0, p_1, \dots, p_k\}$ be a basis composed of small co-prime numbers p_i. Let $p = \prod_{i=0}^{k} p_i$, in a Residue Number System (RNS) any integer in $\mathbb{Z}/p\mathbb{Z}$ is represented in basis β (that is to say by its residues modulo each p_i). Any number represented in basis β can be translated back into an element of $\mathbb{Z}/p\mathbb{Z}$ via the use of the Chinese Remainder Theorem (CRT). In the RNS representation, additions and multiplications can be performed residue-wise, as a reconstruction via CRT will still produce a correct result modulo p.

Through the use of NTT, our scheme now performs n subtractions and multiplications in parallel on $\mathbb{Z}/q\mathbb{Z}$, q being a prime number of size at least the size of n (otherwise NTT would not have been possible). To use an RNS to further decompose our computations, we need to switch to another representation in $\mathbb{Z}/p\mathbb{Z}$ where $p = \prod_{i=0}^{k} p_i$. Fortunately, the well known Montgomery's multiplication algorithm allows us to switch from $\mathbb{Z}/q\mathbb{Z}$ to $\mathbb{Z}/p\mathbb{Z}$ to perform our computations. Once in $\mathbb{Z}/p\mathbb{Z}$ representation, we move to an RNS representation and perform our computations residue-wise (an example of such a combination of RNS and Montgomery's multiplication algorithm can be found in the following article [3]).

The Resulting Scheme After both transformations (NTT and RNS), our scheme has become a series of products involving extremely small integers. For our chosen q, $4-bit$ integers are enough to ensure a practical transformation into lookup tables . Let $\alpha_1^{i,j}$, $\alpha_2^{i,j}$ and $sk^{i,j}$ be the j^{th} residue in RNS of the i^{th} evaluation of respectively α_1, α_2 and sk in NTT/RNS decomposition form. we can now perform:

$$m^{i,j} = \alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j} \forall i, j$$

and then use the Chinese Remainder Theorem and the inverse NTT operation to retrieve the element m. Those elements are small enough so that those computation can be replaced by a series of lookup tables:

$$m^{i,j} = S_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}]$$

With both use of NTT and RNS, such lookup tables are possible in practice. For example, with our chosen parameters, all lookup tables weight is no more than 20 MB. Moreover, doubling our security parameter n only doubles the number of lookup tables (and therefore their size).

Note that many lattice based schemes (BGV [8], FV [12]) use RNS *before* the NTT. This use of both techniques allows for faster polynomial products in those schemes [4]. However, the minimum size of two integers involved in a product, when performing both techniques in that order, is bound by the size of n (otherwise, NTT would not be possible for the lack of n^{th} root of units), which is too much for a practical transformation of our computations in lookup tables. However, both approaches are similar.

It is possible to lower the cost of our lookup tables by chaining RNS decompositions (using Montgomery's algorithm each time to move to another basis composed of smaller primes). However, this comes at a cost of noticeable computation time overhead (as reconstruction requires more and more use of the CRT). Therefore, this technique would only be of interest for embedded softwares that struggles with memory consumption overhead.

Computations involving our secret key are now fully table-based. However, the same problem present in Chow's AES white-box is emerging. Because those tables only depends on the value of the residues of s_k, it is possible to brute force those values until all $sk^{i,j}$ are recovered. Therefore, we now need a technique to hide those residues: encodings.

3.2 Homomorphic Encodings

Unlike the AES white-box proposal in [10], our scheme does not present itself as a big network of lookup tables, but rather in a set of lookup tables in parallel. Therefore, linear and non-linear encodings do not provide a way to prevent the extraction of sk's residues.

However, unlike AES, our scheme present small homomorphic properties. Those open the way to a new type of encodings: homomorphic encodings. We will not discuss in details the proofs of those homomorphic properties, as there is already a lot of research on that topic available (for example see BGV [8] and FV [12] schemes) and it is not the main subject of this article. But we will discuss the extent and implication of those properties.

