Zero-Knowledge Proof of Decryption for FHE Ciphertexts

Christopher Carr[1], Anamaria Costache[2], Gareth T. Davies[1], Kristian Gjøsteen[1] and Martin Strand[1]

[1]Norwegian University of Science and Technology, NTNU, Norway.
{ccarr,gareth.davies,kristian.gjosteen,martin.strand}@ntnu.no
[2]Department of Computer Science, University of Bristol, Bristol, UK.
anamaria.costache@bristol.ac.uk

January 5, 2018

Abstract

Zero-knowledge proofs of knowledge and fully-homomorphic encryption are two areas that have seen considerable advances in recent years, and these two techniques are used in conjunction in the context of verifiable decryption. Existing solutions for verifiable decryption are aimed at the batch setting, however there are many applications in which there will only be one ciphertext that requires a proof of decryption. The purpose of this paper is to provide a zero-knowledge proof of correct decryption on an FHE ciphertext, which for instance could hold the result of a cryptographic election.

We give two main contributions. Firstly, we present a bootstrapping-like protocol to switch from one FHE scheme to another. The first scheme has efficient homomorphic capabilities; the second admits a simple zero-knowledge protocol. To illustrate this, we use the Brakerski et al. (ITCS, 2012) scheme for the former, and Gentry's original scheme (STOC, 2009) for the latter. Secondly, we present a simple one-shot zero-knowledge protocol for verifiable decryption using Gentry's original FHE scheme.

1 Introduction

Consider a number of users with secret inputs who wish to compute some function on those combined inputs. If they are a small group and are online regularly then they can use multi-party computation (MPC), however in the asynchronous or large group setting this will not work. A cryptographic election is an obvious realisation of this scenario but it also covers any computation on highly sensitive data. One solution is for each user to encrypt her input using FHE, and have some semi-trusted entity perform the computation and distribute the resulting value to the users. But how can the users verify that the decryption has been done correctly? What if the FHE scheme used for the computation does not support a proof of decryption?

Verifiable Decryption

Verifiable decryption is well-known for schemes such as ElGamal. To prove that m is the decryption of $(u, v) = (mh^r, g^r)$ in some group with $h = g^a$, one has to prove the relation

$$\log_h m^{-1}u = \log_g v,$$

which can be done with a variant of the standard Schnorr protocol. For soundness, one must prove that there exists an integer value a such that the formula holds. For LWE-based cryptosystems it is no longer sufficient to prove that a certain value exists (working over the integers): it has to be smaller than some threshold. In addition, to hide the a in the Schnorr proof, the randomness used to mask a is uniformly distributed and used in such a way as to make all values of a equally likely.

The naïve approach can leak the secret key directly. Recall the DGHV scheme [49] which does FHE over the integers. The ciphertext is a number $pq + 2r + m$, where q is the key, r is noise and m is the message. If one were to apply a simple Schnorr-like protocol to the scheme, the verifier has to check that something is a lattice point. However, that is equivalent to seeing if it is divisible by the secret key q.

More concretely, consider a general LWE cryptosystem [8] (RLWE cryptosystems are built using the same blueprint). Let q be some modulus. The public key is a matrix A which contains LWE samples and the private key is some vector $\vec{s}$ such that $A\vec{s} = 2\vec{e}$, where $\vec{e}$ is some small noise. To encrypt a message m, set $\vec{m} = (m, 0, \ldots, 0)$, choose a random vector $\vec{r}$ with *small* entries (from $\{-1, 0, 1\}$), and output $\vec{c} = \vec{m} + A^T\vec{r}$. To decrypt, compute $m = [[\langle \vec{c}, \vec{s} \rangle]_q]_2$. If we follow the pattern from above, a naïve proof of correct decryption would be to prove that there exists a vector $\vec{s}$ such that

$$\langle \vec{c} - \vec{m}, \vec{s} \rangle \text{ is small.}$$

The complication is the condition "is small", and typically much smaller than the modulus in the space. One can try to produce a tight proof (with respect to soundness), but that will leak information about the secret (which did not happen in the Schnorr case, as discussed above). To safeguard the secret which is smaller than some β, one can instead prove that it is smaller than $\tau \times \beta$, where τ is large. Then we can achieve honest-verifier zero-knowledge, but at the expense of a large gap between the statement we want to prove, and that the verifier is convinced of. One can mitigate the problem using *rejection sampling*, but only to a certain extent. To sum up, the naïve approach is inadequate.

This problem is further explained in detail by Baum et al. [4]. Their goal is to provide a protocol for proving knowledge of plaintext, which is a problem related to verifiable decryption. They proceed to amortise the cost of the proof across several instances, by letting the verifier assign the ciphertexts into several buckets, and prove the claim for the sum of each bucket. Their technique has been subsequently refined [16, 18].

Baum, Damgård, Oeschsner and Peikert [5] have demonstrated a multiparty computation protocol for distributed threshold decryption. This can be transformed to an zero-knowledge protocol by doing "MPC in the head". However, it is still only efficient when amortised over multiple ciphertexts. Our goal is to start a line of research that will lead to efficient one-shot zero-knowledge protocols.

The analog problem – an *encryptor* wishes to prove that they did in fact encrypt a certain plaintext to a ciphertext – is relatively well studied. Lyubashevsky and Neven [40] show how one can avoid amortisation for proving knowledge of plaintext in a single round. Their technique is dependent on the linearity of encryption, but decryption algorithms are typically not linear.

