Doctoral theses at NTNU, 2018:33

Martin Strand

Fully homomorphic encryption with applications to electronic voting

Doctoral Thesis

ISBN 978-82-326-2864-3 (printed version)
ISBN 978-82-326-2865-0 (electronic version)
ISSN 1503-8181

Martin Strand

Doctoral theses at NTNU, 2018:33

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

NTNU

NTNU
Norwegian University of
Science and Technology

Martin Strand

Fully homomorphic encryption with applications to electronic voting

Thesis for the degree of Philosophiae Doctor

Trondheim, January 2018

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-326-2864-3 (printed version)
ISBN 978-82-326-2865-0 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2018:33

Printed by Skipnes Kommunikasjon as

Preface

This thesis means that I am about to complete a PhD in cryptology. It has been four challenging and instructive years, and as most PhD students experience, my self confidence has swung between 'world champion' and 'impostor'. I was given some rest at Eurocrypt 2017: in his invited talk Nigel Smart called for more research in the area between new theoretical discoveries and ready-to-use products, and I finally felt at home. My results are not at a stage of maturity where they can be used out there. Neither are they new theoretical results. Nonetheless, the hope is that my work can make it easier to design new cryptographic tools, and perhaps help other researchers avoid some blind alleys.

I am indebted to many for making it here. First and foremost, I wish to thank my advisor Kristian Gjøsteen for taking the time to answer any question I've had, and in particular for also answering the questions I should have asked. Despite many obligations, there always seems to be time to spare.

Next, to my coauthors Frederik, Colin, Chris, Kristian, Angela, Christian, Ana and Gareth: it has been a pleasure working with you. I am particularly grateful to Frederik Armknecht and the rest of the group in Mannheim for hosting me in 2015, and making me feel welcome. This acknowledgement extends to the whole crypto community. Even at my first major conference I found it easy to discuss both big problems and banal questions with anyone, be it fellow PhD students or highly merited IACR fellows.

The life of a PhD student is not just about writing papers. My friends and colleagues have given me many opportunities to procrastinate, forget about unreasonable reviewers, play cards, solve world

problems – and in summary given me the breaks I needed to readjust to what after all turned out to be insightful comments from the reviewers.

I owe many thanks to my family for babysitting, careful reading and for providing such great encouragement from the very first day of my formal education. Ragnhild, you deserve the biggest hug of them all. This last year you have put as much effort as anyone into making this possible. Thank you.

Martin Strand
Trondheim, January 2018

Introduction

The connecting theme of the thesis is the theory and applications of fully homomorphic encryption (FHE), in particular with respect to electronic voting and the tools we need in voting protocols.

Fully homomorphic encryption

Fully homomorphic encryption was first envisioned by Rivest, Adleman and Dertouzos in 1978 [13], but it was only theoretically realised by Gentry [9] over 30 years later. The concept of FHE is perhaps best introduced as the solution to an apparent deadlock. Party A has sensitive data that they would like to have analysed. Party B has the best algorithm to analyse the data. However, A does not want to disclose the data to B and B does not want to reveal the algorithm to A, not even through a compiled program: such programs can be reversed engineered. One solution is for party A to encrypt the data, but party B needs to be able to compute on the data. Hence, we need a cryptosystem that can evaluate algebraic operations. We can express it mathematically: let $\mathcal{P}$ and $\mathcal{C}$ denote the plaintext and ciphertext spaces, and let $\oplus_{\mathcal{P}}, \oplus_{\mathcal{C}}$ and $\odot_{\mathcal{P}}, \odot_{\mathcal{C}}$ be operations on the spaces. We then require that for any two ciphertexts c_1 and c_2, we have

$$\begin{aligned}\mathsf{Dec}(c_1 \oplus_{\mathcal{C}} c_2) &= \mathsf{Dec}(c_1) \oplus_{\mathcal{P}} \mathsf{Dec}(c_2)\\ \mathsf{Dec}(c_1 \odot_{\mathcal{C}} c_2) &= \mathsf{Dec}(c_1) \odot_{\mathcal{P}} \mathsf{Dec}(c_2).\end{aligned}$$

It has been known for decades how to compute either one of these operations on encrypted data, but being able to do both at the same time seemed so difficult it eventually became a 'holy grail' of cryptography. Fully homomorphic schemes often express the homomorphic property as a special algorithm Eval that on input of an *evaluation key*, a *circuit* $\mathcal{C}$ and a set of ciphertexts, returns a new ciphertext that encrypts the output of the circuit, as if it had been applied to the plaintexts:

$$\begin{aligned} c_1 \leftarrow \mathsf{Enc}(m_1), \ldots, c_n &\leftarrow \mathsf{Enc}(m_n) \\ c &\leftarrow \mathsf{Eval}(\mathcal{C}, c_1, \ldots c_n) \\ \Rightarrow \quad \mathsf{Dec}(c) &= \mathcal{C}(m_1, \ldots, m_n) \end{aligned}$$

A simple FHE scheme is as follows [16]. Let p be some odd integer, and we will consider it as our key. To encrypt a bit m, select two random numbers r and q from certain ranges, and set the ciphertext as

$$c \leftarrow pq + 2r + m.$$

To decrypt, reduce c modulo p and 2.

For sufficiently large p, q and r, m is hidden. Given two ciphertexts, we see that

$$\begin{aligned} c_1 + c_2 &= p(q_1 + q_2) + 2(r_1 + r_2) + (m_1 + m_2) \\ c_2 \cdot c_2 &= p(pq_1q_2 + 2r_2q_1 + 2r_1q_2 + q_1m_2 + q_2m_1) \\ &\quad + 2(2r_1r_2 + r_1m_2 + r_2m_1) + m_1m_2, \end{aligned}$$

which are both of the same form as the original ciphertext. Observe that the multiplication increases both the ciphertext size and the term multiplied by 2. For the middle term, that is a potential problem. The term grows quadratically in the *noise* r, and will after sufficiently many operations be of the same size as p. Once that happens, it will 'flow over' and may in the process flip the bit m in the $pq + 2r + m$ representation. We say that the noise has grown too much, and the ciphertext can no longer be decrypted reliably.

The second problem of this scheme is its extreme inefficiency. With a naïve implementation of this scheme, encrypting a single bit with (supposedly) 160 bit security, we get a ciphertext of length 317 bits, and it only handles less than five multiplications.

One can also turn this scheme into a public key scheme. The immediate performance drop illustrates the challenge with FHE. Generating key material for 7 (!) bits of security takes several minutes; an encryption of a single bit requires 55023 bits; and the public key is about $3 \cdot 10^9$ bits. We return to the tremendous development that has taken place since this early scheme later in the introduction, as well as in the first paper of the thesis. The thesis contributes new understanding of what can and cannot be achieved using fully homomorphic encryption.

Paper i *A Guide to Fully Homomorphic Encryption.* The paper gives a thorough review of the state of FHE when it was written in 2014–2015. The version included in this thesis is identical to the ePrint version, which seems to have been of use for several authors.

FHE has evolved since our paper was written. One can almost say that the paper was written while FHE still had a ‘sense of adventure’ connected to it, and there were new schemes and problems being proposed at a steady pace. Just a few years later, some ideas have stood out, and we will now briefly consider some of the convergence in the field since the guide was written. The following paragraphs are closely tied to the first paper, so a reader unfamiliar with FHE the reader might find it useful to return to these paragraphs later.

We start by discussing the concept of *hops*. In the survey, we discuss the difference between levels and hops, where the level is the allowed depth of a circuit, whereas the hops describe whether or not a ciphertext output from an evaluation can be used again for new computations. As far as the author is aware, there are no schemes where this separation has been important, since all schemes in reality only describe how to handle separate gates. The recent literature on the field has not mentioned hops.

Gentry’s groundbreaking idea was that one could use a severely limited scheme that could only support a small number of operations, but then apply the decryption circuit with an encrypted key to create new ciphertexts, and in the same process remove the noise that had aggregated. Thus, under certain conditions, one can get unlimited capacity from a noisy scheme. The first challenge was to make the decryption circuit sufficiently shallow to make it possible to evaluate it with the cryptosystem. Secondly, the bootstrapping procedure itself had to be made practical. Starting with the novel paper by Brakerski, Gentry and Vaikuntanathan [6], bootstrapping is often viewed as an optimisation rather than a necessity. The current trend is to select suitable parameters for the concrete application. Larger parameters create bigger ciphertexts and each operation takes slightly longer, but it is still a reasonable trade-off in order to avoid the very costly bootstrapping operation. For even bigger computations it may still be better to use smaller parameters and bootstrapping. In hindsight there is an irony to the development: Realising that an 'exact' solution (like ElGamal is for group homomorphic schemes) was infeasible, the bootstrapping idea to reach the same objective was the real breakthrough. The definitive improvement since 2009 has been on the noisy schemes, and the field is accepting that *levelled* homomorphic encryption – without bootstrapping – can be as useful as schemes supporting arbitrary depth circuits.

The first paper lists a number of schemes and implementations. If we were to redo the table today, it would be far sparser, and essentially only mention two directions: The BGV scheme [6] (starting with the modulus switching introduced by Brakerski and Vaikuntanathan [7]) and the GSW scheme [10] with its variations, in particular the reformulation by Alperin-Sheriff and Peikert [1]. Lately GSW has become completely dominating in the research literature, and other lines of work seem to have been completely abandoned.

We also list a number of security assumptions that early FHE schemes used. These have mostly vanished, and current schemes almost exclusively base their security on the *Learning with errors* problem or its ring variant. When writing the survey paper, we ignored an important catch when it comes to Ring-LWE. Let us quickly revisit the problems here, starting with plain LWE.

Let q be some integer, and let χ be the discrete Gaussian distribution on $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. Fix a dimension n and a vector $\vec{s} \in \mathbb{Z}_q^n$. For each sample, choose a vector $\vec{a}$ from $\mathbb{Z}_q^n$ and a noise term $e \leftarrow \chi$, and output

$$(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

Given m independent samples, the adversary must output $\vec{s}$. There is a reduction from the problem of finding short vectors in the lattice (GapSVP and SIVP) to LWE [12], and these related problems have received much attention and are widely believed to be hard to solve even for quantum computers.

It is natural to view RLWE as an instance where $\mathbb{Z}_q$ is replaced with a larger ring, typically $R = \mathbb{Z}[x]/(x^d + 1)$, $d = 2^l$ for some l, which allows us to set $n = 1$, and thus reduce the overall sample size. Lyubashevsky, Peikert and Regev [11] demonstrate that this direct approach gives a weaker problem. The reason is that the reduction considers the embedding of $\mathbb{Z}[x]/(x^d + 1)$ into $\mathbb{C}^{r_1} \times \mathbb{R}^{r_2}$ for some r_1, r_2, by adjoining the all roots of $x^d + 1$. This will create a skewed distribution. The reader may find a mental image of an ellipse helpful, where the security essentially depends on the shortest radius. The three authors instead explain that one should sample s from the *dual fractional ideal* of R, a from $R_q = R/qR$ and the error term from a *spherical n-dimensional Gaussian distribution* ψ. We refer to the original source for the details. While this is of little significance for the rest of the thesis, we include a mention of it here as a reminder that some of the confusion from the early days of FHE is still present.

One can get an impression of the current status of FHE research by reviewing the programs of some of the recent IACR flagship conferences. Both Crypto and Eurocrypt 2015 had two full sessions dedicated to FHE, but the same venues in 2017 accepted no papers on the field. Lattice-based cryptography is still a hot topic, and FHE is often assumed as a primitive, for instance in papers on Functional Encryption [2].

Fully homomorphic encryption may have hit a brick wall when it comes to efficiency. Smart [15] discussed this as an example of the progression from a theoretical idea, to something that finds practical use in implementations. The second paper in the thesis partly supports this observation.

Paper II *Can there be efficient and natural FHE schemes?* We analyse the possibilities and limitations of FHE, and the analysis partly supports the observation that there seems to be inherent limitations to how good it can get. We either need to rely on solutions with large ciphertext expansion, and any scheme radically outside of the current paradigm is likely to be homomorphic over structures that have no or little use in the computations one would like to do to in practice.

There appears to be a pattern in how new cryptography is introduced. The first results, the proofs of concept, often deal with single bits. It is still common to start by introducing bit commitment schemes, and then zero-knowledge protocols with bit challenges, oblivious transfer of bits, and so on. This holds for FHE as well. The papers introducing new schemes typically demonstrate how to encrypt a single bit. However, in contrast to the other examples, the schemes sometimes rely extensively on this property. For instance, the analysis of the noise growth in the Gentry-Sahai-Waters scheme [10] depends crucially on the size of the message, and a consequence is that it is non-trivial to extend the performance improvements to fields that are sufficiently large to support interesting computations.

There are still routinely posted manuscripts on ePrint trying to realise FHE with noiseless schemes. Just as routinely, they have proven to be insecure, for instance using the techniques developed in the second paper in this thesis.

Electronic voting with FHE

There is a large community researching how to carry out electronic voting. Generally, there are two variations: on-site voting on automats and remote internet voting. Each variant has different challenges, and we focus on the latter in our work.

Remote internet voting is a controversial and challenging field. Ryan [14] has described the problem as a triangle with corners *verifiable* (that the voter can be certain that the vote was counted as intended), *coercion resistance* (anybody else than the voter should not be able to verify how a ballot was cast and counted, to ensure privacy and fight vote-buying) and *useable* (the user should be able to use and understand the system in an easy manner), where one can only satisfy two out of three. He then noted that the Norwegian system seemed to be a tradeoff to all three, with an emphasis on coercion resistance and usability.

If the electorate does not trust the authorities, end-to-end verifiability is more important. End-to-end verifiability means that the voter, without help from any other parties, can check that his ballot was counted as intended. The reader can appreciate how hard this is to combine with coercion resistance, that a third party looking over the voter's shoulder or trying to buy the vote (and wanting a receipt) should have no way of verifying that the voter did as instructed. Combine this with the third requirement, that any non-expert should be able to cast a ballot and understand what he or she is doing.

Norway ran a trial project for e-voting in 2011 and 2013. The government cancelled the project in 2014, but a private company, *New Voting Technology Consulting* has continued helping municipalities with local referendums. We present a completely new way of carrying out the Norwegian election in the thesis.

> **Paper III** *A roadmap to fully homomorphic elections: Stronger security, better verifiability.* We demonstrate how one can deploy a complicated e-voting protocol using FHE, and harvest stronger security properties than one could do previously.

At the end, we provide some numbers to estimate the order of work required to implement this protocol using the BGV scheme [6]. This scheme belongs to what has been called 2nd generation FHE schemes. While less sophisticated than the 3rd generation GSW scheme, it provides more flexibility, and is suitable for reasonably large plaintext spaces. There have been several theorised applications of FHE, some of which are described in the first paper. Many of these are unrealistic in the foreseeable future. This work is unique in its dedication to connect FHE with a highly complex real-world application.

The paper is previously published at VOTING'17. This version is expanded and includes several improvements.

- The presentation has been updated. The algorithms are now presented independently of the cryptosystem, so that the correctness becomes clearer.

- We have added more precise security requirements, and prove that our system satisfy these. Due to page limitations, this discussion had to be omitted in the previous version.
- Several technical improvements to the ballot expansion algorithm which sits at the heart of the work.

Next, a verifiable shuffle is used whenever one wants to shuffle and reencrypt a list of encrypted data such that nobody can tell the correspondence between old and new ciphertexts, yet be fully convinced that the responsible did not replace or modify any of the ciphertexts. Shuffles are particularly useful in certain e-voting applications, but also finds its use in network routing for privacy purposes.

> **Paper IV** *A verifiable shuffle for the GSW cryptosystem.* The paper introduces the primitive of verifiable shuffles to the realm of FHE.

This proof of concept uses the GSW scheme. Bourse, del Pino, Minelli and Wee [5] have provided a very elegant solution to the problem of circuit privacy. (We explore this term in more depth in the initial survey paper.) Circuit privacy is necessary to hide the recryption ahead of the zero-knowledge proof. It turns out that shuffling can be efficient for FHE, but so far only under the assumption that the encryption scheme can handle sufficiently large numbers. As we have already observed, BGV is better for large messages, but has no known efficient method for getting circuit privacy. GSW can handle large plaintexts, but only at the expense of higher noise growth.

The fact that different schemes have their own strengths and weaknesses leads us to the final paper in this collection.

> **Paper V** *Zero-Knowledge Proof of Decryption for FHE Ciphertexts.* The main objective is to provide a good technique for proving correct decryption of a ciphertext.

There is a strong line of research for proving plaintext knowledge, where the efficiency is based on amortisation. Along the same lines, there is a proof of correct decryption [4]. The fundamental problem when working with a scheme based on LWE is that one wants to prove not only that some value exists, but also that it is smaller than some bound. However, the naïve Schnorr-like idea will reveal information about the secret, so one can only safely prove that it is smaller than some constant times the bound. A line of research for the related problem of plaintext knowledge, starting with Baum, Damgård, Larsen and Nielsen [3], with its latest iteration at Crypto 2017 by del Pino and Lyubashevsky [8] mitigates this problem by using many instances, dividing them into baskets by random, and proving the claim for each basket. Our idea is to find an FHE scheme which can support a standard Schnorr proof, and then use the original bootstrapping concept to switch the ciphertext from the original scheme to the scheme that supported simple zero knowledge proofs.

Unfortunately, our proof of concept needs to go back to the first FHE scheme by Gentry, which is impractical by every measure. However, the scheme switching idea is generic, and can be used for any application if the message spaces match and the target scheme can evaluate the decryption circuit of the source.

References

[1] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. `doi:10.1007/978-3-662-44371-2_17`.

[2] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[3] Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. How to prove knowledge of small secrets. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498, Santa Barbara,

CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. `doi:10.1007/978-3-662-53015-3_17`.

[4] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris 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`.

[5] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. `doi:10.1007/978-3-662-53008-5_3`.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. ACM.

[7] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[8] Rafaël del Pino and Vadim Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[9] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[10] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and

Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-40041-4_5`.

[11] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-38348-9_3`.

[12] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[13] Ronald Rivest, Leonard Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[14] Peter Ryan. In the PhD defence of Anders S. Lund, September 2015.

[15] Nigel Smart. Living between the ideal and real worlds. Invited talk at Eurocrypt 2017, May 2017.

[16] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

Paper I

A Guide to Fully Homomorphic Encryption

Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Gorke and Martin Strand

ePrint 2015/1192

A Guide to Fully Homomorphic Encryption

Frederik Armknecht[1], Colin Boyd[2], Christopher Carr[2], Kristian Gjøsteen[3], Angela Jäschke[1], Christian A. Reuter[1], and Martin Strand[3]

[1]University of Mannheim
{armknecht, jaeschke, reuter}@uni-mannheim.de
[2]Department of Telematics, NTNU
{colin.boyd, ccarr}@item.ntnu.no
[3]Department of Mathematical Sciences, NTNU
{kristian.gjosteen, martin.strand}@ntnu.no

Abstract

Fully homomorphic encryption (FHE) has been dubbed the holy grail of cryptography, an elusive goal which could solve the IT world's problems of security and trust. Research in the area exploded after 2009 when Craig Gentry showed that FHE can be realised in principle. Since that time considerable progress has been made in finding more practical and more efficient solutions. Whilst research quickly developed, terminology and concepts became diverse and confusing so that today it can be difficult to understand what the achievements of different works actually are. The purpose of this paper is to address three fundamental questions: What is FHE? What can FHE be used for? What is the state of FHE today? As well as surveying the field, we clarify different terminology in use and prove connections between different FHE notions.

1 Introduction

The purpose of homomorphic encryption is to allow computation on encrypted data. Thus data can remain confidential while it is processed, enabling useful tasks to be accomplished with data residing in untrusted environments. In a world of distributed computation and heterogeneous networking this is a hugely valuable capability. Finding a general method for computing on encrypted data had been a goal in cryptography since it was proposed in 1978 by Rivest, Adleman and Dertouzos [54]. Interest in this topic is due to its numerous applications in the real world. The development of fully homomorphic encryption is a revolutionary advance, greatly extending the scope of the computations which can be applied to process encrypted data homomorphically. Since Gentry published his idea in 2009 [28, 29] there has been huge interest in the area, with regard to improving the schemes, implementing them and applying them.

We look in detail at specific applications in Section 2, but to give a feeling, consider cloud computing. As more and more data is outsourced into cloud storage, often unencrypted, considerable trust is required in the cloud storage providers. The Cloud Security Alliance lists data breach as the top threat to cloud security [61]. Encrypting the data with conventional encryption avoids the problem. However, now the user cannot operate on the data and must download the data to perform the computations locally. With fully homomorphic encryption the cloud can perform computations on behalf of the user and return only the encrypted result.

1.1 What is Fully Homomorphic Encryption?

Principally, FHE allows for arbitrary computations on encrypted data. Computing on encrypted data means that if a user has a function f and want to obtain $f(m_1, \ldots, m_n)$ for some inputs $m_1, \ldots, m_n$, it is possible to instead compute on encryptions of these inputs, $c_1, \ldots, c_n$, obtaining a result which decrypts to $f(m_1, \ldots, m_n)$.

In some cryptosystems the input messages (plaintexts) lie within some algebraic structure, often a group or a ring. In such cases the ciphertexts will often also lie within some related structure, which could be the same as that of the plaintexts. The function f in older homomorphic encryption schemes is typically restricted to be an algebraic operation associated with the structure of the plaintexts. For instance, consider ElGamal. If the plaintext space is a group G, then the ciphertext space is the product $G \times G$, and f is restricted to the group operation on G. Indeed most schemes prior to 2009 fit such a structure. We can express the aim of fully homomorphic encryption to be to extend the function f to be *any function.* This aim can be achieved if the scheme is homomorphic with respect to a functionally complete set of operations and it is possible to iterate operations from that set.

While it is always a requirement that encryption schemes are efficient in a theoretical sense, namely running in polynomial time in the security parameter, practical efficiency was not the first priority in obtaining the first FHE schemes. One reason for the lack of efficiency of these schemes is that they use a plaintext space consisting of a single bit and are homomorphic with respect to addition and multiplication modulo 2. While any function of any complexity can be built up from such basic operations, that may require a large number of such operations.

In order to move towards better efficiency, some recent variants of FHE schemes restrict the functions f in different ways which we will explore later.

Although a theoretical view of FHE cares only about maximising the choices of f, a practical view cares also about keeping this choice only as large as needed, and may also prefer a richer structure for the plaintext and ciphertext spaces that just the binary case.

1.2 Relations of FHE: Functional Encryption and Program Obfuscation

The fundamental idea behind FHE is to be able to apply functions on encrypted data. Two other cryptographic notions formed with functions in mind are *functional encryption* and *obfuscation.* Intriguingly, obfuscation, functional encryption and fully homomorphic encryption seem somehow intertwined, as has been previously recognised [3, 25].

Functional encryption (FE) is similar in essence to identity based encryption and attribute based encryption. Boneh, Sahai and Waters [13] give a concise explanation of the relations between these three notions, as well as some discussion on FHE. The concepts of FHE and FE do indeed have some overlap, and it has been demonstrated that functional encryption can work as FHE, with some slight adaptation [3].

Functional encryption allows a secret key to be issued using a master key, dependent on a function f. Given a ciphertext, the secret key allows the user to learn the value of f applied to the plaintext and nothing else [12]. Computing functions on encrypted data links the two concepts of homomorphic and functional encryption. A notable difference is the way the functions are applied. FE grants control over what functions can be applied to the data via a master key holder, who issues keys based on a decision of the appropriateness of the function. A key can be used to obtain the plaintext result of the function applied on the encrypted data. FHE permits functions to be the run by anyone with the *evaluation key* (see Section 3), however only the owner of the secret key can decrypt the result into plaintext. The user running the function only gets ciphertext.

Obfuscation was originally designed to be conceptually similar to black box computation, where one gains knowledge of inputs to the black box, and outputs from it, but nothing else [9, 25, 38]. With obfuscation, one could place keys within the program to be run without revealing knowledge of the keys. One could thus generate an obfuscated program that contains the public and private keys, and process the input by applying first the decryption algorithm, next the required function, and finally encrypting the result. This would act as a replacement for the homomorphic operation in FHE.

This ability to generate a FHE scheme from an obfuscation scheme and a traditional encryption scheme may seem promising, but practically it remains unclear if this offers an advantage over direct FHE. We would also need to consider the security constraints and implications of *hiding* secret keys inside a published program.

1.3 Need for Systematization

Treatment of FHE can seem very confusing. Sometimes, two definitions seem to say the same thing – for example, at first glance, being able to evaluate an arbitrary circuit and being able to evaluate arbitrarily many circuits consecutively seems to be the same thing. This, however, is not the case as will be explained in Remark 5.

To help understand the distinction, consider the cloud computing example: FHE is usually sold as the solution. However, if we can only evaluate one circuit of arbitrary size, then we cannot use intermediate results for further computations later; everything has to be computed from scratch through the original ciphertexts. This satisfies the usual definition of FHE (Definition 9), but is unintuitive and is hardly an optimal solution. What is needed in this scenario is the ability to evaluate arbitrarily many circuits consecutively.

This highlights another problem in this field: in some cases, definitions do not express what one would intuitively assume. In other cases, one intuition has different definitions in different papers. This, for example, is the case for an attribute called *compactness*, which intuitively says that the ciphertext size should not be growing through homomorphic operations. Gentry defines it through one characterization in his original work, while in subsequent works a different characterization is used. Seeing that both definitions are equivalent is not as straightforward as one may assume, and actually requires an additional assumption.

Sometimes, attributes are not properly defined at all, and sometimes implications are used that have not been mentioned in the same paper. Figure 1 gives an idea of how complex this jungle of definitions is. Starting with the definitions and properties (white rectangles in the figure) we can give classifications of the different kinds of homomorphic schemes (shaded round rectangles). Furthermore, these classifications can again be combined with hop correctness, which yields another set of homomorphic schemes (darker oval shapes).

1.4 Our Contribution

First, in Section 2, we gather existing applications of (fully) homomorphic encryption mentioned in the literature, examine their usefulness both in practice and as building blocks for other cryptographic schemes and point out their limitations.

Next, we provide the much needed organization of terminology. We present existing definitions in a consistent way, reconciling different definitions for one notion when they exist, and explaining points of potential confusion. Furthermore, we introduce new definitions, enabling a better understanding of existing schemes and existing

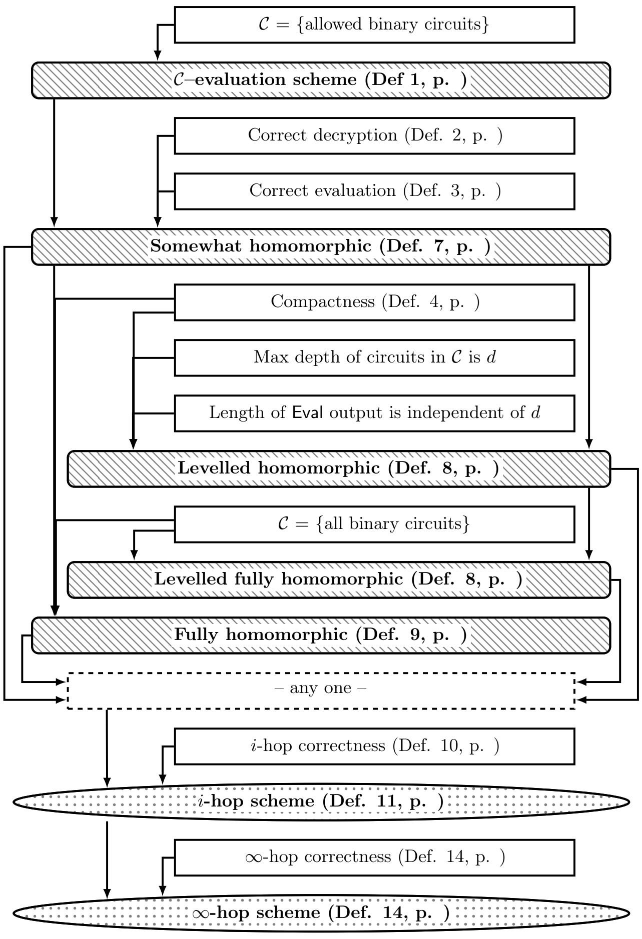
$\mathcal{C}$ = {allowed binary circuits}
$\mathcal{C}$–evaluation scheme (Def 1, p.)
Correct decryption (Def. 2, p.)
Correct evaluation (Def. 3, p.)
Somewhat homomorphic (Def. 7, p.)
Compactness (Def. 4, p.)
Max depth of circuits in $\mathcal{C}$ is d
Length of Eval output is independent of d
Levelled homomorphic (Def. 8, p.)
$\mathcal{C}$ = {all binary circuits}
Levelled fully homomorphic (Def. 8, p.)
Fully homomorphic (Def. 9, p.)
– any one –
i-hop correctness (Def. 10, p.)
i-hop scheme (Def. 11, p.)
∞-hop correctness (Def. 14, p.)
∞-hop scheme (Def. 14, p.)

Figure 1: Classifying FHE. The definitions are white rectangles. The classes are shaded round rectangles. The round shapes below represent the add-on properties of hops. The arrows between the elements of the graph show their dependencies. The integer d specifies the maximum depth of circuits in the set of allowed binary circuits $\mathcal{C}$.

definitions. This contribution constitutes Section 3 and is summarised in Figure 1. In Section 4, we formally prove some elementary relations between notions presented in Section 3. To give an overview of the current landscape in FHE research, Section 5 summarises some existing schemes along with the underlying hardness assumptions and runtimes where available.

2 Applications of FHE

This section explores the numerous applications of the various flavours of homomorphic encryption. Some require *fully* homomorphic encryption, while others just need *somewhat* homomorphic encryption. The distinction will become clear in Section 3. For now, it suffices to know that a fully homomorphic scheme can compute anything on encrypted data, while a somewhat homomorphic scheme is more restricted.

This section is divided into three parts. The first part deals with applications that are feasible today, the second examines constructions that use homomorphic encryption as building blocks, and the third looks at current limitations of FHE.

2.1 Practical Applications of FHE

Although still slow (see Section 5), homomorphic encryption has been proposed for several practical uses. This section lists those applications that are conceivable with the technology we have today.

2.1.1 Consumer Privacy in Advertising

Though often unwanted, advertising can be useful when tailored to user needs, e.g. through recommender systems or through location-based advertising. However, many users are concerned about the privacy of their data, in this case their preferences or location. There have been several approaches to this problem.

Jeckmans et al. [43] sketch a scenario where a user wants recommendations for a product. The scenario is designed around a social network where recommendations are based on the tastes of the user's friends with the condition of confidentiality. The proposed system applies homomorphic encryption to allow a user to obtain recommendations from friends without the identity of the recommender being revealed.

