

# QUANTUM-RESISTANT CRYPTOGRAPHY VIA UNIVERSAL GRÖBNER BASES

SERGIO DA SILVA AND ANIYA STEWART

**ABSTRACT.** In this article, we explore the use of universal Gröbner bases in public-key cryptography by proposing a key establishment protocol that is resistant to quantum attacks. By utilizing a universal Gröbner basis  $\mathcal{U}_I$  of a polynomial ideal  $I$  as a private key, this protocol leverages the computational disparity between generating the universal Gröbner basis needed for decryption compared with the single Gröbner basis used for encryption. The security of the system lies in the difficulty of directly computing the Gröbner fan of  $I$  required to construct  $\mathcal{U}_I$ . We provide an analysis of the security of the protocol and the complexity of its various parameters. Additionally, we provide efficient ways to recursively generate  $\mathcal{U}_I$  for toric ideals of graphs with techniques which are also of independent interest to the study of these ideals.

## 1. INTRODUCTION

Cryptographic systems often rely on the computational difficulty of solving particular mathematical problems. Quantum computing is a rapidly growing industry [22] with reports of capable quantum systems being available by as soon as 2030, making post-quantum cryptography especially relevant while also threatening the security of traditional cryptographic methods [28]. For example, the commonly used RSA cryptosystem, which relies on the difficulty of factoring the product of two (secret) large prime numbers [20], would no longer remain secure using Shor's algorithm on a quantum computer [29]. As the vulnerabilities facing cryptographic systems becomes a reality, it is necessary to explore other approaches and techniques that might prove more useful in resisting quantum attacks.

One promising area of exploration is with primitives that utilize algebraic or combinatorial constructions, especially in the context of lattice-based cryptography. Many algebraic constructions utilize computational aspects of ideals in polynomial rings [6, 7], making Gröbner bases a natural component in their implementation [15]. Gröbner bases are specific generators of a polynomial ideal that allow many algebro-geometric properties to be computed efficiently from an associated monomial ideal. Past attempts to use Gröbner bases in public-key cryptography have failed, such as with Barkee cryptosystems [2, 3]. The main obstacle to these approaches is that a single Gröbner basis is generally too easy to compute to realistically be used to secure a system. A universal Gröbner basis on the other hand is difficult to compute, and involves the computation of high-dimensional

---

*Date:* October 14, 2025.

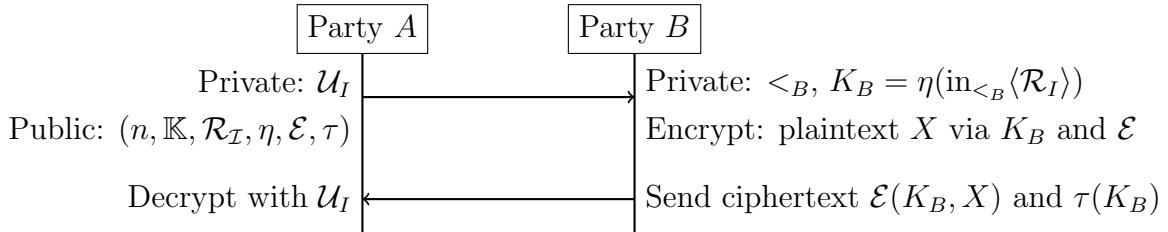
2000 *Mathematics Subject Classification.* Primary: 94A60, 13P10; Secondary: 05E40, 14M25.

*Key words and phrases.* post-quantum cryptography, universal Gröbner bases, toric ideals of graphs.

lattice structures like the Gröbner fan and state polytope of a polynomial ideal. Our approach is to have one party use a universal Gröbner basis to produce a private list of keys while also having a public mechanism for another party to generate one key from that list.

Let  $A$  and  $B$  be two parties who have not previously communicated to share a common encryption key. To establish the protocol  $\mathcal{P}$ , Party  $A$  starts with an ideal  $I \subset \mathbb{K}[x_1, \dots, x_n]$ , a universal Gröbner basis  $\mathcal{U}_I$  of  $I$ , and some generating set  $\mathcal{R}_I$  of  $I$ . Then  $\mathcal{R}_I$ ,  $\mathbb{K}$ , and the number of variables is made public, together with information about the encryption scheme needed to create the ciphertext, including two hash functions  $\eta$  and  $\tau$ . The protocol  $\mathcal{P}$  can be defined without the use of  $\tau$ , in which case we set  $\tau = \emptyset$ .

Now Party  $B$  can send an encrypted message to  $A$  using the information publicly provided by  $A$ . Party  $B$  starts by choosing a random monomial order  $<_B$  of  $\mathbb{K}[x_1, \dots, x_n]$  and then computes the initial ideal  $\text{in}_{<_B}(\mathcal{R}_I)$ . Both  $<_B$  and the initial ideal are kept private. By Dickson's lemma, there is a unique minimal generating set of  $\text{in}_{<_B}(\mathcal{R}_I)$ , and based on a public hash function  $\eta$  provided by  $A$ , Party  $B$  converts this set into a binary sequence which will serve as the encryption key  $K_B$ . Using the predetermined encryption scheme  $\mathcal{E}$  provided,  $B$  encrypts the message into a ciphertext. If  $\tau = \emptyset$ , then only this ciphertext is sent back to  $A$ . Otherwise,  $\tau(K_B)$  is also sent back to  $A$ . The protocol can be described using the following schematic:



First suppose that  $\tau = \emptyset$ . If an attacker were to intercept the ciphertext, they wouldn't know what parameters  $B$  chose to produce the ciphertext, so a brute-force attack would be intractable. For instance, with RSA public-key cryptography, an attacker at least knows the product  $m = pq$  (which is public) and could try to find a brute-force factorization of  $m$ . With our setup, if an RSA encryption algorithm  $\mathcal{E}$  were used to create a ciphertext,  $m$  would remain hidden, so an attacker wouldn't even know what number to attempt to factor.

This added security comes at a cost. Since Party  $A$  will also not know which key Party  $B$  chose, the only option is to try all possible keys to decrypt the message. However,  $A$  has the universal Gröbner basis  $\mathcal{U}_I$ , and therefore possesses the list of all possible initial ideals of  $I$ , and hence has the list of all possible encryption keys that  $B$  could have generated. The list that Party  $A$  needs to exhaustively search to decrypt the message is reasonable compared to the intractable list that an attacker would need to try. The system also relies on  $\mathcal{U}_I$  being extremely difficult to compute directly without prior knowledge of any symmetries used to construct  $I$ .

On the other hand, if  $\tau \neq \emptyset$ , then  $A$  could decrypt the message without iterating through the list of keys since the image of each key under  $\tau$  would already be known

and could be compared with the value  $\tau(K_B)$  provided by  $B$ . However, this reduces the overall security of the system since an attacker intercepting the message would also have knowledge of  $\tau(K_B)$ , and could try attacking  $\tau$  instead of computing  $\mathcal{U}_I$  directly. We summarize the analysis of the protocol  $\mathcal{P}$  presented in this article in the following theorem.