Cryptographic Elections

In a cryptographically-verifiable election a central authority collects encrypted ballots from voters, homomorphically evaluates the election counting circuit and arrives at an encrypted result. The votes are published on some bulletin board so that voters can

also perform the election counting circuit evaluation themselves. The authority then decrypts the ciphertext encrypting the result and appends a proof that the decryption was indeed done correctly. This process of homomorphic tallying [12] is applicable when the counting function used in an election can be efficiently evaluated on encrypted ballots. This approach greatly simplifies public verifiability of a voting system [30], as correctness follows from the homomorphic properties of the cryptosystem. Note that Gjøsteen-Strand [30] uses leveled fully homomorphic encryption with a larger plaintext space, but the approach also works for binary plaintexts. Traditionally, the cryptosystems used in voting have been additively *or* multiplicatively homomorphic [36,44] which places significant restrictions on the kind of counting functions that can be computed. Fully- (or somewhat-) homomorphic encryption (F/SHE) greatly expands the applicability of homomorphic tallying, such as in Gjøsteen-Strand [30]. Unfortunately, this greater functionality comes at a cost: verifiable decryption of the result now becomes an obstacle.

Our Contribution

Gentry's breakthrough [21] came from achieving fully homomorphic capabilities through bootstrapping. Bootstrapping is the homomorphic evaluation of the decryption circuit in order to produce a ciphertext with lower noise. One can formalise this by saying that bootstrapping is an algorithm that takes a ciphertext encrypted under one instance of a scheme, into a new instance. These instances can be identical (using the same key, and requiring a property known as circular security), or they can use different keys. The only requirement for bootstrapping is that the source instance has a decryption algorithm (consider the decryption key as hard-coded into the algorithm, such that each instance has a unique algorithm) that is easy enough for the target instance to evaluate, and still have space left for further computations.

The common state situation – that the source and target instances are using the same underlying scheme – is not intrinsic to the bootstrapping idea. One can therefore generalise it to instances from different cryptosystems. This observation has yielded a number of recent works focused on providing extremely fast bootstrapping [10, 19]. We are less interested in how long this bootstrapping procedure takes since we only ever need to do it once: in this instance to produce the proof of a single decrypted value. In fact, we wish to bootstrap ciphertexts from comparatively efficient modern FHE schemes to (slower) schemes with a particular lattice structure that allows for the zero-knowledge protocol to work.

Assume we are given two homomorphic schemes: one with efficient homomorphic capabilities, the second suitable for zero-knowledge proofs. If the latter can evaluate the former's decryption circuit homomorphically, it follows that we can apply the zero-knowledge proof to the initial scheme. The challenge is to work out the algorithm that binds the two schemes together: one must take an efficient circuit for the source scheme, and formulate in such a way that the target scheme can evaluate it.

As an example of the utility of our main contribution, we provide a one-shot zero-knowledge protocol for verifiable decryption of FHE ciphertexts. In particular, we show how to transform a ciphertext of the BGV [8] encryption scheme to one from the Gentry [21] scheme. We then give a zero-knowledge proof of decryption for the Gentry scheme, and combining the two results yield a proof of decryption for BGV. Our main technical results are illustrated in Fig. 1.

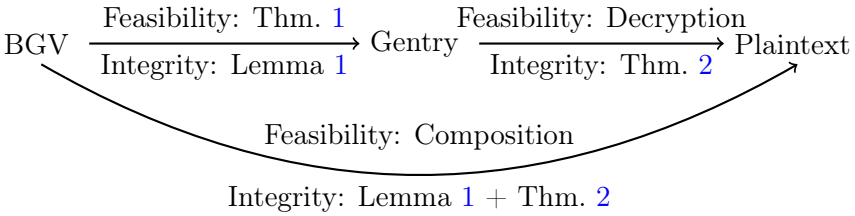

Figure 1: How to obtain a proof of decryption on a BGV ciphertext.

Further Work

Our hope is that this idea can be applied to transformations between other FHE schemes as well. Each such combination requires some precise tailoring, and can have other applications than the one we present here. For example, switching between from FHE scheme B to FHE scheme A, where scheme A is suitable for some specific algorithm, while scheme B is better for general computations.

Additionally, much like the results of decrypting AES homomorphically led to the development of more FHE-friendly symmetric schemes [2, 31, 41], we believe there is a potential to develop specialised and efficient FHE schemes for specific applications, such as simple zero-knowledge proofs. If that scheme is capable of performing the ciphertext transformation from a different scheme, then such a primitive will exist for *all* other such schemes.

To the best of our knowledge, the BGV [8] and original Gentry [21] schemes are the only ones that work in this context. The reason for this is that the Gentry scheme admits very simple and efficient zero-knowledge proofs of correct decryption, and other lattice-based schemes do not or are broken. We use the BGV scheme in the initial phase of our protocol because it appears to be the most efficient [15]. The question of whether zero-knowledge proof of correct decryption via ciphertext-switching can be instantiated with different schemes remains open.

2 Preliminaries

Denote reduction of a modulo b in two ways, either by $[a]_b$ for $a, b \in \mathbb{Z}$ to mean mapping integers to $[-\frac{b}{2}, \frac{b}{2})$ or by $\langle a \rangle_b$ to mean mapping to $[0, b)$. Use $\lceil a \rfloor$ to denote rounding $a \in \mathbb{R}$ to the nearest integer. We use $\cdot$ for scalar multiplication and $\times$ for any other multiplication operation. Denote column vectors as lower case bold $\mathbf{a}$ and matrices as upper case bold $\mathbf{A}$. The inner product of two vectors $\mathbf{a}, \mathbf{b}$ is written $\langle \mathbf{a}, \mathbf{b} \rangle$. We write $a \xleftarrow{\$} S$ to mean that a was chosen uniformly at random from the set S and $a \leftarrow D$ to mean that a was selected according to the distribution D. An NP relation R is a set (x, w) of pairs of inputs x and witnesses w, for which deciding if $(x, w) \in R$ can be checked in time polynomial in the length of x.