Armknecht and Strufe [6] presented a recommender system where a user gets encrypted recommendations without the system being aware of the content. This system builds upon a very simple but highly efficient homomorphic encryption scheme which has been developed for this purpose. This allows a function to be computed which chooses the advertisement for each user while the advertising remains encrypted.

In another approach to personalized advertising [50] a mobile device sends a user's location to a provider, who sends customized ads, such as discount vouchers for nearby shops, back to the user. Of course, this potentially allows the provider to monitor everything about the user's habits and preferences. However, this problem can be solved by homomorphic encryption – provided the advertisements come from a third party (or several) and there is no collusion with the provider.

2.1.2 Data Mining

Mining from large data sets offers great value, but the price for this is the user's privacy. While Yang, Zhong and Wright [64] are often cited as using homomorphic encryption as a solution to this problem, the scheme actually uses *functional encryption*, a common confusion discussed in Section 1.2. However, applying homomorphic encryption is certainly conceivable as a solution.

2.1.3 Medical Applications

Naehrig et al. [50] propose a scenario where a patient's medical data is (continuously) uploaded to a service provider in encrypted form. Here, the user is the data owner, so the data is encrypted under the user's public key and only the user can decrypt. The service provider then computes on the encrypted data, which could consist of things like blood pressure, heart rate, weight or blood sugar reading to predict the likelihood of certain conditions occurring or more generally to just keep track of the user's health. The main benefit here is to allow real-time health analysis based on readings from various sources without having to disclose this data to any one source. Lauter [45] described an actual implementation of a heart attack prediction by Microsoft.

2.1.4 Financial Privacy

Imagine a scenario where a corporation has sensitive data and also proprietary algorithms that they do not want disclosed, e.g. stock price prediction algorithms in the financial sector. Naehrig et al. [50] propose the use of homomorphic encryption to upload both the data and the algorithm in encrypted form in order to outsource the computations to a cloud service. However, keeping the algorithm secret is not something that homomorphic encryption offers, but is rather part of *obfuscation* research (see section 1.2). The attribute that comes closest in fully homomorphic schemes is called *circuit privacy*, but this merely guarantees that no information about the function is leaked by the output – not that one can encrypt the function itself.

What homomorphic encryption offers is the solution to a related problem. Imagine that a corporation A has sensitive data, like a stock portfolio, and another company B has secret algorithms that make predictions about the stock price. If A would like to use B's algorithms (for a price, of course), either A would have to disclose the stock portfolio to B, or B has to give the algorithm to A. With homomorphic encryption, however, A can encrypt the data with a circuit private scheme and send it to B, who runs the proprietary algorithm and only sends back the result, which can only be decrypted by A's secret key. This way, B does not learn anything about A's data, and A does not learn anything about the algorithms used.

2.1.5 Forensic Image Recognition

Bösch et al. [14] describe how to outsource forensic image recognition. Tools similar to this are being used by the police and other law enforcement agencies to detect illegal images in a hard drive, network data streams and other data sets. The police use a database containing hash values of "bad" pictures. in. A major concern is that perpetrators could obtain this database, check if their images would be detected and, if so, change them.

This scheme uses a somewhat homomorphic encryption scheme proposed by Brakerski and Vaikuntanathan [17] to realise a scenario where the police database is encrypted while at the same time the company's legitimate network traffic stays private. The company compares the hashed and encrypted picture data stream with the encrypted database created by the police. The service provider learns nothing about the encrypted database itself, and after a given time interval or threshold, the temporary variable is sent to the police.

2.2 Homomorphic Encryption as a Building Block

Homomorphic encryption schemes can be used to construct cryptographic tools such as zero knowledge proofs, signatures, MACs and multiparty computation implementations.

2.2.1 Zero Knowledge Proofs

Gentry shows in his dissertation [28] that homomorphic encryption can be used in the construction of non-interactive zero knowledge (NIZK) proofs of small size. A user wants to prove knowledge of a satisfying assignment of bits $\pi_1, \ldots, \pi_t$ for a boolean circuit C. The NIZK proof consists of generating a public key, encrypting the π_i's and homomorphically evaluating C on these encryptions. A standard NIZK proof is attached to prove that each ciphertext encrypts either 0 or 1 and that the output of the evaluation encrypts 1.

2.2.2 Delegation of Computation

Outsourcing computation is the second big pillar in cloud computing, besides outsourcing data. A user may want to delegate the computation of a function f to the server. However, the server may be malicious or just prone to malfunctions, meaning the user may not trust the result of the computation. The user wants to have a proof that the computation was done correctly and verifying this proof should also be significantly more efficient than the user doing the computation.

Chung et al. [18] use fully homomorphic encryption to design schemes for delegating computation, improving the results of Gennaro et al. [26], while van Dijk and Juels [63] examine the infeasibility of FHE alone solving privacy issues in cloud computing.

One example for the delegation of computation is *message authenticators*. A user who has outsourced computation on a data set might want to check that the return value is really the correct result. The tag should be independent of the size of the original data set, and only verifiable for the holder of the private key. Gennaro and Wichs [27] propose such a scheme based on a fully homomorphic encryption scheme, which can be considered as a symmetric-key version of fully homomorphic signatures [10]. However, it only supports a bounded number of verification queries.

2.2.3 Signatures

Gorbunov et al. [39] presented a construction of levelled fully homomorphic signature schemes. The scheme can evaluate arbitrary circuits with maximal depth d over signed data and homomorphically produce a short signature which can be verified by anybody using the public verification key. The user uploads the signed data x, then the server runs some function g over the data which yields $y = g(x)$. Additionally, the server publishes the signature $\sigma_{g,y}$ to verify the computation.

This work also introduces the notion of *homomorphic trapdoor functions* (HTDF), one of the building blocks for the signature construction. HTDF themselves are based on the small integer solution (SIS) problem. The first definition of fully homomorphic signatures was given in Boneh and Freeman [10].

2.2.4 Multiparty Computation

Multiparty computation protocols require interaction between participants. Damgård et al. [21] provide a description of how a somewhat homomorphic scheme can be used to construct offline multiplication during the computations. The players use the somewhat homomorphic scheme in a preprocessing phase, but return to the much more efficient techniques of multiparty computation in the computation phase.

2.3 Limitations of FHE

Both in literature and intuitively, there are several applications which permit fully homomorphic encryption as a solution. However, in this subsection, we discuss three main limitations of FHE in real-world scenarios.

The first limitation is support for multiple users. Suppose there are many users of the same system (which relies on an internal database that is used in computations), and who wish to protect their personal data from the provider. One solution would be for the provider to have a separate database for every user, encrypted under that user's public key. If this database is very large and there are many users, this would quickly become infeasible. López-Alt et al. [46] have shown promising directions to address this problem by defining and constructing multi-key FHE.

Next, there are limitations for applications that involve running very large and complex algorithms homomorphically. All fully homomorphic encryption schemes today have a large computational overhead, which describes the ratio of computation time in the encrypted version versus computation time in the clear. Although polynomial in size, this overhead tends to be a rather large polynomial, which increases runtimes substantially and makes homomorphic computation of complex functions impractical. Even if in the future an extremely efficient FHE should be found, other problems remain. For example, for circuits, there is no concept of aborting an algorithm when operating on encrypted data. In the case of comparison, this would require to run the full circuit which is large by itself. In other words, certain mechanisms seems to get significantly more involved just because values remain hidden. One way to solve this problem is suggested by Goldwasser et al. [37] by using Turing machines instead of circuits.

Finally, FHE does not necessarily imply secret function evaluation. We already encountered this in the discussion of the applicability to financial data above. This issue belongs to the research on obfuscation.

3 Definitions

This section gives an overview of the terminology used in the literature on FHE. Some of our definitions come directly from existing papers while others have been rephrased, either because there were no satisfactory formal definitions or to fit the definitions into our formal framework; we give citations in the first case.

We begin with a space $\mathcal{P} = \{0, 1\}$, which we call the plaintext space, and a family F of functions from tuples of plaintexts to $\mathcal{P}$. We can express such a function as a Boolean circuit on its inputs. If we denote this circuit by C, we use ordinary function notation $C(m_1, m_2, \ldots, m_n)$ to denote the evaluation of the circuit on the tuple $(m_1, m_2, \ldots, m_n)$. Our first definition follows Brakerski and Vaikuntanathan [16].

Definition 1 ($\mathcal{C}$–Evaluation Scheme)**.** Let $\mathcal{C}$ be a set of circuits. A $\mathcal{C}$–evaluation scheme for $\mathcal{C}$ is a tuple of probabilistic polynomial–time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ such that:

$\mathsf{Gen}(1^\lambda, \alpha)$ is the key generation algorithm. It takes two inputs, security parameter λ and auxiliary input α, and outputs a key triple (pk, sk, evk), where pk is the key used for encryption, sk is the key used for decryption and evk is the key used for evaluation.

$\mathsf{Enc}(pk, m)$ is the encryption algorithm. As input it takes the encryption key pk and a plaintext m. Its output is a ciphertext c.

$\mathsf{Eval}(evk, C, c_1, \ldots, c_n)$ is the evaluation algorithm. It takes as inputs the evaluation key evk, a circuit $C \in \mathcal{C}$ and a tuple of inputs that can be a mix of ciphertexts and previous evaluation results. It produces an evaluation output.

$\mathsf{Dec}(sk, c)$ is the decryption algorithm. It takes as input the decryption key sk and either a ciphertext or an evaluation output and produces a plaintext m.

Here $\mathcal{X}$ denotes the ciphertext space which contains the *fresh ciphertexts* (see equation (1)), $\mathcal{Y}$ denotes the space of evaluation outputs and $\mathcal{Z}$ is the union of both $\mathcal{X}$ and $\mathcal{Y}$. $\mathcal{Z}^*$ contains arbitrary length tuples made up of elements in $\mathcal{Z}$. The key spaces are denoted by $\mathcal{K}_p, \mathcal{K}_s$ and $\mathcal{K}_e$, respectively, for pk, sk and evk. The public key contains a description of the plaintext and ciphertext spaces. The input to the key generation algorithm Gen is given in unary notation, i.e., 1^λ. Gen may also take another optional input α from the space $\mathcal{A}$, this is the auxiliary input and will become clear in Remark 3. Finally, $\mathcal{C}$ is the set of *permitted circuits*, i.e. all the circuits which the scheme can evaluate.

With these spaces defined, the domain and range of the algorithms are given by

$$\begin{aligned}
\mathsf{Gen}: &\quad \mathbb{N} \times \mathcal{A} \to \mathcal{K}_p \times \mathcal{K}_s \times \mathcal{K}_e \\
\mathsf{Enc}: &\quad \mathcal{K}_p \times \mathcal{P} \to \mathcal{X} \\
\mathsf{Dec}: &\quad \mathcal{K}_s \times \mathcal{Z} \to \mathcal{P} \\
\mathsf{Eval}: &\quad \mathcal{K}_e \times \mathcal{C} \times \mathcal{Z}^* \to \mathcal{Y}
\end{aligned}$$

where $\mathcal{X} \cup \mathcal{Y} = \mathcal{Z}$ and $\mathcal{A}$ is an auxiliary space. Note that in general the evaluation space can be disjoint from the ciphertext space.

Throughout this paper, we treat the ciphertext space $\mathcal{X}$ as the image of encryption, and the evaluation space $\mathcal{Y}$ as the image of evaluation. Therefore $\mathcal{Z}$ cannot contain an element that is not a possible output of the encryption algorithm or the evaluation algorithm. Formally,

$$\mathcal{X} = \{c \mid \Pr[\mathsf{Enc}(pk, m) = c] > 0, m \in \mathcal{P}\} \tag{1}$$

and

$$\mathcal{Y} = \{z \mid \Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) = z] > 0, c_i \in \mathcal{Z}, C \in \mathcal{C}\}.$$

Notably, the evaluation key is often also part of the public key. By defining the scheme this way, with a separate evaluation key, we are not forbidding $pk = evk$ but asserting that it is not strictly necessary. Separate pk and evk is becoming a standard definition [16, § 3.1].

Remark 1 (Ciphertext decryption). Brakerski and Vaikuntanathan [17] mention that running the decryption algorithm on an output of the encryption algorithm is not strictly necessary: "…we do not require that the ciphertexts c_i are decryptable themselves, only that they become decryptable after homomorphic evaluation." They point out that one can always evaluate the encrypted ciphertext with a *blank circuit* (essentially a circuit computing the function $f(x) = x$) before decryption, thus simplifying the allowed inputs to the decryption algorithm. From now on, we allow the decryption of fresh ciphertexts, as this seems a more natural approach and applies to most known FHE schemes. The decryption algorithm can operate on ciphertexts or evaluations (take values from both the ciphertext space and the evaluation space). This choice removes the need for a blank circuit. In general though, this distinction is not necessary, especially when the evaluation space and the ciphertext space are the same.

3.1 Attributes

Here we present the attributes of homomorphic encryption schemes. On the one hand, we need things like correctness to even call this an encryption scheme, and on the other hand we define attributes like compactness and circuit privacy which exclude trivial solutions to the problem of homomorphic encryption.

Definition 2 (Correct Decryption)**.** A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is said to *correctly decrypt* if for all $m \in \mathcal{P}$,

$$\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1,$$

where sk and pk are outputs of $\mathsf{Gen}(1^\lambda, \alpha)$.

This means that we must be able to decrypt a ciphertext to the correct plaintext, without error.

Definition 3 (Correct Evaluation, [16, Def. 3.3])**.** A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) *correctly evaluates* all circuits in $\mathcal{C}$ if for all $c_i \in \mathcal{X}$, where $m_i \leftarrow \mathsf{Dec}(sk, c_i)$, for every $C \in \mathcal{C}$, and some negligible function ϵ,

$$\Pr[\mathsf{Dec}(sk, \mathsf{Eval}(evk, C, c_1, \dots, c_n)) = C(m_1, \dots, m_n)] = 1 - \varepsilon(\lambda)$$

where sk, pk and evk are outputs of $\mathsf{Gen}(1^\lambda, \alpha)$.

This means that with overwhelming probability, decryption of the homomorphic evaluation of a permitted circuit yields the correct result. Note that for Definition 2 and 3 we are intentionally restricting to $\mathcal{X}$ and not to $\mathcal{Y}$. This is developed further in Section 3.3.

From now on, we say a $\mathcal{C}$–evaluation scheme is *correct* if it has the properties of both correct evaluation and correct decryption.

Definition 4 (Compactness [62, Def. 3])**.** A $\mathcal{C}$–evaluation scheme is *compact* if there is a polynomial p, such that for any key-triple (sk, pk, evk) output by $\mathsf{Gen}(1^\lambda, \alpha)$, any circuit $C \in \mathcal{C}$ and all ciphertexts $c_i \in \mathcal{X}$, the size of the output $\mathsf{Eval}(evk, C, c_1, \ldots, c_n)$ is not more than $p(\lambda)$ bits, independent of the size of the circuit.

This means that the ciphertext size does not grow much through homomorphic operations and the output length only depends on the security parameter. This also rules out a trivial homomorphic scheme where the evaluation algorithm is the identity function (that is, it outputs $(C, c_1, \ldots, c_n)$), and the decryption function is defined to decrypt the input ciphertexts $c_1, \ldots, c_n$, apply the appropriate function to the corresponding plaintexts, and output this result [30].

Remark 2 (On compactness)*.* Gentry's original definition was slightly different, which could informally be paraphrased as: *The scheme is compact if there exists a circuit C_D of "reasonable" length that computes the decryption circuit.* This definition relies on the size of the decryption circuit. However, we feel that the first definition, which relies on the length of Eval's output – given intuitively in his work – and used as the definition of compactness in following works [16, 62], provides for a better understanding. We further examine the relationship between these two concepts (and state the latter one formally) in Section 4.1.

In anticipation of following results, we introduce another definition, originally used by Gentry, that groups all of the definitions seen so far in this section [28, Def 2.1.2].

Definition 5 (Compactly Evaluate)**.** A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) *compactly evaluates* all circuits in $\mathcal{C}$ if the scheme is compact and correct.

We now define circuit privacy. One may easily confuse circuit privacy semantically with *circuit obfuscation*, because both seem to keep the circuit secret or private. However, circuit obfuscation deals with the concealing of the circuit. This is important if the used algorithms themselves are valuable and ought to be secret. In contrast, circuit privacy characterizes the distributions of the output of the algorithms Eval and Enc.

Definition 6 (Circuit Privacy [28, Def. 2.1.6])**.** We say that a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is *perfectly/statistically/computationally circuit private* if for any key-triple (sk, pk, evk) output by $\mathsf{Gen}(1^\lambda, \alpha)$, for all circuits $C \in \mathcal{C}$ and all $c_i \in \mathcal{X}$, such that $m_i \leftarrow \mathsf{Dec}(sk, c_i)$, the two distributions on $\mathcal{Z}$

$$D_1 = \mathsf{Eval}(evk, C, c_1, \ldots, c_n)$$

and

$$D_2 = \mathsf{Enc}(pk, C(m_1, \ldots, m_n)),$$

both taken over the randomness of each algorithm, are *perfectly*, *statistically* or *computationally indistinguishable*, respectively.

Why this definition implies that the circuit is *private* may not be immediately clear. Essentially, it states that the output from the evaluation of a specific circuit on ciphertexts looks like the output from the encryption of a plaintext value v, generated in this case by the circuit and the corresponding plaintexts. As v is just another plaintext (i.e. $v = C(m_1, m_2, \ldots, m_n)$), it is *difficult* for determine how it was generated (the level of difficulty is hierarchical from perfect to computational).

Circuit privacy has also been known by the name *strongly homomorphic* [19, 56] in the literature, and there still remains a slight point of divergence within the community on the accurate definition of circuit privacy. Whilst we keep the original definition as given by Gentry [28], a slightly weaker notion exists that is similar, namely *function privacy.* The important difference is that function privacy only requires that evaluating different circuits on encrypted data produces distributions that are statistically close, computationally close or identical. Circuit privacy, on the other hand, requires that these distributions are the same as those of fresh ciphertexts.[1]

3.2 Classifications

Not all homomorphic schemes have the same properties. This part of the paper examines definitions that allow us to classify and distinguish

[1]We would like to thank Shai Halevi for bringing this issue to our attention.

between different types of schemes, depending on what circuits they can evaluate.

Definition 7 (Somewhat Homomorphic)**.** If a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) also satisfies correct decryption and correct evaluation, then it is called a *somewhat homomorphic encryption* scheme (SHE).

There is no requirement for compactness, so the ciphertexts can increase substantially in length with each homomorphic operation. Also, the set $\mathcal{C}$ of permitted circuits consists of *some* circuits; there is no requirement here as to which circuits this must include.

Definition 8 (Levelled Homomorphic [16, Def. 3.6])**.** A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is called a *levelled homomorphic scheme* if it takes an auxiliary input $\alpha = d$ to Gen which specifies the maximum depth of circuits that can be evaluated. Further requirements are correctness, compactness and that the *length of the evaluation output* does not depend on d.

Other than circuit depth, there is no restriction on $\mathcal{C}$. If we require that $\mathcal{C}$ is the set of all binary circuits with depth at most d, the scheme is called *levelled fully homomorphic.*

The difference between somewhat homomorphic and levelled homomorphic schemes is a potential point of confusion. The depth of circuits which a somewhat homomorphic encryption can handle can be increased through parameter choice – this usually means that the ciphertext size will increase with the depth of the circuits allowed. For a levelled homomorphic encryption scheme, the maximum depth is an input parameter and the length of the ciphertext does not depend on it.

Remark 3. The parameter α was introduced in Definition 1 specifically to allow specifying the maximum depth of circuit that can be evaluated. Thus, when we later assume that α is polynomial in λ, this is justified because in all existing schemes $\alpha = d$, a constant. However, we aim to work with the most general framework possible, so we also allow cases where α might have a different functionality and be substantially larger.

Definition 9 (Fully Homomorphic [16, Def. 3.5]). A *fully homomorphic encryption scheme* is a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) that is compact, correct and where $\mathcal{C}$ is the set of all circuits.

This definition means that the scheme can evaluate any circuit of arbitrary size, which does not need to be known when setting the parameters.

3.3 Evaluating in Stages

Sometimes we want to compute a result in two or more stages, where the results from one stage could be used as input for a later stage. In this case, we want to evaluate on ciphertexts that were output by Eval in addition to ciphertexts that were output by Enc.

The definition of correct evaluation (Definition 3) only guarantees that the algorithm Eval works when its input ciphertexts are in $\mathcal{X}$, the set of *fresh ciphertexts* that can be output by the Enc algorithm. We want to study under which conditions we can hope that the evaluation algorithm will work when given evaluation outputs (we present implications in Section 4.2).

Evaluation in stages is known as *i-hop homomorphic encryption* ([56, Section 2.2], [34, Section 1.4]), where i is either an integer or can be replaced by “multi”,“poly” or ∞ (see Definitions 12, 13 and 14 below). We now define *computation in stages* (also *staged computation*).

A computation $\mathbf{C}_{i,n}$ in i stages of width n is defined by a set of circuits $\{C_{k\ell}\}$ indexed by $1 \leq k \leq i, 1 \leq \ell \leq n$, where $C_{k\ell}$ has kn inputs. Given initial plaintexts $m_{01}, m_{02}, \ldots, m_{0n}$, we compute

$$m_{k\ell} = C_{k\ell}(m_{01}, m_{02}, \ldots, m_{0n}, \ldots, m_{k-1,1}, \ldots, m_{k-1,n})$$

for $1 \leq k \leq i$ and $1 \leq \ell \leq n$. The output of the staged computation after Eval and Dec is $m_{i1}, m_{i2}, \ldots, m_{in}$. Denoting the initial plaintexts by $\vec{m}_0$ and the output plaintexts by $\vec{m}_i$, we introduce the natural notation $\vec{m}_i = \mathbf{C}_{i,n}(\vec{m}_0)$.

Let (pk, evk, sk) be a key triple output by Gen, and let c_{01}, c_{02}, …, c_{0n} be a sequence of ciphertexts from $\mathcal{X}$. Compute the ciphertexts $\{c_{k\ell}\}$ for $1 \leq k \leq i, 1 \leq \ell \leq n$ recursively by

$$c_{k\ell} = \mathsf{Eval}(evk, C_{k\ell}, c_{01}, \ldots, c_{0n}, \ldots, c_{k-1,1}, \ldots, c_{k-1,n}).$$

The output of the encrypted staged computation is the sequence of ciphertexts $c_{i1}, c_{i2}, \dots, c_{in}$. Denoting the fresh ciphertexts by $\vec{c}_0$ and the ouput ciphertexts by $\vec{c}_i$, we introduce the natural notation of Eval having multiple outputs

$$\vec{c}_i = \mathsf{Eval}(evk, \mathbf{C}_{i,n}, \vec{c}_0).$$

Remark 4. A slightly narrower view [34, 56] of computation in stages is that the only ciphertext output by one stage can be input for the next stage. Since it is normally considered possible to apply the identity function to a ciphertext, this formulation is usually no weaker than our more general view.

Let $\vec{c} = (c_1, \dots, c_n)$ and $\vec{m} = (m_1, \dots, m_n)$ be tuples of ciphertexts and plaintexts respectively, such that under a secret key sk, $\mathsf{Dec}(sk, c_k) = m_k$ for $1 \le k \le n$. We then introduce the natural notation

$$\vec{m} = \mathsf{Dec}(sk, \vec{c}).$$

Definition 10 (i-Hop Correctness)**.** Let pk, evk, sk be keys output by $\mathsf{Gen}(1^\lambda)$, and let $\mathbf{C}_{i,n} = \{C_{k\ell}\}$ be any staged computation where n is polynomial in λ and $\vec{c}_0 = (c_{01}, \dots, c_{0n})$ in $\mathcal{X}^n$. A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is *i-hop correct* if

$$\Pr[\mathsf{Dec}(sk, \mathsf{Eval}(evk, \mathbf{C}_{i,n}, \vec{c}_0) = \mathbf{C}_{i,n}(\mathsf{Dec}(sk, \vec{c}_0))] = 1 - \varepsilon(\lambda),$$

where ε is a negligible function and the probability is taken over the coins of the Eval algorithm invocations.

While previous definitions of i-hop (and multi-hop, see below) implicitly use a construction like i-hop correctness, it was never clearly defined in the literature. Additionally, we allow Eval to fail, although only with negligible probability. In Figure 2 staged evaluation is illustrated for $i = 2$ and $n = 2$, where each invocation of Eval outputs n results, but not all of them must be used in the next iteration of Eval which is indicated by a dotted arrow. Furthermore, Eval may use $i \cdot n = 4$ different circuits in the whole process.

Now we have defined i-hop correctness, we can define i-hop, multi-hop, poly-hop and ∞-hop. Similar definitions of i-hop and multi-hop can be found in the work of Gentry et al. [34, Section 1.4] and

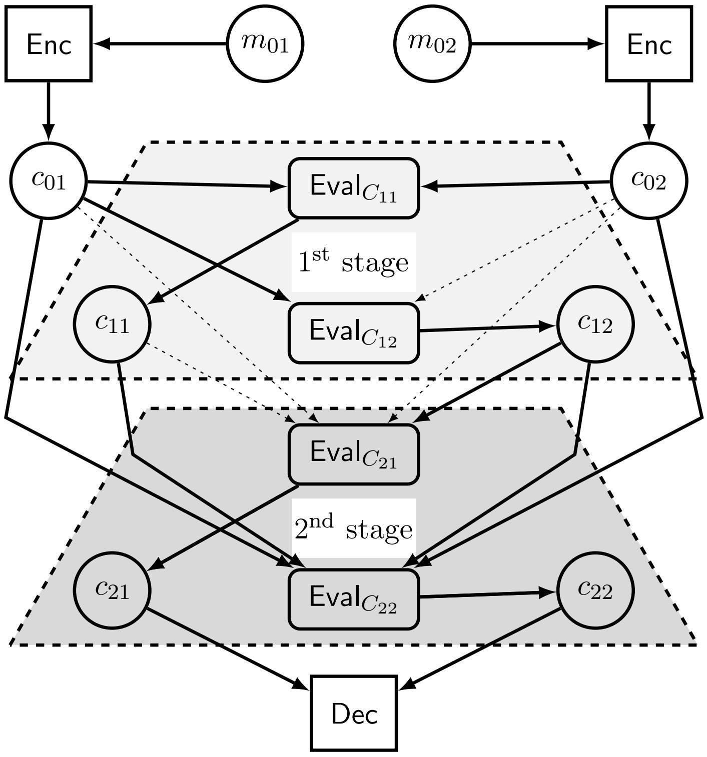

Figure 2: Staged evaluation for $i = 2$ stages and $n = 2$ in- and outputs. After encrypting, a subset of the resulting (fresh) ciphertexts is used for the Eval algorithms as input successively. This is the appropriate diagram for the formula $\vec{c}_2 = (c_{21}, c_{22}) \leftarrow \mathsf{Eval}(evk, \mathbf{C}_{2,2}, \vec{c}_0) = \mathsf{Eval}\,(evk, \mathbf{C}_{2,2}, (\mathsf{Enc}(pk, m_{01}), \mathsf{Enc}(pk, m_{02})))$. A circuit may not use all inputs, the dotted arrows represent the ignored inputs. The light shaded shapes belong to 1-hop and the darker shaded shapes to 2-hop.

Rothblum [56]. The main difference is that we allow inputs from each of the predecessor Eval algorithms as well as fresh ciphertexts. The previous definitions allow only the output of the direct predecessor Eval invocations as input.

Definition 11 (i-Hop, [34,56]). Let $i \in \mathbb{N}$. We say that a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is i-hop if j-hop correctness holds for all j with $1 \leq j \leq i$.

Remark 5 (FHE and i-Hop). The relation between fully homomorphic and i-hop is another possible source of confusion. One may expect that if it is possible to evaluate an arbitrary circuit (fully homomorphic encryption), it would be possible to execute arbitrarily many circuits consecutively. This, however, is not the case. Outputs of Eval might look very different from *fresh* ciphertexts and there is no guarantee that they form valid inputs to Eval. For example, assume we have a 1-hop fully homomorphic encryption scheme, a circuit C that takes as input $c_1, \ldots, c_n$ and outputs $c'_1, \ldots, c'_v$, and a circuit C' that takes as input $c_1, \ldots, c_v$ and outputs $c'_1, \ldots, c'_w$. If we run $\mathsf{Eval}(evk, C' \circ C, c_1, \ldots, c_n)$ (where $C' \circ C$ is the concatenation of the two circuits), this is certainly a valid operation, because we are evaluating *one* circuit (not to be confused with staged computation). However, if we first run $\mathsf{Eval}(evk, C, c_1, \ldots, c_n)$ to obtain $c'_1, \ldots, c'_v$, then attempting to run $\mathsf{Eval}(evk, C', c'_1, \ldots, c'_v)$, is not supported by the 1-hop scheme, because $c'_1, \ldots, c'_v$ will not be valid inputs. This observation is important for applications where two separate entities compute on some encrypted data, and the second entity evaluates the output of the first. In this scenario, the second entity does not have access to the fresh ciphertexts and is forced to operate on the output of the evaluation given by the first. This would be impossible with a 1-hop scheme.

Instead of being bounded by an integer, the hops may be bounded by some polynomial depending on λ which leads us to the next definition.

Definition 12 (Multi-Hop, [34, Sec. 1.4], [56]). Let p be some polynomial. We say that a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is *multi-hop* if j-hop correctness holds for all j with $1 \leq j \leq p(\lambda)$.

Definition 13 (Poly-Hop)**.** Let p be some polynomial and let $\alpha \in \mathcal{A}$. We say that a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is *poly-hop* if j-hop correctness holds for all j with $1 \leq j \leq p(\lambda, \alpha)$.

As far as we are aware, this is the first proposed definition of poly-hop. It seems to be a natural extension to the existing definitions of i-hop and multi-hop. This way, the auxiliary input may influence the number of keys and therefore the number of possible evaluations.

Definition 14 (∞-Hop)**.** We say that a $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is ∞*-hop* if j-hop correctness holds for all j.

Again, as far as we know, ∞-hop was not yet mentioned in the literature. Like poly-hop, ∞-hop is a natural extension of the existing definitions. It allows an unlimited number of hops. Hence, there are direct implications for FHE. See Section 4 for further discussions on this topic.

Remark 6 (Hops and classifications). We are not requiring a fully homomorphic scheme for the notion of hop correctness – the definition is applicable to any homomorphic scheme of Section 3.2 (somewhat homomorphic, levelled homomorphic, levelled fully homomorphic and fully homomorphic).