**Theorem.** *Let  $\mathcal{P} = (n, \mathbb{K}, \mathcal{U}_I, \mathcal{R}_I, <_B, \eta, \mathcal{E}, \tau)$  be a protocol as in Definition 3.1. Then the following statements about the complexity and security of the cryptosystem hold:*

- *Party A can (privately) construct  $\mathcal{U}_I$  in polynomial time by using Theorem 5.12. Given  $\mathcal{U}_I$ , the set  $\mathcal{R}_I$  can be computed in linear time by Theorem 4.5.*
- *Party B can send a message to A by computing a single Gröbner basis for  $I$ , the complexity of which is summarized in Theorem 4.6.*
- *The amount of time that A requires to decrypt the ciphertext when  $\tau = \emptyset$  is summarized in Proposition 4.9.*
- *An attacker without any trapdoor knowledge about how  $I$  was constructed would need to compute the Gröbner fan of  $I$  directly, which is NP-hard with complexity described in Theorem 4.1.*
- *By Theorem 4.3, if lattice-based primitives secured by the Shortest Vector Problem (SVP) are quantum-resistant, then so is the protocol  $\mathcal{P}$  when  $\tau = \emptyset$ .*

A more detailed discussion of the steps involved to initialize the protocol  $\mathcal{P}$  is presented in Section 3. We then consider practical complexity and security issues in Section 4 and subsequently describe how to efficiently construct  $\mathcal{U}_I$  in Section 5 using the toric ideal  $I_G$  of a graph  $G$ . This ideal can be generated by binomials corresponding to primitive closed even walks of  $G$ , which incidentally also define a universal Gröbner basis of  $I_G$ . Using four graph operations, one can recursively generate large graphs for which  $\mathcal{U}_I$  is computable and contains enough primitive closed even walks to ensure the security of  $\mathcal{P}$ .

It is worth noting that the graph constructions presented in Section 5, together with their effect on  $\mathcal{U}_I$ , are of independent importance to combinatorial algebraists [4, 21, 25, 27]. For instance, previous research in this area has revealed connections with algebraic statistics [13], commutative-algebraic techniques [10, 17], and network complexity [11]. In the final section of this article, we will discuss weaknesses and practical concerns regarding the protocol  $\mathcal{P}$ . We also identify areas for potential future research.

**Acknowledgments.** We thank Sarah Arpin for many helpful conversations and references on post-quantum cryptography. Da Silva's research is supported by NSF LEAPS-MPS Grant 2532757.

## 2. PRELIMINARIES

In this paper,  $\mathbb{K}$  will denote any field. In this section, we will provide a very brief overview of Gröbner bases, Gröbner fans and state polytopes. There is a considerable amount of theory involved with these topics, so we provide only what is necessary to understand the subsequent sections, and refer the reader to [8, 15, 24] for further details.

**2.1. Gröbner bases.** Gröbner theory provides a way to associate a monomial ideal  $\text{in}_<(I)$  to an ideal  $I \subset R = \mathbb{K}[x_1, \dots, x_n]$ . This is done in such a way that many algebro-geometric properties of  $I$  can be determined from  $\text{in}_<(I)$ , especially since monomial ideals are generally easier to study. A monomial order  $<$  on  $R$  is a total order on the monic monomials of  $R$  such that  $u \leq v \Rightarrow wu \leq wv$  and  $1 \leq w$  for any monomial  $w \in R$ . Given  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$ , we will use the notation

$$\mathbf{x}^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}.$$

With a monomial order  $<$  on  $R$  and a polynomial  $f \in R$ , we can order all terms of  $f$  and define the initial term of  $f$  as the monomial  $\text{in}_<(f) = c\mathbf{x}^\alpha$ ,  $c \in \mathbb{K}$ , which is greatest term of  $f$  with respect to  $<$ .

**Definition 2.1.** Let  $<$  be a monomial order on  $R$  and let  $I \subseteq R$  be an ideal. The *initial ideal* of  $I$ , denoted by  $\text{in}_<(I)$ , is the monomial ideal in  $R$  defined by

$$\text{in}_<(I) := \langle \text{in}_<(f) \mid f \in I \rangle.$$

Unfortunately, the initial terms of a generating set of  $I$  do not generally constitute a generating set of  $\text{in}_<(I)$ . This leads us to the definition of a Gröbner basis.

**Definition 2.2.** Given an ideal  $I \subseteq R$ , a set  $\mathcal{G} = \{g_1, \dots, g_t\} \subset I$  is a *Gröbner basis* for  $I$  if  $I = \langle g_1, \dots, g_t \rangle$  and  $\text{in}_<(I) = \langle \text{in}(g_1), \dots, \text{in}(g_t) \rangle$ . A *universal Gröbner basis*  $\mathcal{U}_I$  for an ideal  $I$  is a generating set for  $I$  which is a Gröbner basis for  $I$  with respect to *any* monomial order on  $R$ .

*Example 2.3.* Let  $I = \langle ag - bf, ce - dg \rangle \subset \mathbb{K}[a, b, c, d, e, f, g]$  and set  $\mathcal{G} = \{ag - bf, ce - dg\}$ . Given the lexicographic monomial ordering  $<_1$  defined by  $a > b > c > d > e > f > g$  we can show (using `Macaulay2` for example) that

$$\text{in}_{<_1}(I) = \langle ag, ce \rangle = \langle \text{in}_{<_1}(ag - bf), \text{in}_{<_1}(ce - dg) \rangle,$$

which implies that  $\mathcal{G}$  is a Gröbner basis for  $I$  with respect to  $<_1$ .

If however we used the lexicographic monomial ordering  $<_2$  defined by  $d > a > b > c > e > f > g$ , we would get  $\text{in}_{<_2}(I) = \langle ag, bdf, dg \rangle$ . Therefore, for  $<_2$ ,  $\mathcal{G}$  is not a Gröbner basis for  $I$ . To extend  $\mathcal{G}$  to a Gröbner basis of  $I$  with respect to  $<_2$ , we would need to include the polynomial

$$\begin{aligned} S(dg - ce, ag - bf) &= \frac{\text{in}_{<_2}(ag - bf)}{\gcd(dg, ag)} \cdot (dg - ce) - \frac{\text{in}_{<_2}(dg - ce)}{\gcd(dg, ag)} \cdot (ag - bf) \\ &= a(dg - ce) - d(ag - bf) \\ &= bdf - ace. \end{aligned}$$

which is found by applying Buchberger's algorithm. This involves computing the  $S$ -polynomials between pairs of generators, finding the remainder after polynomial division by  $\mathcal{G}$ , and extending  $\mathcal{G}$  by adjoining any non-zero remainders. This process is repeated until all remainders are 0. For specifics about the algorithm, refer to [8].  $\square$

Although Buchberger's algorithm provides a method for constructing a Gröbner basis for any given  $I \subseteq R$  with respect to some given monomial order  $<$ , modern techniques have become more sophisticated and differ from the  $S$ -polynomial computation above [1].

A fundamental result in the theory of monomial ideals is Dickson's Lemma which states that every monomial ideal has a unique minimal monomial generating set. If some fixed monomial order of  $R$  is also given, then every monomial ideal of  $R$  will have a unique minimal *ordered* generating set.

**Lemma 2.4.** *Given a monomial order  $<$  on  $R = \mathbb{K}[x_1, \dots, x_n]$  and a monomial ideal  $M \subseteq R$ , there exists a unique minimal monomial ordered generating set of  $M$ .*

We will be applying hash functions to sets of monomial generators, so having a unique way to write a given list of monomials is necessary. In defining the cryptosystem presented in the next section, we will also discuss sets of minimal monomial ideal generators which have bounded exponents, so we conclude this section with the following definition.