2.1 Zero-Knowledge Protocols

In a zero-knowledge protocol a prover attempts to prove to a verifier that she knows a proof that a statement is true – usually a witness for an instance of an NP relation. An *accepting conversation* is one for which the verifier outputs accept. Later on we will need the following two definitions relating to zero-knowledge protocols, and we use the notation of Damgård [17].

Definition 1 (Special Soundness)**.** There exists an efficient algorithm A s.t. if (a, c, z) and (a, c', z'), with $c \neq c'$, are accepting conversations for x, then $\mathsf{A}(\Delta, x, a, c, z, c', z') = w$ (where Δ are the system parameters) such that $(x, w) \in R$, for some binary relation R.

Definition 2 (Special Honest-Verifier Zero Knowledge (S-HVZK))**.** There exists a polynomial-time simulator S, which on input x and a challenge c, outputs an accepting conversation of the form (a, c, z), with the same probability distribution as conversations between the honest prover and verifier on input x.

A three-message protocol satisfying completeness and these properties is called a Σ-*protocol*.

2.2 Lattices and Ideal lattices

An n-dimensional lattice is a discrete subgroup of $\mathbb{R}^n$. Lattices that form a discrete subgroup of $\mathbb{Z}^n$ are called integral lattices, and we mainly focus on these. For a set of linearly-independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{Z}^n$, the set

$$L = \Lambda(\{\mathbf{b}_i : 1 \leq i \leq n\}) = \left\{ \sum_{i=1}^{n} x_i \cdot \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$$

is a lattice with basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Lattices are often given by providing the basis in the form of a matrix $\mathbf{B}$, with basis vectors $\mathbf{b}_i$ as columns. The rank d of a lattice L is the dimension of the subspace $\mathsf{span}(L) \subseteq \mathbb{Z}^n$, and we only consider full-rank (i.e. $d = n$) lattices.

A Hermite normal form (HNF) basis for a lattice is a basis such that $b_{i,j} = 0$ for all $i < j$, $b_{j,j}$ for all j and if $i > j$, then $b_{i,j} \in [-b_{j,j}/2, +b_{j,j}/2)$. For any basis B of L one can compute the HNF(L) efficiently using Gaussian elimination. In some older lattice-based cryptosystems, the secret key is set as a 'good' basis for a lattice where the vectors are short, and the public key is set to be the HNF of the same lattice.

Let $\Phi(X)$ be a monic irreducible polynomial of degree n. We will often use the $2n^{\text{th}}$ cyclotomic polynomial $\Phi(X) = X^n + 1$ with $n = 2^k$ for some $k \in \mathbb{Z}$. Define R as the ring of integer polynomials modulo $\Phi(X)$, $R = \mathbb{Z}[X]/(\Phi(X))$. Elements of R can be considered as polynomials or as vectors; since elements of the ring R are polynomials of degree at most $n - 1$, they can be associated with coefficient vectors in $\mathbb{Z}^n$.

A non-empty subset $I \subset R$ is called an ideal of R if I is an additive subgroup of R, and for all $r \in R$ and all $x \in I$, $x \cdot r \in I$. One can define the ideal generated by the set $E \subset R$ as the intersection of all ideals containing E (i.e. the smallest ideal containing E), which we denote by I_E if E contains more than one element. If $E = \{x\}$ contains a single element, we denote the ideal generated by it by (x). An ideal I is called principal if it is generated by a single element x, and then consists of all multiples of x in R. A lattice is an *ideal lattice* if it corresponds to an ideal of R. If the ideal lattice corresponds to a principal element (x) we can represent it using a single element $x \in R$ or its coefficient vector $\mathbf{x} \in \mathbb{Z}^n$. This allows very compact representation of each component of schemes based on ideal lattices.

2.3 Homomorphic Encryption

A homomorphic encryption scheme E is a public-key encryption scheme with an additional Eval algorithm that operates on ciphertexts. For the purposes of this paper, we focus on

schemes that are both additively and multiplicatively homomorphic. We have a security parameter λ, and the algorithms are as follows.

$$\begin{aligned}(pk, sk) &\leftarrow \mathsf{KeyGen}(1^\lambda)\\ c &\leftarrow \mathsf{Enc}(pk, m)\\ c &\leftarrow \mathsf{Eval}(pk, \mathcal{F}, c_1, \cdots, c_n)\\ m &\leftarrow \mathsf{Dec}(sk, c).\end{aligned}$$

The Eval algorithm takes as input the public key pk, ciphertexts and a (circuit representing some) function $\mathcal{F}$ and must respect a correctness requirement, namely that applying $\mathcal{F}$ to the ciphertexts is equivalent to applying it to the underlying plaintext messages,

$$\mathsf{Dec}(sk, \mathsf{Eval}(pk, \mathcal{F}, c_1, \cdots, c_n)) = \mathcal{F}(m_1, \cdots, m_n).$$

We say E is *fully homomorphic* if it can support arbitrary functions $\mathcal{F}$, and *somewhat homomorphic* otherwise.

Since many of the schemes we refer to are very complex but have been detailed extensively elsewhere, we emphasise only the key points that we are interested in, and refer the reader to the original papers and the references therein for further details. Halevi [32] points out that there have been three distinct generations of FHE schemes: i) Gentry's scheme and its variants, which require special assumptions in order to bootstrap successfully, ii) Brakerski et al.'s work that creates levelled schemes capable of evaluating any fixed (polynomial) depth, based on standard assumptions and iii) Work beginning with Gentry, Sahai and Waters [28] that has asymmetric multiplication, allowing small noise growth, at the cost of not being able to use some of the optimisations available to second-generation schemes.