Remark 7 (Poly-hop vs. multi-hop). What is the difference between poly-hop and multi-hop? If you need a security parameter to output a public key, then any bound on the security parameter is also a bound on the public key. This makes sense if a user cannot increase the size of the public key independently (or to some degree of independence) of the security parameter. This, however, is allowed by Definition 1, as some form of auxiliary input to the key generation algorithm. There is nothing stopping an auxiliary input defining the public key size, independent of the security parameter. This relates to poly-hop because in practice, levelled homomorphic schemes can be achieved by having several key pairs with which one can perform the recrypt operation (see Section 5.1 for a more detailed explanation). This means the public key size increases multiplicatively by this number of keys. Of course, if the number of key pairs is polynomial in λ (or more generally, if α is polynomial in λ), poly-hop and multi-hop are the same.

4 Implications

We now detail the implications of the definitions given in the previous section. First we return to the issue of compactness and its two, seemingly separate, definitions.

4.1 Consolidating compactness

As noted, there is a difference between Definition 4 and the definition of compactness originally given by Gentry [28]. This section is devoted to reconciliation of these two definitions of compactness. The definition presented by Gentry is given below, and also the definition of compact evaluation, which is important for Lemma 1.

Remark 8. Here, many results only hold if the auxiliary input to the key generation algorithm, α, is polynomially bounded by λ. For all meaningful applications (and for all homomorphic encryption schemes known to date) this appears to be the case, but we cannot formally guarantee it (see also Remark 7). Thus, we state explicitly when we need this requirement for a statement to hold.

Definition 15 (G-Compactness [28, Def. 2.1.2]). A $\mathcal{C}$–evaluation scheme is *G-compact* if there is a polynomial f such that, for every value of the security parameter λ, the decryption algorithm can be expressed as a circuit C_D of size at most $f(\lambda)$.

Definition 16 (G-Compact Evaluation [28, Def. 2.1.3]). A $\mathcal{C}$–evaluation scheme (Gen, Enc, Eval, Dec) is said to *G-compactly evaluate* all permitted circuits in $\mathcal{C}$ if the scheme is G-compact and is correct for all permitted circuits.

Recall that the size of a circuit is just the total number of gates it has. Picturing a circuit as a directed graph, this is the sum of all the vertices minus the sum of the input vertices [44, §1.2, p.13]. It is not immediately clear that this definition of compactness is the same as the definition we gave earlier.

Theorem 1. *Let α be bounded by a polynomial in λ. A $\mathcal{C}$–evaluation scheme (*Gen*,* Enc*,* Eval*,* Dec*) G-compactly evaluates $\mathcal{C}$ if and only if the scheme compactly evaluates $\mathcal{C}$.*

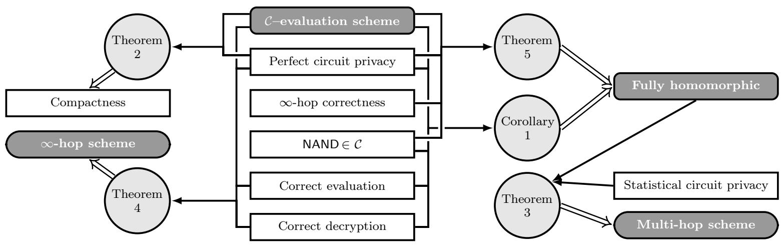

Figure 3: Overview of Theorem 2 through Theorem 5 and Corollary 1 and their dependencies. White rectangles are definitions, gray rounded rectangles are classifications, black shapes are hop schemes and light gray circles are theorems. Simple arrows pointing towards a theorem or corollary represent the requirements for a theoreom/corollary while double arrows represent the implication.

The proof can be found in Appendix A.

Theorem 2. *A $\mathcal{C}$–evaluation scheme (*Gen, Enc, Eval, Dec*) with perfect circuit privacy implies compactness if α is polynomially bounded in λ.*

The proof can be found in Appendix B.

4.2 FHE and Hop Results

We now present results relating to FHE schemes and hop correctness, assuming that α is polynomially bounded by λ. Figure 3 shows a comprehensive overview of these results, also including Theorem 2. The diagram is formulated as in Figure 1, where there are two different kind of arrows. The simple black arrow is a requirement, so for example Theorem 4 has the two requirements of perfect circuit privacy and fully homomorphic. All of the requirements are needed for each theorem. The second arrow type is double-lined, which represents the implication of the theorem. As for Theorem 4, the given requirements yield an infinity-hop scheme.

Theorem 3. *A fully homomorphic encryption scheme (*Gen, Enc, Eval, Dec*) that is statistically circuit private is multi-hop.*

The proof can be found in Appendix C. We now investigate the relationship between fully homomorphic and i-hop, noting what properties a somewhat homomorphic encryption scheme needs to be be *fully.* First, we examine under which conditions a fully homomorphic scheme allows infinite stages of computation (∞-hop):

Theorem 4. *A somewhat homomorphic encryption scheme (*Gen*,* Enc*,* Eval*,* Dec*) which is perfect circuit private is* ∞*-hop.*

Proof. Since the scheme has perfect circuit privacy, the outputs of Eval are distributed identically to fresh encryptions. This means that they are of exactly the same form ($\mathcal{X} = \mathcal{Y} = \mathcal{Z}$) and decrypt *correctly.* So they are ciphertexts and constitute a valid input to Eval again. This holds no matter how often we apply evaluate, as the output is always of the same form as the input. □

The following theorem considers the other direction – when an ∞-hop scheme is fully homomorphic.

Theorem 5. *A somewhat homomorphic encryption scheme (*Gen*,* Enc*,* Eval*,* Dec*) with* NAND *in* $\mathcal{C}$ *that is perfect circuit private and* ∞*-hop is fully homomorphic.*

Proof. Since the scheme has perfect circuit privacy, it has compactness by Theorem 2. Thus, all we need to show is that the scheme can evaluate any circuit. Assume that this is not the case, so there exists a circuit C which the scheme cannot correctly evaluate. But then we can express C as a circuit composed only of NAND-gates. Since the scheme is ∞-hop and $\mathsf{NAND} \in \mathcal{C}$, we can correctly evaluate each NAND-gate on the corresponding input, no matter what level of evaluation iteration this input has. Thus, we have found a way to correctly evaluate this circuit with the scheme, meaning $C \in \mathcal{C}$. This is a contradiction to our assumption and thus shows that the scheme is fully homomorphic. □

Corollary 1. *A somewhat homomorphic encryption scheme (*Gen*,* Enc*,* Eval*,* Dec*) that is perfect circuit private and has* $\mathsf{NAND} \in \mathcal{C}$ *is fully homomorphic.*

Proof. By Theorem 2 and Theorem 4, then Theorem 5. □

5 Existing schemes

In this section we briefly survey existing fully homomorphic encryption schemes. Only limited steps towards full homomorphism were made before Gentry's breakthrough. Fellows and Koblitz's Polly Cracker [24] is fully homomorphic except that it lacks compactness. It was anyway not intended to be practical. Albrecht et al. [2] show that nearly all SHE schemes are variants of Polly Cracker. The Boneh-Goh-Nissim scheme [11] is compact, but can only handle a single multiplication. Obviously these schemes are not fully homomorphic by Definition 9, or by contemporary treatments of FHE [28].

Table 1 lists a number of prominent fully homomorphic encryption schemes, starting with Gentry's 2009 scheme. For each scheme the table mentions the underlying computational assumption (described below) and an indication of the asymptotic or concrete runtime where available.

5.1 Bootstrapping and Alternatives

A key concept in the development of the first fully homomorphic scheme is Gentry's *bootstrapping* technique. Schemes based on Gentry's blueprint are noise-based, which means that the plaintext is hidden by noise which can be removed by decryption. However, this noise increases with each homomorphic evaluation, and once it exceeds a certain threshold, decryption will fail.

To overcome this problem, Gentry introduced the notion of *re-cryption* which works by encrypting a ciphertext anew (so that it becomes doubly encrypted) and then removing the inner encryption by homomorphically evaluating the doubly encrypted plaintext and the encrypted decryption key using the decryption circuit. As long as the evaluation algorithm can handle the decryption process plus one more gate, progress can be made in evaluating the circuit of interest.

Definition 17 (Bootstrappable)**.** A $\mathcal{C}$–evaluation scheme is called *bootstrappable* if it is able to homomorphically evaluate its own decryption circuit plus one additional NAND gate.

This informal definition essentially captures the more precise one in

Scheme	**Under-lying Prob-lems**	**Asymptotic Runtime**	**Concrete Runtime**
Gentry: A Fully Homomorphic Encryption Scheme [28]	BDDP & SSSP	$\mathcal{O}(\lambda^{3.5})$ per gate for ciphertext refreshing [60]	-
van Dijk, Gentry, Halevi, Vaikuntanathan: FHE over the Integers [62]	AGCD & SSSP	Public key size: $\mathcal{O}(\lambda^{10})$, no gate cost given	-
Coron, Naccache, Tibouchi: Public Key Compression and Modulus Switching for FHE over the Integers [20]	DAGCD & SSSP	Public key size: $\mathcal{O}(\lambda^5 \log(\lambda))$, no gate cost given	Recryption takes about 11 minutes.
Brakerski, Vaikuntanathan: Efficient FHE from (standard) LWE [16]	DLWE	Evaluation key size: $\mathcal{O}(\lambda^{2C} \log(\lambda))$ where C is a very large parameter that ensures bootstrappability.	-
Brakerski, Vaikuntanathan: FHE from Ring-LWE and Security for Key Dependent Messages [17]	PLWE	Very cheap key generation, unknown for bootstrapping	-
Brakerski, Gentry, Vaikuntanathan: FHE without Bootstrapping [15]	RLWE	Per-gate computation overhead $\mathcal{O}(\log \lambda \cdot \lambda \cdot d^3)$ (where d is the depth of the circuit) without bootstrapping, $\mathcal{O}(\log \lambda \cdot \lambda^2)$ with bootstrapping.	36 hours for AES encryption on supercomputer [32]. Updated impl. [33] runs AES-128 enc with 2 s/block (3 GB RAM). With bootstrapping 6 sec/block (3.7 GB RAM). Vectors of 1024 elements from GF(2^{16}) were recrypted in 5.5 min at sec. level ≈ 76, single CPU core [40].
Smart, Vercauteren: FHE with Relatively Small Key and Ciphertext Sizes [58]	PCP & SSSP	Key gen. is $\mathcal{O}(\log n \cdot n^{2.5})$ where n is the lattice dimension [31]	Key gen. took several hours even for small parameters which do not deliver a fully homomorphic scheme, for larger parameters the keys could not be generated.
Rohloff, Cousins: A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU [55]	SVP & RLWE	-	Recryption at 275 seconds on 20 cores with 64-bit security.
Gentry, Halevi: Implementing Gentry's Fully-Homomorphic Encryption Scheme [31]	SVP & BDD	Key generation is $\mathcal{O}(\log n \cdot n^{1.5})$ where n is the dimension of the lattice	Bootstrapping: From 30 s for small setting, to 30 min for large setting.

Table 1: Selected FHE schemes with underlying security problems. Authors often provide different runtime analyses for schemes, so figures may not be comparable. The concrete experiments have been run by the respective authors on different hardware, but still give an indication. Blank cells are, to the best of our knowledge, not publicly known.

the literature [28]. Now the question is: Does publishing an encryption of the secret key under its own public key impair security?

If we assume it is safe to publish the encryption of the secret key under its corresponding public key, we achieve fully homomorphic encryption and even i-hop [17, 28, 58, 62]. This assumption is called *circular security.* However, if circular security does not hold then one possibility is to use a chain of public key/secret key pairs, where the secret key is always encrypted under the next public key. This allows suitable somewhat homomorphic schemes to become levelled homomorphic, where the level depends on the number of key pairs.

An alternative way to achieve homomorphic encryption is due to Brakerski et al. [15]. The challenge is still how to manage the noise, but this time it is achieved by reducing the modulus of the ciphertext space along with the noise. A security parameter that dictates how small the modulus can be gives a bound on the number of levels. This line of work yields native levelled homomorphic schemes [15, 16, 55]. However, authors usually note that one can apply bootstrapping as an optimisation, as well as a means to get to a fully homomorphic i-hop scheme, again assuming circular security.

5.2 Security Assumptions

We now give a brief overview of the problems that existing schemes are based on. The formal definitions are often taken directly from the corresponding papers, but simplified by omitting parameters whenever possible. Many of these problems were studied by Ajtai [1].

Most of the problems below have reductions to either the *Shortest Vector Problem* (SVP) or the *Closest Vector Problem* (CVP), which informally requires a player to provide a shortest possible vector in the lattice and the vector closest to a point respectively. These problems have decisional variants as well. For instance, $GapSVP_\gamma$ is the problem of proving that there is a vector shorter than 1, or that all vectors are longer than γ. In addition, we mention the *shortest independent vector problem* (SIVP), which is essentially to compute a lattice basis with only short vectors.

We first consider the Learning With Errors problem family. All of these problems exist in both search and decision variants, just like the computational Diffie-Hellman and decisional Diffie-Hellman problems.

LWE: [53] The *Learning With Errors problem* is a generalization of the “learning parity with noise” problem. For an integer $q = q(n)$ and an error distibution $\chi = \chi(n)$ on $\mathbb{Z}_q$, define the distribution $A_{s,\chi}$ for some $s \in \mathbb{Z}_q^n$ as the distribution obtained by choosing a vector $a \leftarrow \mathbb{Z}_q^n$ uniformly at random and a noise term $e \leftarrow \chi$ and outputting $(a, \langle a, s\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Then the (n, m, q, χ)-LWE problem is to output s, given m independent samples from $A_{s,\chi}$.

The decisional version is to distinguish between m samples chosen according to $A_{s,\chi}$ for some uniformly random s and m samples from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

PLWE: [16] The *Polynomial LWE problem* is a variant of the Ring Learning With Errors Problem (RLWE) and is closely related to DLWE. For a parameter λ, let $f(x) \in \mathbb{Z}[x]$ be a polynomial of degree $n = n(\lambda)$, and let $q = q(\lambda) \in \mathbb{Z}$ be a prime. Consider the rings $R = \mathbb{Z}[x]/\langle f(x)\rangle$ and $R_q = R/qR$, and let χ denote the Gaussian distribution over R. Then the $\mathrm{PLWE}_{f,q,\chi}$ assumption states that for all λ and for all $l = \mathrm{poly}(\lambda)$, the two distributions $D_1 = \{(a_i, a_i \cdot s + e_i)\}$ and $D_2 = \{(a_i, u_i)\}, i = 1, \ldots, l$, are computationally indistinguishable. Here, s, a_i and u_i are uniform in R_q and the e_i are sampled from χ.

RLWE: [49] This is the same problem as PLWE where $f(x) = x^d + 1$ and $d = d(\lambda)$ is a power of 2. There also exists a variant with augmented data in the error term, named *Augmented LWE*. For certain parameters, A-LWE is as hard as LWE [8].

There exists a quantum reduction from LWE to SVP and SIVP by Regev [53]. It is also known that the search and decision variants are equally hard [47].

SSSP: [28] This is called the *Sparse Subset Sum Problem.* In his original work, Gentry “squashed the decryption circuit”, which reduces the size of the decryption circuit such that it is in the set of circuits that the scheme can homomorphically evaluate. The idea is that the secret key is written as the sum of some elements, and these elements are “hidden” in a much larger set of elements. This large set becomes part of the public key, and the secret key includes an indicator vector of which elements belong to the smaller set (i.e. sum up to the secret key). This gives the adversary information about the secret key, so we must ensure that it cannot be extracted from the public key. SSSP formalizes this requirement. Since all papers that follow Gentry's blueprint use this squashing technique, the SSSP problem appears several times in the table. It is formally defined as follows:

Let S and T be two natural numbers with $S \ll T$, and let q be a prime number. The challenger sets $b \leftarrow \{0, 1\}$. If $b = 0$, it generates a set τ with cardinality $|\tau| = T$ of uniformly random integers in $[-q/2, q/2]$ such that there exists a subset of cardinality S whose elements sum to 0 mod q. If $b = 1$, the set τ is generated without this requirement. The challenge is to guess b.

BDD: The *Bounded Distance Decoding problem* is identical to the *Closest Vector Problem*, except that for BDD, there is a guarantee that the vector t is very close to the lattice. The Closest Vector Problem is a problem from lattice theory that informally asks for the point on a lattice that is closest to a given vector $t \in \mathbb{R}^n$.

There exists quantum reductions proving that GapSVP, BDD and SIVP are equally hard, as well as an equivalence between SVP and CVP [53]. An equivalence between SVIP and BDD, however, remains an open problem [48].

AGCD: [41] The *Approximate Greatest Common Divisor problem* is the task of given near multiples of a number p, to find that number p. Given polynomially many numbers of the form $x_i = q_i \cdot p + r_i$ where r_i is much smaller than $q_i \cdot p$, output p.

The decisional variant includes an an additional integer $z = x + b \cdot \alpha$. Here, x is of the same form as the x_i's, b is either 0 or 1, and α is from an appropriate interval depending on the parameters. The task is to find b.

PCP: The *Polynomial Coset Problem* [58] is a decisional problem that can informally be described as having to decide whether a given value is the evaluation of a small polynomial mod$p$, or randomly sampled from $\mathbb{F}_p$. More formally, we can describe the problem in a challenge scenario:

The challenger first selects $b \leftarrow \{0, 1\}$ randomly and runs the key generation algorithm of the scheme, which outputs a prime p and a value $\alpha \in \mathbb{F}_p$, derived under some constraints which we will not go into here. If $b = 0$, the challenger randomly chooses a polynomial $R(x)$

with coefficients in a certain range and computes $r = R(\alpha) \bmod p$. If $b = 1$, the challenger chooses $r \leftarrow \mathbb{F}_p$ randomly. The problem now is: Given (p, α, r), decide whether $b = 0$ or $b = 1$.

Notably, this problem differs from other problems in that it is defined with respect to a corresponding scheme, making it less natural and harder to explaining in outline. The PCP problem is related to the Ideal Coset Problem as defined by Gentry [28].

5.3 Implementations

It is fair to say that FHE mostly exists on paper. However, there also exist implementations, as suggested in the above table. The foremost among those is Halevi and Shoup's `HElib` [40], which implements the BGV scheme [15] along with optimisations such as ciphertext packing [59], which allows several plaintexts to be encoded in a single ciphertext. In 2014, bootstrapping was introduced to the library [40]. However, at the time of writing the secure Gaussian randomness distribution is not yet implemented, hence there are no security proofs for `HElib`. The library was used by the IBM, Microsoft and Stanford/MIT teams at the 2015 iDASH Secure Genome Analysis Contest [45].

There exists another library called FHEW [22] which is based on the FHE scheme of Ducas and Micciancio [23].

6 Conclusion

In this paper we have simplified and structured the jungle of definitions in the field of homomorphic encryption. We investigated whether existing applications need homomorphic encryption as a solution to their problems, both in theory and in practice. Furthermore, we reviewed the current state of the art and presented it systematically.

There is still much work to be done. Current schemes have some way to go to be practical in daily applications. Thus we can expect continuing focus on making existing schemes more efficient and on constructing new efficient schemes. In fact, given that several applications do not require *fully* homomorphic encryption, an important and promising line of research is to identify applications which would benefit from appropriate homomorphic encryption schemes and afterwards tailor schemes for respective use cases.

Then there is the more theoretical line of work. While a framework for *group* homomorphic encryption schemes has been presented [5] and to some extent for FHE [4], an equivalent result is lacking for fully (or at least somewhat) homomorphic encryption schemes in full generality. As explained in Section 5, all secure schemes which go beyond simple group-homomorphic operations are noise-based and one of the main challenges is to control the noise. In fact this is often the reason why fully homomorphic encryption schemes are considerably less efficient. A unified view on somewhat/fully homomorphic encryption schemes may be very useful in gaining a better understanding of the expected security and on the possible design space.

All in all, the topic of FHE is an interesting and challenging research area with great potential, and there is much to be done. However, if research (specifically the advancement of efficiency) continues at its current pace, we are confident that real-world applications may be right around the corner.

Acknowledgements This work was supported by the German Academic Exchange Service (DAAD), project number 57068907 and the Juniorprofessoren-Programm Baden-Württemberg 2013.

References

[1] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108. ACM, 1996.

[2] Martin R. Albrecht, Jean-Charles Faugère, Pooya Farshim, Gottfried Herold, and Ludovic Perret. Polly cracker, revisited. *Designs, Codes and Cryptography*, pages 1–42, 2015.

[3] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully

homomorphic encryption. In Martijn Stam, editor, *Cryptography and Coding – 14th IMA International Conference, IMACC 2013*, volume 8308 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2013.

[4] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. Shift-type homomorphic encryption and its application to fully homomorphic encryption. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology – AFRICACRYPT 2012*, volume 7374 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2012.

[5] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. Group homomorphic encryption: characterizations, impossibility results, and applications. *Des. Codes Cryptography*, 67(2):209–232, 2013.

[6] Frederik Armknecht and Thorsten Strufe. An efficient distributed privacy-preserving recommendation system. In *The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2011*, pages 65–70. IEEE, 2011.

[7] Sanjeev Arora and Boaz Barak. Computational complexity: A modern approach (draft). Accessed 27 Oct 2014, 2007.

[8] Rachid El Bansarkhani, Özgür Dagdelen, and Johannes A. Buchmann. Augmented learning with errors: The untapped potential of the error term. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security, FC 2015*, volume 8975 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2015.

[9] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[10] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Paterson [51], pages 149–168.

[11] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Ishai [42], pages 253–273.

[13] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[14] Christoph Bösch, Andreas Peter, Pieter H. Hartel, and Willem Jonker. SOFIR: securely outsourced forensic image recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014*, pages 2694–2698. IEEE, 2014.

[15] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.

[16] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE, 2011.

[17] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[18] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Rabin [52], pages 483–501.

[19] Michael Clear, Arthur Hughes, and Hitesh Tewari. Homomorphic encryption with access policies: Characterization and new constructions. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella

Hassanien, editors, *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings*, volume 7918 of *Lecture Notes in Computer Science*, pages 61–87. Springer, 2013.

[20] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

[21] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Safavi-Naini and Canetti [57], pages 643–662.

[22] Léo Ducas and Daniele Micciancio. Fhew: Fastest homomorphic encryption in the west. a fully homomorphic encryption library. `https://github.com/lducas/FHEW`. Accessed 2016-01-18.

[23] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015.

[24] Michael Fellows and Neal Koblitz. Combinatorial cryptosystems galore! In *Finite fields: theory, applications, and algorithms*, volume 168 of *Contemp. Math.*, pages 51–61. Amer. Math. Soc., Providence, RI, 1994.

[25] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[26] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin [52], pages 465–482.

[27] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 2013.

[28] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[29] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 169–178. ACM, 2009.

[30] Craig Gentry. Computing on the edge of chaos: Structure and randomness in encrypted computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:106, 2014.

[31] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Paterson [51], pages 129–148.

[32] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [57], pages 850–867.

[33] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. *IACR Cryptology ePrint Archive*, 2012:99, 2012.

[34] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable Yao circuits. In Rabin [52], pages 155–172.

[35] Henri Gilbert, editor. *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[36] Oded Goldreich. Introduction to complexity theory, online lecture notes. Accessed 27 Oct 2014, 1999.

[37] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2013.

[38] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.

[39] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 469–477. ACM, 2015.

[40] Shai Halevi and Victor Shoup. Bootstrapping for helib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.

[41] Nick Howgrave-Graham. Approximate integer common divisors. In Joseph H. Silverman, editor, *Cryptography and Lattices, International Conference, CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2001.

[42] Yuval Ishai, editor. *Theory of Cryptography – 8th Theory of Cryptography Conference, TCC*, volume 6597 of *Lecture Notes in Computer Science*. Springer, 2011.

[43] Arjan Jeckmans, Andreas Peter, and Pieter H. Hartel. Efficient privacy-enhanced familiarity-based recommender system. In Jason Crampton et al., editors, *Computer Security – ESORICS 2013*,

volume 8134 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2013.

[44] Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[45] Kristin Lauter. Practical applications of homomorphic encryption, 2015.

[46] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 1219–1234. ACM, 2012.

[47] Vadim Lyubashevsky. Search to decision reduction for the learning with errors over rings problem. In *2011 IEEE Information Theory Workshop, ITW 2011*, pages 410–414. IEEE, 2011.

[48] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2009.

[49] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [35], pages 1–23.

[50] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW*, pages 113–124. ACM, 2011.

[51] Kenneth G. Paterson, editor. *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.

[52] Tal Rabin, editor. *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[53] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.

[54] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[55] Kurt Rohloff and David Bruce Cousins. A scalable implementation of fully homomorphic encryption built on NTRU. In Rainer Böhme et al., editors, *Financial Cryptography and Data Security – FC 2014 Workshops, BITCOIN and WAHC 2014*, volume 8438 of *Lecture Notes in Computer Science*, pages 221–234. Springer, 2014.

[56] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Ishai [42], pages 219–234.

[57] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012.

[58] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[59] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.

[60] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.

[61] Top Threats Working Group. The notorious nine: Cloud computing top threats in 2013. Report, Cloud Security Alliance, February 2013. Accessed 7 Apr 2015.

[62] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [35], pages 24–43.

[63] Marten van Dijk and Ari Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In Wietse Venema, editor, *5th USENIX Workshop on Hot Topics in Security, HotSec'10*. USENIX Association, 2010.

[64] Zhiqiang Yang et al. Privacy-preserving classification of customer data without loss of accuracy. In Hillol Kargupta et al., editors, *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, pages 92–102. SIAM, 2005.

A Proof of Theorem 1

We prove Theorem 1 with the following lemmas.

Lemma 1. *A scheme with G-compactness and correctness is compact.*

This lemma holds only when we assume correct decryption and correct evaluation. To understand why we must have correctness, consider G-compactness without it. In this case, we have a decryption circuit of polynomial size but it does not have to *actually* decrypt. This means we could form a trivial decryption circuit that takes any Eval output, trims the bits to a specified maximum size and runs the decryption circuit on that. Obviously, this would not bound the output of Eval, so G-compactness without correctness does not imply compactness.

Proof. Given the security parameter λ, we can find a decryption circuit with size at most $p(\lambda)$. As each gate takes at most 2 inputs, the bound on the number of inputs to the circuit is less or equal to $2p(\lambda) = q(\lambda)$

where q is a polynomial.[2] This bound on the input length means that the output from the Eval algorithm must output a ciphertext less than $q(\lambda)$ bits in length. Otherwise, we would be unable to run the decryption algorithm *correctly*, contradicting our assumption. Noting that $q(\lambda)$ is a bound independent of the size of the circuit being evaluated, this gives us compactness. □

Lemma 2. *A scheme with compactness and correctness is G-compact when α is polynomially bounded by λ.*

Proving this lemma relies on results from complexity theory, allowing us to construct a polynomially sized circuit from a polynomial algorithm. For details on the relationship between algorithm running time and circuit size see Gentry [30, §2.1 Circuits], Goldreich [36, Ch. 2, Ch. 20 §1.2] or Arora & Barak [7, Ch. 6].

Proof. All outputs of Eval are no more than $b(\lambda)$ bits long, meaning input to the decryption algorithm after evaluation is at most $b(\lambda)$ bits in length. Eval is a poly-time algorithm, and so its running time is bounded polynomially by the length of its input.

The algorithm Enc also has a polynomial bound on the length of its output. Firstly note, the number of outputs from an algorithm cannot be greater than the algorithm running time. Next, we know Enc is a poly-time algorithm taking two inputs, pk and $m \in \mathcal{P}$, where pk is an output from Gen, a poly-time algorithm taking the input parameters λ and $\alpha = \alpha(\lambda)$. $\mathcal{P}$ is described in pk. Now, the running time of Enc is bounded by some polynomial on the input parameters, themselves bounded polynomially by the input parameter λ. Thus, the output from Enc is bounded by a polynomial, $a(\lambda)$ say. Since sk is again an output of Gen, we can also bound its length by a polynomial $c(\lambda)$. Taking $d(\lambda) = \max\{a(\lambda), b(\lambda)\}$, which is also a polynomial in λ, we can define $v = d + c$. Clearly v is a polynomial in λ and bounds the size of the inputs to Dec. Thus, Dec has a running time of $p(v(\lambda))$ for some polynomial p.

[2]Any circuit where each gate takes n inputs (for some bounded n) can be constructed as a circuit with gates taking at maximum two inputs, with only a constant factor increase in size. [36, §1.2]

Using the results cited above, we can now construct a decryption circuit that replicates the algorithm, where the circuit size will be some polynomial q on the running time of the algorithm. Thus the size of the decryption circuit is $q(p(v(\lambda)))$, which is a polynomial, independent of $\mathcal{C}$. This completes the proof. $\square$

B Proof of Theorem 2

We prove Theorem 2 with the following lemmas.

Lemma 3. *A $\mathcal{C}$–evaluation scheme (*Gen*,* Enc*,* Eval*,* Dec*) with perfect circuit privacy implies $\mathcal{X} = \mathcal{Y}$.*

Lemma 4. *A $\mathcal{C}$–evaluation scheme (*Gen*,* Enc*,* Eval*,* Dec*) with $\mathcal{X} = \mathcal{Y}$ implies compactness when α is polynomially bounded in λ.*

Recall that $\mathcal{X}$ is the set of all possible outputs of the encryption algorithm and $\mathcal{Y}$ is the set of all possible outputs of the evaluation algorithm.

When we are dealing with perfect circuit privacy, perfect indistinguishability means that

$$|\Pr[\mathsf{Enc}(pk, C(\mathsf{Dec}(sk, c_1), \ldots, \mathsf{Dec}(sk, c_n))) = x] \\ - \Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) = x]| = 0$$

holds for every key tuple (pk, sk, evk) output by $\mathsf{Gen}(1^\lambda, \alpha)$, for all $c_i \in \mathcal{X}$ and all $C \in \mathcal{C}$.

Proof of Lemma 3. $\mathcal{Y} \subseteq \mathcal{X}$:

Assume $\mathcal{Y} \nsubseteq \mathcal{X}$, then there exists $a \in \mathcal{Y}$ such that $a \notin \mathcal{X}$. Hence a is a possible output of Eval with probability $p > 0$, but is not a possible output of Enc. This means that for some $c_i \in \mathcal{X}$, and some $C \in \mathcal{C}$

$$|\Pr[\mathsf{Enc}(pk, C(\mathsf{Dec}(sk, c_1), \ldots, \mathsf{Dec}(sk, c_n))) = a] \\ - \Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) = a]| = |0 - p| = p > 0,$$

which is a contradiction to perfect circuit privacy.