Let m_1 and m_2 be two plaintexts and c_1 and c_2 their respective ciphers for private key sk. An encryption scheme is homomorphic for an operator $\otimes$ if and only if:

$$Decrypt(c_1 \otimes c_2, sk) = m_1 \otimes m_2$$

Lattice based scheme of the BGV/FV family are not fully homomorphic. They are homomorphic for a specific amount of additions and multiplications, depending on their parameters. If too many operations are performed on ciphertexts, decryption will fail to produce the correct plaintext. The homomorphic properties (the amount of additions and/or multiplication that can be performed before it fails) heavily depends on the size of q. With our parameters, we can easily perform several homomorphic additions, but at most two homomorphic products.

Zero Encodings Let $z = (z_1, z_2)$ be a static encryption of the plaintext 0, we can use homomorphic properties to modify our lookup tables:

$$S'^{i,j}_{sk}[\alpha_2^{i,j}][\alpha_1^{i,j}] = \alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j} + z_2^{i,j} - z_1^{i,j} \cdot sk^{i,j}$$

where $z_1^{i,j}$ and $z_2^{i,j}$ are, respectively, the residues of z_1 and z_2 in NTT/RNS form. Because z is an encryption of 0, homomorphic properties will ensure that:

$$m'[i] = m[i] + 0$$

Therefore masking our lookup tables.

Note that by design, there are many different ciphertexts resulting from an encryption of 0 (This is due to IND-CCA properties of lattice based schemes). Therefore an attacker in a white-box setting would not only have to understand that an encryption of 0 has been used to encode our tables, but *which one*.

One Encodings A similar technique consists in using a static encryption of the plaintext 1, and applying a multiplication of its decryption function instead of an addition. Because of the small amount of multiplicative homomorphic properties, this encoding can only be performed twice while still guaranteeing correct decryption. In our proposition, we recommend using only one of this type of encoding as, while it adds confusion, we will need one more multiplication for the last type of encodings. Let $u = (u1, u2)$ be an encryption of 1, we can modify our lookup tables:

$$S'^{i,j}_{sk}[\alpha_2^{i,j}][\alpha_1^{i,j}] = (\alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j}) \cdot (u_2^{i,j} - u_1^{i,j} \cdot sk^{i,j})$$

Both techniques can and should be used together to provide a better masking of our lookup tables.

Masking Encodings Let *mask* be a binary mask of size n, we can use the encryption method to encrypt each bit of this mask into one static cipher $\beta = (\beta_1, \beta_2)$. If we now modify our lookup tables in a way similar to the additive zero masking:

$$S'^{i,j}_{sk}[\alpha_2^{i,j}][\alpha_1^{i,j}] = \alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j} + \beta_2^{i,j} - \beta_1^{i,j} \cdot sk^{i,j}$$

This time, once decryption is finished, we get the following relation:

$$m'[i] = m[i] \oplus mask[i]$$

The user can then get knowledge of *mask* and remove it from the resulting plaintext, as knowledge of *mask* does not allow for the removal of the encoding provided by β on the lookup tables.

Rotating Encodings Let $r = (r_1, r_2)$ be the encoding of the plaintext X^{index}. Because our elements lie in R_q, the computation:

$$m \cdot X^{index}$$

will simply result modulo 2 in a rotation of the coefficients of our plaintext. Modifying our lookup tables in the following manner:

$$S'^{i,j}_{sk}[\alpha_2^{i,j}][\alpha_1^{i,j}] = (\alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j}) \cdot (r_2^{i,j} - r_1^{i,j} \cdot sk^{i,j})$$

will yield after decryption a plaintext whose coefficients have been rotated. The user can then get knowledge of $X^{n-index}$ and perform the inverse rotation:

$$m' \cdot X^{n-index}$$

as only the knowledge of its encrypted form allows for the removing of this encoding from the lookup tables.

As explained in the section covering zero-encodings, the use of multiplicative homomorphic encodings is however limited due to the small amount of homomorphic properties our scheme possess.

3.3 On the security and limitations of our basic design

First, we would like to mention the relation between the security level of our scheme and the memory consumption overhead of its white-box design. Usually, in lattice based cryptography, security level depends on the number of dimensions n. In our proposal, doubling the size of n only doubles the number of lookup tables, which in turn doubles the size of our white-box design.

This article proposes a design for an asymmetric white-box based on a lattice based cryptographic algorithm. All computations involving the secret key sk occurring during decryption are replaced with lookup tables. A series of homomorphic encodings are then applied to prevent trivial key extraction from those lookup tables. As a result of those encodings, an extra step of decoding is required to reverse the masking and rotation of the plaintext.