2.3.1 Gentry-like Schemes

The three components that we can consider separately are the underlying SHE scheme, the procedure that 'squashes' the decryption circuit and the bootstrapping procedure. The main trick involved in Gentry's original method to reduce the degree of the decryption polynomial is to add to the public key a hint of the secret key: a large set of vectors, of which a very sparse subset adds up to the secret key (SSSP). An improvement to this step was given by Stehlé and Steinfeld [48] who showed how to reduce the number of vectors required. Gentry and Halevi [22] showed a way to do bootstrapping without squashing the decryption circuit by expressing the decryption function of the SHE scheme as a special depth-3 arithmetic circuit.

We give a high-level overview of what we refer to as a Gentry encryption. Gentry and Halevi gave an implementation of the scheme [23] which has a simpler presentation compared to the original scheme, however it was shown to be insecure [11]. There are other 'Gentry-like schemes' that have ciphertexts which are suitable for our purposes and have a simpler presentation. By this we mean not only Gentry's original scheme [21] but also the variants/implementations by Smart and Vercauteren [46] van Dijk et al. [49] and others [14, 27]. However, all of these implementations are currently broken and therefore we will use the original Gentry construction.

The algebraic set-up of the ring R is the same as mentioned above. We have a cyclotomic polynomial ring $R = \mathbb{Z}[X]/(\Phi(X))$. We also have two ideals I and J that are coprime in R, i.e. $I + J = R$. We also have two bases of the ideal J; one "good" basis, which plays the role of the secret key, and one "bad" basis, which plays the role of the

public key. Messages are binary, and we view the plaintext space as embedded into R/I. To encrypt, sample some error $\mathbf{r}$ and output $\mathbf{c} = 2\mathbf{r}+\mathbf{m} \pmod{\mathbf{B}_{\text{pk}}}$, so that the ciphertext is

$$\mathbf{c} = 2\mathbf{r} + \mathbf{b} + \mathbf{m},$$

where $\mathbf{b} \in \mathbf{B}_{\text{pk}}$, which is the "bad" basis – i.e. public key – of J. Here m is encoded as the constant polynomial 0 or 1.

To decrypt, use the good basis to identify and remove $\mathbf{b}$ (using the modulo operation), and then the inner noise in the same manner.

2.3.2 Brakerski-Gentry-Vaikuntanathan-like Schemes

We can separately consider another class of FHE schemes that do not use assumptions for bootstrapping, but instead employ modulus switching [7–9, 37]. This class of schemes can be optimised in a number of ways [3, 24–26, 33, 35, 47]. In this work, we choose to focus on the original BGV scheme [8]. This is because it is currently the most efficient scheme [15], as well as one of the two FHE schemes widely implemented [34].

Decryption of a BGV ciphertext $\mathbf{c}$ carrying m is performed with secret key s by evaluating

$$m \leftarrow [\langle \mathbf{c}, \mathbf{s} \rangle]_q \pmod 2,$$

where $\mathbf{s} = (1, -s)$. For a more detailed description and analysis of this scheme, we refer to the original presentation [8]. The construction is a levelled scheme and applies modulus switching at each level. A fresh ciphertext is encrypted at the 'top' level, modulo q_L. With each multiplication, we appropriately scale the resulting ciphertext by q_{i-1}/q_i to go from a ciphertext at level q_i to one at level q_{i-1}. This reduces the noise growth and allows for L multiplications. In an implementation, the number of multiplications L we allow for is specified in the set-up phase.

3 Ciphertext Switching

This section represents our main technical contribution on FHE: an algorithm which transforms a BGV ciphertext into a Gentry one. In Section 4 we will detail how to give a zero-knowledge proof of decryption for Gentry-style schemes, and in combination with our *ciphertext switching* technique this resolves our motivating scenario: use a BGV-type scheme to efficiently perform computations, then a Gentry-type one for verifiable decryption. Our approach takes inspiration from Gentry's original work [21]: bootstrapping is simply a homomorphic evaluation of the decryption circuit. This is done by adding one layer of encryption on a ciphertext, then evaluating the decryption circuit, resulting in a ciphertext with one layer of encryption. The added layer of encryption can be under the same scheme, or a different one. In the first case, the result of the homomorphic decryption is a ciphertext in the initial scheme. In the latter case, a ciphertext in the second scheme. We are interested in the latter case. We will use the bootstrapping procedure in order to perform a ciphertext-switch between a BGV ciphertext and a Gentry one. We pick the Gentry and BGV schemes because of their efficient capabilities to perform verifiable decryption and homomorphic operations, respectively. The idea of ciphertext-switching is relatively straightforward, and it is easy to imagine a context in which two different schemes would be used.

The results presented in this section could not be implemented in practice since the original Gentry construction [21] is not (securely) implementable. Therefore, in this section we provide a proof of concept that the ciphertext switching procedure can be instantiated.

We use the notation $\{m\}_{\mathsf{E}}$ to mean that the plaintext message m is encrypted under the scheme E. In particular, we will write $\{m\}_{\mathsf{G}}$ and $\{m\}_{\mathsf{BGV}}$ to mean that the message m is encrypted under the Gentry and the BGV scheme, respectively. We assume that all messages and keys are honestly generated.

Assume we are working in the ring $R = \mathbb{Z}[X]/(\Phi(X))$, where $\Phi(X)$ is monic irreducible, taken to be a power of two cyclotomic polynomial. Let $\deg(\Phi) = n$ and fix this ring for the remaining of this section. For simplicity, we will assume that s is a binary polynomial in the ring R. This has an impact on security, and one would need to carefully select s as described by Albrecht [1].