For $\mathcal{X} \subseteq \mathcal{Y}$, we use the strategy from above. So $\mathcal{X} \subseteq \mathcal{Y}$ and $\mathcal{Y} \subseteq \mathcal{X}$, hence $\mathcal{X} = \mathcal{Y}$. $\square$

Proof of Lemma 4. If $\mathcal{X} = \mathcal{Y}$ then all outputs of Eval are also outputs of Enc. Thus, by the argument in Lemma 1 on the length of outputs of Enc,

$$\text{length}(c) \leq a(\lambda)$$

for some polynomial a for all fresh ciphertexts c, where λ is the input parameter. Hence, all evaluation outputs are bounded by some polynomial on the input parameter. □

C Proof of Theorem 3

Proof. We first show that the probability that the evaluation algorithm outputs a ciphertext that is not fresh or does not decrypt correctly is negligible. The result then follows from the structure of a computation in stages.

Since the scheme is fully homomorphic, there is a negligible function ε' such that if pk, evk, sk have been output by $\mathsf{Gen}(1^\lambda, \alpha)$, then for any circuit $C \in \mathcal{C}$ and ciphertexts $c_1, c_2, \ldots, c_n \in \mathcal{X}$, we have

$$\Pr[\mathsf{Dec}(sk, \mathsf{Eval}(evk, C, c_1, \ldots, c_n)) = \\ C(\mathsf{Dec}(sk, c_1), \ldots, \mathsf{Dec}(sk, c_n))] = 1 - \varepsilon'(\lambda).$$

Since the scheme has statistical circuit privacy, there is a negligible function ε'' such that if pk, evk, sk have been output by $\mathsf{Gen}(1^\lambda, \alpha)$, then for any circuit $C \in \mathcal{C}$ and ciphertexts $c_1, c_2, \ldots, c_n \in \mathcal{X}$, we have

$$\Delta = \sum_{y \in \mathcal{Z}} |\Pr[\mathsf{Enc}(pk, C(\mathsf{Dec}(sk, c_1), \ldots, \mathsf{Dec}(sk, c_n))) = y] \\ - \Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) = y]| \leq \varepsilon''(\lambda).$$

Since Enc will never output ciphertexts in $\mathcal{Z} \setminus \mathcal{X}$, we have

$$\begin{aligned} &\Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) \in \mathcal{X}] \\ &\geq 1 - \sum_{y \in \mathcal{Z} \setminus \mathcal{X}} \Pr[\mathsf{Eval}(evk, C, c_1, \ldots, c_n) = y] \\ &\geq 1 - \Delta \geq 1 - \varepsilon''(\lambda). \end{aligned}$$

It is then clear that there is a negligible function ε such that the probability for an evaluation to be correct and the resulting ciphertext is in $\mathcal{X}$ is at least $1 - \varepsilon(\lambda)$.

Suppose pk, evk, sk have been output by $\mathsf{Gen}(1^\lambda, \alpha)$, that $\mathbf{C}_{i,n}$ is a computation in i stages of width n and that $c_{01}, c_{02}, \ldots, c_{0n} \in \mathcal{X}$. We are interested in the probability

$$\Pr[\mathsf{Dec}(sk, \mathsf{Eval}(evk, \mathbf{C}_{i,n}, \vec{c}_0)) = \mathbf{C}_{i,n}(\mathsf{Dec}(sk, \vec{c}_0))].$$

Let E_j be the event that after the jth stage, all of the ciphertexts computed so far are in $\mathcal{X}$ and decrypt to the correct value. It is clear that $\Pr[E_i]$ is no greater than the probability we are interested in. Since different executions of Eval are independent, we have:

$$\Pr[E_j] \geq \Pr[E_j \mid E_{j-1}] \Pr[E_{j-1}] \geq (1 - \epsilon(\lambda))^n \Pr[E_{j-1}].$$

It quickly follows that

$$\Pr[E_i] \geq (1 - \epsilon(\lambda))^{in}.$$

To conclude the proof, we must show that $1 - (1 - \epsilon(\lambda))^{in}$ is negligible when in is polynomial in λ. For simplicity, we will write ϵ instead of $\epsilon(\lambda)$ and show that $1 - (1 - \epsilon)^k$ is negligible when k is polynomial in λ.

We will show this by induction:

1. $k = 1 : 1 - (1 - \epsilon)^1 = \epsilon$, which is negligible by definition.

2. Now let $\mu := 1 - (1 - \epsilon)^k$ be negligible. Then we have:
   $1 - (1 - \epsilon)^{k+1} = 1 - (1 - \epsilon)^{k+1} - \mu + \mu$
   $= 1 - (1 - \epsilon)^{k+1} - (1 - (1 - \epsilon)^k) + \mu$
   $= (1 - \epsilon)^k - (1 - \epsilon)^{k+1} + \mu$
   $= (1 - \epsilon)^k \cdot (1 - (1 - \epsilon)) + \mu$
   $= (1 - \epsilon)^k \cdot (-\epsilon) + \mu$
   This shows that by increasing the exponent from k to $k + 1$, we get an increase of $(1 - \epsilon)^k \cdot (-\epsilon)$. If we can show that this increase is negligible, we have completed our proof. We again show this by induction:

   (a) $k = 1 : (1 - \epsilon) \cdot (-\epsilon) = \epsilon^2 - \epsilon$. Noting that $|\epsilon^2 - \epsilon| < \epsilon$ and ϵ negligible, the claim holds for $k = 1$.
   (b) Let $\alpha := (1 - \epsilon)^k \cdot (-\epsilon)$ be negligible. Then we have:
   $(1 - \epsilon)^{k+1} \cdot (-\epsilon) = (1 - \epsilon) \cdot (1 - \epsilon)^k \cdot (-\epsilon)$
   $= (1 - \epsilon) \cdot \alpha < \alpha$,
   so for the case of $k + 1$ it is also negligible.

3. Thus, we have shown that we start out with something negligible and add something negligible in each step. So, as long as we take polynomially many steps, the result will always also be negligible. $\square$

Paper II

Can there be efficient and natural FHE schemes?

Kristian Gjøsteen and Martin Strand

ePrint 2016/105, submitted to J. Math. Cryptology

Can there be efficient and natural FHE schemes?

Kristian Gjøsteen and Martin Strand

Department of Mathematical Sciences, NTNU
{kristian.gjosteen, martin.strand}@ntnu.no

January 7, 2018

Abstract

In 1978, Rivest, Adleman and Dertouzos asked for algebraic systems for which useful privacy homomorphisms exist. To date, the only acknowledged result is noise based encryption combined with bootstrapping. Before that, there were several failed attempts.

We prove that fully homomorphic schemes are impossible for several algebraic structures. Then we develop a characterisation of all fully homomorphic schemes and use it to analyse three examples. Finally, we propose a conjecture stating that secure FHE schemes must either have a significant ciphertext expansion or use unusual algebraic structures.

1 Introduction

In 1978 Rivest, Adleman and Dertouzos [32] posed two questions about *privacy homomorphisms*:

Q1 Does this approach have enough utility to make it worthwhile in practice?

Q2 For what algebraic systems does a useful privacy homomorphism exist?

A privacy homomorphism was defined to be an encryption function that would permit direct computations on the encrypted data. Gentry's construction of the first fully homomorphic encryption (FHE) scheme [19] answers this goal, but in a slightly different manner than what Rivest et al. envisioned. The original problem considered clear algebraic structures and mappings between them, while Gentry succeeded using noise-based constructions, putting less emphasis on the mappings.

One could say that both questions are still open. Gentry's 2009 breakthrough [19] in creating the first fully homomorphic encryption scheme has been followed by a number of much more efficient schemes [8,10,14,15,21,26]. There have been several earlier attempts at finding privacy homomorphisms, but none have been successful. Instead, there have been some negative results. Ahituv et al. [1] proved that a vector space isomorphism on $\mathbb{F}_2^n$ cannot be secure.

The answer to the first question seems to be positive. Some FHE applications have been demonstrated, but the list of theoretical applications is far longer than the list of viable implementations.

The first part of this work aims at closing some doors for the second question. To do this, we try to analyse the possibility for *natural* schemes based on automorphisms or isomorphisms on various structures. An example of such a scheme is the Pohlig-Hellman blockcipher. For completeness, we also survey previous results in this line of research.

Note that the analysis in the first part does not include modern noise-based FHE schemes. Although one can define operations on the ciphertext spaces by always performing a bootstrapping operation after an addition or multiplication, the ciphertext spaces do not become rings or any other well-known structure.

A second approach to achieve security is to embed the plaintexts into a larger set, such as for instance in ElGamal. We extend our arguments to show that this kind of scheme is also impossible for some of the structures we study. In this part of the work we do not assume a public *encryption* key, only that the adversary can perform evaluation of ciphertexts.

Furthermore, we extend the earlier characterisation of Armknecht, Katzenbeisser and Peter [4] to handle all public-key FHE schemes over any algebraic structure. This yields a simple transformation from any scheme to a suitable decision problem in order to analyse whether the scheme can be secure or not. Our results immediately allow us to prove that two proposed schemes are insecure, while our analysis supports the existing work on a third scheme.

The sum of our results make us propose the (informal) conjecture that FHE schemes either need to have a rather big expansion – which to date has not been efficient – although the plaintext spaces are usually nice fields, or alternatively, some suggested noise-free schemes have a homomorphic structure which will not easily permit the operations that the user would like to perform. Such structures will normally not look very natural to practitioners. The conjecture is an initial answer to Gentry's challenge to the mathematical community [20, p. 616]. The overall pursuit for an FHE scheme which both admits a natural-looking algebraic structure and maintains good efficiency is still open.

The paper is organised as follows. In Section 3, we analyse the possibility of fully homomorphic schemes on a number of common algebraic structures, and reach negative answers for the most useful structures. This technique does not allow us to consider the current FHE schemes, so we proceed to extend Armknecht et al.'s characterisation in Section 4. A brief recap of the algebra used in this paper is provided in the next section. The appendix contains a section with arguments aimed at refining the characterisation with topology. While the argument seems fruitless so far, it indicates that noise-based FHE schemes truly need an embedding in an infinite space.

1.1 Our contribution

All of the acknowledged FHE schemes in existence today are based on lattices, and usually feature a large ciphertext expansion. While one could say that some schemes are almost practical [13], we should not expect to see them in widespread use just yet. The main reason is often the communication cost, which consequently influences the computational cost. The big question is to find out if this can be reduced significantly while maintaining usefulness and security.

We are not proposing new schemes, and any discussion of existing schemes is only to demonstrate our new techniques for analysis. Our contribution can be summarised in two points.

- We provide a general tool for analysing new public-key FHE schemes, and we believe that the technique will easily distinguish between secure and insecure constructions.
- We extend existing results by proving that a number of possible FHE schemes must be insecure.

1.2 Related work

Following the original problem, there were a few attempts at creating privacy homomorphisms, usually followed by attacks: see Yu et al. [35] for a brief survey. Earlier impossibility results include Ahituv et al. [1] as mentioned above and Yu et al. who proved that a FHE scheme cannot achieve IND-CCA2 security.

Boneh and Lipton [7] demonstrated that any deterministic fully homomorphic encryption scheme over $\mathbb{Z}/n\mathbb{Z}$ can be broken in subexponential time.

Finally, Armknecht, Gagliardoni, Katzenbeisser and Peter [3] have proven that a group-homomorphic scheme will be vulnerable against a quantum adversary. A subset of the same authors also showed that no group-homomorphic scheme with a prime-order ciphertext group can be IND-CPA secure [5]. Our characterisation extends the construction provided by Armknecht, Katzenbeisser and Peter [4].

2 Preliminaries

We assume that the reader has some familiarity with fully homomorphic encryption, so we only give a brief overview. The interested reader should look up the survey by Armknecht et al. [2]. The term FHE has come to mean two things: Either that the scheme can evaluate both addition and multiplication, or that it can evaluate any circuit of *any* multiplicative depth. The scheme is i-hop if it can evaluate i circuits after each other, or ∞-hop if there is no limit. Note that all *somewhat* homomorphic schemes with a *sufficiently small decryption circuit* can be made fully homomorphic and ∞-hop using Gentry's original bootstrapping theorem [19]. A levelled scheme can compute any circuit with multiplicative depth up to its designated level.

Throughout the text, $\mathcal{P}$ will denote the plaintext space and $\mathcal{C}$ will denote the ciphertext space.

2.1 Algebraic structures

We assume that the reader is familiar with the definitions of groups, rings, fields and vector spaces. Recall that a division ring is a ring where every non-zero element has an inverse, and that all finite division rings are commutative, hence fields. We assume that all rings have an identity element. Furthermore, we also need two more concepts, namely *modules* and *algebras*. A module is a generalization of vectors spaces where the coefficients come from a ring. For a ring R, a (left) R-module is an additive abelian group M together with a scalar multiplication with ring elements on the left. We also have right R-modules, but for commutative rings, left and right R-modules are the same.

It is well known that any vector space of dimension n is isomorphic to n copies of the field. This is not true for modules. In particular, even the word "dimension" is not well-defined, and not all modules have a basis. Those that have are called *free* modules, and if every basis has the same number of elements, some n, we say that the module is of rank n.

- If I is an ideal of R, then I is also an R-module. In particular, R itself is an R-module.
- Any vector space over a field or division ring is also a module.
- Matrices over a ring R form an R-module.

We are interested in mappings between modules. Let M and N be R-modules. An R-homomorphism f is a function $f : M \to N$ such that for all $r \in R$ and $m, m_1, m_2 \in M$,

$$\begin{aligned} f(rm) &= rf(m) \\ f(m_1 + m_2) &= f(m_1) + f(m_2). \end{aligned}$$

For a field k, a *k-algebra* A is a k-vector space which is also equipped with a multiplication operation compatible with the scalar multiplication, such that A is a ring in its own respect. An algebra mapping is a function which is both a linear transformation on A as a vector space and a ring homomorphism on A.

2.2 The Wedderburn-Artin theorem

Recall that the structure theorem for finitely generated abelian groups states that any such group is isomorphic to a direct sum of copies of $\mathbb{Z}$ and cyclic groups of prime order. Rings generally lack a corresponding theorem. However for certain classes of rings we know the structure in detail. For instance, every finite field with the same cardinality is isomorphic. In the case of *semisimple* rings, we have the Wedderburn-Artin theorem.

Theorem 1 ([6], p. 382). *Let R be a left (or right) artinian ring with unity and no non-zero nilpotent ideals. Then R is isomorphic to a finite direct sum of matrix rings over division rings.*

The first sentence of the theorem is one of several equivalent definitions of a semisimple ring. An artinian ring is one where any descending chain (under inclusion) of ideals becomes constant after a finite number. Any finite ring is trivially artinian, while the integers $\mathbb{Z}$ are not. Consider this chain of ideals

$$(2) \supseteq (2^2) \supseteq (2^3) \supseteq \cdots$$

to see that it need not stabilise.

Also recall that an ideal I is nilpotent if there exists an integer n such that $I^n = (0)$.

3 Isomorphisms on structures

We now consider specific structures. In this section, we treat them in the symmetric case. This is partly for a practical reason – one motivation behind this work was to explore the possibility for very efficient schemes without any expansion. The conclusion seems to be that they are unlikely or at best, not practical. We divide our potential schemes into four types.

1. Identical spaces $\mathcal{P} = \mathcal{C}$
2. Isomorphic spaces $\mathcal{P} \simeq \mathcal{C}$ (but possibly with different descriptions)
3. $\mathcal{C}$ is larger than $\mathcal{P}$, but only with a constant expansion
4. $\mathcal{C}$ is larger than $\mathcal{P}$, and the expansion depends on the parameters

In all scenarios, we assume that both $\mathcal{P}$ and $\mathcal{C}$ share the same kind of algebraic structure. For instance, if $\mathcal{P}$ is a vector space over a field k, then $\mathcal{C}$ must also be a k-vector space. For schemes of the second type, we stress that the spaces really must be isomorphic. The next section will deal with the situation where $\mathcal{P}$ is isomorphic to the residue classes of $\mathcal{C}$.

Since we are primarily looking for very efficient schemes, we will not use any energy on schemes of Type 4. Note that there is still a certain jump from that to the state of art today. The ciphertext spaces of modern noise-based schemes are not rings, even if you consider bootstrapping as a part of every multiplication operation. The underlying reason is that the noise generation hinders the space from being closed under even the normally "cheap" addition operation. Also, one is not guaranteed to have the associative property for either operation.[1] One should rather use the techniques of Section 4 to analyse these schemes.

From now on, we will refer to these cases by their numbers. It is straightforward to observe that any scheme of Type 1 or 2 must be

[1]The GSW scheme [21] is a good example of this, where noise propagates very differently based on how the multiplication is performed.

deterministic, and hence not achieve semantic security, while those of Type 3 typically should be randomised, such as ElGamal or Paillier, which always feature data doubling.

3.1 Achievable security

As a consequence of the discussion from the previous section, we need a weaker security notion than that of semantic security for schemes of Type 1 or 2. We want to describe a notion where it is hard to distinguish encryption from a random isomorphism. Typically, the former will be a subset of all isomorphisms, but may not be exhaustive. Furthermore, any encryption isomorphism must be hard to invert: even if a scheme trivially achieves indistinguishability with two sets that are identical, it could be practically insecure, since it may be easy to invert.

Thus we want a definition that catches the same principle as pseudorandom permutations, taking algebraic structure into account, and also adding the invertibility requirement. This is the closest we can come to semantic security under these restricted conditions.

We assume that we have access to a black box that can efficiently compute any mapping ϵ into $\mathcal{C}$ such that there exists another mapping $\delta : \mathcal{C} \to \mathcal{P}$ such that $\delta \circ \epsilon$ acts as the identity on $\mathcal{P}$. For instance in Type 1 above, ϵ would be any isomorphism on $\mathcal{P}$. Essentially, the black box should be able to compute everything that could have been an encryption, whereas the scheme is limited to the isomorphisms indexed by the keys.

Definition 1. Let Π be an encryption scheme of Type 1 or 2 with plaintext space $\mathcal{P}$ and some isomorphic ciphertext space $\mathcal{C}$. Let ϵ_K be the mapping induced by the encryption algorithm under a key K, and let $\mathcal{O}$ denote the collection of all such mappings. Finally, let $\mathrm{Iso}(\mathcal{P}, \mathcal{C})$ denote all isomorphisms from $\mathcal{P}$ to $\mathcal{C}$. We say that Π is secure if

1. the adversary can only distinguish $\mathcal{O}$ and $\mathrm{Iso}(\mathcal{P}, \mathcal{C})$ with negligible probability, and
2. any mapping $\epsilon_K \in \mathcal{O}$ is hard to invert.

There is a straightforward real-or-random game for the first property: the challenger randomly selects whether to use a random isomorphism or an instance of the cryptosystem. The adversary wins if it can distinguish with non-negligible probability. To align with the IND-CPA notion, we allow the adversary to query a number of encryptions before giving its answer, but no decryptions.

3.2 Groups

There exist secure group homomorphic schemes of Type 1, and therefore of Type 2–4 as well.

This statement is straightforward to prove constructively by providing a concrete example, for instance the Pohlig-Hellman exponentiation cipher [30]. Let G be a cyclic group of secret order n, and choose e, d such that $ed \equiv 1 \pmod{\phi(n)}$. Encryption and decryption is as with RSA. Then all automorphisms on the group are indexed by e, so the scheme trivially satisfies the first condition since the sets are identical. The second condition holds under standard number-theoretic assumptions.

It is straightforward to construct an example to show that this implies the existence of a secure scheme when the groups are isomorphic but with different descriptions, or if we add further copies of the group. However, ElGamal is a better example in that respect.

Textbook RSA is not secure according to our definition. For simplicity, assume that $n = pq = (2p' + 1)(2q' + 1)$ with p, q, p', q' prime, and let G be the group of multiplicative units in $\mathbb{Z}/n\mathbb{Z}$. We then know that, as groups

$$G \simeq \mathbb{Z}/p'\mathbb{Z} \times \mathbb{Z}/q'\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}.$$

This group has $6p'q'$ automorphisms [22], where the factor 6 comes from the permutation of the order 2 elements in the Klein 4-subgroup V. RSA is able to produce $p'q'$ of these automorphisms, all of which leave the four elements of V fixed. The automorphisms outside the RSA subset either leave -1 alone, or swap it with one of the other order 2 elements. In the latter case, it is possible to distinguish. In the first case, a distinguisher must find one of the two other special elements. That will allow the adversary to factor n, using the same idea as in Rabin's oblivious transfer [31].

3.3 Vector spaces

Ahituv, Lativ and Neumann [1] showed that a homomorphism $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ cannot be secure, by using f on the basis to essentially compute the inverse. Their argument easily extends to a vector space over any field.

For spaces of different sizes, fix a field k, and let $\mathcal{P}$ and $\mathcal{C}$ be k-vector spaces, with $p = \dim \mathcal{P} \leq \dim \mathcal{C} = q$. Get encryptions of at least pq linearly independent vectors. There exists a matrix D such that $Dc_i = m_i$ for each pair (m_i, c_i). Solve the resulting system for D, which is essentially the decryption key. Hence, such a scheme cannot be secure.

The reader might come up with an apparent counterexample to the claim, the Goldwasser-Micali cryptosystem. Let n be a product of two primes, let y be a non-quadratic residue modulo n and let x_i be units modulo n. To encrypt a vector m in $\mathbb{F}_2^\ell$, compute a tuple $(c_1, \cdots, c_\ell)$ where $c_i = y^{m_i} x_i^2$ modulo n. The ciphertext space is then a $\mathbb{Z}/n\mathbb{Z}$-module where addition is given by pointwise multiplication and scalar multiplication is done by exponentiation. Decrypt by checking whether each component is a quadratic residue or not. However, C is not an $\mathbb{F}_2$-vector space, so it does not fit in our system. It shows that one doesn't need to go far away from these constructions to find something that is secure, although with a certain expansion factor.

This argument proves the following theorem.

Theorem 2. *Let k be a field and let $\mathcal{P}$ and $\mathcal{C}$ be k-vector spaces. Then there are no secure encryption schemes between $\mathcal{P}$ and $\mathcal{C}$.*

3.4 Fields

We only consider finite fields, and we can consider prime fields and extension fields separately. For prime fields, there are only fields of the kind $\mathbb{Z}/p\mathbb{Z}$, possibly with some strange description, but nonetheless straightforward to convert into the canonical form. There is only the identity automorphism.

For extension fields, where $\mathcal{P} \simeq \mathcal{C}$, an algorithm by Lenstra [23] demonstrates that it is feasible to convert the original presentation to practical descriptions of the fields, and then compute any isomorphisms between those fields. Hence, there can be no secure homomorphic schemes between fields of the same size.

Let us tackle the third type. First note that the fields must have the same characteristic, otherwise there would be no structure preserving mappings between them. We can therefore assume that both $\mathcal{P}$ and $\mathcal{C}$ are extensions of the same prime field $\mathbb{F}_p$. We can consequently also view $\mathcal{P}$ and $\mathcal{C}$ as $\mathbb{F}_p$-vector spaces, reducing the problem to the one in the previous subsection.

Alternatively, one can adjoin extra elements to $\mathcal{P}$ to make it isomorphic to $\mathcal{C}$, and then compute an isomorphism based on known plaintext-ciphertext pairs.

If the encryption is merely a deterministic injection into a larger ciphertext space it can still not be secure, as we can view the image as a field in its own right.

We summarise the discussion in a theorem.

Theorem 3. *Let $\mathcal{P}$ and $\mathcal{C}$ be k fields. Then there are no secure encryption schemes between $\mathcal{P}$ and $\mathcal{C}$.*

3.5 Rings

Rings are far harder to tackle than the previous structures, and we are not able to give a definitive answer. The key information is the set of automorphisms on a ring. We start by getting an overview of those for reduced rings, that is, rings with no non-zero nilpotent elements.

3.5.1 Reduced rings

Note that since R is finite, it is in particular left (and right) artinian. Assume that R has no non-zero nilpotent ideals, then R is semisimple. By the Wedderburn-Artin theorem, R is then isomorphic to a finite direct product of matrix rings over division rings,

$$R \simeq M_{n_1}(D_1) \oplus \cdots \oplus M_{n_\ell}(D_\ell).$$

We have assumed that R has no nilpotent ideals. Since R is artinian, nil and nilpotent ideals are the same, so we have simultaneously assumed there are no non-zero nil ideals. In particular, this means that the radical of R, which is precisely all nil elements, is the zero ideal.

Now, if at least one $n_i \geq 2$, then one can create a nilpotent element in that matrix ring, a contradiction. Furthermore, a finite division ring is always a field, so we reach the simplification

$$R \simeq k_1 \oplus \cdots \oplus k_\ell.$$

Assume that there are m distinct fields up to isomorphism, and with ℓ_i fields in each set, $1 \leq i \leq m$. Gather isomorphic fields together. For each i, let d_i be the degree of the field extension. Let ϕ be an automorphism on R. We can write it as $(\phi_{1,1}, \ldots \phi_{1,\ell_1}, \ldots, \phi_{1,\ell_m})$. There are d_i automorphisms on each field, and one can use any permutation on each set of isomorphic fields. This yields a total of

$$\prod_{i=1}^{m} (d_i - 1)!\, \ell_i!$$

automorphisms on R, which bounds the size of the key space.

To see why there are no more isomorphisms, consider $R = k_1 + k_2$, where $k_1 = k_2$ are extension fields over k. Define a function $\phi : R \to R$ which is the identity on $k \times k$, and swaps the components otherwise. Take $(x_1, x_2) \in k^2$ and $(x_1', x_2') \notin k^2$. If one applies ϕ on the sum, one can see that it differs from $\phi(x_1, x_2) + \phi(x_1', x_2')$, so ϕ cannot be a ring homomorphism.

Now consider the special case where we let R be a k-algebra. Friedl and Rónyai and others [11, 16] have described polynomial time algorithms to compute a Wedderburn-Artin decomposition explicitly. We say “a” rather than “the”, since there can be several isomorphic decompositions, given by permutations on isomorphic summands. One can therefore compute the automorphisms for each field separately by the techniques described above. The feasibility of the computation of any automorphism therefore depends only on $\prod_{i=1}^{m} \ell_i!$ being sufficiently large.

3.5.2 Semisimple k-algebras

We shift the assumptions slightly. We now allow more nilpotent elements, but only for algebras over a field k. The aforementioned algorithms also work in this setting; one can compute orthogonal idempotents in the centre of each summand, and next a basis for each matrix ring. Computing the mapping between the algebra and the decomposition is feasible: multiply with the idempotents to decompose the element, and then use linear algebra. We therefore have a canonical description of any ring element, up to permutations of components and bases.

In particular, this setting includes many matrix rings. Note that we cannot automatically use our reasoning about vector spaces here since linear mappings are different from ring homomorphisms. However, vector transformations is a subset of all algebra homomorphisms. Brakerski [9] has proven that whenever decryption is an inner product computation involving the secret key, then the scheme cannot be CPA secure. In particular, this involves all schemes using inner automorphisms on matrix rings, i.e. where a square matrix M (possibly encoding the ciphertext from a subring) is conjugated, $C = A^{-1}MA$. If the ring is commutative, one can rewrite the above equation and the corresponding decryption formula as inner products. By the Skolem-Noether theorem, every automorphism of a matrix ring over a field is inner.

One attempt to avoid this problem is by first embedding the plaintext space in the noncommutative quaternions or octonions, four- or eight-dimension division algebras constructed by adjoining three or seven extra elements to a ring in a similar manner as when constructing the two-dimensional complex numbers by adjoining the special symbol $i = \sqrt{-1}$ to the real numbers. The construction is also valid if performed over rings like $\mathbb{Z}/q\mathbb{Z}$. However, there exist straightforward embeddings of the quaternions and the octonions over a ring R as subrings of the 4×4 and 8×8 matrix rings over R, respectively.

Hence, one can always ignore the extra noncommutative structure by considering larger matrices.

3.5.3 Large ciphertext rings

Finally, consider the case where both $\mathcal{P}$ and $\mathcal{C}$ are rings, and let $I = \ker \mathsf{Dec} = \mathsf{Dec}^{-1}(0)$. By the reasoning in Section 4 and the first isomorphism theorem, we therefore know that $\mathcal{P} \simeq \mathcal{C}/I$, so we can identify $\mathcal{P}$ as a subring of $\mathcal{C}$. If $\mathcal{P}$ is commutative, then $\mathcal{C}$ becomes an associative $\mathcal{P}$-algebra. When $\mathcal{P}$ is a field, this reduces to the vector space scenario above, hence not secure. The general challenge is to distinguish elements of I and R.

3.6 Modules

Modules can be quite similar to vector spaces, so in particular for free modules over well-behaved rings, one would expect that the same techniques should apply. On the other hand, an abelian group is a $\mathbb{Z}$-module, and $\mathbb{Z}$ is certainly a very well-behaved ring, despite its lack of nontrivial units.

4 Characterisation

In this section we treat fully homomorphic schemes with as much generality as we can, by considering various algebraic structures. An algebraic structure is an object that consists of a set of elements, is closed under at least one operation and satisfies certain axioms on the operations. Examples include groups, rings, vector spaces and so on. Note in particular that we ignore somewhat and levelled homomorphic schemes, although some of the reasoning can also be applied to those. The reason is that the spaces involved in somewhat homomorphic schemes are neither closed under addition nor multiplication, since once the noise level grows too large, the ciphertext will no longer decrypt to the correct value.

The short version of this section is that any homomorphic encryption consists of a mapping into the ciphertext space and an addition with a random encryption of zero. For the scheme to be secure, at least one of those operations must protect the message.

In this section, we use the more conventional security definition of IND-CPA.

Definition 2 (IND-CPA). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Define the following experiment $\text{IND-CPA}_{\Pi}^{b}(\mathcal{A}, \lambda)$ between an adversary $\mathcal{A}$ and a challenger.

1. The challenger runs $(sk, pk) \leftarrow \mathsf{Gen}(1^{\lambda})$ and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ outputs to messages m_0, m_1 of the same length.
3. The challenger computes $\mathsf{Enc}(pk, m_b)$ and gives it to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b', and the challenger returns b' as the output of the game.

The scheme Π is IND-CPA secure if for any λ and for any probabilistic polynomial time adversary $\mathcal{A}$,

$$\begin{aligned}\text{Adv}(\mathcal{A}) = |\Pr\left[\text{IND-CPA}_{\Pi}^{0}(\mathcal{A}, \lambda) = 1\right] - \\ \Pr\left[\text{IND-CPA}_{\Pi}^{1}(\mathcal{A}, \lambda)\right] = 1| \leq \varepsilon(\lambda)\end{aligned}$$

Let S be an algebraic structure with $n \geq 1$ operations $*_1, \ldots, *_n$, where $*_1$ is a binary operation such that for any object O with structure S, we have a neutral element 0 with respect to $*_1$, and that $*_1$ is a bijection when the second coordinate is fixed. This implies that all elements have inverses with respect to the first operation. We stress that we do not put any assumptions on the other operations.

Any structure that contains a group structure satisfies this requirement when $*_1$ is taken to be the group operation, typically addition. For the binary case, it requires either XOR or AND. If the set of binary operators is functionally complete, we can add either to the set and rearrange.