**Definition 2.5.** We say that a set of monomials  $\{\mathbf{x}^{\alpha_1}, \dots, \mathbf{x}^{\alpha_r}\} \subset R = \mathbb{K}[x_1, \dots, x_n]$  is minimal if it is the unique minimal monomial generating set of the ideal  $\langle \mathbf{x}^{\alpha_1}, \dots, \mathbf{x}^{\alpha_r} \rangle$ . Then define  $\mathcal{M}_k$  to be the set

$$\mathcal{M}_k := \left\{ \{\mathbf{x}^{\alpha_1}, \dots, \mathbf{x}^{\alpha_r}\} \subset R \mid \{\mathbf{x}^{\alpha_1}, \dots, \mathbf{x}^{\alpha_r}\} \text{ is minimal; } \alpha_{i,j} < k, 1 \leq i \leq r, 1 \leq j \leq n \right\}.$$

**2.2. Gröbner Fans.** Given an ideal  $I \subset R = \mathbb{K}[x_1, \dots, x_n]$ , there is a formal combinatorial structure for enumerating all possible initial ideals of  $I$ . We first need to define what it means for two monomial orders to be equivalent, which is best done in the more general setting of weight orders.

**Definition 2.6.** Given a polynomial ring  $R = \mathbb{K}[x_1, \dots, x_n]$  and weight vector  $w = (w_1, \dots, w_n) \in \mathbb{R}^n$ , we can define a *weight order*  $<_w$  on  $R$  by

$$x_1^{a_1} \cdot \dots \cdot x_n^{a_n} \leq_w x_1^{b_1} \cdot \dots \cdot x_n^{b_n} \text{ if and only if } a_1w_1 + \dots + a_nw_n \leq_w b_1w_1 + \dots + b_nw_n.$$

*Remark 2.7.* For a fixed  $I$  and monomial order  $<$  on  $R$ , there exists a weight  $w \in \mathbb{R}^n$  such that  $\text{in}_{<}(I) = \text{in}_{<_w}(I)$ . In fact, every monomial order is a weight order, but not every weight order is a monomial order. In particular,  $\text{in}_{<_w}(\cdot)$  does not necessarily yield a unique initial monomial term for every choice of  $w$ . As an example, let  $f = x^2y + x^{10} - y^2 \in \mathbb{K}[x, y]$  and  $w = (2, 10)$ . Then  $\text{in}_{<_w}(f) = x^{10} - y^2$ .

Given an ideal  $I$ , we can define an equivalence relation  $\sim$  on  $\mathbb{R}^n$  as  $w_1 \sim w_2$  if and only if  $\text{in}_{<_{w_1}}(I) = \text{in}_{<_{w_2}}(I)$ . If the initial ideal is a monomial ideal, then the collection of weight vectors in that equivalence class define a maximal cone in  $\mathbb{R}^n$  and the union of these cones is called the *Gröbner fan of  $I$* , denoted by  $GFan(I)$ . We will refer to a maximal cone in  $GFan(I)$  as a *Gröbner region* (or *Gröbner cone*), and denote it by  $GR_{<}(I)$ . Here,  $GR_{<}(I)$  has a representative weight order  $<$  and corresponds to a distinct monomial initial ideal of  $I$  together with a marked reduced Gröbner basis  $\mathcal{G}_{<}$  for  $I$ . The union of these reduced Gröbner bases defines a universal Gröbner basis of  $I$ . See [24] for more information about Gröbner fans, and for proofs of these facts. The Gröbner fan is the normal fan of a polytope [24, Theorem 2.5] called the *state polytope of  $I$* , and is denoted by  $State(I)$ . See [9] for more information on correspondence between normal fans and convex polytopes.

**Proposition 2.8.** [24, Corollary 1.3] Let  $I \subset R = \mathbb{K}[x_1, \dots, x_n]$ , and suppose that  $\mathcal{G}_< \subset R$  such that  $\text{in}_<(\mathcal{G}_<)$  is the initial ideal of  $I$  associated to the Gröbner region  $GR_<(I)$ . Then

$$\bigcup_{GR_<(I) \subseteq GFan(I)} \mathcal{G}_<$$

defines a universal Gröbner basis of  $I$ .

With this structure, we now turn to the question of how to compute a Gröbner fan of an ideal  $I$ . Our protocol uses keys which are defined by initial ideals of  $I$ , so each Gröbner region of  $GFan(I)$  defines a distinct key that can be used for encryption. Indeed, if an attacker with no knowledge about how  $I$  or  $\mathcal{U}_I$  were constructed wanted to compute  $\mathcal{U}_I$  directly from  $I$ , they would need to first compute the Gröbner fan of  $I$ , and then take a union of all Gröbner basis representatives for the cones in the fan.

In [24, Section 7], Sturmfels proposed an algorithm for computing  $GFan(I)$ , which was later implemented with the `Gfan` package [19] of `Macaulay2`. An analysis of the complexity of the algorithm was considered in [16]. In this algorithm, there are three main steps that are iterated at each vertex of  $State(I)$  (i.e. at each vertex of the graph of the polytope  $State(I)$ ). We refer the reader to the aforementioned sources for the technical details of the algorithm, and only provide a very brief summary of the nature of each step below.

In the algorithm, we start with one known Gröbner region of the fan (e.g. by directly computing a Gröbner basis with respect to some order). We then travel from one maximal cone to a neighboring maximal cone while also computing a Gröbner basis representative for the new cone by a mutation of the Gröbner basis for the current one.

- (1) Each fan can be viewed as the normal fan to a dual object called a state polytope  $State(I)$ . The graph  $(V, E)$  of the Gröbner fan is the 1-skeleton of  $State(I)$ . Each vertex in the polytope corresponds to a cone in  $GFan(I)$ . Computing normal vectors to facets for this polytope requires time denoted by  $T_{\text{facets}}$ .
- (2) Starting at a vertex, we choose a sequence of admissible “shoot” edges in  $State(I)$  until all vertices have been reached (using a reverse search algorithm). Selecting these admissible search edges requires time denoted by  $T_{\text{shoot}}$ .
- (3) With an admissible edge chosen, there exists an operation to change the marked Gröbner basis for one vertex to a marked Gröbner basis for another vertex. This is called a “flip” procedure and is the most time consuming step to complete. The time for this procedure is denoted by  $T_{\text{flip}}$ .

*Remark 2.9.* We will later compare the complexity of computing  $GFan(I)$  with other lattice-based primitives. Since the construction of  $GFan(I)$  involves cones in  $\mathbb{R}^n$ , we can always choose a weight vector  $w$  in each cone with only integer entries. This allows us to view  $GFan(I)$  and  $State(I)$  as lattice constructions by intersecting with  $\mathbb{Z}^n$ .

### 3. A QUANTUM-RESISTANT KEY ESTABLISHMENT PROTOCOL

In this section we will introduce a public encryption protocol that uses Gröbner bases for symmetric key establishment. The main idea is to leverage the difficulty in computing a universal Gröbner basis of a polynomial ideal compared to the computational time

needed to compute a single Gröbner basis. We start by providing an example which illustrates the spirit of the algorithm, and in the subsequent section, we formally describe the protocol parameters and implementation. Experts who wish to only consider the formal description of the protocol rather than a particular small-scale implementation may skip to Section 3.2.