As far as the security of our scheme in a white-box model is concerned, since we do not have knowledge of any public work specifically considering lattice based cryptography in a white-box model, we will consider the two following type of attacks: side-channel attacks on lattice based cryptography, as those can easily be adapted to the white-box model (consider for example the adaptation of side-channel attacks on AES white-box designs [7]), and algebraic attacks, as those proved equally potent when regarding AES white-box designs. Side-channel attacks aim to gain information on the secret key of a design by observing physical manifestation of the underlying computation (power consumption, electromagnetic emanations) when subject to different inputs and/or in the presence of faults. In a white-box setting, this can easily be adapted to other measures, like the hamming weight of data used in a computation or the control flow of the implementation of a white-box design. Algebraic attacks on the other hand, aim to represent a white-box design as a system of equations with various unknowns, one of them being the secret key or parts of it. If the system can be solved, the secret key can be recovered.

Side-Channel Attacks on Lattice Based Cryptography: Various articles regarding the protection of lattice-based schemes against side-channel attacks have been published in the last couple of years [19] [5] [20]. While side-channel attacks are not strictly speaking related to the white-box model (belonging instead to the grey-box model), many attacks on AES white-box designs shows that those attacks can easily be adapted and improved in a white-box setting. In particular, power analysis (DPA) and fault attacks (DFA) benefit greatly from a white-box setting where you have full control of the device performing the cryptographic operations (for example, the article [7] adapts classic side-channel type attacks against AES to the white-box model). In the articles [19] and [5], masked implementations of lattice based designs are proposed as a countermeasure against side-channel attacks. Our homomorphic encodings, in particular the masking encoding, perform a similar function in our design. The article [20] discuss side-channel attacks that can be performed on such designs. They conclude that while masking seems to be an effective countermeasure against DPA attacks, a Simple Power Analysis (SPA) attack is still possible. However, because our proposal is fully table-based, we do not believe such an attack is trivially adaptable to our design. Moreover, the addition of multiplicative encodings adds a layer of security against such attacks. As far as fault attacks are concerned, we have currently no knowledge of any fault attack against a lattice based design that could function on our specific proposal. However, our design is a simple lattice-based scheme that is significantly different to many state of the arts lattice-based proposals. Therefore fault attacks could very well be worth investigating further in future work.

Algebraic Attacks: As far as algebraic attacks specific to the white-box model are concerned, like the BGE attack on AES white-box designs [6], our initial design seems particularly vulnerable. Indeed, while using various combinations of homomorphic encodings in our white-box design adds a layer of security against side-channel oriented white-box attacks, extraction of the secret key could still be possible. Mixing our key computation with static encryption of various values (zero, masks, rotations) increases the complexity of the computation being performed within our lookup tables. However, the resulting table still represents the result of a linear equation. While we increase the number of values unknown to the attacker (as opposed to a non-encoded version of the scheme where sk is the only unknown), an attacker will eventually be able to solve such a system. Indeed, an attacker not only has full knowledge of our lookup tables, he also has full control over their inputs/outputs. Those tables can be seen as oracles providing as many equation as the attacker may need to recover all encodings. Once those encodings are known, they can easily be removed from our lookup tables, making key extraction trivial. While we have not managed in practice to create and solve such a system when all homomorphic encodings are used in conjunction together, the linear properties of the computations performed within our lookup tables is still a huge red flag regarding security against that type of attack.

To prevent key extraction against algebraic attacks, we need one of two countermeasures: non-linear encodings to prevent the solvability of our systems, or to be able to perform the final step of our decryption process, the reduction modulo 2, within our lookup tables. Both countermeasures are achievable and will be discussed in the following section. However, they both present severe restrictions to be used in practice.

A depiction of our design so far can be found in Appendix A

4 Additional Countermeasures

In this section, we will cover two additional countermeasures that can be used to improve the resilience of our scheme against white-box attacks. Unlike the different transformations applied to our scheme so far, those countermeasures are either specific to a particular setting of white-box cryptography or present considerable overhead to both memory consumption and computation time. However, both those techniques provide a huge gain in security in a white-box setting.

4.1 External Encodings

In article [10], external encodings are mentioned as encodings applied to encrypted messages before they are sent to the party utilizing the white-box. In the case of a white-box performing decryption, for example, a message is encrypted and then encoded before it is send to the party performing decryption. This means that the white-box used by said party needs to be constructed with those encodings in mind. In the case of AES, the first set of lookup tables therefore contains an input encoding reversing the effect of those external encodings.