Let $\mathcal{P}$ be an object with structure S, let $\mathcal{C}$ be a set (without any structure) with operations $*'_1, \ldots, *'_n$, and let $1_{\mathcal{P}}$ be the identity map on $\mathcal{P}$. Let $\epsilon : \mathcal{P} \to \mathcal{C}$ and $\delta : \mathcal{C} \to \mathcal{P}$ be maps such that $\delta \circ \epsilon = 1_{\mathcal{P}}$, and for all $1 \leq i \leq n$ and all $c_1, \ldots, c_{a_i}$ where a_i is the number of elements that $*_i$ takes as input, we have

$$\delta(*'_i(c_1, \ldots, c_{a_i})) = *_i(\delta(c_1), \ldots, \delta(c_{a_i})).$$

We call $(\mathcal{P}, \mathcal{C}, \epsilon, \delta)$ an S-homomorphic tuple.

Now assume we have a set $(\mathsf{Instance}, \mathsf{Epsilon}, \mathsf{Sample}, \mathsf{Delta}, \mathsf{Op})$ of algorithms such that:

$\mathsf{Instance}$ takes in security parameter λ and returns a S-homomorphic tuple

Sample takes in $(\mathcal{C}, \epsilon)$ and returns a random element from $\delta^{-1}(0)$

$\mathsf{Epsilon}$ takes in $(\mathcal{C}, \epsilon, m)$ and computes $\epsilon(m)$

Delta takes in $(\mathcal{C}, \delta, c)$ and computes $\delta(c)$

Op computes $*'_i(c_1, \ldots, c_{a_i})$ on input $(\mathcal{C}, \epsilon, i, c_1, \ldots, c_{a_i})$.

Based on this, we can construct an abstract encryption scheme.

Definition 3. Define the Abstract Homomorphic Encryption Scheme (AHES) as an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with the following algorithms

Gen Run $\mathsf{Instance}(1^\lambda)$ to get $(\mathcal{C}, \epsilon, \delta)$. Output $pk = evk = (\mathcal{C}, \epsilon)$ and $sk = (\mathcal{C}, \delta)$,

Enc On input pk, m, return

$$\mathsf{Op}(\mathcal{C}, \epsilon, 1, \mathsf{Epsilon}(pk, m), \mathsf{Sample}(pk)),$$

Dec On input sk, c, return $\mathsf{Delta}(\mathcal{C}, \delta, c) = \delta(c)$,

Eval On input $evk, *_i, c_1, \ldots, c_{a_i}$, output

$$\mathsf{Op}(\mathcal{C}, \epsilon, i, c_1, \ldots, c_{a_i}).$$

To see why this encryption makes sense, observe that we can induce the structure from $\mathcal{P}$ onto a subset of $\mathcal{C}$ through ϵ and δ. Define a equivalence relation $\sim$ on $\mathcal{C}$ by $c_1 \sim c_2$ if $\delta(c_1) = \delta(c_2)$. Each equivalence class now corresponds to a unique plaintext, so we can identify $\mathcal{P}$ with $\mathcal{C}/{\sim}$. For each m, we mark $\epsilon(m)$ as a distinguished representative of its class, creating the subset. Encryption is therefore to go to the corresponding equivalence class, and then using the operator $*_1'$ with something that decrypts to 0. That amounts to using $*_1$ with the neutral element in the second coordinate, i.e. nothing ("adding encryptions of zero"). The result will be in the same class, but randomly distributed.

Let c_0 be sampled from $\delta^{-1}(0)$ using Sample. Correctness is ensured since

$$\begin{aligned}\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) &= \delta(\mathsf{Op}(evk, 1, \mathsf{Epsilon}(pk, m), \mathsf{Sample}(pk))) \\ &= \delta(*_1'(\epsilon(m), c_0)) \\ &= *_1(\delta(\epsilon(m)), \delta(c_0)) = m *_1 0 = m.\end{aligned}$$

Evaluation is well-defined by the properties of the S-homomorphic tuple and the same arguments as above.

Theorem 4. *The Abstract Homomorphic Encryption Scheme is a homomorphic ∞-hop scheme. Any homomorphic ∞-hop scheme E over a given algebraic structure S can be expressed in terms of the above.*

Proof. The scheme is fully homomorphic. Express any evaluation in terms of the operations on $\mathcal{P}$. Note that the evaluation and ciphertext spaces are the same, so it is ∞-hop.

Now for a given homomorphic ∞-hop encryption scheme ($E.\mathsf{Gen}$, $E.\mathsf{Enc}, E.\mathsf{Eval}, E.\mathsf{Dec}$), we need to construct the algorithms for our abstract scheme. Define operations $*_1, \dots, *_n$ on $\mathcal{P}$ based on the allowed computations, and identify the neutral element 0 with respect to $*_1$.

$\mathsf{Instance}$ Run $E.\mathsf{Gen}$ to get $E.\mathcal{C}$ and keys. Fix some a, and define $\epsilon(m) = E.\mathsf{Enc}(E.pk, m)$ using a as randomness, and $\delta(c) = E.\mathsf{Dec}(E.sk, c)$. Return $(\mathcal{C}, \epsilon, \delta)$.

Sample Compute a random encryption of 0.

Op Use E.Eval.

The algorithms Epsilon and Delta follows from ϵ and δ. The homomorphic properties of ϵ and δ are satisfied by definition, and we can use the construction above to create an instance of the AHES. $\square$

The security is based on an assumption that all the equivalence classes are of about the same size. The following lemma ensures that this is the case in certain special cases.

Lemma 1. *If $*_1'$ is a bijection when the first element is fixed, and the whole of $\delta^{-1}(0)$ is samplable, then all equivalence classes as described above in the ciphertext space of the Abstract Homomorphic Encryption Scheme have the same cardinality.*

Proof. Let m be an arbitrary message, $c = \epsilon(m)$ and let c_0 be the element sampled from $\delta^{-1}(0)$ in the encryption algorithm. Recall that the encryption is then defined as

$$\mathsf{Op}(\mathcal{C}, \epsilon, 1, \mathsf{Epsilon}(pk, m), \mathsf{Sample}(pk)) = *_1'(c, c_0).$$

Then the restricted function $f_c : \delta^{-1}(0) \to \delta^{-1}(m)$ given by $f_c(x) = c *_1' x$ is an injection (as it is a bijection on all of $\mathcal{C}$), so it is clear that $|\delta^{-1}(0)| \leq |\delta^{-1}(m)|$.

Now let m' be such that $m' *_1 m = 0$, which we know exists. Consider the images $A = f_c(\delta^{-1}(0))$, $B = \delta^{-1}(m)$ and $C = f_{\epsilon(m')}(B)$. By the homomorphic property, $f_{\epsilon(m')}$ maps elements of $\delta^{-1}(m)$ to $\delta^{-1}(0)$, so $C \subseteq \delta^{-1}(0)$.

Since $f_{\epsilon(m')}$ is an injection, then $|\delta^{-1}(m)| = |B| = |f_{\epsilon(m')}(B)| = |C| \leq |\delta^{-1}(0)|$, but then $|\delta^{-1}(0)| = |\delta^{-1}(m)|$ for any m. $\square$

The scheme is semantically secure if it is hard to decide if an element is in a given equivalence class.

Definition 4 (Subset membership problem (SMP))**.** Let $\mathcal{C}$ be an efficiently samplable set, and let R be a subset of $\mathcal{C}$. The challenger selects a random bit b. If $b = 0$, c is sampled uniformly from R, else from $\mathcal{C}$, and sent to the adversary. The adversary wins if it outputs the correct b.

We modify the standard problem slightly. Replace the subset R with an efficient structure preserving mapping $\delta : \mathcal{C} \to \mathcal{P}$ as above. We now sample from the subset $\delta^{-1}(0)$.

The following theorem should not come as a surprise, as it is to a large extent a reformulation of IND-CPA security. However, its proof contains a transformation that can be used on any FHE scheme, and that should return a suitable problem to study. As the following examples will show, this enables us to quickly analyse schemes proposed in good faith, and thus may be a valuable tool.

Theorem 5. *Let* Π *be a fully homomorphic encryption scheme with ciphertext space* $\mathcal{C}$ *and a mapping* δ *induced by the decryption algorithm. Assume that the equivalence classes in* $\mathcal{C}$ *are of about the same size: for all plaintexts* m_1, m_2 *the inequality* $|1 - \frac{|\delta^{-1}(m_1)|}{|\delta^{-1}(m_2)|}| \leq \varepsilon$ *holds where* ε *is negligible. The scheme* Π *is then semantically secure if and only if the subset membership problem is hard.*

Proof. First we show that an adversary $\mathcal{A}_{\text{AHES}}$ with non-negligible advantage ε against a real-or-random game implies a distinguisher for the subset membership problem SMP. Our adversary $\mathcal{A}_{\text{SMP}}$ receives the challenge $(\mathcal{C}, \delta^{-1}, x)$ from SMP. We create an instance of AHES with $\mathcal{C}$ and δ^{-1} as a part of the public key. Note that we can construct ϵ by selecting an element from $\delta^{-1}(m)$ for each m.[2] It is clear that $\delta \circ \epsilon = 1_{\mathcal{P}}$, and this will induce the required structure on the equivalence classes $\delta^{-1}(m)$ on $\mathcal{C}$. The homomorphic property holds since δ is defined to be structure preserving.

Transmit the public key. Upon receiving the plaintext m from the adversary, we encrypt it as $c \leftarrow \mathsf{Op}(evk, 1, \mathsf{Epsilon}(pk, m), x)$, and return the challenge ciphertext. When $\mathcal{A}_{\text{AHES}}$ returns a d with $d = 0$ if c is an encryption of m, we simply set $b' = d$ and replies with b' to the SMP instance.

If $b = 0$, then x is an encryption of 0, and the encryption is as usual. Otherwise, x can be interpreted as a random ciphertext, so c encrypts a random element, and the distribution is near uniform

[2]This could potentially take a long time for a large plaintext space. Luckily, we note it can usually be done by encrypting with zero randomness, hence ϵ is typically fully described in constant time.

since the equivalence classes are of approximately the same size. It is clear that $\mathcal{A}_{\text{SMP}}$ has the same advantage against SMP as $\mathcal{A}_{\text{AHES}}$ has against Abstract Homomorphic Encryption Scheme.

For the opposite direction we again play a real-or-random game with a similar idea as AKP are using. Set up an Abstract Homomorphic Encryption Scheme instance (AHES), and send the public key to the adversary $\mathcal{A}_{\text{AHES}}$. Then select a message m. Note that the message will be invertible with respect to $*_1$. AHES responds with a ciphertext c, and $\mathcal{A}_{\text{AHES}}$ computes $x \leftarrow c *_1' \mathsf{Epsilon}(pk, m^{-1})$ and submits x to $\mathcal{A}_{\text{SMP}}$ along with $\mathcal{C}$ and ϵ. Upon receiving d from $\mathcal{A}_{\text{SMP}}$, $\mathcal{A}_{\text{AHES}}$ forwards $b' = d$ to the AHES instance.

If $b = 0$, then the ciphertext was real, which means that x encrypts 0, and hence selected from the inverse image of δ. Otherwise, c will be a random encryption, and it will just be shifted by $\mathcal{A}_{\text{AHES}}$, so it will still be a random element selected from $\mathcal{C}$. This means that $\mathcal{A}_{\text{AHES}}$ will have the same advantage as $\mathcal{A}_{\text{SMP}}$. $\square$

Remark 1. We define the *Splitting Oracle-Assisted Subgroup Membership Problem* (SOAP) problem in the same way as in AKP [5], replacing “subgroup” with “subset”.

Informally, the SOAP problem is the same as the SMP problem, except that before receiving the challenge, the adversary has access to an oracle that on input c outputs $(\epsilon(m), c_0)$ where $c = \epsilon(m) *_1' c_0$, thus ”splitting“ a ciphertext into the distinguished representative of its equivalence class and its corresponding encryption of 0. Theorem 3 from AKP will then hold with only minor changes to the proof. To find out if the SOAP problem is hard outside a generic group remains an interesting problem.

We give the theorem with the few relevant words changed.

Theorem 6 (Characterization of IND-CCA Security, Theorem 3, [5]). *Let* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be an* ∞*-hop fully homomorphic encryption scheme. Then:*

$$\mathcal{E} \textit{ is IND-CCA1 secure} \Leftrightarrow \textit{SOAP is hard}$$

4.1 Examples

We now apply our construction to one successful and two less successful schemes to prove the power of our approach. Our attacks spring directly from our characterisation. For the Nuida-Kurosawa scheme [29], it does not provide new insight, as the scheme is already quite close to this construction.

4.1.1 FHE over integers with non-binary space

Nuida and Kurosawa have suggsted a simple scheme over the integers [29]. Let Q be a fixed prime, and let p and q_0 be large, secret primes such that $N = pq_0$ is hard to factor. A base encryption of m is of the form $m + pk + Qr$, and decryption is given by taking modulo p and then Q. Let x' be an encryption of 1 and $\{x_i\}$ be a set of encryptions of 0. The public key is $(N, x', \{x_i\})$. To encrypt using the public key, transform the message m from $\mathbb{Z}_Q$ into $\mathbb{Z}_N$ by multiplying with x', and add a random sum of encryptions of 0. See the original paper for details on the bootstrapping.

Note that this scheme by design is similar to our general description. The base problem that needs to be hard is therefore easily isolated, namely distinguishing the following two distributions,

$$\{m + pk + Qr \mid m < Q \ll p\} = \mathbb{Z}_{pq_0}$$

and

$$\{pk + Qr \mid Q \ll p\},$$

along with additional data available for bootstrapping. This is essentially the same problem studied by Cheon et al. [12].

4.1.2 Liu's scheme

In May 2015, Liu published a candidate scheme on IACR's ePrint archive [25]. Although it was quickly proven insecure [34], it provides a series of valuable lessons. We refer to the original paper for details about the scheme.

The instance algorithm outputs $\mathcal{P} = \mathbb{Z}/q\mathbb{Z}$ as a field and $\mathcal{C} = (\mathbb{Z}/q\mathbb{Z})^{n+1}$ as a $\mathcal{P}$-algebra with ordinary addition and scalar multiplication, but with a key-dependent multiplication operation. Let $*_1$ be addition. The public key is the tuple $(\Theta, \Phi, \{\mathsf{Enc}(sk_i sk_j)\}_{i,j=1,\dots n+1})$, and the private key is the vector $sk = (sk_1, \dots, sk_{n+1})$.

Define ϵ as encryption using 0 for all randomisers. Now, δ is just the linear mapping $\mathcal{P}^{n+1} \to \mathcal{P}$ given by $x \mapsto \langle sk, x\rangle$, and the problem is to decide whether the given element is in the kernel. It is clear that the kernel is an n-dimensional subspace of $\mathcal{P}^{n+1}$. Sample $m \geq n$ vectors from the subspace, and append x to form a matrix, and compute the rank. If $x \notin \delta^{-1}(0)$, then the rank will be $n+1$ with high probability, giving the adversary an advantage. The scheme can therefore not be secure.

4.1.3 Li-Wang scheme

Wang and Li proposed a new scheme based on multiplication of matrices over noncommutative rings [24]. As key, select a secret invertible matrix H and compute H^{-1}. To encrypt, place the message m in the top left corner of a upper triangular matrix M. The remaining places are filled with random values. Then compute the ciphertext as $C = HMH^{-1}$. Addition and multiplication works in the natural way.

The authors speculate that the scheme may be IND-CCA1 secure. However, we believe that it is insecure. Note that the scheme is symmetric, but the same reasoning as above implies that we only need to be able to distinguish encryptions of 0 from a random encryption.

Observe that the diagonal of M completely determines the invertibility of C, and that an encryption of 0 cannot be invertible. However, there is not an equivalence, since it can also be non-invertible if any element of the diagonal is a non-unit. To improve the probability, we can ask the encryption oracle for additional encryptions of 0, and add them to C, checking the invertibility for each.

With high probability one can then distinguish encryptions of units from encryptions of non-units, which is already sufficient to win a left-or-right game. The advantage against a real-or-random game will depend on the number of non-units in the ring. If the underlying ring is a division ring, then there are no other non-units than 0. One can efficiently compute inverses by using a variant of LU decomposition suited for noncommutative rings.

Another tool for deciding invertibility, and which could give further information on the linear dependencies of the rows and columns, is the notion of quasideterminants, introduced by Gelfand and Retakh in 1991 [17]. In contrast to determinants for matrices over commutative rings there is not just one quasideterminant for an $n \times n$ matrix, but n^2. They may not always be computable, but provide useful information whenever they are.

Proposition ([18], Proposition 1.4.6)**.** *If the quasideterminant $|A|_{ij}$ is defined, then the following statements are equivalent.*

1. $|A|_{ij} = 0$
2. *the i-th row of the matrix A is a left linear combination of the other rows of A*
3. *the j-th column of the matrix A is a right linear combination of the other columns of A*

The multiplicative identities for quasideterminants could also provide additional equations in order to perform a message-recovery attack.

5 Conclusion

We summarise the findings of Section 3 in Table 1. We cannot give a definitive answer to Rivest et al.'s questions, but the trend seems to be that nice structures also make it easier to break schemes. This is unfortunate, as we would like to be able to perform our computations over nicely behaved structures. The contemporary solution to this is to allow a nice plaintext space – even fields – but compensate by having ciphertext spaces with very little structure. In particular, they are not closed under addition and multiplication until the expensive bootstrapping procedure is introduced.

Structure	Abelian groups	Vector spaces	Fields	Rings
Identical sets	✓	×	×	?
Non-trivially isomorphic	✓	×	×	?
Constant expansion	✓	×	×	?

Table 1: ×: No secure schemes. ?: No final conclusion. ✓: Examples of secure schemes exist.

Nuida [28] has proposed a framework for noise-free FHE schemes, but he also notes that the schemes look "somewhat 'artificial' ", and continues: "more 'natural' constructions of the underlying groups [...] would be more desirable [...]. Here we note, however, that such a natural instantiation of our schemes seems not easy to find." [28, p. 3] We summarise this in the following informal conjecture, which agrees with what seems to be the consensus. Still, it remains to be proven.

Conjecture. *The security of a fully homomorphic encryption scehme either depends on a massive and noisy ciphertext expansion or using an algebraic structure which admits few of the computations one would like to perform in real world applications.*

Acknowledgements The work was partially financed by the Norwegian Research Council grant no. 233977 and the German Academic Exchange Service (DAAD), project number 57068907. The authors wish to thank Frederik Armknecht, Angela Jäschke, Colin Boyd, Christopher Carr and Christian Reuter for valuable discussion. Parts of this work was done while the second author visited Armknecht in Mannheim.

References

[1] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing encrypted data. *Commun. ACM*, 30(9):777–780, 1987.

[2] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. `http://eprint.iacr.org/`.

[3] Frederik Armknecht, Tommaso Gagliardoni, Stefan Katzenbeisser, and Andreas Peter. General impossibility of group homomorphic encryption in the quantum world. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 556–573. Springer, 2014.

[4] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. Shift-type homomorphic encryption and its application to fully homomorphic encryption. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology – AFRICACRYPT 2012*, volume 7374 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2012.

[5] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. Group homomorphic encryption: characterizations, impossibility results, and applications. *Des. Codes Cryptography*, 67(2):209–232, 2013.

[6] P. B. Bhattacharya, S. K. Jain, and S. R. Nagpaul. *Basic abstract algebra.* Cambridge University Press, Cambridge, second edition, 1994.

[7] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 1996.

[8] Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and*

Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.

[9] Zvika Brakerski. When homomorphism becomes a liability. In *TCC*, pages 143–161, 2013.

[10] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.

[11] Murray R. Bremner. How to compute the wedderburn decomposition of a finite-dimensional associative algebra. *Groups Complexity Cryptology*, 3(1):47–66, 2011.

[12] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2013.

[13] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 325–340. Springer, 2016.

[14] Yarkın Doröz, Yin Hu, and Berk Sunar. Homomorphic aes evaluation using the modified ltv scheme. *Designs, Codes and Cryptography*, pages 1–26, 2015.

[15] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[16] Katalin Friedl and Lajos Rónyai. Polynomial time solutions of some problems in computational algebra. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 153–162. ACM, 1985.

[17] I.M. Gel'fand and V.S. Retakh. Determinants of matrices over noncommutative rings. *Functional Analysis and Its Applications*, 25(2):91–102, 1991.

[18] Israel Gelfand, Sergei Gelfand, Vladimir Retakh, and Robert Lee Wilson. Quasideterminants. *Advances in Mathematics*, 193(1):56 – 141, 2005.

[19] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[20] Craig Gentry. Computing on the edge of chaos: Structure and randomness in encrypted computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:106, 2014.

[21] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[22] Christopher J. Hillar and Darren L. Rhea. Automorphisms of finite abelian groups. *The American Mathematical Monthly*, 114(10):917–923, 2007.

[23] H. W. Lenstra, Jr. Finding isomorphisms between finite fields. *Math. Comp.*, 56(193):329–347, 1991.

[24] Jing Li and Licheng Wang. Noise-free symmetric fully homomorphic encryption based on noncommutative rings. Cryptology ePrint Archive, Report 2015/641, 2015. `http://eprint.iacr.org/`.

[25] Dongxi Liu. Practical fully homomorphic encryption without noise reduction. Cryptology ePrint Archive, Report 2015/468, 2015. `http://eprint.iacr.org/`.

[26] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey

fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234. ACM, 2012.

[27] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[28] Koji Nuida. Candidate constructions of fully homomorphic encryption on finite simple groups without ciphertext noise. Cryptology ePrint Archive, Report 2014/097, 2014. `http://eprint.iacr.org/`.

[29] Koji Nuida and Kaoru Kurosawa. (Batch) fully homomorphic encryption over integers for non-binary message spaces. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 537–555. Springer, 2015.

[30] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[31] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[32] Ronald Rivest, Leonard Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[33] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[34] Yongge Wang. Notes on two fully homomorphic encryption schemes without bootstrapping. Cryptology ePrint Archive, Report 2015/519, 2015. `http://eprint.iacr.org/`.

[35] Yu Yu, Jussipekka Leiwo, and A. Benjamin Premkumar. A study on the security of privacy homomorphism. *I. J. Network Security*, 6(1):33–39, 2008.

A Can we do better?

The characterisation could still be more powerful by modelling noise-based encryption schemes in a precise way. Our work has not resulted in a concrete result for this case, but this section contains the arguments that have been tried. The main finding is that noise-based systems may depend on an embedding in some infinite space.

Remark 2 (Topology and metrics). This is a short introduction to the main tools in this section.

A topology is a selection of subsets that are defined to be open, and they – roughly speaking – generalise the open intervals from the real line and the interior of circles in the plane, that is, all points strictly within some distance from a set centre. Open sets in a topology can be used to measure concepts such as nearness and connectedness, although one could claim that the main objective is to study what continuous functions can do.

A metric is real-valued symmetric function which is positive for all pairs of points x and y unless $x = y$. Furthermore, we also require that the triangle inequality holds.

We are unable to model the concept of growing noise using our algebraic characterisation. Hence, it cannot be used to give a natural description of somewhat homomorphic schemes. This could be solved if we were able to describe metrics or perhaps a topology on the ciphertext space. Any metric will describe a topology, but the converse is not true. The most general approach is therefore to search for a topology, and hope that it has a corresponding metric.

The motivation for this thought is Gentry's construction of a metric in his thesis [19]. Define the distance $\mathrm{d}(L, t)$ from the lattice L to a vector t by taking the minimum distance in $\mathbb{R}^n$ from a lattice vector to t. Each lattice vector is an encryption of 0, so we can measure how far a ciphertext is from being the canonical 0 encryption. Consider the following construction.

To equip $\mathcal{C}$ and $\mathcal{P}$ with a reasonable topology, we make a few assumptions. All operations should be continuous (this corresponds with the definition of topological groups and rings), and encryption and decryption should be continuous functions between the spaces. It is also reasonable to require that each equivalence class is an open set, i.e. having a neighbourhood around any ciphertext inside it. If encryption can reach the whole corresponding equivalence class, then $\mathcal{P}$ has the discrete topology, since the preimage of an open set must be open. The discrete topology hardly gives any information.

This also proves that $\mathcal{C}$ is disconnected. The complement of an equivalence class is closed by definition. It is also expressible as the union of all other classes, hence open. Hence there are $|\mathcal{P}|$ clopen sets in $\mathcal{C}$. For the moment being, we don't know anything more about the structure inside these sets.

For the sake of argument, assume that $\mathcal{C}$ is finite, as it is in classical group based constructions, or would be if we were working in some finite extension of $\mathbb{Z}/n\mathbb{Z}$.

It is a fact that a finite topological space is metrisable if and only if it is discrete. Hence, we can only measure how far a ciphertext is from its distinguished representative if we sacrifice all other topological structure on the space.

To relax the requirements, consider a pseudometric instead. A pseudometric does not require non-degeneracy, i.e. the distance between two points may be 0 even if they are different points. One can then define a pseudometric on $\mathcal{C}$ by $\mathrm{d}(x, y) = 1$ only if the points have neighbourhoods not contained in each other, otherwise 0. This, however, means that the distance between all points in the same equivalence classes have distance 0, so the topology will not give any information about the noise.

These arguments together seem to suggest that it is too restrictive to assume that $\mathcal{C}$ is finite. Further noise-based constructions of FHE schemes should therefore continue to focus on ciphertexts inside some infinite set. This is precisely what happens when existing schemes use rounding, as well as describing secret keys as rational numbers instead of inverses in a finite ring.

We ignore the topology for now, and restrict our attention to metric spaces. Assume that our scheme has a function f that can embed the noise into $\mathbb{R}^n$ for some n. In the case of ElGamal, this would be to output the discrete log of the first coordinate, whereas for the DGHV [33] scheme, it could be to identify the small random noise, by reducing modulo the key and subtracting the plaintext. Generally, it is (part of) the random input used to sample the random encryption of 0.

Now define $\mathrm{d} : \mathcal{C} \times \mathcal{C} \to [0, \infty]$ as follows

$$\mathrm{d}(c_1, c_2) = \begin{cases} \infty & \text{if } \delta(c_1) \neq \delta(c_2) \\ \|f(c_1) - f(c_2)\| & \text{otherwise} \end{cases}$$

A metric that can also output infinite distances is sometimes called an extended metric, and does not alter any of the usual properties of ordinary metrics. As it is becoming more common, we will simply call it a metric. Notice that d is a metric if any only if f is injective. That may not be the case, and then d is a pseudometric, which is sufficient, albeit not optimal.

One could circumvent this problem by stating that f should output all randomness that went info the ciphertext. Then it is injective. However, then we might lose our main goal, which is to measure the noise. In the case of the symmetric DGHV scheme, it would be a tuple with the coefficient to the key along with the small random noise. For the ordinary 2-norm, the first component would dominate the second, leaving the distance function useless.

For the special case of RLWE, we have the canonical embedding into $\mathbb{C}^n$ [27], in which one can compute distances properly.

Paper III

A roadmap to fully homomorphic elections: Stronger security, better verifiability

Kristian Gjøsteen and Martin Strand

ePrint 2016/105, published in the proceedings of Voting'17: 2nd Workshop on Advances in Secure Electronic Voting

A roadmap to fully homomorphic elections: Stronger security, better verifiability

Kristian Gjøsteen and Martin Strand

Department of Mathematical Sciences, NTNU
{kristian.gjosteen, martin.strand}@ntnu.no

Abstract

After the trials of remote internet voting for local elections in 2011 and parliamentary elections in 2013, a number of local referendums has renewed interest in internet voting in Norway.

The voting scheme used in Norway is not quantum-safe and it has limited voter verifiability. In this case study, we consider how we can use fully homomorphic encryption to construct a quantum-safe voting scheme with better voter verifiability.

While fully homomorphic cryptosystems are not efficient enough for the the system we sketch to be implemented and run today, we expect future improvements in fully homomorphic encryption which may eventually make these techniques practical. This study explains which improvements are needed, both in terms of performance and features.

1 Introduction

Norway conducted trials of remote internet voting for the 2011 local elections and the 2013 parliamentary elections. The government discontinued the trials in 2014, but a number of local referendums in 2016 has caused renewed interest in remote internet voting, especially for referendums.

There are two issues with the scheme used in 2013 that should be improved. The scheme is not quantum-safe, and voter verifiability is weak due to an auditing protocol that can only be run by accredited organisations. This is a study to see if we can improve on both of these shortcomings concurrently.

It is unclear if a sufficiently large and reliable quantum computer will ever be built to threaten the security of discrete logarithm-based systems, but the mere possibility that the encryption protecting ballot confidentiality may be compromised in 10–30 years from now is a serious problem that needs to be adressed.

Verifiability is difficult in Norway for two reasons. Revoting is used as an anti-coercion tactic, and Norwegian ballots are sufficiently complicated to allow Italian attacks, i.e., marking a ballot with a number of insignificant yet unique changes. Also, the entire ballot is required for the count, so the election can not be considered as a collection of independent races. Voter verifiability is in general not considered to be important by the Norwegian electorate (polls and other studies generally finds high levels of trust in Norwegian elections [21]), but given recent discussions about machines counting paper ballots, better voter verifiability than in the 2013 scheme would still be useful.

There are many schemes in the literature that achieve better voter verifiability than the 2013 scheme, but in general, these are not quantum-safe and do not facilitate very complicated ballots. All of the mainstream fully homomorphic schemes are believed to be quantum-safe.

While the 2013 protocol [14] exploited the multiplicative structure of the ElGamal scheme, a fully homomorphic scheme can allow us to use both addition and multiplication. This enables much more flexible computations, which means that we can arrange the decryption and counting process such that it is more voter verifiable.

1.1 Alternative approaches

There have been earlier attempts at completing election tallies while the ballots are still encrypted, but not at this level of complexity. Salamonsen [24] tried to apply Pailler encryption to Norwegian county elections, possibly the easiest variant, and timed the effort needed to compute ciphertexts and the necessary zero knowledge proofs, clocking in at between 2 and 5 hours of work for the voter. Peeking ahead to Section 8, we see that our solution is far more efficient than this.

Benaloh et al. [4] have described how one can use single-operation homomorphic encryption to tally a single transferable vote election. However, we tackle a more intricate problem in this work that cannot be solved with the same techniques. Chilotti et al. [8] have constructed a LWE based voting system in detail, but assume that their bulletin board is honest. We remove that restriction, and also get a scheme that can handle more complex (yet compact) ballots.

1.2 Contributions

At a theoretical level, we are exploring the limits of practical applications of fully homomorphic encryption. The idea of FHE was first proposed in 1978 [23], but was first properly realised with Gentry's 2009 breakthrough [12]. There have been several proposed applications [1, 20], but many of those are purely theoretical due to the tremendous amount of redundant data that would be needed per user. This application is realistic, yet pushing the boundaries of what FHE at its current state can be applied to.