**3.1. An illustrative example.** Let  $A$  and  $B$  be two parties who wish to securely communicate but have not previously communicated to share a common key. In this example,  $B$  would like to send an encrypted message to  $A$  using information publicly provided by  $A$ . Specific (simple) choices of protocol parameters have been chosen for this example.

Suppose that  $A$  has an ideal  $I$  for which a universal Gröbner basis  $\mathcal{U}_I$  is known. For this example, suppose that

$$I = \langle ag - bf, ce - dg, ace - bdf \rangle \subset \mathbb{K}[a, \dots, g]$$

is an ideal, where  $\mathbb{K}$  is any field and  $a, \dots, g$  are indeterminates of the polynomial ring. This list of generators is kept private and provides Party  $A$  with a fast way to compute initial ideals of  $I$  (see Section 2.2 for a background on Gröbner fans). A second list of minimal generators  $\mathcal{R}_I$  is also computed for  $I$ . For instance,  $A$  may choose

$$\mathcal{R}_I = \{ag - bf, ce - dg\}.$$

It is worth noting that  $ace - bdf = a(ce - dg) + d(ag - bf)$ , so these generators still define the ideal  $I$ , but do not always form a Gröbner basis of it, depending on the monomial order. Party  $A$  then makes  $\mathcal{R}_I$ ,  $\mathbb{K}$ , and the number of variables public, together with information about the encryption scheme.

Suppose that  $B$  would like to send  $A$  a secure message.  $B$  proceeds to produce an encryption key using that public information that  $A$  has provided. They do this by randomly selecting some monomial order  $<_B$  of  $\mathbb{K}[a, \dots, g]$ , and then proceeds to compute  $\text{in}_{<_B}(\mathcal{R}_I)$ . Both  $<_B$  and the initial ideal are kept private. There is a unique minimal generating set for this monomial ideal by Lemma 2.4, and based on a public hash function  $\eta$  provided by  $A$ , converts this monomial ideal into an encryption key. For this example,  $B$  chooses the lexicographic order  $c > d > e > f > g > b > a$  so that

$$\text{in}_{<_B}(\mathcal{R}_I) = \langle bf, ce \rangle.$$

At this point,  $B$  would apply  $\eta$  to  $\{bf, ce\}$ . As an overly simplified method for generating an encryption key for this example, let us concatenate the exponent vectors of each minimal monomial generator (using a predefined order from the function provided by  $A$ ). Then

$$\begin{aligned} bf &= a^0 b^1 c^0 d^0 e^0 f^1 g^0 \longrightarrow 0100010 \\ ce &= a^0 b^0 c^1 d^0 e^1 f^0 g^0 \longrightarrow 0010100 \end{aligned}$$

Therefore,  $B$ 's private encryption key is  $K_B = 01000100010100$  which is used to encrypt a message using whatever encryption scheme  $\mathcal{E}$  Party  $A$  has made public. For instance, the encryption scheme may involve using  $K_B$  to generate two large primes from a hash function attached to  $\mathcal{E}$  and encrypt a message using the RSA encryption scheme (without making the product of the two primes public – even to  $A$ ). In the first version of the protocol, only this ciphertext is sent to  $A$ .

With  $\mathcal{U}_I$ ,  $A$  will have the list of all possible initial ideals of  $I$ , and therefore will have a list of all possible encryption keys that  $B$  could have generated. For this example, those keys are:

$$\begin{aligned}\langle ag, ce \rangle &\longrightarrow 10000010010100 \\ \langle ag, bdf, dg \rangle &\longrightarrow 100000101010100001001 \\ \langle ace, ag, dg \rangle &\longrightarrow 101010010000010001001 \\ \langle bf, ce \rangle &\longrightarrow 01000100010100 \\ \langle bf, dg \rangle &\longrightarrow 01000100001001\end{aligned}$$

Party  $A$  then tries each key (by a brute-force search) until the message is decrypted. Using some symmetric encryption algorithm  $\mathcal{E}$  instead of an asymmetric one would reduce the computational time for Party  $A$ .

In an alternate version of the protocol, a hash function  $\tau$  is also made public that is used to map  $K_B$  to some binary sequence. Then, together with the ciphertext, Party  $B$  also sends  $\tau(K_B)$ . Party  $A$ , having all possible values of  $\tau$  applied to each key, would then know which  $K_B$  was chosen, and could decrypt the message immediately. This comes at the cost of reduced security for  $\mathcal{P}$ .

We will now summarize each step of the protocol and briefly highlight security issues which are elaborated on in Section 4:

- (1) Party  $A$  is assumed to have a universal Gröbner basis  $\mathcal{U}_I$  for some ideal  $I \subset \mathbb{K}[x_1, \dots, x_n]$  and some trimmed generating set  $\mathcal{R}_I$ . This trimmed list is made public, together with a list of conventions needed for  $B$  to establish an initialization vector. A method for  $A$  to quickly find examples of  $I$  and  $\mathcal{U}_I$  can be found in Section 5. The complexity of computing  $\mathcal{R}_I$  can be found in Theorem 4.5.
- (2) If  $B$  wants to send a message to  $A$ , then a monomial order  $<_B$  needs to be chosen, and the initial ideal  $\text{in}_{<_B}(\mathcal{R}_I)$  needs to be computed. This can be done relatively quickly, as described in Theorem 4.6. By Lemma 2.4, there exists a unique minimal generating set of  $\text{in}_{<_B}(\mathcal{R}_I)$ .  $B$  uses this unique generating set to produce an encryption key using a function  $\eta$  provided by  $A$ . The message is then encrypted using the encryption scheme  $\mathcal{E}$  (also publicly provided by  $A$ ).
- (3) If  $\tau$  is not provided, then Party  $B$  only sends the ciphertext to  $A$ , keeping all other parameters private. If an attacker intercepts the message, they would not know which encryption key was used to create the message, and reducing the list of possible keys to a manageable list would require knowledge of  $\mathcal{U}_I$ . An analysis of the complexity of doing this and the general security of the system can be found in Section 4.1. If  $\tau$  is provided, then Party  $B$  would also send  $\tau(K_B)$  together with the ciphertext.
- (4) Since  $A$  has a list of all possible initial ideals of  $I$  (which defines a Gröbner fan – see Section 2.2), they also have a list of all possible encryption keys. The message can then be decrypted by trying each one (in the case that  $\tau$  is not provided). The time complexity for this is described in Section 4.2. Variations that help cut

down  $A$ 's decryption time are also discussed there. In the case when  $\tau$  is provided, Party  $A$  could decrypt the ciphertext without iterating over all keys.

This protocol relies on the fact that it is both time consuming to enumerate all potential encryption keys and to directly try and generate a universal Gröbner basis for  $I$ . Therefore, an attacker would need to spend an unreasonable amount of time and resources to decrypt the message, unless they already had a universal Gröbner basis for  $I$ . If this is the case, then how is it possible for  $A$  to even initialize the system? Using a combinatorial approach, we will show how to efficiently generate examples with symmetries that are kept private but for which  $\mathcal{U}_I$  is easy to compute.