Using this technique imposes a very specific set up where the party providing the white-box and the one encrypting the data are both trusted parties. For example, one can imagine a company developing a software for a specific platform. Said company sends to legitimate users encrypted software updates and a white-box performing a decryption of those encrypted updates without leaking any cryptographic key used in that process. This way, the company can ensure legitimate users are able to update their product, but also aim at preventing them from manufacturing fake updates that could alter the product in unforeseen ways (unlocking paid content for example). In this specific set up, external encodings can be used for the party providing the white-box and the one encrypting messages are both trusted (in fact, they are one and the same).

On the contrary, let us consider a setup in which the administrator of a secure environment wants to provide for its users a way to exchange encrypted data. This administrator wants to prevent its users from being able to extract their respective cryptographic keys, for they could then easily use those keys to communicate from outside the secure environment with people inside. In this set up, external encodings cannot be used, for the party encrypting data is itself not trusted and could collaborate with the one decrypting messages to gain knowledge of secret cryptographic keys.

Let us consider the use of external encodings in our scheme. Because such encodings will be applied and removed by different parties, external encodings do not need any linear or homomorphic properties. Let G (respectively H) be a set of non-linear encodings over Galois' field GF_{2^4} (one encoding for each residue derived from the NTT transform and the RNS decomposition). Let the tuple (α_1, α_2) be an encryption of a message m. Let $\alpha_1^{i,j}$ (respectively $\alpha_2^{i,j}$) be the set of residues obtained after using an NTT transform and an RNS decomposition on α_1 (respectively α_2) on α_1 using the exact same root of unity ω (for the NTT transform) and primed basis β (for the RNS decomposition) as those used in our white-box performing decryption. The encrypting party can apply external encodings by computing:

$$\bar{\alpha_1}^{i,j} = G_{i,j}(\alpha_1^{i,j})$$

respectively:

$$\bar{\alpha_2}^{i,j} == G_{i,j}(\alpha_2^{i,j})$$

Before computing the new values $\bar{\alpha_1}$ and $\bar{\alpha_2}$ by using the CRT and inverse NTT transform on the resulting residues.

As far as our white-box is concerned, decoding is pre-computed within pre-computation tables. Let $G_{i,j}^{-1}$ (respectively $H_{i,j}^{-1}$) be the inverse encodings of $G_{i,j}$ (respectively of $H_{i,j}$):

$$S_{sk}^{i,j}[\bar{\alpha_2}^{i,j}][\bar{\alpha_1}^{i,j}] = H_{i,j}^{-1}(\bar{\alpha_2}^{i,j}) - G_{i,j}^{-1}(\bar{\alpha_1}^{i,j}) \cdot sk^{i,j}$$

This transformation adds a layer of resistance against white-box attacks as the encodings of both sets G and H do not need to be linear functions, as opposed to the homomorphic encodings covered in the previous section. Therefore, our pre-computation tables cannot be seen by an attacker as a solvable linear system. However,This new type of encodings is only usable in a specific context that may not be accessible to all applications.

4.2 A 3-table scheme

This next countermeasure aims to include the last part of our decryption process, the reduction modulo 2, in our lookup tables. Reducing the result of our lookup tables modulo 2 would effectively erase most of the information accessible to an attacker in a white-box context. However, this operation cannot be performed while our lookup tables use as input residues derived from an NTT transform coupled to an RNS decomposition. To solve this issue, we propose an alternative representation of our table-based scheme.

Let S_{sk} be a set of lookup tables using a combination of homomorphic encodings:

- a rotating encoding $r = (r_1, r_2)$,
- an encoding of 0 $z = (z_1, z_2)$.
- an encoding of 1 $u = (u_1, u_2)$

For each key residue $sk^{i,j}$, the table $S_{sk}^{i,j}$ pre-computes the following equation:

$$\begin{aligned} S_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] =& ((\alpha_2^{i,j} - \alpha_1^{i,j} \cdot sk^{i,j}) \cdot (r_2^{i,j} - r_1^{i,j} \cdot sk^{i,j}) + \\ & (z_2^{i,j} - z_1^{i,j} \cdot sk^{i,j})) \cdot (u_2^{i,j} - u_1^{i,j} \cdot sk^{i,j}) \end{aligned}$$