Next, this is a case study on how FHE could be used to make future Norwegian elections both quantum-safe and more voter verifiable. Our proposed protocol is borderline practical, at least taking into account the number of zero knowledge proofs the existing protocol must check, and it can be further optimised by implementation experts. We provide some experimental data to give a rough estimate of the computation efforts needed. There are still some primitives lacking before this roadmap can be implemented completely. We expect further progress in fully homomorphic encryption, which means that this protocol can eventually become practical in the not-too-distant future.

1.3 Organisation

The paper is organised as follows. The next section introduces lattice cryptography, fully homomorphic encryption and the BGV scheme in particular. We also discuss some of the primitives we will be using throughout the paper.

Section 3 discusses Norwegian local elections and the security expectations by the public. There are many ways to count ballots, and in Section 4 we describe one algorithm that consists of a mix of easily auditable procedures and some fairly simple circuits, where some of the circuits carry out a partial audit of other procedures. This is followed by a more formal discussion of the security model the final protocol must satisfy. In Section 6, we use the already developed counting algorithm in a larger protocol, and apply encryption to the data and evaluating the circuits homomorphically. After that, in Section 7, we argue that this scheme is secure, and discuss how it improves the previous versions. Finally, we discuss parameter selection, and give a rough idea of the order of magnitude of runtime if this system is to be implemented.

2 Preliminaries

Lattices have long been important in cryptography, both as a tool to attack systems and as basis for new cryptographic systems. Two recent developments have made such lattice-based cryptography even more important, namely the development of *fully homomorphic encryption* (FHE) and the renewed interest in *quantum-safe cryptographic schemes.*

Fully homomorphic encryption is a form of encryption that allows one to do certain computations on encrypted data. While first defined in 1978 [23], the first theoretical solution was Gentry's breakthrough 2009 construction [12].

2.1 Fully homomorphic encryption

Fully homomorphic encryption allows us to evaluate a function described by a *circuit* on a set of encrypted inputs, resulting in an encryption of the result we would have gotten if we instead just computed the circuit on (unencrypted) inputs. *Compactness* – that the size of the output ciphertext is independent of the number of inputs and the circuit evaluated – is needed to get interesting solutions.

Lattice problems such as *(ring) learning with errors* ((R)LWE) are generally considered to be hard to solve, even for a large quantum computer. Lattice cryptography has seen a tremendous development since Regev [22] found a quantum reduction from the natural lattice problems of finding the shortest vector (SVP) or finding a short basis of independent vectors (SIVP), to LWE.

Several authors have used LWE and RLWE to create fully homomorphic encryption. The main ideas remain the same as in Gentry's original construction. The plaintext is masked with *inner* and *outer* randomness, where the innermost is denoted as noise. One can then typically perform additions and multiplications, though sometimes a NAND gate must be used. However, for each operation, the noise level grows. When it reaches the same size as the outer randomness, the ciphertext is no longer decryptable. Multiplications are usually expensive in terms of noise, causing the noise to grow quickly, while additions are cheap.

The noise problem can sometimes be solved using a technique called *bootstrapping*. Since the scheme is homomorphic, one can apply the decryption circuit to the ciphertext, but using the key in an encrypted state. The result is a new encryption of the same value, but where the noise is independent of the original noise level. One can fine-tune the parameters such that the resulting ciphertext has a lower noise level than the original one. However, the bootstrapping process is computationally expensive, so it is more common to select parameters based on the function one wants to compute, so as to achieve a designated *[multiplicative] depth* (so-called levelled fully homomorphic encryption). Many schemes have also provided solutions for limiting the noise growth, increasing the practical depth that can be reached without bootstrapping.

Formally, a FHE scheme consists of algorithms (Gen, Enc, Eval, Dec). The unusual member of the set is Eval, which accepts a special evaluation key evk, a circuit $\mathcal{C}$ and a number of ciphertexts $c_1, \dots c_n$ such that

$$\mathsf{Dec}(sk, \mathsf{Eval}(evk, \mathcal{C}, c_1, \dots, c_n)) = \mathcal{C}(\mathsf{Dec}(sk, c_1), \dots, \mathsf{Dec}(sk, c_n)).$$

We will simplify this notation whenever it is convenient, and often just express the circuit (or function) directly on the ciphertexts, even when we really want them to be applied to the encrypted data.

Remark 1. We clarify some terminology for the benefit of the reader. The term *fully homomorphic encryption* has two meanings, either that the scheme in question can process any circuits of any depth (typically by using bootstrapping) or that it can evaluate two operations, in contrast to group homomorphic schemes like ElGamal. In principle, we only need a levelled homomorphic scheme, but will use the word fully to denote the concept.

2.2 The BGV cryptosystem

The presentation of the circuits in this paper is fairly general, but we have made sure to only use features supported by the BGV scheme. We refer to the original publication [5] for the technical details, but quickly introduce some of the high-level features provided by the scheme.

Plaintext slots Following an idea of Smart and Vercauteren [26], one can pack several plaintexts into a single ciphertext and do SIMD operations (single-instruction multiple-data) on the vector of plaintexts. The advantage is that one saves space, and that one can perform operations on tuples of data in the time it would to do it on a single value. All slots must have the same size. The plaintext space of the BGV scheme can thus be set to any space $\mathbb{F}_{q^\ell}^n$ for some integers q, ℓ and n, where n denotes the number of slots.

Noise management The authors use a system of modulus reduction for each multiplication, such that the noise increases slower than it would otherwise do. Hence, one can have smaller ciphertexts. The number of times one can do the modulus reduction decides the maximal multiplicative depth.

Key switching In addition to reducing the modulus, one can also efficiently transform ciphertexts from one key to another. The transformation key is dependent on both of the private keys, so knowledge of one private key is not sufficient to produce the switch key.

Note that these features together satisfy the setup requirements in Section 5.1.

FHE has reached a level of maturity where it is practical for some applications and security levels [10]. We expect performance to increase still further. The BGV [5] cryptosystem has been implemented by Halevi and Shoup [16], and among others, Microsoft has also worked with implementations [19].

2.3 Tools

We will describe our counting algorithm in terms of a small set of standard algorithms:

Eq On two values from $\mathbb{F}$, it outputs 1 if they are equal, otherwise 0.

In If given a value $a \in \mathbb{F}$ and a set $A \subset \mathbb{F}$, it outputs 1 if $a \in A$, otherwise 0.

Normalise For any nonzero input from $\mathbb{F}$, return 1. On 0, return 0.

It is possible to express these algorithms as circuits over any field.

A circuit for equality checking for the BGV scheme has been provided by Kim et al. [17]. We will denote it as a function

$$\mathsf{Eq}(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise.} \end{cases}$$

The multiplicative depth for equality checking in the BGV scheme is given as $\lceil \log(q-1) \rceil + \lceil \log \ell \rceil$ where q is the characteristic of the field and ℓ is the order of the extension. Although this will be a fairly high number, most equality checks will turn out to run in parallell, which will help keep the noise under control.

Define $\mathsf{In}(a,S) = \sum_{s \in S} \mathsf{Eq}(a,s)$, which will return 1 if and only if a is a member of set the S. Note that checking the $\leq$ operator can be interpreted as a special membership test.

Normalisation can be done using the algorithm of Kim et al. [17], which depends on the Frobenius automorphism $x \mapsto x^q$. This particular exponentiation can be done for free in the BGV system.

The most intricate primitive we need is an efficient zero knowledge proof or argument for correct decryption. These are well known for schemes such as ElGamal [7], but for FHE, they only become efficient when applied to many ciphertexts concurrently [2,3]. However, much of the work can be done ahead of time, and the protocol also supports distributed decryption, which will essentially guarantee the security of the complete scheme. Note that one possible instantiation of the following protocol would only require the verifiable decryption of a single ciphertext. Providing an efficient zero knowledge proof for that case can still be considered an open problem.

We later provide optional sections sketching how to compute the distribution of representatives using FHE. For that we need algorithms for sorting and division by rational numbers. Emmadi et al. [11] analysed a number of algorithms and concluded that Odd-Even Merge sort would work best for the Smart-Vercauteren scheme [25]. It is reasonable that some of their results will apply to the BGV scheme as well. Chung and Kim proposed that one can use a continued fraction representation of rational numbers to reduce the storage requirement for rational numbers with a given precision, and also described how to perform divisions [9]. Çetin et al. [6] have demonstrated that it is possible to compute fractions and even square roots by applying numerical methods to the encrypted data.

3 Norwegian elections

The main idea in this paper is to do most of the ballot processing as computations on encrypted data. This means that we need to give an arithmetic circuit for counting. In order for this circuit to make sense, we first need to explain the mechanics of Norwegian elections in some detail. In all Norwegian elections, each district elects multiple members roughly as follows. Parties nominate lists of candidates for each district. The voter chooses one of these party lists as their ballot. Here we only discuss the details of the local elections, since these are the most complicated and are also most realistic for renewed interest in internet voting.

To vote in a municipal election, the voter must first pick a party list. Choosing a given party list gives that party a certain number of *list votes*. The total number of list votes in a district will determine the number of representatives each party gets.

The voter can then give *person votes* to zero or more candidates on the list. The number of person votes each candidate gets determines which candidates are actually elected as members for that party.

The party may also *prefer* a subset of their candidates. These candidates will then automatically get an additional number of person votes equal to 25 % of the number of ballots submitted for that party.

Optionally, the voter can also *write in* a certain number of candidates from other party lists. Arbitrary write-ins are not allowed. These candidates will then receive person votes. However, writing in a candidate from a different party list will also transfer a list vote from the voter's party of choice to the party that the write-in candidate belongs to.

Consider the example ballot from Figure 1. If the number of members to be elected is 29, each submitted ballot will initially give 29 list votes to the indicated party, in this case the Crypto Party. But on this ballot, the voter has listed four candidates from other lists, which means that the Crypto Party only gets 25 list votes, while the Hacker Party (HP) gets two list votes, and the Analyst Party (AP) and the Eavesdropper Party (EP) get one list vote each. When tallying, one first counts the list votes each party gets, and decide how many representatives each party gets using a modified Sainte-Laguë's method. The original Sainte-Laguë's method is to create a table with one column for each party, and with each party's number of list votes written in the first row. In the ith row, the number from the first row of the same column divided by $2i - 1$ is written. The k representatives to be elected are then distributed to the parties with the k largest numbers. The modification used in Norway is that the numbers in the first row are divided by 1.4 before distributing candidates, a modification that slightly favours larger parties.

The next step when tallying is to decide which candidates are

Crypto Party
Candidate list for the local elections 2019

1. ⊠ **Gaius Julius Caesar**
2. ☐ **Giovan Battista Bellaso**
3. ⊠ Al-Khalil ibn Ahmad al-Farahidi
4. ☐ Auguste Kerckhoffs

⋮

34. ☐ Charles Wheatstone

Candidates from other lists

NAME	PARTY
Alan Turing	HP
Konrad Zuse	HP
Charles Babbage	AP
Eve Mallory	EP

Figure 1: An example ballot for a local election

actually elected. To do this, one counts all person votes given by voters (either to a candidate on the party list, or by writing in a candidate from another party list) and the person votes resulting from party preference. The candidates are then ranked according to the number of person votes received. In the event of a tie, the order of the candidates on the party list is used.

Today's manual counting is being done so that as to minimise the number of discarded ballots. Inconsistent alterations to the ballot are ignored, as long as the voter's main intent remains clear. We strive to achieve the same in this work. Our general rule will be that any ballot that can be tied to a party, is registered as such. It will only be completely invalidated if it is impossible to decide which party list the voter wanted to choose. Interestingly, we get a large performance benefit from this practice, in particular by stating that if the voter writes in the same candidate from a different list multiple times, then the candidate only receives a one extra vote, but multiple list votes are transferred to the candidate's party. We define the ideal counting functionality to follow the same rule.

3.1 General security context

Privacy is important in Norwegian elections. The ballot should obviously be confidential, but even the list of who voted is considered confidential in Norway. In particular, this means that any voter verifiable scheme that reveals the identities of the voters is unacceptable in Norway.

A second constraint is coercion resistance. It seems the main defence against coercion in Norway must be revoting, and the revoting could possibly be paper revoting. Paper voting cannot involve any secrets or other material from previous electronic voting, and a paper ballot should also supercede any subsequent electronic ballot submission. Italian attacks are easy in Norway, since adding a random set of marks to a ballot will most likely make it unique and have negligible electoral effect. This means that we must avoid publishing complete ballots, even for verification purposes. Today, the election authorities publish lists for each municipality consisting of the number of ballots for each party, list votes received and sent from the list, and the number of write-ins and extra votes for each person. There are also separate lists tracing the number of write-ins for each candidate from the other party lists [18].

It seems that electronic voting must coexist with paper voting for the forseeable future. This means that the electronic count must somehow combine with the paper count before the final result is declared. All paper votes must be digitalised and encrypted if one wants to do an all-encrypted computation of the election result.

4 Counting algorithm

Recall the algorithms Eq, In and $\mathsf{Normalise}$ that we discussed in Section 2. This section is dedicated to explaining how one can count the ballots in the election, and expressing these algorithms as polynomials over a sufficiently large field $\mathbb{F}$, also known as *circuits*. These circuits will be suitable for fully homomorphic encryption, which we will apply in Section 6.

Let m be the number of parties taking part in the election and n be the total number of candidates from all the parties. Furthermore, let v be the maximal number of voters from the voting district, and let k be the number of candidates. The voter may list up to $k/4$ candidates from other lists, which also places an upper bound on how many list votes that may be transferred from the chosen party to another.

4.1 Ballot expansion and verification

The first algorithm we describe is one that will take as input a ballot tuple

$$b = (p, s_1, \ldots, s_{n_p}, e_1, \ldots, e_{n'}),$$

where p is the index of the chosen party list. Assume there is a canonical list of all candidates from all parties, and that each party has no more than n_p candidates. Then s_1 through s_{n_p} are slots that can be filled with (unique) indices for candidates belonging to party p and that the voter wants to give a person vote. Finally, $e_1, \ldots, e_{n'}$

are the indices of the candidates from other lists representing ballot write-ins.

This format is unsuitable for tallying, so we want to change it to an expanded form. However, we prefer having the voter submitting his ballot in the above way, due to ease of verification. Almost all combinations of values will constitute a valid, or at least not harmful, ballot, so we design the algorithm to tolerate small errors, and turn a completely malformed ballot into a blank vote.

As output, we get the expanded ballot

$$eb = (p_1, \dots, p_m, p'_1, \dots, p'_m, p''_1, \dots, p''_m, c_1, \dots, c_n).$$

A valid ballot has the following characteristics

- Exactly one of the first m coordinates is 1, and the rest are 0. The position of the 1 determines which party the voter selected. Let the single 1 be in the ith position.
- At most one of the following m places is non-zero, and if so holds an integer counting the number of list votes to deduct from the party. The non-zero number must be in position $m + i$.
- The third section of m shows to which parties the deducted list votes are distributed, under the conditions that they sum up to the number of list votes given away, that the sum is smaller than $k/4$, and that no list votes are distributed back to the party chosen by the voter. In particular, this means that position $2m + i$ must be 0.
- Finally, the last section consists of values in $\{0, 1\}$, where a 1 indicates that the voter have given the candidate a person vote or as a write-in from a different list.

The zero vector is called the blank ballot.

We now develop the ballot expansion circuit, which is summarised in Algorithm 1.

- We first want to identify the correct party on the ballot submitted by the voter, and place a 1 in the correct coordinate. This is

Algorithm 1 Transform tuples and verify ballot

```
1:  for $i \leftarrow 1$ to $m$ do
2:      $p_i \leftarrow \mathsf{Eq}(i, p)$
3:  $\rho \leftarrow \sum_{i=1}^{m} p_i$
4:  for $i \leftarrow 1$ to $m$ do
5:      $p''_i \leftarrow \rho\left(\sum_{j=1}^{n'} \mathsf{In}(e_j, P_i)\right)$
6:  for $i \leftarrow 1$ to $m$ do
7:      $p'_i \leftarrow \mathsf{Eq}(i, p) \cdot \sum_{j=1}^{m} p''_j$
8:  for $i \leftarrow 1$ to $n$ do
9:      $t_1 \leftarrow \mathsf{Eq}(i, e_1) + \ldots + \mathsf{Eq}(i, e_{n'})$
10:     $t_2 \leftarrow \mathsf{Eq}(i, s_1) + \ldots + \mathsf{Eq}(i, s_{n_p})$
11:     $c_i \leftarrow \rho \cdot \mathsf{Normalise}($
            $(1 - \mathsf{Eq}(p, P_{c_i}))t_1 +$
            $(\mathsf{Eq}(p, P_{c_i}))t_2)$
12: return $(p_1, \ldots, p_m, p'_1, \ldots, p'_m, p''_1, \ldots, p''_m, c_1, \ldots, c_n)$
```

straightforward since there can be at most one match between an increasing index and a number provided by the voter. Note that if the party indicated by the voter is out of range, then the ballot will effectively become blank.

- We compute a value ρ, which is 1 if and only if the ballot contains a valid party, and 0 otherwise. Since a ballot with an invalid party should result in a blank ballot, this intermediate result greatly simplifies the remaining computations.

- Next we compute all list votes for other parties, as described in the third section above. Let $\{P_i\}$ denote all parties taking part in the election, and let p''_i be the number of list votes transferred to party P_i. By abuse of notation, let P_i also denote the set of indices for the candidates of that party. The number p''_i of list votes transferred to party P_i is therefore a count of how many of the candidates on the ballot that belong in P_i. We multiply with ρ to ensure that the value is non-zero only if the voter has selected a party.

- The number of outgoing votes is the sum of all the incoming list votes to other parties. To ensure that the sum is in the correct position, we do the equivalent of an `if`-statement. Using the same technique as we did to place a 1 at the right party, we can use that bit to select the right slot, and then scale it with the sum of all p''_i.

- Finally, we correctly place the person votes and write-ins on candidates. For each candidate i, let the bit $t_1 = 1$ if and only if i is equal to any of the write-in candidates $e_1, \ldots, e_{n'}$, and let bit $t_2 = 1$ if and only if i is equal to any of the party candidates $s_1, \ldots, s_{n_p}$. These can be computed using two sums. The first will be 0 or 1 by the same reason that only one party can be selected. The second can be greater if the voter lists the same candidate multiple times, so we eventually need to normalise the whole sum. One can only give write-in votes if the candidate belongs to a different party, so we must perform an inequality check in front of t_1, so ensure that the candidate i does not belong to the party p. Likewise, only candidates belonging to the party can receive person votes, so we do an equality check in front of t_2, which is 1 if and only if candidate i belongs in the selected party p. Sum these two and normalise to 1, which means that even if the voter wrote the same candidate multiple times, it still only counts once, and a multiplication by ρ verifies that a party list is chosen.

To summarise, the only way to invalidate a ballot is to choose a party index out of bounds, and then the whole ballot is expanded into the blank ballot. Assume p was out of range. Then $\rho = 0$, so all $p''_i = 0$. Also, p'_i can only be nonzero if some i equals the party index. Hence, also all p'_i will be zero when the party was invalid. For the person votes, ρ will null all votes unless a valid party is chosen. Otherwise, $\rho = 1$, and we verify that $\sum_{i=1}^{m} p''_i - \sum_{i=1}^{m} p'_i = 0$ by noting that exactly one of $\mathsf{Eq}(i, p)$ will be 1.

Finally, we notice that each candidate at most gets a single person vote, due to the normalisation, so the expanded ballot satisfies the above validity requirements. This discussion can be summarised in the following lemma.

Lemma 1. *Let eb be the expansion of b under Algorithm 1. Then eb is always a valid (or blank) ballot and represents the same vote as b if b is valid.*

4.2 Optional feature: Reject all malformed ballots

There is a possibility of inconsistency in the ballot if the voter lists the same person from another party twice. The result will be that the person is given only a single person vote, while several party votes are transferred to that party. However, the voter is still bounded by the maximal number of votes that can be transferred, and he could achieve the same effect by listing a different candidate which was either almost guaranteed a seat, or one with hardly any chance of getting one.

This is not in keeping with the current voting regulations, but showcases that variations with hardly any consequence, can influence the computational complexity. We can catch this scenario by computing

$$\mathsf{Normalise}\left(\prod_{i \le j}(e_i - e_j)\right)$$

and store the result in a verification bit u_e that can be used to select valid ballots.

Jumping ahead, we can accept the situation given in the previous subsection if the behaviour is predictable and return codes are generated by the expanded ballot so that the voter can inspect the change. The official software should never allow this malformed ballot to be posted so a user would need to go out of its way to be able to submit such a ballot.

4.3 Selecting ballots and tallying

Whenever a new ballot is submitted, the ballot box expands it using Algorithm 1 and stores it with the identity of the voter and a strictly increasing sequence number. Recall that we need to cancel all current and future electronic votes if a person votes on paper. If the voter submits a paper ballot, the ballot box creates a blank ballot under the voter's identity and assigns a greater sequence number than will be used for any real ballot. For simplicity, we denote that number by ∞.

The ballot box replaces all identities with pseudonyms once voting has closed, and sorts the list of tuples lexicographically, i.e. first by pseudonym *pv* and then sequence number s. The list is then randomly shifted, and one should consider the first element as the 'next' element of the last item.

At the end, the latest ballot from each person is to be selected as the final, and all others should be discarded. We therefore introduce a function to verify that all entries from the identity are correctly sorted by sequence number. For identities v_i, v_j and timestamps s_i, s_j, define

$$f(v_i, v_j, s_i, s_j) = \begin{cases} 1 & \text{if } v_i \neq v_j \text{ or } s_i < s_j\text{; and} \\ 0 & \text{otherwise} \end{cases}$$

and apply f to consecutive tuples in the list. We get 0 if there is an error in the sorting by timestamps. All values should be 1, so we can multiply all u, and the product will be 1 if and only if the list is correctly sorted. We return to how we verify that the ballots from the same voter have not been partitioned and spread out in order to count more than one.

Next we want to select the last entry for each person. Define

$$g(v_i, v_j) = \begin{cases} 1 & v_i \neq v_j\text{; and} \\ 0 & \text{otherwise,} \end{cases}$$

and run g over all consecutive entries. It will only output 1 for the last entry of each identity, and we can multiply the ballots with the output of g to select exactly the ballots those that are to be counted-

To complete the tally now, one can simply compute the sum of *all* submitted ballots, where each is scaled by the bit described above.

Lemma 2. *Consider the following ideal functionalities. Let* select *be a function from the set of paper ballots and lists of* (v, s, b)*, to lists of* b*. It correctly chooses the ballots that should be counted. The*

Algorithm 2 Storing and selecting ballots to be tallied

Precondition: The ballot box has initialised a list L of ballots
Postcondition: $u = 1$

```
 1: function Add-Ballot($v, eb$)
    $v$ is voter identity, $b$ is the ballot, $eb$ is the expanded ballot
 2:     Create tuple $(v, s, b, eb)$, s.t. $s$ greater than any finite $s$ in $L$
 3:     Insert $(v, s, b, eb)$ in $L$
 4: function Cancel-Electronic-Ballots($\{v\}$)
 5:     Let $b$ be a blank vote and $eb$ be the expanded blank ballot
 6:     for all $v$ in $\{v\}$ do
 7:         Add tuple $(v, \infty, b, eb)$ to $L$.
 8: function Prepare-Ballot-List
    $L_0$ is the list of ballots
 9:     for all unique identities $v$ do
10:         $pv \leftarrow$ (random pseudonym)
11:         Replace all occurrences of $v$ with $pv$
    $L$ is the new list $\{(v, s, b, eb)\}$ of length $N$
12:     Sort $L$ lexicographically and send privately to the auditor
13:     $k \leftarrow \mathsf{Random}(0, \dots, N-1)$
14:     Set $L = (\mathsf{tuple}_k, \mathsf{tuple}_{k+1}, \dots, \mathsf{tuple}_{N-1}, \mathsf{tuple}_0, \dots, \mathsf{tuple}_{k-1})$
15: function Check-Correctness-Of-Ballot-List
16:     for all $v_i$ do
17:         $u_i \leftarrow f(v_i, v_{i+1 \bmod N}, s_i, s_{i+1 \bmod N})$
18:     $u \leftarrow \prod_i u_i$
19:     for all $(v_i, b_i)$ do
20:         $z_i \leftarrow g(v_i, v_{i+1 \bmod N})$
21:         $eb'_i \leftarrow z_i \, eb_i$
22:     return $(eb'_1, \dots, eb'_l, u)$
23: function Tally($L' = (eb'_1, \dots, eb'_l)$)
24:     return $\sum_{i=1}^{l} eb'_i$
```

function tally *takes as input a list $\{b\}$ of ballots and returns the correct result. Their concatenation then computes the correct result from all the electronically submitted ballots after cancelling with respect to paper ballots. Let* res *denote their result.*

Let $\mathcal{C}_1$ be the circuit described in Algorithm 1, let $\mathcal{C}_2$ be the circuit we get from Algorithm 2 up to and including Check-Correctness-Of-Ballot-List*, and let $\mathcal{C}_3$ be the final tallying circuit. Futhermore, let L_1 be a list $\{(v, s, b)\}$. Let $L_2 = (\{(v, s, b, eb)\} = \mathcal{C}_1(L_1)$, $(L_3, u) = \mathcal{C}_2(L_2, \{v\}_{\mathsf{paper}})$ and $r = \mathcal{C}_3(L_3)$. If $u = 1$ and L_3 is correctly sorted, then* $\mathsf{res} = r$.

The correctness of $\mathcal{C}_1$ follows from Lemma 1, the rest follows from the above discussion.

4.4 Optional: Computing the distribution of representatives

Recall from Section 2.3 and 3 that it is theoretically possible to compute the media-ready outcome of the election. We briefly sketch how.

From previously in the section we have a tuple

$$eb = (p_1, \ldots, p_m, p'_1, \ldots, p'_m, p''_1, \ldots, p''_m, c_1, \ldots, c_n),$$

which now is summed. The first block of coordinates now says how many times each list was chosen by the voters. We can convert this into list votes by multiplying by the number of seats k in the council. The final number of list votes for party i is then

$$l_i = k \cdot p_i - p'_i + p''_i.$$

Compute all such numbers, divide by the Sainte-Laguë numbers 1.4, 3, 5, 7, ... and create a long list (with all parties) consisting of pairs of a party identifier p and the divisions of l_i. We can do this since there exists a circuit to handle rational numbers. Finally sort the list descending by the divisions using the sorting circuit. The top k pairs then identify the seat allocation.

To pair seats and candidates, cut the list of candidates into party lists. Some candidates receive the extra votes, 25 % of the number p_i. Add this to the respective candidates and their person votes and write-in votes, and sort by number of votes.

5 Security model

We model our system with the same players that already existed in the Norwegian e-voting project, namely the voter V with her computer and mobile phone, a ballot box B, a receipt generator R, a decryption service D and an auditor A. We quickly explain the existing motivation before proceeding. For more details, see Gjøsteen [14].

A premise for Gjøsteen's work is that the users may not be in control of their own equipment, for instance due to malware. One should therefore distinguish the voter's intention and what the computer actually does. When the ballot box receives an encrypted ballot from the voter's computer, it transforms and partially decrypts the ballot, and forwards it to the receipt generator. Then the transformed ballot is completely decrypted, and the correct receipt code is sent by SMS to the voter's mobile phone.

Both the ballot box and the receipt generator give the auditor a log of everything they have seen, so that the auditor can compare logs and make sure no one of them is ignoring information seen by the other. Any information dropped by the ballot box should ideally be detected by the voter, because of a text message. (The soundness of this protocol is based on an assumption that the phone is independent of the device used to vote. While this may have been an acceptable assumption when the system was first introduced, it is less so today. Finding another solution may be necessary, but is outside the scope of this paper.)

Next consider what happens when the election closes. Then the ballot box should provide ciphertexts to the decryption service, which outputs the public result of the election. The auditor verifies that the decryption service got the right ciphertexts from the ballot box, and that the output was correct.

5.1 Setup assumptions

We assume the existence of the following algorithms.

$\mathcal{K}$ takes in a security parameter 1^λ, and outputs a tuple (pk_1, sk_1, sk_2, evk, tk), where pk_1 is the public encryption key, sk_1 and

sk_2 are private decryption keys, evk is an evaluation key and tk is a transformation key to switch from decryption under sk_1 to sk_2.

$\mathcal{S}$ takes in all voters $\{v_i\}$ within the vote counting district and system key material and outputs a family of personalised functions $\{h_{v_i}\}$ and other receipt material described in Section 6.1.1.

$\mathcal{E}$ takes in pk_1 and a ballot b and outputs a ciphertext c.

$\mathcal{T}$ takes in a ciphertext c which can be decrypted by sk_1 and a transformation key tk, and outputs a ciphertext c' that can be decrypted by sk_2.

$\mathcal{E}val$ takes in an allowed circuit $\mathcal{C}$, ciphertexts $c_1, \ldots, c_n$ and the evaluation key evk, and outputs a ciphertext c' which encrypts the output of $\mathcal{C}$ on the decryptions of $c_1, \ldots, c_n$.

$\mathcal{D}$ takes in a ciphertext and a corresponding decryption key and outputs a message.

5.2 Security requirements for counting

The security requirements can be summarised in the following list. The definitions are close to those of the original protocol, in order to be able to compare apples and apples. However, some properties must necessarily be changed to correspond to our updated model. The formal definitions are available in Gjøsteen [14].

D-privacy The decryption service should not be able to correlate its input to voter identities.

Generate parameters. Let V be a voter, let $\{c_i\}$ be all ciphertexts containing ballots, and let c' be the output of the tally circuit. The distribution of c' should be independent of V.

B-privacy An adversary that knows the transformation key should not be able to say anything about the content of the ciphertext he sees.

R-privacy An adversary that controls the pre-code decryption key and sees many transformed encryptions of valid ballots should not be able to say anything non-trivial about the content of those encryptions.

One can formalise the notion by letting an adversary with access to the encryption key and the decryption key of transformed ballots submit a sequence of ballots to the simulator, who in turn encrypts either the sequence or the sequence under some permutation. The simulator transforms the ballots, and sends back original ciphertexts and transformed ciphertexts to the adversary, who must distinguish.

A-privacy An adversary that sees the insertion of blank ballots, can correlate all ciphertexts to real identities, and verifies all decryptions should not be able to learn how anyone voted.

This definition differs considerably from previous work. We play the following game between a simulator and an adversary, and probability of the adversary winning should be close to 1/2. The simulator selects a random bit b, and runs $\mathcal{K}$ and $\mathcal{S}$. The adversary gets pk_1, and produces a set $\{V_i\}$ of voters and two sets $\{m_i^0\}, \{m_i^1\}$ of corresponding ballots that will create identical election outcomes. The simulator uses the set $\{(V_i, m_i^b)\}$ to simulate an election with the adversary playing the part of the auditor. The adversary outputs a bit b' and wins if $b = b'$. The probability that the adversary wins should be close to 1/2.