Before looking at the ciphertext-switching procedure, we need to ensure that it can be set up. This means that we need to match up the plaintext and ciphertext spaces. Both spaces have a very simple presentation for the BGV scheme. These are, respectively, R_p and R_q, where $R_p = R/pR$, and similarly for R_q. The integers p and q are referred to as the plaintext and ciphertext moduli, respectively. We can set the plaintext modulus p to be 2, without loss of generality. The ciphertext modulus q is chosen in accordance with security requirements [1].

For the Gentry scheme, the plaintext space can also be taken to be binary, so matching the two schemes is not complicated. Matching the ciphertext spaces is more intricate. Simplifying greatly, a Gentry ciphertext is an element of the form $(\mathbf{c} \mod J) \mod I$, where both I and J are ideals of the ring R. The exact set-up and definition of these ideals is very complex, and we refer the reader to the original presentation. This simplified presentation of a ciphertext is of course not enough to give a deep understanding of the scheme; however it is enough to see that a Gentry ciphertext lies in the intersection of ideals $I \cap J$, which we will call K. Therefore, to ensure that the ciphertext spaces match, we will require that $R_q \subseteq K$.

In order to perform the ciphertext-switching procedure, we encrypt each of the coefficients s_i of s under the Gentry scheme. This is performed by sampling errors $\mathbf{r}_i$ and $\mathbf{b}_i$ and outputting

$$\mathbf{a}_i = 2 \cdot \mathbf{r}_i + \mathbf{b}_i + s_i.$$

As mentioned previously, many bootstrapping procedures make use of another homomorphic scheme. Typically, this is a GSW or LWE encryption scheme [10, 19]. The method is then: bootstrap a ciphertext with a GSW or LWE-encrypted secret key, which gives a GSW or LWE-encrypted message, according to which scheme was used in the procedure. The bootstrapping operation is then achieved by extracting the encryption of the message in the desired form. Since we will not want to recover a ciphertext in BGV form, this allows us to dispense of the last operation. Thus, our method becomes: encrypt the BGV secret key under the Gentry scheme, and decrypt homomorphically.

The procedure is as follows: we start with a BGV ciphertext $\mathbf{c} = (c_0, c_1) \in R^2$ under a secret key s. We ignore the issues of modulus or key switching, and also note that there is an implicit mod $\Phi(X)$ performed with each homomorphic operation. The BGV secret key will have the form

$$s(X) = \sum_{i=0}^{n-1} s_i \cdot X^i,$$

where each $s_i \in \{0, 1\}$. The first step is to encrypt each s_i in the Gentry scheme. Each $\{s_i\}_{\mathsf{G}}$ is an n-vector. We form a GSW-type matrix of all the encryptions $\{s_i\}_{\mathsf{G}}$:

$$\mathbf{S} = (\{s_i\}_{\mathsf{G}})_{i \in \{0 \cdots n-1\}}.$$

For a more detailed analysis of this technique, see for example Alperin-Sheriff and Peikert [3]. Recall that the ciphertext $\mathbf{c} = (c_0, c_1)$ consists of two polynomials of degree (at most) $n-1$. Write them in their coefficient vector representation and evaluate

$$\mathbf{c}_0 - \mathbf{c}_1 \cdot \mathbf{S}.$$

Recall that bootstrapping is the process of homomorphically evaluating the decryption circuit. Thus, evaluating the BGV decryption circuit on a BGV ciphertext with a Gentry-encrypted secret key results in a Gentry-encrypted ciphertext. We have the following theorem.

Theorem 1. *Ciphertext-switching a BGV ciphertext* $\mathbf{c}$ *under its Gentry-encrypted secret key* $\{s\}_{\mathsf{G}}$ *results in a Gentry-encrypted ciphertext.*

Proof. We will assume that we have set up our schemes in a correct manner, i.e. that the plaintext/ ciphertext spaces match up, as explained earlier in this section. Suppose we have an encryption scheme E which is linearly homomorphic, i.e. the following is true

$$b - a \cdot \mathsf{E}(s) = \mathsf{E}(b - a \cdot s).$$

This is trivially true of most homomorphic schemes in the literature, including the Gentry scheme, and the proof relies on this fact. This is true of any encryption scheme E which supports affine transformations.

Indeed, the decryption circuit of BGV is

$$m \leftarrow [\langle \mathbf{c}, \mathbf{s} \rangle]_q \pmod 2,$$

where by abuse of notation $\langle \mathbf{c}, \mathbf{s} \rangle = c_0 - c_1 \cdot s$, for a ciphertext $\mathbf{c} = (c_0, c_1)$. Now if the scheme E does indeed support affine transformations, we have that

$$c_0 - c_1 \cdot \{s\}_{\mathsf{G}} = \{c_0 - c_1 \cdot s\}_{\mathsf{G}}.$$

More precisely, writing each row i in $\mathbf{S}$ as $2 \cdot \mathbf{r}_i + \mathbf{b}_i + s_i$, in evaluating the above we get the following. Notice we now switch to vector coefficient notation, writing $\mathbf{c}_i$ for the polynomial c_i.

$$\begin{aligned}
\mathbf{c}_0 - \mathbf{c}_1 \cdot \{s\}_{\mathsf{G}} &= \sum_i c_{0,i} - c_{1,1} \cdot (2 \cdot \mathbf{r}_i + \mathbf{b}_i + s_i) \\
&= \sum_i (c_{0,i} - c_{1,1} \cdot s_i) + (2 \cdot \mathbf{r}_i + \mathbf{b}_i) \\
&= \mathbf{c}_0 - \mathbf{c}_1 \cdot \mathbf{s} + 2 \cdot \mathbf{r} + \mathbf{b} \\
&= m + k \cdot q + 2 \cdot \mathbf{r} + \mathbf{b} \\
&= \{m\}_{\mathsf{G}} \pmod q.
\end{aligned}$$