By simply developing this equation, we get:

$$\begin{aligned} S_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] =& -\alpha_1^{i,j} \cdot r_1^{i,j} \cdot u_1^{i,j} \cdot (sk^{i,j})^3 + \\ & (\alpha_1^{i,j} \cdot r_1^{i,j} \cdot u_2^{i,j} + \alpha_2^{i,j} \cdot r_1^{i,j} \cdot u_1^{i,j} + \\ & \alpha_1^{i,j} \cdot r_2^{i,j} \cdot u_1^{i,j} + z_1^{i,j} \cdot u_1^{i,j}) \cdot (s_k^{i,j})^2 - \\ & (\alpha_2^{i,j} \cdot r_1^{i,j} \cdot u_2^{i,j} + \alpha_1^{i,j} \cdot r_2^{i,j} \cdot u_2^{i,j} + \\ & z_1^{i,j} \cdot u_2^{i,j} + \alpha_2^{i,j} \cdot r_2^{i,j} \cdot u_1^{i,j} + z_2^{i,j} \cdot u_1^{i,j}) \cdot (sk^{i,j}) + \\ & \alpha_2^{i,j} \cdot r_2^{i,j} \cdot u_2^{i,j} + z_2^{i,j} \cdot u_2^{i,j} \end{aligned}$$

Let us define two random elements $v \in R_q$ and $w \in R_q$ that will serve as masking for the following design. Let us then define two set of lookup tables V_{sk} and W_{sk} performing the following operation for all residues:

$$\begin{aligned} V_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] =& -\alpha_1^{i,j} \cdot r_1^{i,j} \cdot u_1^{i,j} \cdot (sk^{i,j})^3 + \\ & (\alpha_2^{i,j} \cdot r_1^{i,j} \cdot u_1^{i,j} + z_1^{i,j} \cdot u_1^{i,j}) \cdot (s_k^{i,j})^2 - \\ & (\alpha_2^{i,j} \cdot r_1^{i,j} \cdot u_2^{i,j} + \alpha_1^{i,j} \cdot r_2^{i,j} \cdot u_2^{i,j} + z_1^{i,j} \cdot u_2^{i,j}) \cdot (sk^{i,j}) + \\ & z_2^{i,j} \cdot u_2^{i,j} + v^{i,j} \end{aligned}$$

$$\begin{aligned} W_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] =& (\alpha_1^{i,j} \cdot r_1^{i,j} \cdot u_2^{i,j} + \alpha_1^{i,j} \cdot r_2^{i,j} \cdot u_1^{i,j}) \cdot (s_k^{i,j})^2 - \\ & (\alpha_2^{i,j} \cdot r_2^{i,j} \cdot u_1^{i,j} + z_2^{i,j} \cdot u_1^{i,j}) \cdot (sk^{i,j}) + \\ & \alpha_2^{i,j} \cdot r_2^{i,j} \cdot u_2^{i,j} + w^{i,j} \end{aligned}$$

such that for all residues $(\alpha_2^{i,j}, \alpha_1^{i,j})$:

$$V_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] + W_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}] - v^{i,j} - w^{i,j} = S_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}]$$

The last computation required to perform decryption is an addition of the result of both tables and the removal of encoding values v and w. Note that unlike the computations represented by V_{sk} and W_{sk}, this last part of the decryption process is free of products. Let $a^{i,j}$ and $b^{i,j}$ be examples of residues obtained through the use of, respectively, lookup tables $V_{sk}^{i,j}$ and $W_{sk}^{i,j}$ such that:

$$a^{i,j} = V_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}]$$
$$b^{i,j} = W_{sk}^{i,j}[\alpha_2^{i,j}][\alpha_1^{i,j}]$$

Let a (respectively b) be an element of R_q reconstructed from the residues$a^{i,j}$ (respectively $b^{i,j}$). Let T be a set of n lookup tables performing the following operation:

$$T_i[a[i], b[i]] = (m[i] = a[i] + b[i] - v[i] - w[i]) \mod 2$$

Such a lookup table T not only compute the addition of our two intermediate elements and the removal of masking values v and w, it also performs the last step of the decryption process that is the reduction modulo 2. However, such a table far exceed the size of our previous lookup tables V and W. As an example, our implementation manages to produce a set of tables T of size approximately 200MB. Therefore, a full set of tables V, W and T weight around 240MB, increasing the size of our white-box design by a factor 10.