**3.2. Protocol Parameters.** As seen in the previous section, there are a number of parameters which can be chosen when setting up this cryptosystem. We provide the theoretical framework for this protocol and leave choices of specific parameters unspecified, recognizing that there may be practical implementation concerns that may require flexibility.

**Definition 3.1.** Let  $\mathcal{P} = (n, \mathbb{K}, \mathcal{U}_I, \mathcal{R}_I, <_B, \eta, \mathcal{E}, \tau)$  be a tuple of parameters defined by:

- $n \in \mathbb{N}$  and  $\mathbb{K}$  is a field.
- $\mathcal{U}_I$  is a universal Gröbner bases for an ideal  $I \subset \mathbb{K}[x_1, \dots, x_n]$ .
- $\mathcal{R}_I$  is some (usually minimal) generating set of  $I$ .
- $<_B$  is a monomial order on  $\mathbb{K}[x_1, \dots, x_n]$ .
- $\eta : \mathcal{M} \rightarrow \{0, 1\}^s$  is a hash function on sets of monomials whose domain contains  $\mathcal{M}_k$  (see Definition 2.5) to binary strings of fixed length  $s$ .
- $\mathcal{E}$  is an encryption algorithm with an encryption key  $K$  and plaintext  $X$  as input, and whose output is a ciphertext  $\mathcal{E}(K, X)$ .
- $\tau$  is either a hash function on binary sequences  $\{0, 1\}^s$ , or is set to  $\emptyset$  if not being utilized.

Let  $A$  and  $B$  be two parties who wish to securely communicate but have not previously communicated to share a common key.

- (I) **Initialization:** Party  $A$  generates an ideal  $I \subset \mathbb{K}[x_1, \dots, x_n]$  and a list of polynomials  $\mathcal{U}_I \subset \mathbb{K}[x_1, \dots, x_n]$  which define a universal Gröbner basis of  $I$ . A reduced generating set  $\mathcal{R}_I$  for  $I$  is also computed. The set  $\mathcal{U}_I$  is kept private. The information  $(n, \mathbb{K}, \mathcal{R}_I, \eta, \mathcal{E}, \tau)$  is made public, and is sent to  $B$ :

$$A \longrightarrow B : \mathcal{P}_{\text{public}} = (n, \mathbb{K}, \mathcal{R}_I, \eta, \mathcal{E}, \tau).$$

- (II) **Key Generation:**  $B$  uses  $\mathcal{P}_{\text{public}}$  to generate a private key  $K_B$  by first selecting a monomial order  $<_B$  on  $\mathbb{K}[x_1, \dots, x_n]$ .  $B$  then computes  $\text{in}_{<_B}(\mathcal{R}_I)$  which has a unique minimal monomial generating set  $\mathcal{G}$ . Then  $K_B = \eta(\mathcal{G})$ , which is kept private.  
(III) **Encryption:**  $B$  can now encrypt any plaintext  $X_B$  using  $\mathcal{E}$  and  $K_B$ . If  $\tau = \emptyset$ , only this ciphertext is sent to  $A$ :

$$B \longrightarrow A : \mathcal{E}(K_B, X_B).$$

For a protocol defined with  $\tau \neq \emptyset$ , the value  $\tau(K_B)$  is also sent:

$$B \longrightarrow A : (\mathcal{E}(K_B, X_B), \tau(K_B)).$$

- (IV) **Decryption:** Since  $A$  has a universal Gröbner basis  $\mathcal{U}_I$ , they possess all possible initial ideals  $\text{in}_<(I)$ , and hence possess all possible encryption keys  $\mathcal{K}_{\mathcal{I}}$ . If  $\tau = \emptyset$ , then for each  $\kappa \in \mathcal{K}_I$ ,  $A$  can compute  $\mathcal{E}^{-1}(\kappa, \mathcal{E}(K_B, X_B))$  until the message is decrypted by brute force (choosing  $\mathcal{E}$  to be a symmetric encryption algorithm will reduce this computational time). For a protocol defined with  $\tau \neq \emptyset$ , Party  $A$  also has the list  $\tau(\mathcal{K}_I)$ , and hence knows which  $K_B$  was used to produce  $\tau(K_B)$ , and can therefore immediately compute  $\mathcal{E}^{-1}(K_B, \mathcal{E}(K_B, X_B))$ .

*Remark 3.2.* When  $\tau = \emptyset$  and Party  $A$  needs to iterate over all encryption keys, there is a question of how they will know when the ciphertext has been correctly decrypted. Besides being a readable message, one method is to require the message to contain some embedded marker defined from  $\mathcal{E}$ .

With this language, there are infinitely many cryptosystems that can be initialized, depending on the parameters of  $\mathcal{P}$ . We will discuss the security of this system and how to compute each component in Section 4. For now, we show that even if a breach were to occur where  $\mathcal{U}_I$  were made public, the cryptosystem still provides a reasonable amount of security.

**Proposition 3.3.** *Suppose that  $\mathcal{P} = (n, \mathbb{K}, \mathcal{U}_I, \mathcal{R}_I, <_B, \eta, \mathcal{E}, \tau)$  is a set of protocol parameters as in Definition 3.1 such that  $\mathcal{U}_I = \mathcal{R}_I$ . Denote  $r = |\mathcal{U}_I|$  and let  $m$  be the maximum number of monomial terms of a polynomial in  $\mathcal{U}_I$ . Then*

$$\# \text{ of possible encryption keys of } \mathcal{P} \leq m^r.$$

*In particular, the number of operations needed to conduct a brute-force attack of a system where  $\mathcal{U}_I$  is public is  $O(m^r)$ .*

*Proof.* Any potential initial ideal of  $I$  can be read off from  $\mathcal{U}_I$  by picking one term from each  $f \in \mathcal{U}_I$ . There are at most  $m$  such terms for each  $f$ , leaving at most  $m^r$  many possibilities.

The remaining operations of converting each list of monomials to an encryption key via  $\eta$ , and decrypting the message using  $\mathcal{E}^{-1}$  is some constant multiple of  $m^r$  (for bounded key and message sizes).  $\square$

*Remark 3.4.* Note that each potential list of monomials from the proof of Proposition 3.3 may not produce a valid initial ideal since there may not exist a monomial order of  $\mathbb{K}[x_1, \dots, x_n]$  which yields that particular list of monomials (for example, there is no monomial order which picks  $x_1x_2$  as an initial term of  $x_1^2 + x_1x_2 + x_2^2$ ). We also saw this in Section 3.1 where there were a priori  $2^3$  possible lists of monomials, but only 5 defined actual keys.

In another extreme, suppose that  $\mathcal{U}_I$  remains private and an attacker wanted to generate all potential encryption keys for a system  $\mathcal{P}$  without computing a universal Gröbner basis for  $I$ . They would need to generate all encryption keys using  $\eta$  associated to  $\mathcal{M}_k$ . With the assumption that the allowed exponent vectors have entries strictly smaller than  $k$ , the number of possible encryption keys is a doubly exponential in  $n$ . Even in the square-free case where  $k = 2$ , a brute-force attack would be unwieldy.