Providing a complete noise analysis here would require the introduction of all the formalism from Gentry's scheme, which is beyond our scope. Instead, we notice that our resulting Gentry ciphertext is the sum of n such ciphertexts, and we refer to the original construction for a thorough analysis. □

3.1 Switching Integrity

Returning to the election example, assume that the election authority has prepared for some likely, but unfavourable, outcomes. Resourcefully, they alter the transformation key $\{s\}_{\mathsf{G}}$ during key generation to encrypt a different key, which will transform the correct result into one they prefer, which they can then prove the correct decryption of.

Multiparty computation can be used for key generation in answer to this problem. However, there is a simpler method to check the validity of the transformation key. We give the result in its full generality, only assuming linearity of decryption and that cancellation holds in the plaintext space.

In the following lemma, the reader might find it helpful to think of sk as the "real" decryption key, and sk' as a key the dishonest decryptor has found to change the output favourably.

Lemma 1. *Let $\mathcal{C}$ and $\mathcal{P}$ denote the ciphertext and plaintext spaces, and let $\mathsf{Dec}_{sk}, \mathsf{Dec}_{sk'} : \mathcal{C} \to \mathcal{P}$ be functions indexed by two different keys. Assume that the adversary wants to target messages in a set $A \subset \mathcal{C}$, such that for all $c \in A$ we have $\mathsf{Dec}_{sk}(c) \neq \mathsf{Dec}_{sk'}(c)$. Assuming the decryption algorithm is additively homomorphic and that cancellation works in the plaintext space, then for ciphertexts outside of A, Dec_{sk} and $\mathsf{Dec}_{sk'}$ will also differ.*

Proof. We use a contrapositive argument. Assume that $c_1 \in A$ and $c_2 \notin A$. Then there are two cases for $c_1 + c_2$, either in or not in A. We assume the latter, the former is analogous. There are some corner cases for FHE ciphertext spaces where the linearity does not hold, but then, for the sake of the argument one can just select different values.

Assume that decryption under sk' agrees with sk for all values outside A. It follows that

$$\begin{aligned}\mathsf{Dec}_{sk}(c_1 + c_2) &= \mathsf{Dec}_{sk'}(c_1 + c_2) = \mathsf{Dec}_{sk'}(c_1) + \mathsf{Dec}_{sk'}(c_2) \\ &= \mathsf{Dec}_{sk'}(c_1) + \mathsf{Dec}_{sk}(c_2).\end{aligned}$$

By linearity and cancellation, we must have $\mathsf{Dec}_{sk'}(c_1) = \mathsf{Dec}_{sk}(c_1)$. Hence, Dec'_{sk} cannot modify any value in A without also modifying all values outside A. □

Verification Protocol: To verify that the secret key is correct, one only needs to verify the decryption of a single non-zero plaintext. Let E_S be the source scheme and E_T be the target scheme.

1. The challenger sends a tuple $(m, \{m\}_{\mathsf{E}_S})$ to the decryptor.
2. Both parties agree on a representation of $\{m\}_{\mathsf{E}_T}$ using the public transformation algorithm.
3. The decryptor proves the correctness of the decryption $\{m\}_{\mathsf{E}_T} - m = 0$.
4. The challenger verifies the proof.

Note that this protocol can be carried out independently (and possibly long before) the zero-knowledge protocol that we describe in the next section.

4 One-Shot Verifiable Decryption

As we discussed in the introduction, there are no generic and efficient zero-knowledge proofs for correct decryption of an FHE ciphertext. Currently, the most promising approach is based on MPC, and is only efficient when the computational cost is amortised over a large number of instances [6]. In this section we detail a zero-knowledge proof for decryption of Gentry ciphertexts, and thus when combined with the ciphertext switching procedure discussed in the previous chapter we can transform a ciphertext from any scheme with a sufficiently simple decryption circuit to a "proof-friendly" scheme.

4.1 The Zero-Knowledge Proof

Let $\hat{\mathbf{c}}$ be a ciphertext encrypted under a scheme whose decryption circuit is sufficiently simple for Gentry, and let $\mathbf{c}$ be the corresponding ciphertext under Gentry. Recall that $\mathbf{c}$ can be written as $\mathbf{c} = 2\mathbf{r} + \mathbf{b} + \mathbf{m}$, where $\mathbf{b}$ represents a lattice point and $\mathbf{r}$ is some noise vector. As long as $\mathbf{r}$ is inside some set D, the ciphertext is decryptable. Letting the prover have access to the decryption key, we get a simple Schnorr-like Σ-protocol to prove that $m = 0$. Notice that the decryption algorithm can be modified to output both the message and the noise vector.

The generic protocol, between a prover P and a verifier V, is as follows.

P1 Choose an encryption $\mathbf{c}' = \mathbf{b}' + \mathbf{r}'$ of zero such that the noise $\mathbf{r}'$ can hide $\mathbf{r}$, and send $\mathbf{c}'$ to the verifier.

V1 Select $e \xleftarrow{\$} \{0, 1\}$ and send e to the prover.

P2 If $e = 0$, set $\mathbf{d} = \mathbf{b}'$, or if $e = 1$, set $\mathbf{d} = \mathbf{b} + \mathbf{b}'$. Transmit $\mathbf{d}$.