B-integrity An adversary that knows all the key material, and can choose the per-voter key material, should not be able to create an identity, a ciphertext and a transformed ciphertext such that the transformed ciphertext decryption is inconsistent with the decryption of the ciphertext.

D-integrity The decryption service must not be able to alter the election outcome.

The focus in this work will be on D/A-privacy and D-integrity.

6 The voting protocol

The previous sections established a *correct* circuit for computing the election result. However, the sensitive data were left completely unprotected, and there is no overall protocol. In this section, we describe that protocol and apply a fully homomorphic encryption scheme to the data, thus securing them, while making it possible to still do all computations. The main idea is that anyone can compute the algorithms themselves, and verify that they reach the same results.

6.1 Protocol description

The protocol consists of a the same players as in Gjøsteen's original protocol [14].

- The voter's PC encrypts the ballot and sends it to the ballot box over a private channel.
- The ballot box relays the ballot to the return code generator, which signs it, and returns the signature to the ballot box, which also signs it, and returns both signatures to the voter's PC. Hence, the ballot has been seen by two independent players.
- The ballot box expands the ballot and stores it in accordingly. The return code generator and the ballot box runs the subprotocol to generate return codes to the voter, as sketched below.
- The ballot box runs the functions in Algorithm 2, and finally sends the output from Tally to the decryption service.
- The decryption service decrypts using multiparty computations and produces a proof of decryption [3].
- The auditor receives logs from all other centralised players, verifies any proofs, and runs the verification procedures outlined below.

6.1.1 Return codes

The voter is to get a receipt to prove that the ballot has been received and processed correctly. Without specification, we assume that there for each voter v exists a one-way function h_v that produces what we call pre-codes.

Ahead of the election, the key generation service applies h_v privately to all possible choices made by the voter. Next, human readable codes are associated to the output of h_v, and the table of choices and human readable codes is sent to the voter. Send the table of pre-codes and human readable codes to the receipt generator. Finally, the key generation service erases all information.

Upon receiving a ballot b from a voter v, the ballot box applies h_v to the expanded ballot eb, and sends $h_v(eb)$ to the receipt generator. The receipt generator looks up the human readable code, and sends it to the voter, who verifies.

Return codes are not the main focus of this paper, so they will not be discussed further.

6.1.2 Verification

To verify the outcome is correct, namely that the adversary has not altered any of the ballots, an auditor should complete the following steps. At the current point, we assume that all computations have been performed honestly, and we will return to that assumption at a later stage in the paper.

1. Verify that blank records with sequence number ∞ have been inserted for each voter submitting a paper ballot.

2. Check that u generated by f is 1, which guarantees correct sorting.

6.2 Encrypting data

The following list describes what data should be encrypted, and which data that must be kept secret. The complete overview is shown in Figure 6.2.

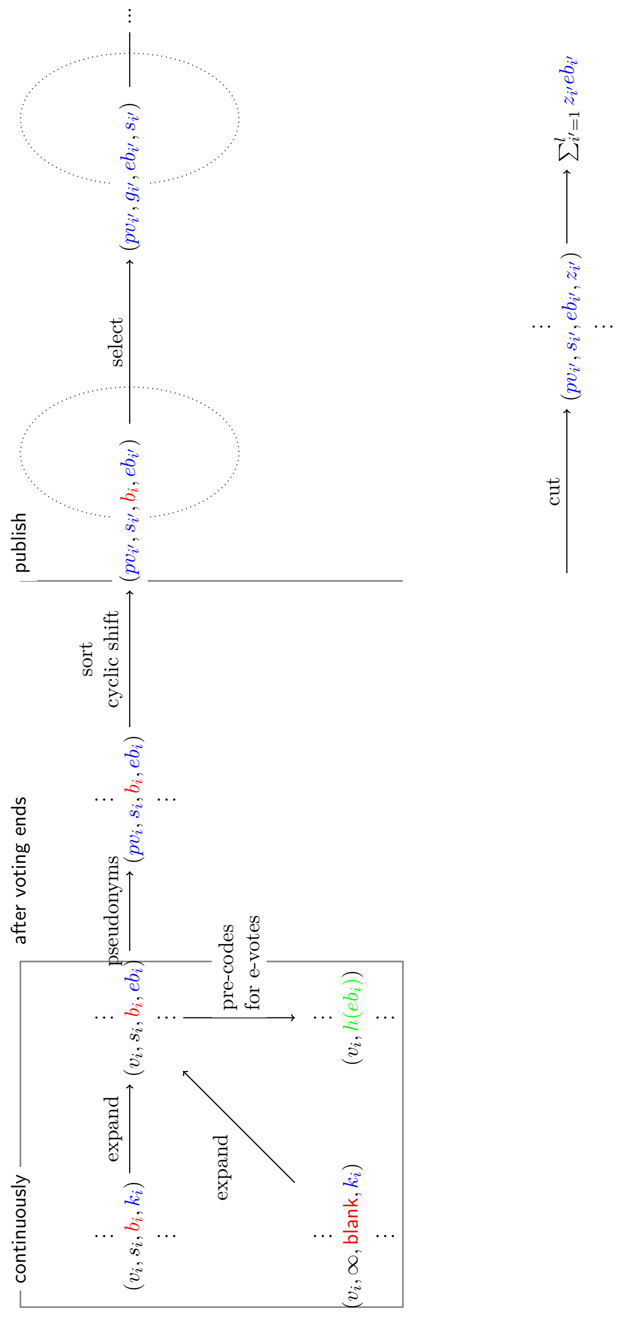

Figure 2: An overview over the complete system. Legend: Red – AES; blue – BGV; Green – BGV under new key

1. The voter encrypts his ballot using AES, and appends the AES key k encrypted under BGV.
2. The ballot box converts from AES to BGV [13], and computes Algorithm 1 on the encrypted data.
3. In Algorithm 2, the voter identity is kept secret, while the ballot is stored in its encrypted form. The shuffled list of identities is sent to the auditor through a secure channel. The sequence numbers must be encrypted, and be published along with the original AES ciphertexts with encrypted keys.
4. Finally, perform a verifiable decryption of the result and the u generated from the sorting verification.

Remark 2. If one opts to compute the validity bit u_e, one should consider decrypting it for each ballot, to save multiplicative depth.

7 Security

We verify that our scheme satisfies the security requirements from Section 5.2.

Theorem 1. *The voting system satisfies A-privacy and D-privacy.*

Proof. Recall the definitions in Section 5.2. D-privacy follows from the fact that the tally circuit outputs a single ciphertext that contains the result, as well as the sorting verification bit. Since the circuits are correct, the decryption reveals nothing else than what is revealed by the election result, which is independent of identities.

For A-privacy, assume we have an adversary with a non-negligible advantage. We can then build a distinguisher with non-negligible advantage in the IND-CPA game for the BGV cryptosystem. Upon receiving the two sets og ballots from the adversary $\{m_i^0\}, \{m_i^1\}$, use the IND-CPA simulator to encrypt the ballots on behalf of the A-privacy simulator instead of using AES and then decrypting homomorphically into BGV. The adversary can then detect which set was used to generate the ballots. Use the response from the adversary as response to the IND-CPA challenge. □

The integrity follows directly from the correctness of the circuits.

Theorem 2. *If the auditor checks that the blank ballots have been added correctly, that the sorting is correct, all public computations and the proof of decryption, then the election result is correct. This implies D-integrity.*

We now discuss some of the less quantifiable security properties we achieve. Consider public verifiability, which possibly was the largest drawback in the original system used in Norway. Originally, the users could verify that the ballot was correctly stored in the ballot box, but after that, they had to trust the auditor completely. Our gap in verifiability lies in the selection of votes to be counted, where a corrupt ballot box may insert fake votes that a corrupt auditor may choose to ignore. An honest auditor should either notice that electronic votes lack a valid digital signature, or that fake paper votes have been inserted. Even with this gap, however, our proposal is a significant improvement on previous schemes used in Norway, since any voter now can compute the election result based on the published ciphertexts, and then get a verified decryption of that result.

We defend against coercion by letting a voter revote electronically any number of times, and decreeing that a paper ballot will override any earlier or later electronic votes. This is within the requirement of the Norwegian Election Act, which states that "[t]he purpose of this Act is to establish such conditions that citizens shall be able to elect their representatives to the Storting, county councils and municipal councils by means of a secret ballot in free and direct elections." [27] Let us now consider how a coercer could be able to succeed.

It is clear that any coercer cooperating with a corrupt ballot box or auditor will be able to defeat revoting as an anti-coercion strategy. We therefore assume a coercer that sits next to the voter as she casts her ballot, and assume that the coercer is also able to record the precise ciphertext, and himself transform it into the FHE ciphertext c_i that will appear in the public records.

If the voter now revotes electronically, and the coercer afterwards returns and forces her to vote under surveillance again, then the public list of ciphertexts will reveal that the voter revoted. However, any paper vote will be sorted after the last electronic vote, so it cannot be discovered by the adversary. Also note that since the identities are replaced with pseudonyms and then sorted (while still encrypted) the attacker cannot guarantee that a paper vote will be sandwiched between an electronic vote and the first vote of someone of whom he knows the identity, making the paper ballot truly anonymous.

8 Parameter selection

Using the above algorithms, we can estimate the parameters needed for the protocol. For a conservative estimate, we can look at the numbers for the largest municipality, Oslo. There are about 500,000 eligible voters, and the last local election saw 17 different party lists with a total of 659 candidates. The city council consists of 59 members. This means that the voter can list at most 15 names from other parties on her ballot, so the greatest number we need to handle is about 7,500,000 $\approx 2^{23}$. Then equality checks will need a depth 23 circuit. We can now compute how much depth we will need after converting from the symmetric ciphertext.

- p''_i can be computed with many equality checks in parallell, and must be multiplied to a second equality check, hence depth 24. For p_i, we need a maximum depth of 23.
- p'_i is one equality check multiplied with a sum of p''_i, so we need 24 multiplications. The candidate slots are also the result of a multiplication of two equality checks.
- If we wish to compute the validity check, we require $\binom{15}{2}$ multiplications. However, they can be arranged in a tree of depth 7.
- Each value $\tilde{u}_i$ requires an equality check and an inequality check. After that, all such ciphertexts must be multiplied, which can be done with depth of approximately 20.
- The selection bit $\tilde{z}_i$ takes a single comparison, and is multiplied to the rest of the ballot, adding one level to some of the previous results.

In addition comes the depth required to send the receipt, but that is dependent of the function employed to generate the return codes.

Finally, we can conclude that no part of the computation requires a depth greater than 50.

The number of slots needed in the Oslo case is $3 \cdot 17 + 659 = 710$.

We ran the bundled general test program of `HElib` [15, 16] with the above parameters on a server running Ubuntu 14.04 on Intel Xeon 2.67 GHz processors with a total of 24 cores and 256 GB of memory. The program ran the key generation on a single core, and used a maximum of 8 cores for some sample ciphertext operations. The maximum memory usage was in the order of 20 GB. The complete process took 4:52 minutes, with key generation taking about half of that time. While this order of magnitude is unreasonable for a single voter, it may be feasible for an election system, as long as the feedback to the voter is sufficiently quick. Implementing the above algorithms efficiently is an open problem.

Acknowledgements The authors wish to thank the anonymous reviewers of Voting'17 for constructive and useful suggestions.

References

[1] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. `http://eprint.iacr.org/`.

[2] Carsten Baum, Ivan Damgård, Tomas Toft, and Rasmus Winther Zakarias. Better preprocessing for secure multiparty computation. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016*, volume 9696 of *Lecture Notes in Computer Science*, pages 327–345. Springer, 2016.

[3] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris 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`.

[4] Josh Benaloh, Tal Moran, Lee Naish, Kim Ramchen, and Vanessa Teague. Shuffle-sum: coercion-resistant verifiable tallying for STV voting. *IEEE Trans. Information Forensics and Security*, 4(4):685–698, 2009.

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.

[6] Gizem S. Cetin, Yarkin Doroz, Berk Sunar, and William J. Martin. Arithmetic using word-wise homomorphic encryption. Cryptology ePrint Archive, Report 2015/1195, 2015. `http://eprint.iacr.org/2015/1195`.

[7] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[8] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based e-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, volume 9606 of *Lecture Notes in Computer Science*, pages 245–265. Springer, 2016.

[9] HeeWon Chung and Myungsun Kim. Encoding rational numbers for FHE-based applications. Cryptology ePrint Archive, Report 2016/344, 2016. `http://eprint.iacr.org/`.

[10] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical report, Microsoft Research, February 2016.

[11] Nitesh Emmadi, Praveen Gauravaram, Harika Narumanchi, and Habeeb Syed. Updates on sorting of fully homomorphic encrypted data. Cryptology ePrint Archive, Report 2015/995, 2015. `http://eprint.iacr.org/`.

[12] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[13] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[14] Kristian Gjøsteen. The Norwegian internet voting protocol. Cryptology ePrint Archive, Report 2013/473, 2013. `http://eprint.iacr.org/`.

[15] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014.

[16] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.

[17] M. Kim, H. T. Lee, S. Ling, and H. Wang. On the efficiency of FHE-based private queries. *IEEE Transactions on Dependable and Secure Computing*, PP(99), 2016.

[18] Oslo kommune. Valgresultater 2015 - statistikk, 2015. `https://www.oslo.kommune.no/politikk-og-administrasjon/politikk/valg-2017/statistikk/valgresultater-2015`, accessed 2017-10-11.

[19] Kristin Lauter. Practical applications of homomorphic encryption, 2015.

[20] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW*, pages 113–124. ACM, 2011.

[21] OSCE Office for Democratic Institutions and Human Rights. Norway, Parliamentary Elections 9 September 2013, Final Report. Technical report, dec 2013.

[22] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.

[23] Ronald Rivest, Leonard Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[24] Kristine Salamonsen. A security analysis of the helios voting protocol and application to the norwegian county election, 2014.

[25] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[26] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.

[27] Lov om valg til stortinget, fylkesting og kommunestyrer (valgloven). `http://lovdata.no`, sep 2002. Translation at `https://www.regjeringen.no/globalassets/upload/KRD/Kampanjer/valgportal/Regelverk/Representation_of_the_People_Act170609.pdf`.

Paper IV

A verifiable shuffle for the GSW cryptosystem

Martin Strand

ePrint 2018/027, To appear in the proceedings of Voting'18:
3rd Workshop on Advances in Secure Electronic Voting

A verifiable shuffle for the GSW cryptosystem

Martin Strand

Department of Mathematical Sciences, NTNU
martin.strand@ntnu.no

Abstract

We provide the first verifiable shuffle specifically for fully homomorphic schemes. A verifiable shuffle is a way to ensure that if a node receives and sends encrypted lists, the content will be the same, even though no adversary can trace individual list items through the node. Shuffles are useful in e-voting, traffic routing and other applications.

We build our shuffle on the ideas and techniques of Groth's 2010 shuffle, but make necessary modifications for a less ideal setting where the randomness and ciphertexts admit no group structure.

The protocol relies heavily on the properties of the so-called gadget matrices, so we have included a detailed introduction to these.

Keywords: verifiable shuffle, fully homomorphic encryption, post-quantum

1 Introduction

A verifiable shuffle is used to prove that two sets of ciphertexts will decrypt to the same values, but without revealing how the sets relate. Such shuffles are well-known for group homomorphic schemes, and are still being developed and improved. Today, shuffles are particularly useful in e-voting and mixnets, in order to make it hard to correlate the input and the output of a node.

Fully homomorphic encryption has also been suggested as a useful primitive for both e-voting and private network routing. Complex voting systems in particular can take advantage of the features of fully homomorphic encryption, but the FHE toolbox is still missing a number of useful protocols. Recent development have brought shuffling for FHE within reach.

Our result starts from Groth's 2010 shuffle [10], which uses an idea from Neff [14]. A polynomial $(X-x_1)(X-x_2)\cdots(X-x_n)$ is obviously unchanged when the roots are permuted. One can then ask the prover to evaluate the polynomial at random points. The probability of two nonidentical polynomials evaluating to the same value at a random point is negligible. While later development have resulted in even more efficient protocols, Groth's 2010 approach has the advantage of simplicity and that there are few compromises: the protocol satisfies a standard soundness condition and it is honest-verifier zero knowledge. We return to the details in Section 2.

The polynomial mentioned above is hidden in a subprotocol which is used to prove correctness of a shuffle of known content. The subprotocol is completely independent of the encryption scheme, and uses only a homomorphic commitment scheme. It is important to note that the commitment scheme need only be homomorphic with respect to a single operation. We return to the selection of such schemes. The protocol is then completed by binding the secret data – which we want to prove the claim for – to the known content for which one can prove the relation.

Groth's protocol depends crucially on the fact that some group homomorphic schemes are homomorphic both with respect to the message and the randomness. For instance, the product of two ElGamal ciphertexts with messages m_1 and m_2 and randomness r_1 and r_2, will be a new ciphertext encrypting m_1m_2 using $r_1 + r_2$ as randomness. Generally, a necessary requirement for the original shuffle is that the equation

$$\mathsf{Enc}(m_0 \oplus_{\mathcal{M}} m_1; r_0 \oplus_{\mathcal{R}} r_1) = \mathsf{Enc}(m_0; r_0) \oplus_{\mathcal{C}} \mathsf{Enc}(m_1; r_1).$$

holds, where $\oplus_{\mathcal{M}}$, $\oplus_{\mathcal{R}}$ and $\oplus_{\mathcal{C}}$ are the algebraic operations used in the message, randomness and ciphertext groups respectively. Note that $r_1 \oplus r_2$ is an equally likely randomness as either r_1 or r_2.

The noise-based homomorphic schemes do not satisfy the above requirements, since the ciphertext spaces usually are far from being groups at all. The reason is the noise management; even sufficiently many additions will eventually make the ciphertext decrypt to the wrong value, there need not be an identity element, and associativity may not hold, especially for multiplication in combination with noise management techniques. In fact, the Gentry-Sahai-Waters scheme even exploits this property to minimise the noise growth [9].

Furthermore, the homomorphic property does in general not hold concurrently for the messages and the randomisers. It can, however, be possible to compute the noise after an operation for certain simple cases. We show that the abelian group requirement is not necessary, so that a variant of the original protocol is secure also for a noise-based homomorphic scheme.

The final issue is to take advantage of the quantum security of the lattice based encryption schemes, and then make the protocol future-proof. The secrecy requirements for verifiable shuffles is long-term, while soundness is only short-term. This allows us to achieve security against a potential future quantum adversary using a perfectly hiding commitment scheme, since the computational binding property is only necessary until the proof has been verified.

However, using a lattice based commitment scheme by Baum et al. [3], we can also clear the protocol completely of classic cryptography.

1.1 A naive approach

Recall the polynomial $(X - x_1)(X - x_2) \cdots (X - x_n)$, where the roots $x_1, x_2, \ldots, x_n$ are the secret data to be shuffled. Assume we have two sets of ciphertexts, say $\{E_i\}$ and $\{e_i\}$, and some secret permutation π such that $\mathsf{Dec}(E_i) = \mathsf{Dec}(e_{\pi(i)})$. The straightforward approach to shuffling using fully homomorphic encryption is to compare the two polynomials

$$\begin{aligned} P_1(X) &= (X - e_1)(X - e_2) \cdots (X - e_n) \\ P_2(X) &= (X - E_1)(X - E_2) \cdots (X - E_n) \end{aligned}$$

by requiring the prover to demonstrate that the ciphertext $P_2(e_i)$ for one or more i given by the verifier decrypts to 0. Such proofs exist [3, 5], given that the prover has decryption capabilities. Also, it would be straightforward to verify this protocol using multilinear maps [7] with their zero-test abilities. However, at the time of writing, all multilinear map candidates are broken for this application [1]. Additionally, this computation would require a very *deep* circuit, i.e. high degree polynomials, which with today's FHE techniques is forbiddingly expensive.

1.2 Related work

Independently, Costa, Martínez and Morillo [6] have published a shuffle for lattice based schemes. Their shuffle is based on an idea of Wikström using *permutation matrices*. Unfortunately, one cannot guarantee the secrecy of the shuffle due to lack of circuit privacy for their re-encryption procedure. They observe that the RLWE scheme is additively homomorphic, and suggest to re-encrypt by adding an encryption of 0. This idea is sound for group homomorphic schemes since the randomness is near-uniformly distributed over the group. With noise-based schemes, the randomness is typically a Gaussian, so decryption and a following analysis of the noise term can reveal extra information about the ciphertext. As a consequence, the permutation can leak from the ciphertexts regardless of the properties of the zero-knowledge protocol for which the authors provide a proof.

1.3 Our contribution

Our main contribution is the first adaptation of a verifiable shuffle specifically for a FHE cryptosystem under the assumption of equal noise levels in the input ciphertexts. The efficiency is mostly affected by the inherent limitations of the cryptosystem. The assumption can if necessary be met using bootstrapping.

A second contribution is a detailed exposition of the properties of the gadget matrix, and it is our hope that it can be useful for others who need to work with the details of the GSW ciphertexts.

1.4 Outline

We have introduced the main ideas here in Section 1. The upcoming section will in turn describe the concepts we need to build our protocol, such as *gadget matrices*, the GSW cryptosystem, commitment schemes, zero-knowledge protocols, and Groth's original shuffle. Then, in Section 3, we describe our modifications and present the shuffle in full, including a proof of it being a zero-knowledge argument.

2 Preliminaries

This section introduces the concepts and technical terms needed in this paper to successfully use the Groth shuffle on GSW ciphertexts. Before we discuss the cryptosystem, we look at gadget matrices in detail.

Following the the description of GSW, we will discuss commitment schemes, zero-knowledge proofs and Groth's verifiable shuffle protocol.

2.1 Gadget matrices

Much of the notation will follow Alperin-Sheriff and Peikert [2]. Assume we work in a field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, and let $\ell = \lceil \log_2 q \rceil$. One can then define the gadget vector $\vec{g} \in \mathbb{Z}_q^\ell$ as

$$\begin{pmatrix} 1 \\ 2 \\ 4 \\ \vdots \\ 2^{\ell-1} \end{pmatrix}$$

For any $a \in \mathbb{Z}_q$, it is clear that there exist many vectors $\vec{x} \in \mathbb{Z}_q^\ell$ such that $\langle \vec{g}, \vec{x} \rangle = a$. In particular, the vector $\vec{x}$ can be the binary decomposition of a, with all entries 0 or 1. The binary decomposition is the output of the function or algorithm denoted $\vec{g}^{-1}_{\text{det}}$.

Sometimes, we want a random preimage of $\vec{g}$ rather than the binary decomposition. Let $X = \{\vec{x} \mid \langle \vec{x}, \vec{g} \rangle = 0\}$, the set of all preimages of 0. Let $\vec{g}\,^{-1}_{\text{rand}}$ denote an algorithm that computes $\vec{g}\,^{-1}_{\text{det}}$ and samples a value $\vec{x}$ from X, typically from a Gaussian distribution with a prescribed radius, and outputs the sum $\vec{g}\,^{-1}_{\text{det}} + \vec{x}$.

From now on, we will use subscripts when it is necessary to distinguish between the two variants of the algorithm. If no subscript is given, then the discussion applies equally to both.

Next, we can expand the whole process to handle an n-dimensional vector $\vec{a}$ rather than a single value. Define the sparse matrix

$$G = \begin{pmatrix} 1 & \dots & 2^{\ell-1} & & & & & & & \\ & & & 1 & \dots & 2^{\ell-1} & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & 1 & \dots & 2^{\ell-1} \end{pmatrix} \in \mathbb{Z}_q^{n \times n\ell}.$$

The matrix G is known as the *gadget matrix*, and the literature often express it in shorthand as $\vec{g}\,^T \otimes I_n$.

The map from $\mathbb{Z}_q^{n\ell}$ to $\mathbb{Z}_q^n$ induced by the matrix G is not invertible, but it is easy to find preimages. As with g, the binary decomposition of each coordinate is a preimage. In line with the literature, let G^{-1}_{det} denote this function. It is not linear, since the sum of two binary decompositions need not be all binary again. The output of G^{-1} is a right-inverse for the map G, and we can extend it to $\mathbb{Z}_q^{n \times m}$ by applying G^{-1} column-wise to some $n \times m$ matrix A. As with $\vec{g}\,^{-1}$, we sometimes want random samples, and denote the resulting sampling algorithm by G^{-1}_{rand}. We use the notation $X \leftarrow G^{-1}_{\text{rand}}(A)$ when we want to indicate that we sample from some distribution imposed on the algorithm.

The following properties are straightforward to derive from the above construction.

Lemma 1. *Assume all operations are modulo some q, and let $A \in \mathbb{Z}_q^{n \times m}$ and $\lambda \in \mathbb{Z}_q$ be some scalar. Then,*

1. $G \cdot G^{-1}(I) = I = I_n \in \mathbb{Z}_q^{n \times n}$
2. $G \cdot G^{-1}(A) = A \in \mathbb{Z}_q^{n \times m}$

3. In particular, $G \cdot G^{-1}(\lambda G) = \lambda G$

We get a particularly nice structure when applying the G^{-1} algorithm on multiples of G.

Lemma 2. *Assume all operations are modulo some* q*, and let* $\lambda \in \mathbb{Z}_q$ *be some scalar with binary decomposition* $\sum_{i=0}^{\ell-1} \lambda_i 2^i$*. Then*

$$G_{\text{det}}^{-1}(\lambda G) = \begin{pmatrix} \Lambda & & & \\ & \Lambda & & \\ & & \ddots & \\ & & & \Lambda \end{pmatrix} \in \mathbb{Z}_q^{n\ell \times n\ell}$$

where

$$\Lambda = \begin{pmatrix} \lambda_0 & \lambda_{\ell-1} & \cdots & \lambda_1 \\ \lambda_1 & \lambda_0 & \cdots & \lambda_2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{\ell-1} & \lambda_{\ell-2} & \cdots & \lambda_0 \end{pmatrix} \in \mathbb{Z}_q^{\ell \times \ell}$$

In particular, $G_{\text{det}}^{-1}(G)$ *is the* $n\ell \times n\ell$ *identity matrix.*

The pattern comes from the fact that multiplying by 2 corresponds with one-step shifts in the binary expression of a number.

One can view $G_{\text{det}}^{-1}(\lambda G)$ as a representation of λ, and consider all representations as equivalent (modulo the kernel of G). Then the G^{-1} algorithm is homomorphic on equivalence classes, which is crucial for the GSW cryptosystem. Recall that linear mappings can be represented by matrices. Consider the mapping $G : \mathbb{Z}_q^{n\ell} \to \mathbb{Z}_q^n$ given by $\vec{x} \mapsto G\vec{x}$. This mapping have several right-inverses $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\ell}$ such that

$$(G \circ H)(\vec{x}) = GH\vec{x} = \vec{x},$$

so $G \circ H = \mathrm{id}_{\mathbb{Z}_q^n}$. Fix G^{-1} as one specific such right-inverse. Then, for all H,

$$G^{-1}(\vec{x}) - H(\vec{x}) \in \ker G.$$

As explained above, we can expand the map G to $\mathbb{Z}_q^{n\ell \times m} \to \mathbb{Z}_q^{n \times m}$, and we can expand the right-inverses as well. By the above relation, for each H we then get

$$G^{-1}(A) = HA + B_A, \qquad G(B_A) = 0,$$

and so for scalars $a, b \in \mathbb{Z}_q$, we have

$$\begin{aligned} G^{-1}(aG)G^{-1}(bG) &= (aHG + B_a)(bHG + B_b) \\ &= ab(HGHG + b^{-1}HGB_b + a^{-1}B_aHG + (ab)^{-1}B_aB_b) \\ &= ab(HIG + b^{-1}H \cdot 0 + a^{-1}B_aHG + (ab)^{-1}B_aB_b) \\ &= abHG + B' \qquad (\text{with } GB' = 0) \\ &= G^{-1}(abG) + B' - B_{ab}. \end{aligned}$$

As a consequence, $G^{-1}(aG)G^{-1}(bG)$ and $G^{-1}(abG)$ can be said to encode the same scalar ab, but with a difference which lies in the kernel of G. A similar computation holds for the sum $G^{-1}(aG) + G^{-1}(bG)$.

We can illustrate this with a toy example. Let $q = 7, \ell = 3$ and $n = 3$. Then

$$G = \begin{pmatrix} 1 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix}.$$

Consider $G_{\det}^{-1}(5G)$ and $G_{\det}^{-1}(3G)$, which will have blocks $\left[\begin{smallmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{smallmatrix}\right]$. Both their sum and product will be $\left[\begin{smallmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{smallmatrix}\right]$ which contains a 2, something we cannot avoid since we are computing modulo 7. However, a new encoding of $5 \cdot 3 \equiv 5 + 3 \equiv 1 \pmod 7$ is just the identity matrix.

This is an effect one has to take into account when computing. Still, it is certainly so that the different matrices represent the same value. In particular,

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

and note that

$$\begin{pmatrix} 1 & 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \equiv \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \pmod 7.$$

In other words, $(1, 1, 1)$ is in the kernel of G.

2.2 The GSW cryptosystem and circuit privacy

The 2013 cryptosystem by Gentry, Sahai and Waters (GSW) [9] is based on hiding the message as an eigenvalue of the ciphertext. The private key is an *approximate eigenvector*. For simplicity, we will use the symmetric formulation by Alperin-Sheriff and Peikert [2], and at the end explain how to make the scheme public key. Let n be an integer, let q be a modulus and $\ell = \lceil \log_2 q \rceil$. Finally, let χ be a subgaussian distribution (Gaussian with very small tails) over $\mathbb{Z}$.

Key generation Let $\vec{\bar{s}} \leftarrow \chi^{n-1}$ coordinate-wise, and output $\vec{s} = (\vec{\bar{s}}, 1)$ as the private key.

Encryption To encrypt a message $\mu \in \{0,1\}$, choose a random matrix $\bar{C}$ from $\mathbb{Z}_q^{(n-1)\times m}$, where $m = n\ell$, an error vector $\vec{e} \leftarrow \chi^m$ and set $\vec{b}^T = \vec{e}^T - \vec{\bar{s}}^T \bar{C} \pmod q$. Let

$$C = \begin{pmatrix} \bar{C} \\ \vec{b}^T \end{pmatrix} + \mu G.$$

Decryption Given $\vec{s}$ and C, let $\vec{c}$ be the penultimate column of C, and output 0 if $\langle \vec{s}, \vec{c} \rangle \pmod q$ is closer to 0 than 2^{l-2}. Otherwise, output 1.[1]

Addition Add the matrices C_1 and C_2.