**Proposition 3.5.** *Let  $\mathcal{P} = (n, \mathbb{K}, \mathcal{U}_I, \mathcal{R}_I, <_B, \eta, \mathcal{E}, \tau)$  be as in Definition 3.1 such that  $\eta$  is a map whose domain contains  $\mathcal{M}_k$ . Then*

$$\# \text{ of possible encryption keys of } \mathcal{P} \leq 2^{k^n}.$$

*In particular, the number of operations needed to conduct a brute-force attack of a system  $\mathcal{P}$  is  $O(2^{k^n})$ .*

*Proof.* There are exactly  $k^n$  monomials in  $\mathbb{K}[x_1, \dots, x_n]$  whose exponent vector entries are bounded above by  $k$ . A minimal generating set of a monomial ideal is a subset of this list of monomials, so the number of possible subsets is  $2^{k^n}$ . Different choices of a subset may result in the same monomial ideal, but each subset results in one case that needs to be checked. The complexity bound follows similarly to the proof of Proposition 3.3.  $\square$

*Remark 3.6.* An attacker could also try to compute all possible initial ideals of  $I$  by enumerating all monomial orders on  $\mathbb{K}[x_1, \dots, x_n]$  and computing a Gröbner basis directly for each one. Using only the lexicographic orders shows that this approach has at least  $n!$  cases to check (there are also other monomial orders which are not equivalent to a lexicographic order). This, together with the amount of time needed to perform one Gröbner basis computation would make this approach untenable.

#### 4. SECURITY ANALYSIS

The security of the protocol in the last section depends on the fact that it is generally difficult to compute the universal Gröbner basis for an ideal. Much work has been done to describe procedures for this computation (for example, see [16, 24]), and there exist programs to compute such a basis. However, even for small ideals, the computational complexity can be a barrier to finding an explicit list. For example, let  $\text{Det}_{t,m,n}$  denote the ideal in the polynomial ring in  $mn$  variables generated by the  $t \times t$  minors of an  $m \times n$  generic matrix. For  $\text{Det}_{3,4,4}$  alone, there are over 160,000 possible initial ideals [16], so a brute-force universal Gröbner basis construction would require over 160,000 individual (albeit simplified) Gröbner basis computations.

This section is dedicated to showing that an attacker with no prior knowledge of how  $\mathcal{U}_I$  was constructed would require an unreasonable amount of time to compute  $\mathcal{U}_I$ . In Section 5, we contrastingly show that Party *A* can construct  $\mathcal{U}_I$  with relative ease if  $I$  is chosen with some symmetries. In this section, we will also show that Party *B* does not require much time to encrypt a message, and that the required time scales well with any additional complexity that *A* adds to the system  $\mathcal{P}$ .

**4.1. Quantum Resistance.** In Section 2.2, we outlined the steps needed to compute  $GFan(I)$ , the Gröbner fan of  $I$ , and by extension the state polytope of  $I$ ,  $State(I)$ . Each maximal cone in  $GFan(I)$  (or dually, any vertex of  $State(I)$ ), defines one possible initial ideal of  $I$ , and hence defines a possible encryption key that Party *B* could have used. Based on how Party *A* decrypts messages using  $\mathcal{P}$  when  $\tau = \emptyset$ , a rational attacker without any prior knowledge of the symmetries of  $I$  would need to compute  $GFan(I)$  to guarantee the decryption of the message (see Propositions 3.3 and 3.5 for other less effective attacks).

The algorithm proposed in [16] is a practical implementation of the theoretical algorithm proven in [24, Section 7]. It involves three main steps that are iterated at each vertex of  $\text{State}(I)$  (i.e. on the graph of the polytope  $\text{State}(I)$ ). These steps are technical, and are only briefly outlined in Section 2.2. The complexity of performing these steps was analyzed in [16] and is highlighted in the next theorem. It ultimately shows that an attacker would require an inordinate amount of resources to compute the Gröbner fan directly, assuming  $I$  is sufficiently complex.

**Theorem 4.1.** *Let  $(V, E)$  be the graph of the Gröbner fan of  $I$ . The time complexity for computing this graph given a marked reduced Gröbner basis is the class of functions*

$$O\left(\sum_{\mathcal{G} \in V} T_{\text{facets}}(\mathcal{G}) + \sum_{(\mathcal{G}_1, \mathcal{G}_2) \in E} T_{\text{shoot}}(\mathcal{G}_1) + \sum_{(\mathcal{G}_2, \mathcal{G}_1) \in E} T_{\text{flip}}(\mathcal{G}_1, \mathcal{G}_2)\right),$$

which can be written as

$$O\left(\sum_{\mathcal{G} \in V} T_{\text{lp}}(n, r(\mathcal{G}))r(\mathcal{G}) + \sum_{(\mathcal{G}_1, \mathcal{G}_2) \in E} r(\mathcal{G}_1)n^2 + T_{\text{lp}}(n, \mathcal{G}_1) + \sum_{(\mathcal{G}_2, \mathcal{G}_1) \in E} T_{\text{flip}}(\mathcal{G}_1, \mathcal{G}_2)\right),$$

where  $r(G)$  is the number of non-leading terms in the marked reduced Gröbner basis  $G$ , and  $T_{\text{lp}}$  is the time complexity related to finding an interior point to a cone. The first two terms are bounded by a polynomial in the size of the output. The computational complexity of the third term is NP-hard.

*Proof.* The complexity description and bound for the first two terms can be found in [16, Theorem 5.1]. The statement about the third term can be found in [18, Section 5], using a result from tropical geometry. More specifically, a tropical variety is a union of Gröbner regions, making it a subfan of a Gröbner fan. In [26, Section 3], several decision problems related to tropical varieties (which could be solved given a Gröbner fan [26, Remark 3.4]) are shown to be NP-hard.  $\square$

*Example 4.2.* Let  $\text{Det}_{t,m,n}$  denote the ideal in  $\mathbb{K}[x_{11}, \dots, x_{mn}]$  generated by the  $t \times t$  minors of the matrix:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}.$$

Using the **Gfan** software package [19] of **Macaulay2**, the authors in [16] were able to show that on a standard 2.4 GHz Pentium processor at the time (i.e. 2005),  $\text{Det}_{3,4,4}$  has 163,032 many Gröbner regions in its Gröbner fan, and hence 163,032 possible initial ideals. Note that  $\text{Det}_{3,4,4}$  is an ideal that initially is only generated by 16 polynomials in 16 variables. Without the use of the symmetries of  $\text{Det}_{3,4,4}$ , the full computation took approximately 14 hours. Using the symmetries of  $\text{Det}_{3,4,4}$ , the computation time for the full-dimensional cones took only 7 minutes. While a modern computer could do this considerably faster, the number of computations remains unchanged, so scaling  $m$  and  $n$  would produce similar results.  $\square$

In light of the recent threats that quantum computers pose to traditional public-key cryptosystems, it becomes increasingly important to develop new techniques to secure data that can resist attacks from a quantum computer. Post-quantum cryptography is a relatively new field, but there are six main classes of algorithms which are considered to be resistant to quantum attacks [5]. Therefore, to show that our protocol  $\mathcal{P}$  is quantum-resistant, it suffices to demonstrate that it belongs to one of these categories.