V2 Verify that $\mathbf{d}$ is a lattice point, and check that the noise $e\mathbf{c} + \mathbf{c}' - \mathbf{d}$ is well-formed and sufficiently small.

We use rejection sampling [38,39] to improve the parameters of our proposal. Rejection sampling is useful in a scenario where one has a distribution D, but lacks a good way of sampling from it. Instead, one can sample from a larger distribution $E \supset D$, and simply reject any value not in D. The usefulness becomes apparent when the sampling takes part over several rounds in a protocol. Typically, a value outside D will leak information, but only after combining everything in the final step can one decide whether it will be out of bounds, so we avoid this problem.

We can generalise the protocol to a larger e, which will result in an arbitrarily good soundness bound (but may require increased parameters for the Gentry scheme). To instantiate the above idea specifically for the Gentry scheme, all that remains is to add the following lines to the respective steps:

P2 If $e\mathbf{r} + \mathbf{r}'$ is outside the set D, reject.

V2 Verify that $\mathbf{d}$ is a lattice point: check that $(B_J^{\mathrm{pk}})^{-1}\mathbf{d}$ is an integer vector. Verify that $e\mathbf{c} + \mathbf{c}' - \mathbf{d}$ is within bounds and the correct randomness to generate $\mathbf{d}$ in the encryption algorithm.

Notice that $e\mathbf{c} + \mathbf{c}' - \mathbf{d} = 2e\mathbf{r} + \mathbf{r}'$, which is the combined randomness used to generate the ciphertexts $\mathbf{c}$ and $\mathbf{c}'$, so $\mathbf{d}$ is the lattice part of $e\mathbf{c} + \mathbf{c}'$.

The modification of step P2 is an application of rejection sampling. The probability of the prover rejecting depends on the parameters: a large set D (possibly exponentially

large) requires larger parameters, whereas a small set will result in more rejections. We leave the choice of concrete parameters for specific applications.

Remark 1. When creating a Schnorr-like zero-knowledge proof for modern LWE-based schemes, one usually has to prove that they can extract a value through rewinding and this value will be small, like we have discussed earlier. However, the extraction equation normally requires that one divides by the difference of the challenges. If that is different from ± 1 then there is no longer a guarantee that the extracted value is small, so one is prevented from using a large challenge space. Alternatively, one can use some values outside ± 1, with the consequence that one can only make a guarantee for a bound larger than that one is interested in. The difference may be acceptable, and is called the "soundness slack" [4]. Here, we do not need to reconstruct short vectors. We are not limited by the size of the challenge space, and so the main advantage of the above protocol is the lack of soundness slack.

Theorem 2. *The above protocol is complete and achieves special soundness and special honest-verifier zero knowledge.*

Proof. To verify completeness, notice that $(B_J^{\mathrm{pk}})^{-1}(\mathbf{b}') + e\mathbf{b}$ will be an integer vector since $\mathbf{b}'$ and $\mathbf{b}$ are lattice points. Next, we have $e\mathbf{c} + \mathbf{c}' - \mathbf{d} = 2e\mathbf{r} + \mathbf{r}'$, which is exactly the values used to generate the lattice points.

For special soundness, assume we have gathered challenges e_0 and e_1 such that $e_0 - e_1$ is invertible in R, and that the prover has responded successfully with $\mathbf{d}_0$ and $\mathbf{d}_1$. Then compute $(e_0 - e_1)^{-1}(\mathbf{d}_0 - \mathbf{d}_1)$ to get $\mathbf{b}$ and extract the message as $\mathbf{c} - \mathbf{b} \pmod 2$.

Depending on the ring R, $e_0 - e_1$ may not always be invertible. If so, let ϕ denote the probability that an arbitrary element is a unit. After rewinding, $e_0 - e_1$ is invertible with probability ϕ. After rewinding $k - 1$ times, the probability that no $e_i - e_j$ is invertible is just $(1 - \phi)^{\binom{k}{2}}$, which quickly becomes negligible.

Next we prove special honest-verifier zero knowledge. Note that the transcript of a correct protocol run is $\{\mathbf{c}', e, \mathbf{d}\}$, where the ciphertext is a fresh random encryption of 0, e is a uniformly random value from some finite set and $\mathbf{d}$ is distributed based on the distribution on $\mathbf{r}$ in the encryption algorithm. The simulator is initiated with an e, then proceeds to select a random noise vector and computes the corresponding lattice point $\mathbf{d}$. Then $\mathbf{c}'$ is given by $e\mathbf{c}+\mathbf{c}'-\mathbf{d} = \mathbf{r}\,\mathbf{c}'$, which guarantees that the verification step is satisfied. The distribution of the simulated transcript essentially only depends on the distribution of the randomness $\mathbf{r}$ as a function of e, so the simulator can choose it accordingly. □ □

In the protocol we took advantage of the lattice structure of Gentry's cryptosystem to create an elegant proof – structure that is not available in more efficient schemes such as BGV. The challenge in providing a compact ZK proof of decryption for other schemes (not just BGV) remains but we believe this will be a fruitful direction given that one can design a proof-friendly FHE scheme that is only geared towards a single circuit: the decryption of ciphertexts under a different scheme. Looking at the greater picture, verifiable decryption need not be the only application of the generalised bootstrapping idea.

Acknowledgements. We would like to thank Nigel Smart for valuable discussions and feedback. This research was partially funded by the Research Council of Norway under Project No. 248166. Part of this work was conducted while the second author visited NTNU.

References

[1] M. R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany. Cited on page 8.

[2] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In Oswald and Fischlin [43], pages 430–454. Cited on page 4.

[3] J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In Garay and Gennaro [20], pages 297–314. Cited on pages 7 and 9.