Multiplication Define $C_1 \odot C_2$ as $C_1 \cdot G^{-1}(C_2)$.

The cryptosystem is usually only defined for a binary plaintext space, but the definition can be modified even up to the large space $\mathbb{Z}_q$ by modifying the decryption algorithm to extract bits from more columns than the penultimate, and then building the message from the bits. However, this has a strong negative impact on the noise behaviour. When two ciphertexts encrypting binary messages are multiplied, the noise grows far less than with previous FHE cryptosystems. The growth is a function of the encrypted value of the first ciphertext, so larger plaintext spaces can potentially also give worse noise problems.

[1]See Alperin-Sheriff and Peikert [2] for a justification of this algorithm.

Nonetheless, we will have to assume a large message space for our application, in the order of 160–180 bits, in order to facilitate the scalar multiplications we will perform. This fact is the main drawback of our work.

The original GSW scheme does not achieve *circuit privacy*. Informally, this property guarantees that nobody are able to deduce which circuit output a given ciphertext. Gentry [8] defined the notion by requiring that an encryption of an evaluation of a circuit should be indistinguishable from an encryption of evaluation of the circuit on encrypted data. In other words, evaluate-then-encrypt should be the same as encrypt-then-evaluate. Bourse et al. [4] provide a simulation based definition – capturing mostly the same intuition – and prove that the GSW cryptosystem is circuit private if the multiplication algorithm is slightly modified and all input ciphertexts have low noise from the same distribution. The definitions only differ in that Bourse et al. allow the length of the circuit to leak.

Alperin-Sheriff and Peikert [2] proposed to use the G_{rand}^{-1} algorithm instead of G_{det}^{-1} for performance reasons. Bourse et al. go one step further, and also add a matrix which is 0 everywhere except for the bottom row, which constitutes *gaussian shift* on the ciphertext,

$$C_1 \odot C_2 = C_1 G_{\text{rand}}^{-1}(C_2) + \begin{pmatrix} 0 \\ \vec{y}^T \end{pmatrix},$$

where C_1 and C_2 are ciphertexts and $\vec{y}$ is a vector drawn from χ^m. In particular, one can scale C_1 by α by letting $C_2 = \alpha G$. Also note that $\begin{pmatrix} 0 \\ \vec{y}^T \end{pmatrix}$ is a valid encryption of 0. We will use this fact in the upcoming protocol.

Finally, we note that the GSW scheme can be made public-key by publishing the above $\begin{pmatrix} \bar{C} \\ b^T \end{pmatrix}$ as, say, $\hat{A}$ and define encryption as

$$C \leftarrow \hat{A}R + \mu G,$$

where R is a random matrix with entries in $\{-1, 0, 1\}$.

2.3 Commitment schemes

A commitment scheme is an important tool in protocols. The concept allows a player to make a binding promise to use certain values, but without revealing them at the time of the promise. The commitment can later be verified when the committer reveals the opening information. Formally, a commitment scheme consists of three algorithms:

KeyGen On input 1^ℓ, output a public key pk

Commit On input (pk, m, r), return c.

Verify On input (pk, m, r, c), return accept if c is a valid commitment to m, otherwise reject.

We say that (m, r) is an opening of c. The key material will normally be omitted to simplify notation.

Any public key cryptosystem can be turned into a commitment scheme which is unconditionally binding. Pedersen commitments [15] is an example of a scheme that is unconditionally hiding and where binding depends on the discrete logarithm problem being hard. A particularly nice property about the Pedersen scheme is that it is homomorphic; we have

$$\mathsf{Commit}(m_1, r_1) \cdot \mathsf{Commit}(m_2, r_2) = \mathsf{Commit}(m_1 + m_2, r_1 + r_2).$$

A commitment scheme must satisfy two security properties. The scheme must be *hiding*, which means that a commitment to some message m_1 is indistinguishable from a commitment to some other message m_2. Next, it must be *binding*, which means that it is hard to find two openings for distinct messages for a single commitment. At most one of these properties may hold unconditionally, but both may hold only computationally.

Lately, Baum et al. [3] proposed a new additively homomorphic commitment scheme based on the Ring-SIS problem. This is conjectured to be safe also against quantum computers. Recall from the introduction that also classical commitment schemes that are unconditionally hiding and computationally binding will remain secure and usable until the adversary has quantum computers readily available, since the binding property is only needed during the protocol execution to provide soundness.

2.4 Zero-knowledge protocols

Zero-knowledge protocols capture the intuition of being able to convince someone else about the validity of some claim, but without revealing any other information.

Definition 1. Let R be a relation, and let $(x, w) \in R$. An *honest-verifier zero-knowledge protocol* $(\mathcal{P}, \mathcal{V})$ for R is a two-party game between a prover $\mathcal{P}$ on input (x, w) and a verifier $\mathcal{V}$ on input x, satisfying

Completeness Whenever $(x, w) \in R$, $\mathcal{V}$ accepts.

Soundness If $(x, w) \notin R$, then for any $\mathcal{P}^*$, $\mathcal{V}$ will only accept with negligible probability.

Honest-verifier zero knowledge (HVZK) There exists a simulator $\mathcal{M}$ running in expected polynomial time on input x such that the output is indistinguishable from the transcripts of $(\mathcal{P}, \mathcal{V})$ run with (x, w) as input to $\mathcal{P}$.

The w is called a *witness* for the relation.

If the prover is only given bounded computationally resources, the protocol is usually called an *argument*, otherwise we call it a *proof*. The zero-knowledge property can be varied by requiring the indistinguishability to hold computationally, statistically or unconditionally.

The simulator can choose all messages arbitrarily. A proof or argument is *special* honest-verifier zero knowledge (SHVZK) if the output of the simulator is indistinguishable from a real transcript if it has to use truly random messages as simulated challenges from the verifier (as opposed to being able to choose such challenges arbitrarily).

To guarantee that a zero-knowledge protocol can be used as a subprotocol in a larger context, Lindell [11] introduced the notion of witness-extended emulation, which loosely speaking requires that there exists a machine that on basis of sufficiently many rounds of the protocol (reusing the prover's commitments) is able to both output a valid witness for the relation as well as a valid simulated transcript. Our use of this property is limited to noting that the shuffle of known content satisfies it in order for us to be able to use it for our protocol, so we refer to the original source for details.

2.5 Groth's shuffle

In 2010, following up on Neff's idea [14] of proving the validity of a shuffle by using the fact that a polynomial $\prod(X - x_i)$ is stable under permutation of the roots x_i, Groth presented an efficient, yet conceptually simple shuffle [10]. The idea is two-fold. First, one uses the polynomial idea to prove that some c is a commitment to a permutation of messages $m_1, \dots, m_n$. The values are known by the verifier, but the permutation remains hidden. Next, one binds the secret data to the known data, and proves that the same permutation is still used.

It is important to note that the shuffle of known content is independent of the encryption scheme employed in the main protocol, and only requires a group homomorphic commitment scheme. Later, we can therefore reuse the shuffle of known content completely, and rely on the following properties [10, Theorem 1].

- The shuffle satisfies special honest-verifier zero knowledge with witness-extended emulation.
- If the commitment scheme is statistically hiding we get statistical HVZK.
- If the commitment scheme is unconditionally binding we get unconditional soundness.

Recall that the property of witness-extended emulation guarantees that we can use the SKC protocol as a building block of the full shuffle.

While the SKC protocol only uses the commitment scheme, the outer protocol depends heavily on the encryption scheme, in particular rerandomisation and cancellation of original randomness. Both of these features are less straightforward in FHE schemes than in classical group homomorphic schemes, and call for some modifications to the original protocol.

Remark 1. Groth introduces two security parameters, ℓ_e and ℓ_s, subject to the conditions that

- ℓ_e must be sufficiently large to make it hard to break soundness, i.e. it must be hard to predict a challenge of length ℓ_e,
- For any a sampled from the uniform distribution on $[0, 2^{\ell_e}] \cap \mathbb{Z}$, d and $a + d$, must be statistically indistinguishable whenever d is sampled from the uniform distribution on $[0, 2^{\ell_e+\ell_s}] \cap \mathbb{Z}$, and
- If the commitment space has message space $\mathbb{Z}_q^n$, then $2^{\ell_e+\ell_s} \leq q$.

The second bullet point is to avoid leakage of information whenever $a + d < 2^{\ell_e}$ or $2^{\ell_e+\ell_s} \leq a + d$. Notice that we can achieve the same result with smaller parameters if we employ rejection sampling [12,13]. The probability of $2^{\ell_e} \leq a+d \leq 2^{\ell_e+\ell_s}$ is approximately $1-\frac{1}{2^{\ell_s}}-\frac{1}{2^{\ell_s+\ell_e}}$.

The third bullet point is to avoid overflow that would require modular reductions. However, Groth notes that "[w]hen the cryptosystem has a message space where $m^q = 1$ for all messages, this requirement can be waived".

Concretely, Groth suggests $\ell_e = \ell_s = 80$ for the interactive variant, and $\ell_e = 160$ and $\ell_s = 20$ if the protocol is made non-interactive using the Fiat-Shamir heuristic and rejection sampling. We will keep the same parameters for our protocol.

We reached the above probability by considering the uniform distributions on $X = [0, 2^{\ell_e}] \cap \mathbb{Z}$ and $Y = [0, 2^{\ell_e+\ell_s}] \cap \mathbb{Z}$, with probability density functions (PDF) $m_1(X = k) = \frac{1}{2^{\ell_e}}$ and $m_2(Y = k) = \frac{1}{2^{\ell_e+\ell_s}}$. The PDF of $Z = X + Y$ is then the convolution $m_3(Z = j) = \sum_{k=-\infty}^{\infty} m_1(k) m_2(j - k)$. For each j up to $2^{\ell_e} - 1$, there are $j + 1$ combinations, so we get $m_3(j) = \frac{j+1}{2^{2\ell_e+\ell_s}}$ for the first part. For j between 2_e^{ℓ} and $2^{\ell_e+\ell_s}$, there are constantly $2^{\ell_e} + 1$ combinations, and it decreases by 1 for each j in the tail that follows, down to $j = 2^{2\ell_e+\ell_s}$, which is the greatest value we can sample. From this, we can compute the cumulative density function F_3 at the two points $j = 2^{\ell_e}$ and $j = 2^{\ell_e+\ell_s}$, which is approximately $\frac{1}{2^{\ell_s}}$ and $1 - \frac{1}{2^{\ell_e+\ell_s}}$, respectively.

3 Verifiable shuffle for GSW

Now we can combine the tools and ideas above to get a verifiable shuffle for GSW ciphertexts. Let n denote the number of ciphertexts, and recall that ℓ_e and ℓ_s denote security parameters for the zero-knowledge protocol. We now briefly describe the changes that must be made to Groth's shuffle.

The first part of a shuffle is to permute and rerandomise the ciphertexts. Given an ElGamal ciphertext $(a = g^r, b = \mu h^r)$, a new ciphertext will typically look like $(ag^{r'}, bh^{r'})$, and one can easily prove that it is hard to find the correct correspondence between the old and new set as long as r' is random. The fundamental reason is that the randomness of ElGamal forms a group, and that any rerandomisation is indistinguishable from a fresh encryption.

This is not the case for FHE in general and GSW in particular. The randomness is not bounded, and the Eval algorithm will result in a new ciphertext with randomness being a function of both the messages and the randomness of the inputs. We need to employ Bourse et al.'s technique for circuit privacy. Let the old and new ciphertexts be denoted by $\{e_i\}$ and $\{E_i\}$, and the permutation by π.

Ideally, the shuffling circuit should have all old ciphertexts and the permutation as input, such that all old ciphertexts contribute to each new ciphertext,

$$E_i = \sum_{j=1}^{n} e_{\pi(i)} G^{-1}_{\text{rand}}(\delta_{\pi(i),j} G) + \begin{pmatrix} 0 \\ \vec{y}_i^T \end{pmatrix}$$

where $\delta_{a,b}$ is 1 if $a = b$ and 0 otherwise, and $\vec{y}$ is some vector chosen by the circuit privacy algorithm [4].

However, this is causing problems for achieving the completeness property of the protocol, so we have opted for a simpler version. For each i, sample $X_i \leftarrow G^{-1}_{\text{rand}}(G)$, and let

$$E_i = e_{\pi(i)} X_i + \begin{pmatrix} 0 \\ \vec{y}_i^T \end{pmatrix}$$

which is sufficient under the condition that all $\{e_i\}$ have the same noise level and equal-length decryptions. The order is not coincidental. The shuffling circuit is essentially included in X_i, and this order of multiplication hides it [4]. If necessary, enforce the noise-level condition by bootstrapping the ciphertexts before shuffling them. Bootstrapping is an deterministic operation which only requires the public key. Note that one can only measure the noise by using the decryption key.[2]

The original protocol was expressed using multiplications. Since we are using the additive structure of the GSW scheme, we switch from multiplications and exponentiations to additions and scalar multiplications. This is in itself a favourable move, as the efficiency of the original protocol was measured in exponentiations, while additions and scalar multiplications are almost for free in FHE schemes.

Finally, one of the verifications step in the original protocol involved creating a ciphertext using randomness provided by the prover. Since we lack the nice structure on the randomness in the GSW cryptosystem, we need to provide complete ciphertexts instead of just randomness. This requires us to convince the verifier that the ciphertext is "innocent", in the sense that it doesn't encrypt a value that allows the prover to cheat. Fortunately, we can observe that the ciphertext in question will be all zeros except for the bottom row, which guarantees that it can only encrypt 0.

The complete protocol follows.

Precomputation Start with fresh ciphertexts $\{e_i\}$ with equal noise levels. Bootstrap each ciphertext to achieve near-freshness if necessary. Shuffle using a random permutation π and re-encrypt to get new ciphertexts $\{E_i\}$.

Common input Fresh ciphertexts $\{e_i\}$ and shuffled ciphertexts $\{E_i\}$.

Private input to $\mathcal{P}$ Permutation π, matrices $X_i \leftarrow G^{-1}_{\text{rand}}(G)$ and vectors $\vec{y}$ such that for each i,

$$E_i = e_{\pi(i)} X_i + \begin{pmatrix} 0 \\ \vec{y}_i^T \end{pmatrix}$$

[2]An anonymous reviewer pointed out that it is important to ensure that a malicious mix server cannot mark the ciphertexts, typically by using randomness of different size, resulting in more noise. This may lead to a DoS attack unless one employ bootstrapping, but should not compromise secrecy since only the decryption service can measure noise.

Protocol

$\mathcal{P}1$ Select randomness r and r_d for the commitment scheme, and select n random values d_i of length $\ell_e + \ell_s$.

Let

$$c \leftarrow \mathsf{Commit}(\pi(1), \ldots, \pi(n); r)$$
$$c_d \leftarrow \mathsf{Commit}(-d_1, \ldots, -d_n; r_d).$$

Set $D_i \leftarrow G^{-1}_{\text{rand}}(d_i G)$, $\vec{y}_d \leftarrow \chi_{\mathbb{Z}^m}$ and $E_d \leftarrow \sum_{i=1}^{n} E_i D_i + \left(\begin{smallmatrix} 0 \\ \vec{y}_d^T \end{smallmatrix}\right)$

Send c, c_d and E_d to the verifier.

$\mathcal{V}1$ Return a set of random numbers $\{t_i\}$ of length ℓ_e.

$\mathcal{P}2$ Set $f_i \leftarrow t_{\pi(i)} + d_i$, compute X'_i such that

$$X'_{\pi(i)} = G^{-1}_{\text{det}}(t_{\pi(i)} G) - X_i G^{-1}_{\text{det}}(f_i G) + X_i D_i,$$

and set $Z = \sum_{i=1}^{n} \left(\left(\begin{smallmatrix} 0 \\ \vec{y}_i^T \end{smallmatrix}\right) G^{-1}_{\text{det}}(f_i G) - \left(\begin{smallmatrix} 0 \\ \vec{y}_i^T \end{smallmatrix}\right) D_i \right) + \left(\begin{smallmatrix} 0 \\ \vec{y}_d^T \end{smallmatrix}\right)$. Cancel if not $2^{\ell_e} \leq f_i \leq 2^{\ell_e + \ell_s}$ for all i.

Send $\{f_i\}, \{X'_i\}, Z$ to the verifier.

$\mathcal{P}$–$\mathcal{V}$ Run the shuffle of known content to prove that

$$c^{\lambda} c_d \mathsf{Commit}(f_1, \ldots, f_n) = \mathsf{Commit}(\lambda\pi(1) + t_{\pi(1)}, \ldots, \lambda\pi(n) + t_{\pi(n)}),$$

where λ is a challenge from the verifier.

$\mathcal{V}2$ Verify the following

- The elements c and c_d are in the commitment space
- For all i, $2^{\ell_e} \leq f_i \leq 2^{\ell_e + \ell_s}$
- $GX'_i = 0$ for all i
- The shuffle of known content
- The matrix Z is of the form $\left(\begin{smallmatrix} 0 \\ \vec{y}^T \end{smallmatrix}\right)$

- $\sum_{i=1}^{n} E_i G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_i (G_{\text{det}}^{-1}(t_i) + X_i') - E_d = Z$

Theorem 1. *Assume that $\{e_i\}$ is a set of fresh ciphertexts. Then the above protocol is a special honest-verifier zero-knowledge argument for correctness of a shuffle of fully homomorphic ciphertexts. If the commitment scheme is statistically binding, then the scheme is an SHVZK proof of a shuffle.*

Proof. Completeness

As shown in Remark 1, the probability of $\mathcal{P}$ aborting is $\frac{1}{2^{\ell_s}} + \frac{1}{2^{\ell_e+\ell_s}}$, which can be made arbitrarily small with a suitable choice of ℓ_s.

We need to check two of the verification equations, the rest is straightforward. Note that X_i comes from the G^{-1} algorithm and encodes a 1. By the discussion in Section 2.1, one can see that $X'_{\pi(i)}$ must encode $-f_i + t_{\pi(i)} + d_i = 0$ for all i, hence $GX_i' = 0$, all i.

Next, we verify that $\sum_{i=1}^{n} E_i G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_i (G_{\text{det}}^{-1}(t_i) + X_i') - E_d = Z$. This is a tedious, but uncomplicated computation:

$$
\begin{aligned}
&\sum_{i=1}^{n} E_i G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_i (G_{\text{det}}^{-1}(t_i) + X_i') - E_d \\
&= \sum_{i=1}^{n} E_i G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_{\pi(i)} (G_{\text{det}}^{-1}(t_{\pi(i)}) + X'_{\pi(i)}) - \sum_{i=1}^{n} E_i D_i + \begin{pmatrix} 0 \\ y_d^T \end{pmatrix} \\
&= \sum_{i=1}^{n} \left(e_{\pi(i)} X_i + \begin{pmatrix} 0 \\ y_i^T \end{pmatrix} \right) G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_{\pi(i)} (G_{\text{det}}^{-1}(t_{\pi(i)}) \\
&\quad - X_i G_{\text{det}}^{-1}(f_i G) + G_{\text{det}}^{-1}(t_\pi(i)) + X_i D_i) \\
&\quad - \sum_{i=1}^{n} \left(e_{\pi(i)} X_i + \begin{pmatrix} 0 \\ y_i^T \end{pmatrix} \right) D_i + \begin{pmatrix} 0 \\ y_d^T \end{pmatrix} \\
&= \sum_{i=1}^{n} e_{\pi(i)} \Big(X_i (G_{\text{det}}^{-1}(f_i G) - G_{\text{det}}^{-1}(f_i G) + D_i - D_i) - G_{\text{det}}^{-1}(t_{\pi(i)}) \\
&\quad + G_{\text{det}}^{-1}(t_{\pi(i)}) \Big) + \sum_{i=1}^{n} \left(\begin{pmatrix} 0 \\ y_i^T \end{pmatrix} G_{\text{det}}^{-1}(f_i G) - \begin{pmatrix} 0 \\ y_i^T \end{pmatrix} D_i \right) + \begin{pmatrix} 0 \\ y_d^T \end{pmatrix} \\
&= Z
\end{aligned}
$$

Soundness

We need to prove that there exists a permutation π, such that $\mathsf{Dec}(e_{\pi(i)}) = \mathsf{Dec}(E_i)$ for all $1 \leq i \leq n$. We can extract the permutation using rewinding, but we will not extract the matrices X_i used to rerandomise the ciphertexts (although we can prove that they must exist, and have been generated in an honest way).

Run the protocol $(\mathcal{P}^*, \mathcal{V})$ until the prover outputs a transcript. Due to the rejection sampling, the prover may try several times. If the verifier would reject the transcript, we output $\perp$. Following the exact same argument as in Groth's original proof, we can extract π and $\{-d_i\}$ using two valid transcripts [10, p. 562].

Because of the commitment we now know that $f_i = t_{\pi(i)} + d_i$, and since $GX'_i = 0$, we know that $\mathsf{Dec}(X'_i) = 0$. Also, we know that $\mathsf{Dec}(Z) = 0$. Recall that we scale a ciphertext C by computing $CG^{-1}(\lambda G)$, hence if we apply the decryption function to

$$\sum_{i=1}^{n} E_i G_{\text{det}}^{-1}(f_i G) - \sum_{i=1}^{n} e_i (G_{\text{det}}^{-1}(t_i) + X'_i) - E_d = Z,$$

we get

$$\begin{aligned}
&\sum_{i=1}^{n} f_i \mathsf{Dec}(E_i) - \sum_{i=1}^{n} (t_i + 0)\mathsf{Dec}(e_i) - \mathsf{Dec}(E_d) \\
&= \sum_{i=1}^{n} t_i \mathsf{Dec}(E_{\pi^{-1}(i)}) + \sum_{i=1}^{n} d_i \mathsf{Dec}(E_i) - \sum_{i=1}^{n} t_i \mathsf{Dec}(e_i) - \mathsf{Dec}(E_d) \\
&= \sum_{i=1}^{n} t_i (\mathsf{Dec}(E_{\pi^{-1}(i)}) - \mathsf{Dec}(e_i)) + \sum_{i=1}^{n} d_i \mathsf{Dec}(E_i) - \mathsf{Dec}(E_d) \\
&= \mathsf{Dec}(Z) = 0.
\end{aligned}$$

Since only one sum depends on $\{t_i\}$, both sums must be 0 individually. Furthermore, since each t_i is unpredictable, each summand must be 0. Hence, $\mathsf{Dec}(E_{\pi^{-1}(i)}) = \mathsf{Dec}(e_i)$, which we wanted to prove.

Note that we can apply the decryption function without actually being able to compute it for unknown ciphertexts.

Special honest-verifier zero knowledge

Let π_0 and π_1 be two permutations, and let $\mathcal{C}_0$ and $\mathcal{C}_1$ be the corresponding shuffle circuits. By circuit privacy, the adversary cannot decide whether $\mathcal{C}_0$ or $\mathcal{C}_1$ was used to generate $\{E_i\}$ from $\{e_i\}$. Hence, the precomputation step does not leak any information.

To prove that the shuffle itself is HVZK given the challenges, we construct a simulator whose output will be indistinguishable from a real protocol transcript. We provide the simulator through a hybrid argument.

Sim I Simulate the shuffle of known content, and select c and c_d as random commitments.

It follows from the properties of the shuffle of known content that Sim I is indistinguishable from a real transcript.

Sim II Construct a random Z from the same distribution as the original. The distribution is hard to give explicitly, but does not depend on secret data, so it can be simulated by choosing the fundamental terms of the sum independently, and adding. Likewise, choose $\{X_i'\}$ by choosing $\{(\bar{X}_i, \bar{d}_i)\}$ under the same distributions as the prover would, and some permutation $\bar{\pi}$. Compute $\{X_i'\}$ by the equation in $\mathcal{P}2$, such that $GX_i' = 0$ for all i. Choose $\{f_i\}$ from the sum of the uniform distributions over $[0, 2^{\ell_e}] \cap \mathbb{Z}$ and $[0, 2^{\ell_e+\ell_s}] \cap \mathbb{Z}$ under the constraint that $2^{\ell_e} \leq f_i \leq 2^{\ell_e+\ell_s}$. Then choose E_d to fit.

The simulated values for Z, $\{X_i'\}$ and $\{f_i\}$ have the same distribution since they are computed from the same formulas as the original, but using new (but identically distributed) random values instead of X_i and π. Then E_d becomes a valid ciphertext by the homomorphic property of GSW. Circuit privacy makes a simulated E_d indistinguishable from a real E_d, and the IND-CPA property of the cryptosystem will finally provide computational SHVZK. $\square$

4 Further work

We have presented a verifiable shuffle for fully homomorphic schemes. Shuffling techniques have evolved further since the protocol we have chosen to forge from, and we believe it would be interesting to see adaptions of newer shuffles.

Furthermore, Groth's original shuffle can be used with a large family of group homomorphic encryption schemes. The result in this paper can only use the GSW scheme, due to the existence of the efficient and computationally simple circuit privacy technique. However, one should pick one's scheme based on what the application needs, so the shuffling primitive should be available for more schemes. This requires more research on techniques for circuit privacy.

Finally, it would be interesting to see an implementation of verifiable shuffling for FHE schemes, coupled with a real-life application. Only then will one be able to see if the parameters and the runtime of the proof will be acceptable. For instance, we predict that the scheme will be unsuitable for applications with many shuffles, such as onion routing. However, for elections, where one can spend minutes or even hours on the process, this protocol may already be mature.

Acknowledgements The author wishes to thank Jens Groth for his useful comments to an early version of this manuscript, as well as to the anonymous reviewers of Voting'18.

References

[1] Martin Albrecht and Alex Davidson. Are graded encoding scheme broken yet? `http://malb.io/are-graded-encoding-schemes-broken-yet.html`, 2017. Accessed 2017-08-30.

[2] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014.

[3] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris 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`.

[4] Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89. Springer, 2016.

[5] Christopher Carr, Anamaria Costache, Gareth T. Davies, Kristian Gjøsteen, and Martin Strand. Zero-knowledge proof of decryption for FHE ciphertexts. Cryptology ePrint Archive, Report 2018/026, 2018. `https://eprint.iacr.org/2018/026`.

[6] Núria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography (full version). Cryptology ePrint Archive, Report 2017/900, 2017. `http://eprint.iacr.org/2017/900`.

[7] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[8] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[9] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[10] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *J. Cryptology*, 23(4):546–579, 2010.

[11] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 16(3):143–184, 2003.

[12] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In Ronald Cramer, editor, *Public Key*

Cryptography - PKC 2008, volume 4939 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2008.

[13] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[14] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[15] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

Paper V

Zero-Knowledge Proof of Decryption for FHE Ciphertexts

Christopher Carr, Anamaria Costache, Gareth T. Davies, Kristian Gjøsteen and Martin Strand

ePrint 2018/026

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

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.

[*]Work partially conducted while visiting NTNU.

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 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, but 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 (for LWE, 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 $\langle \vec{c} - \vec{m}, \vec{s} \rangle$ 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

$$\begin{tikzcd}[column sep=huge]
\text{BGV} \arrow[r, "\text{Feasibility: Thm. 1}", "\text{Integrity: Lemma 1}"'] \arrow[rr, bend right, "\text{Feasibility: Composition}", "\text{Integrity: Lemma 1 + Thm. 2}"'] & \text{Gentry} \arrow[r, "\text{Feasibility: Decryption}", "\text{Integrity: Thm. 2}"'] & \text{Plaintext}
\end{tikzcd}$$

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

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, and 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.

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 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.

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 $a \overset{\$}{\leftarrow} S$ 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 $\mathsf{A}(\Delta, x, a, c, z, c', z') = w$ 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.

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 [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 [47] van Dijk et al. [49] and others [13, 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. We sample some error $\mathbf{r}$ and output $\mathbf{c} = 2\mathbf{r} + \mathbf{m} \pmod{\mathbf{B}_{\mathrm{pk}}}$, so that the ciphertext is

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

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

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, 46]. 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.

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 setup 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. *Let* $\mathbf{c}$ *be a BGV ciphertext. Ciphertext-switching* $\mathbf{c}$ *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 $\langle \mathbf{c}, \mathbf{s} \rangle = c_1 - c_0 \cdot s$, for a ciphertext $\mathbf{c} = (c_0, c_1)$. Now if the scheme E does indeed support affine transformations, we have that

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

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.

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

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. $\square$

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 $e \overset{\$}{\leftarrow} \{0, 1\}$ 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 the correct randomness to generate $\mathbf{d}$.

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.

References

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

[2] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Oswald and Fischlin [43], pages 430–454.

[3] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Garay and Gennaro [20], pages 297–314.

[4] Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. How to prove knowledge of small secrets. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[5] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris 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`.

[6] Carsten Baum, Ivan Damgård, Tomas Toft, and Rasmus Winther Zakarias. Better preprocessing for secure multiparty computation. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 327–345, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany.

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

[8] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. ACM.

[9] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [42], pages 97–106.

[10] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[11] Gu 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.

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

[13] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[14] Jean-Sébastien Coron and Jesper Buus Nielsen, editors. *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.

[15] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.

[16] Ronald Cramer, Ivan Damgård, Chaoping Xing, and Chen Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In Coron and Nielsen [14], pages 479–500.

[17] Ivan Damgård. On σ-protocols, v2. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2010.

[18] Rafaël del Pino and Vadim Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[19] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Oswald and Fischlin [43], pages 617–640.

[20] Juan A. Garay and Rosario Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[21] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[22] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [42], pages 107–109.

[23] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[24] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 1–16, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

[25] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[26] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [45], pages 850–867.

[27] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In Gilbert [29], pages 506–522.

[28] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[29] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[30] Kristian Gjøsteen and Martin Strand. A roadmap to fully homomorphic elections: Stronger security, better verifiability. Cryptology ePrint Archive, Report 2017/166, 2017. `http://eprint.iacr.org/2017/166`.

[31] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 430–443, Vienna, Austria, October 24–28, 2016. ACM Press.

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

[33] Shai Halevi and Victor Shoup. Algorithms in HElib. In Garay and Gennaro [20], pages 554–571.

[34] Shai Halevi and Victor Shoup. Helib – an implementation of homomorphic encryption., 2014.

[35] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Oswald and Fischlin [43], pages 641–670.

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

[37] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.

[38] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 162–179, Barcelona, Spain, March 9–12, 2008. Springer, Heidelberg, Germany.

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

[40] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In Coron and Nielsen [14], pages 293–323.

[41] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 311–343, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[42] Rafail Ostrovsky, editor. *52nd FOCS*, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[43] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[44] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 61–72, Chennai, India, December 20–22, 2004. Springer, Heidelberg, Germany.

[45] Reihaneh Safavi-Naini and Ran Canetti, editors. *CRYPTO 2012*, volume 7417 of *LNCS*, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

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

[47] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

[48] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[49] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [29], pages 24–43.