Since  $State(I)$  has vertices in  $\mathbb{Z}^n$  [24, Chapter 2], it can be viewed as a convex polytope in the standard integer lattice  $\mathbb{Z}^n$ . Computing  $GFan(I)$  directly is NP-hard by Theorem 4.1, and would generally require millions of simplified Gröbner basis calculations, in addition to polytope computations (like finding normal vectors to the facets, a Gröbner walk through the graph of  $State(I)$ , etc.). Therefore, these structures would be considered lattice-based primitives. Lattice-based cryptography is one of the previously mentioned approaches considered resistant against quantum attacks [5]. To prove the security of  $\mathcal{P}$ , we will need to show that computing  $GFan(I)$  is at least as difficult as a lattice-based problem which is considered quantum-resistant. Additionally, if  $\tau$  is used, then it needs to be chosen to be quantum-resistant too.

**Theorem 4.3.** *Assume that lattice-based primitives secured by the Shortest Vector Problem (SVP) are quantum-resistant. If  $\tau \neq \emptyset$ , then also assume that  $\tau$  is quantum-resistant. Then a protocol  $\mathcal{P}$  for which  $|\mathcal{U}_I|$  is sufficiently large is quantum-resistant.*

*Proof.* We need to demonstrate that computing a Gröbner fan is as difficult as solving a certain SVP. To do this, we will show that when properly rephrased, knowing  $\mathcal{U}_I$  will provide a solution to a certain SVP.

Recall that  $GFan(I)$  is a union of cones, called Gröbner regions, and in the interior of each cone  $GR_<(I)$ , there is at least one integer lattice point  $x \in GR_<(I) \cap \mathbb{Z}^n$  such that  $|x| \leq |y|$  for all  $y \in \text{int}(GR_<(I) \cap \mathbb{Z}^n)$  where  $|\cdot|$  is the usual Euclidean norm on  $\mathbb{R}^n$ . Consider the following question: Among all Gröbner regions  $GR_<(I) \subset GFan(I)$ , which one has an interior integer lattice point  $x \in GR_<(I) \cap \mathbb{Z}^n$  which is closest to the origin?

Suppose that  $GFan(I)$  is known. Then the normal vectors to facets of  $State(I)$  have been computed, and therefore lattice generators for each rational polyhedral cone  $GR_<(I) \cap \mathbb{Z}^n$  are known. For simplicial cones, given integer vectors  $v_1, \dots, v_n$  that generate the cone  $GR_<(I) \cap \mathbb{Z}^n$ , the interior points have the form  $c_1v_1 + \dots + c_nv_n$  with  $c_i > 0$  and  $c_i \in \mathbb{Z}$ . The interior lattice point with the closest distance is precisely  $v_1 + \dots + v_n$  (recall that the cones computed in the algorithm are contained in the positive orthant [16, Definition 2.8]). The non-simplicial case can be computed using integer linear programming techniques for rational convex polytopes, which runs in polynomial time for a fixed dimension [12]. In particular,  $GFan(I)$  has more information than what is needed to answer the question posed in the last paragraph, and the question can be answered in polynomial time.

On the other hand, given  $GFan(I)$ , we could take the collection of all appropriately scaled lattice vectors generating all Gröbner regions, and select some minimal generating set  $\mathcal{C}$  from this collection. Note that the shortest vector that answers the above question is a rational combination of some of these generators. By scaling, we can assume that the rational combination yields an integer combination. The (now scaled) vectors in

$\mathcal{C}$  generate a lattice, and we can ask what the shortest vector is in that lattice, which is a specific independent SVP. The solution to this SVP is exactly equal to the vector computed in the previous paragraph, showing that computing  $GFan(I)$  is at least as difficult as an SVP.

Finally, when  $\tau \neq \emptyset$ , an attacker could try to compute  $\tau^{-1}(\tau(K_B))$  directly, circumventing the security that  $GFan(I)$  provides, so  $\tau$  needs to be a hash function which is resistant to quantum attacks.  $\square$

*Remark 4.4.* There are additional ways in which computing  $GFan(I)$  emulates the spirit of other lattice-based problems used in post-quantum security. For example, given a vertex  $\mathbf{v} \in \mathbb{Z}^n$  of  $State(I)$ , as a face of the polytope, is characterized by the inequality  $w \cdot \mathbf{v} > w \cdot \mathbf{u}$  for all other  $\mathbf{u} \in State(I)$ , where  $w$  is the weight order associated to the vertex  $v$ . There is another vertex  $v'$  of  $State(I)$  which has the furthest distance from  $v$ , and is precisely the point of  $State(I)$  which minimizes the functional  $f(P) = w \cdot P$  over  $State(I)$ .

Many algebraic problems haven't been studied in the context of post-quantum cryptography, so many of the established lattice-based problems that have been formulated do not immediately translate to algebraic settings. Further research is needed to establish independent algebraic or combinatorial problems which are considered quantum resistant.

**4.2. Other complexities associated with  $\mathcal{P}$ .** We will start with the complexity of the one-time computation of  $\mathcal{R}_I$  by Party  $A$ . There are numerous ways to trim the ideal  $I$ , given that  $\mathcal{U}_I$  is already known. One possibility is to choose some monomial order  $<$  of  $\mathbb{K}[x_1, \dots, x_n]$  and compute a Gröbner basis for  $I$  using  $\mathcal{U}_I$ . This also generates the ideal  $I$ , and generally involves far fewer than  $|\mathcal{U}_I|$  many generators.

**Theorem 4.5.** *Let  $\mathcal{U}_I$  be a universal Gröbner basis for an ideal  $I \subset R = \mathbb{K}[x_1, \dots, x_n]$  and  $<$  some monomial order on  $R$ . Then a Gröbner basis for  $I$  with respect to  $<$  is one possible choice of  $\mathcal{R}_I$ . Furthermore, if  $|\mathcal{U}_I| = N$ , then  $\mathcal{R}_I$  can be computed with  $O(N)$  many operations.*

*Proof.* Since  $\mathcal{U}_I$  is a universal Gröbner basis, selecting Gröbner generators of  $I$  for a fixed  $<$  is a simple procedure which involves marking the initial terms of each of the  $N$  many elements of  $\mathcal{U}_I$ , and then eliminating redundancies and any elements whose initial terms are not needed to generate  $\text{in}_{<}(I)$ .  $\square$

Next we shift to the complexity of Party  $B$  finding a Gröbner basis of  $I$ . In not having access to  $\mathcal{U}_I$ ,  $B$  will need to compute a Gröbner basis directly. The next theorem provides the complexity of this computation.

**Theorem 4.6.** [1, Proposition 1] *Let  $(f_1, \dots, f_m)$  be a system of homogeneous polynomials in  $\mathbb{K}[x_1, \dots, x_n]$  with  $\mathbb{K}$  an arbitrary field. The number of operations in  $\mathbb{K}$  required to compute a Gröbner basis of the ideal  $I$  generated by  $(f_1, \dots, f_m)$  for a graded monomial ordering up to degree  $D$  is bounded by*

$$O\left(mD\binom{n+D-1}{D}^\omega\right), \text{ as } D \rightarrow \infty$$

where  $\omega$  is the exponent of matrix multiplication over  $\mathbb{K}$ .