However, as far as security is concerned, this design is far more resilient than the one described in the section 2. The distribution of computation between V and W can be randomised from one white-box implementation to the other. Furthermore, from an attacker point of view, both tables are indistinguishable from each other (they have the same inputs and produce a result of similar entropy), opening the door for various obfuscation techniques impeding the effort of an attacker trying to isolate one table from the other. As far as tables T are concerned, the reduction modulo 2 is a point of no return as most of the information on secret key sk is erased by this last step of decryption.

In conclusion, this 3-table design is far more resistant to the simple scheme presented in section 2, but it comes at a heavy price in memory consumption and computation time (as we now require two uses of the inverse-NTT operation per encrypted message).

5 Conclusion

We have presented a design of asymmetric lattice based white-box that can be used in practice. It is a table-based implementation which takes advantage of the use of both NTT and RNS techniques. The security against white-box analysis is based on the use of homomorphic encoding techniques to prevent the extraction of secret key residues.

In white-box cryptography, one concern for usability in practice is always space as using lookup tables instead of computing an operation is extremely costly. Through the use of NTT and RNS decomposition techniques, we manage to create a set of lookup tables performing all operations involving the key while remaining under 20 MB. This target of 20 MB matches the limitations imposed during the two WhibOx challenges [1] [2] that aimed to gather state of the art AES white-box implementations. Therefore, we consider this proposal to be a legitimate practical challenger for white-box cryptography. While this is still a lot of memory consumption overhead for most applications in the industry (for example, D.R.M. or mobile software), our proposal can be adjusted by chaining RNS representations to reduce the memory consumption overhead. However, this comes at a cost of non-negligible computation time overhead.

Moreover, we discuss the limitations in term of security of our basic scheme. Therefore, We also present two additional countermeasures that increase the resistance of our scheme in a white-box context. However, both those countermeasures present drawbacks that should be considered for practical use.

While there is still work to be done for optimizations and evaluation of security, this design could serve as a ground work for asymmetric white-box scheme. Asymmetric white-boxes can already be found in the wild, and poorly implemented white-box schemes can have serious consequences on software security. We encourage the community to break those types of designs in order to understand how they could be vulnerable, and how we could improve their implementations.

In addition, as far as security is concerned, because research on asymmetric white-box (and therefore its cryptanalysis) is extremely scarce, we do not recommend the usage of our proposal in practice.

While encoding techniques discussed in this article are heavily bounded to the homomorphic properties of our scheme (apart from external encodings), the use of NTT and RNS to shift toward a table-based implementation is generic enough to be applicable to other asymmetric schemes. This means that other asymmetric cryptosystems may be worth considering in the future, as those could have different interesting properties for white-box development (for example, resistance to side-channels, dynamic encodings).

Furthermore, techniques described in this work could be adapted and improved to offer resilience against grey-box attacks. In particular, homomorphic encodings could be made dynamic in a grey-box context (as the use of randomness is extremely delicate in the white-box context) and table-based implementation is already a strong tool against some side-channel techniques.

All techniques described in this article have been fully implemented and tested in python. While this language is quite helpful for proof of concept work, a C++ implementation would probably yield less overhead.

Our future work will be to improve encodings by making them dependant of some pseudo-randomness source derived from the context. We are also searching for a different technique of decomposition that could support the final computation performed by our last lookup table (in the case of our 3-table scheme). Furthermore, we are trying to find new ways to attack our current proposal. Finally, we are working on adapting those techniques to other cryptosystems to study their potential as white-box implementations.