[4] C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498, Santa Barbara, CA, USA, Aug. 14–18, 2016. Springer, Heidelberg, Germany. Cited on pages 2 and 12.

[5] C. Baum, I. Damgård, S. Oechsner, and C. Peikert. Efficient commitments and zero-knowledge protocols from ring-SIS with applications to lattice-based threshold cryptosystems. Cryptology ePrint Archive, Report 2016/997, 2016. http://eprint.iacr.org/2016/997. Cited on page 2.

[6] C. Baum, I. Damgård, T. Toft, and R. W. Zakarias. Better preprocessing for secure multiparty computation. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 327–345, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany. Cited on page 11.

[7] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Safavi-Naini and Canetti [45], pages 868–886. Cited on page 7.

[8] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *ITCS 2012*, pages 309–325, Cambridge, MA, USA, Jan. 8–10, 2012. ACM. Cited on pages 2, 3, 4, and 7.

[9] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [42], pages 97–106. Cited on page 7.

[10] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg, Germany. Cited on pages 3 and 8.

[11] G. Chunsheng. Cryptanalysis of the smart-vercauteren and gentry-halevi's fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/328, 2011. http://eprint.iacr.org/2011/328. Cited on page 6.

[12] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th FOCS*, pages 372–382, Portland, Oregon, Oct. 21–23, 1985. IEEE Computer Society Press. Cited on page 3.

[13] J. Coron and J. B. Nielsen, editors. *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany. Cited on pages 14 and 16.

[14] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504, Santa Barbara, CA, USA, Aug. 14–18, 2011. Springer, Heidelberg, Germany. Cited on page 6.

[15] A. Costache and N. P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340, San Francisco, CA, USA, Feb. 29 – Mar. 4, 2016. Springer, Heidelberg, Germany. Cited on pages 4 and 7.

[16] R. Cramer, I. Damgård, C. Xing, and C. Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In Coron and Nielsen [13], pages 479–500. Cited on page 2.

[17] I. Damgård. On σ-protocols, v2. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2010. Cited on page 4.

[18] R. del Pino and V. Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394, Santa Barbara, CA, USA, Aug. 20–24, 2017. Springer, Heidelberg, Germany. Cited on page 2.

[19] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Oswald and Fischlin [43], pages 617–640. Cited on pages 3 and 8.

[20] J. A. Garay and R. Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, Santa Barbara, CA, USA, Aug. 17–21, 2014. Springer, Heidelberg, Germany. Cited on pages 13 and 15.

[21] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. Cited on pages 3, 4, 6, and 7.

[22] C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [42], pages 107–109. Cited on page 6.

[23] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. Cited on page 6.

[24] C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 1–16, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany. Cited on page 7.

[25] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg, Germany. Cited on page 7.

[26] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [45], pages 850–867. Cited on page 7.

[27] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In Gilbert [29], pages 506–522. Cited on page 6.

[28] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Heidelberg, Germany. Cited on page 6.

[29] H. Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. Cited on pages 15 and 16.

[30] K. Gjøsteen and M. Strand. A roadmap to fully homomorphic elections: Stronger security, better verifiability. Cryptology ePrint Archive, Report 2017/166, 2017. http://eprint.iacr.org/2017/166. Cited on page 3.

[31] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. MPC-friendly symmetric key primitives. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 430–443, Vienna, Austria, Oct. 24–28, 2016. ACM Press. Cited on page 4.

[32] S. Halevi. Homomorphic encryption. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 219–276, Cham, 2017. Springer International Publishing. Cited on page 6.

[33] S. Halevi and V. Shoup. Algorithms in HElib. In Garay and Gennaro [20], pages 554–571. Cited on page 7.

[34] S. Halevi and V. Shoup. Helib – an implementation of homomorphic encryption., 2014. Cited on page 7.

[35] S. Halevi and V. Shoup. Bootstrapping for HElib. In Oswald and Fischlin [43], pages 641–670. Cited on page 7.

[36] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. Cited on page 3.

[37] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In H. J. Karloff and T. Pitassi, editors, *44th ACM STOC*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press. Cited on page 7.

[38] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In R. Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 162–179, Barcelona, Spain, Mar. 9–12, 2008. Springer, Heidelberg, Germany. Cited on page 11.

[39] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616, Tokyo, Japan, Dec. 6–10, 2009. Springer, Heidelberg, Germany. Cited on page 11.

[40] V. Lyubashevsky and G. Neven. One-shot verifiable encryption from lattices. In Coron and Nielsen [13], pages 293–323. Cited on page 2.

[41] P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 311–343, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. Cited on page 4.

[42] R. Ostrovsky, editor. *52nd FOCS*, Palm Springs, CA, USA, Oct. 22–25, 2011. IEEE Computer Society Press. Cited on pages 13 and 14.

[43] E. Oswald and M. Fischlin, editors. *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, Sofia, Bulgaria, Apr. 26–30, 2015. Springer, Heidelberg, Germany. Cited on pages 13, 14, and 15.

[44] K. Peng, R. Aditya, C. Boyd, E. Dawson, and B. Lee. Multiplicative homomorphic e-voting. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 61–72, Chennai, India, Dec. 20–22, 2004. Springer, Heidelberg, Germany. Cited on page 3.

[45] R. Safavi-Naini and R. Canetti, editors. *CRYPTO 2012*, volume 7417 of *LNCS*, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Heidelberg, Germany. Cited on pages 13 and 14.

[46] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany. Cited on page 6.

[47] N. P. Smart and F. Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014. Cited on page 7.

[48] D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394, Singapore, Dec. 5–9, 2010. Springer, Heidelberg, Germany. Cited on page 6.

[49] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [29], pages 24–43. Cited on pages 2 and 6.