A A Basic Asymmetric White-Box Scheme

Algorithm 4: Decrypt

```
input  : $\alpha = (\alpha_1, \alpha_2)$, $sk$
output: $m$

gotoNTT($\alpha_2$, $\alpha_1$)                                   // going into NTT
for $i$ in range($n$): do
    $x[i]$ = gotoRNS($\alpha_1[i]$)                              // going into RNS
    $y[i]$ = gotoRNS($\alpha_2[i]$)                              // using basis $\beta$
    for $j$ in range($\beta$): do
        $z[i][j]$ = fetch_data($i, j, x[i][j], y[i][j]$)          // access lookup table
    end
    $m[i]$ = gobackRNS($z[i]$)                                  // reverse RNS
end
$m$ = gobackNTT($m$)                                           // reverse NTT
for $i$ in range($n$): do
    $m[i] = (m[i] > q/2)?\ m[i]\%2\text{: } (1 - m[i])\%2$
end
decode(m)                    // remove encodings (depends on construction)
```

References

1. Ches 2017 capture the flag challenge (2017), https://whibox-contest.github.io/
2. Ches 2019 capture the flag challenge (2019), https://www.cyber-crypt.com/whibox-contest/
3. Bajard, J.C., Laurent-Stéphane, D., Kornerup, P.: An rns montgomery modular multiplication algorithm **47**, 766 – 776 (08 1998)
4. Bajard, J.C., Eynard, J., Hasan, A., Zucca, V.: A full rns variant of fv like somewhat homomorphic encryption schemes. Cryptology ePrint Archive, Report 2016/510 (2016), https://eprint.iacr.org/2016/510
5. Beirendonck, M.V., D'Anvers, J.P., Karmakar, A., Balasch, J., Verbauwhede, I.: A side-channel resistant implementation of saber. Cryptology ePrint Archive, Report 2020/733 (2020), https://eprint.iacr.org/2020/733
6. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a white box aes implementation. In: Handschuh, H., Hasan, M.A. (eds.) Selected Areas in Cryptography. pp. 227–240. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
7. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: Hiding your white-box designs is not enough. Cryptology ePrint Archive, Report 2015/753 (2015), https://eprint.iacr.org/2015/753
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277 (2011), https://eprint.iacr.org/2011/277
9. Bringer, J., Chabanne, H., Dottax, E.: White box cryptography: Another attempt. Cryptology ePrint Archive, Report 2006/468 (2006), https://eprint.iacr.org/2006/468
10. Chow, S., Eisen, P.A., Johnson, H., Oorschot, P.C.v.: White-box cryptography and an aes implementation. In: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography. pp. 250–270. SAC '02, Springer-Verlag, London, UK, UK (2003), http://dl.acm.org/citation.cfm?id=646558.694920
11. van Emde-Boas, P.: Another NP-complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice. Report. Department of Mathematics. University of Amsterdam, Department, Univ. (1981)
12. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
13. Guinet, Aguilar, Guelton, Lepoint: Quatre millions d'échanges de clés par secondes. SSTIC Archive, 2015 (2015), https://www.sstic.org/2015/presentation/4M_kx_per_sec
14. Hoogerbrugge, Jan (Eindhoven, N.M.W.E.N.: Protecting the input/output of modular encoded white-box rsa (November 2016), http://www.freepatentsonline.com/y2016/0328543.html
15. Karroumi, M.: Protecting white-box aes with dual ciphers. In: Rhee, K.H., Nyang, D. (eds.) Information Security and Cryptology - ICISC 2010. pp. 278–291. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
16. Lepoint, T., Rivain, M.: Another nail in the coffin of white-box aes implementations. Cryptology ePrint Archive, Report 2013/455 (2013), https://eprint.iacr.org/2013/455
17. Lex Aaron Anderson (Auckland), Alexander Medvinsky (San Diego, C.R.S.S.D.C.: Protecting the input/output of modular encoded white-box rsa (July 2018), https://patents.justia.com/patent/20180198613
18. Longa, P., Naehrig, M.: Speeding up the number theoretic transform for faster ideal lattice-based cryptography. Cryptology ePrint Archive, Report 2016/504 (2016), https://eprint.iacr.org/2016/504
19. Oder, T., Schneider, T., Pöppelmann, T., Güneysu, T.: Practical cca2-secure and masked ring-lwe implementation. Cryptology ePrint Archive, Report 2016/1109 (2016), https://eprint.iacr.org/2016/1109

20. Primas, R., Pessl, P., Mangard, S.: Single-trace side-channel attacks on masked lattice-based encryption. Cryptology ePrint Archive, Report 2017/594 (2017), https://eprint.iacr.org/2017/594
21. Wilhelmus P.A.J. Michiels, P.M.G.: White-box implementation (February 2009), https://patents.google.com/patent/US20110150213A1/
22. Xiao, Y., Lai, X.: A secure implementation of white-box aes (12 2009)