Recall that the exponent of matrix multiplication is a constant  $\omega$ , depending on  $n$ , which is used to bound the complexity of matrix multiplication of  $n \times n$  matrices. Suppose that Party  $A$  has already chosen a preset number of variables, so that  $n$  is fixed, as well as the number of elements in a generating set (which is  $m$ ). If we allow the degrees of the  $f_i$ 's to vary, then we are allowing the degree of the polynomials used in the Gröbner basis computation to vary, and thus the maximum degree of the Gröbner basis elements (i.e. the variable  $D$ ) may also vary. Then, as a function of  $D$ ,

$$\begin{aligned} \binom{n+D-1}{D} &= \frac{(n+D-1)(n+D-2) \cdots (D+1)D!}{D!(n-1)!} \\ &= \frac{(n+D-1)(n+D-2) \cdots (D+1)}{(n-1)!} \end{aligned}$$

which is a polynomial of degree  $n-1$  in  $D$ . Therefore,

$$\begin{aligned} mD \binom{n+D-1}{D}^\omega &= mD \left( \frac{(n+D-1)(n+D-2) \cdots (D+1)D!}{D!(n-1)!} \right)^\omega \\ &= O(D^{\omega(n-1)+1}) \end{aligned}$$

Even if  $A$  changes the complexity of a protocol  $\mathcal{P}$  based on the degree of the generators considered (e.g. changes  $D$  to  $D+1$ ), it only affects  $B$ 's Gröbner basis computation polynomially.

**Corollary 4.7.** *With the same notation as Theorem 4.6, let  $n$  and  $m$  be fixed positive integers. Then the number of operations in  $\mathbb{K}$  required to compute a Gröbner basis of the ideal  $I$  is polynomial in  $D$ . More precisely, it is bounded by*

$$O(D^{\omega(n-1)+1}), \text{ as } D \rightarrow \infty$$

*Remark 4.8.* By a theorem of Dubé, the maximum degree  $D$  of polynomials appearing in a Gröbner basis is bounded by  $(d^2 + 2d)^{2^{n-1}}$  where  $d$  is the maximum degree of the polynomials in  $\{f_1, \dots, f_m\}$  (see [14]). Note that this bound grows very large even for small  $n$  and  $d$ , but in our case, Party  $B$  is not computing a random Gröbner basis for a random ideal. On the contrary, the maximum degree of an element in  $\mathcal{U}_I$  is an upper bound for the  $D$  that Party  $B$  would encounter.

On the other hand, since  $\omega$  is dependent on  $n$ , the complexity of computing a Gröbner basis is doubly exponential in  $n$ . This complexity assumes the worst case however. It is expected that only a mild increase in the complexity will result from adding new generators with a similar structure to the current  $f_i$  (for instance, adding binomial generators). In summary, Theorem 4.6 tells us that:

- An increase in the maximum degree of the generators  $f_1, \dots, f_m$  results in a polynomial increase in time for  $B$ .
- An increase in the number of generators  $m$  for fixed  $D$  and  $n$  also results in a polynomial increase in time for  $B$  (with complexity  $O(m)$ ).

- An increase in  $n$  could result in a doubly exponential jump in the worst case for Party  $B$ 's computation time [18, Section 5], so care in selecting  $I$  should be taken.

We conclude this section with a brief statement about the maximum time needed for Party  $A$  to decrypt a message when  $\tau = \emptyset$ .

**Proposition 4.9.** *Let  $\mathcal{P} = (n, \mathbb{K}, \mathcal{U}_I, \mathcal{R}_I, <_B, \eta, \mathcal{E}, \tau)$ . Suppose that  $N$  is the number of Gröbner regions in the Gröbner fan  $GFan(I)$ , and let  $C(\mathcal{E})$  be the maximum amount of time needed to decrypt a ciphertext (of bounded length) using  $\mathcal{E}$ . If  $\tau = \emptyset$ , then the maximum amount of time needed to decrypt a message (of bounded length) sent using  $\mathcal{P}$  is  $N \cdot C(\mathcal{E})$ .*

*Example 4.10.* Let us continue with Example 4.2 when  $\tau = \emptyset$ . We will let  $\mathcal{E}$  be the RSA encryption/decryption scheme. Suppose that the decryption time using  $\mathcal{E}$  is approximately  $C(\mathcal{E}) \sim 0.00055$  seconds on a standard laptop. For this example, Party  $A$  would have to check at most  $N = 163,032$  keys, taking at most 90 seconds to decrypt the message. Having partial information from Party  $B$  in the message about the key  $K_B$  would reduce this time significantly and may be needed to bring down the decryption time to a more reasonable number, for practical purposes.

It is worth noting that in this example, Party  $B$  would only take about 0.3 seconds to compute a single Gröbner basis for a fixed order  $<_B$  (found by dividing the 14 hours needed to compute the Gröbner fan by the number of Gröbner regions).  $\square$

*Remark 4.11.* When  $\tau = \emptyset$ , the protocol  $\mathcal{P}$  offers greater security since no information about  $K_B$  is sent publicly, but this comes at a cost to Party  $A$  who now needs to iterate through all possible encryption keys. Choosing a symmetric  $\mathcal{E}$  can help reduce this time. On the other hand, if  $\tau \neq \emptyset$ , then Party  $A$  can decrypt a ciphertext very quickly, at the cost of information about  $K_B$  being public using  $\tau$ . In this case, the security of the system is also dependent on the security of  $\tau$ .

## 5. EFFECTIVE INITIALIZATION OF $\mathcal{U}_I$

In the last section, we saw that the problem of an attacker trying to compute  $\mathcal{U}_I$  directly is intractable if the set is sufficiently large, leading one to wonder how it is possible for Party  $A$  to even initialize the system  $\mathcal{P}$ . Here we will introduce a way to easily compute examples of  $\mathcal{U}_I$  using toric ideals of graphs. The knowledge of which graph was used in the construction would provide a trapdoor to an attacker computing  $\mathcal{U}_I$ , so it is imperative that this information be kept private. The content of this section is also of independent interest to combinatorial algebraists, especially those working with toric ideals of graphs and geometric vertex decomposition (see [13] for example).

**5.1. The toric ideal of a graph.** Let  $G = (V(G), E(G))$  be a finite simple graph where  $V(G) = \{v_1, \dots, v_n\}$  is the set of vertices of  $G$ , and  $E(G) = \{e_1, \dots, e_r\}$  is the set of edges of  $G$  with  $e_i = \{v_{i_j}, v_{i_k}\}$  an unordered pair of vertices which we call the endpoints of  $e_i$ . Given  $G$ , we can associate an ideal  $I_G$  to it. Let  $\mathbb{K}[E(G)] = \mathbb{K}[e_1, \dots, e_r]$  and  $\mathbb{K}[V(G)] = \mathbb{K}[v_1, \dots, v_n]$  be two polynomial rings over  $\mathbb{K}$  with the edges and vertices

viewed as indeterminates, respectively. Then consider the  $\mathbb{K}$ -algebra homomorphism,

$$\varphi_G : \mathbb{K}[E(G)] \rightarrow \mathbb{K}[V(G)]$$

defined on the indeterminates  $e_i$  by  $\varphi_G(e_i) = v_{i_j}v_{i_k}$  where  $e_i = \{v_{i_j}, v_{i_k}\}$  for all  $i \in \{1, \dots, r\}$ . The kernel of the map  $\varphi_G$  will be denoted by  $I_G$  and is called the *toric ideal of the graph  $G$* .

There is a convenient graph-theoretic description of the elements of  $I_G$ . First, recall that a *walk* of length  $k$  in a graph  $G$  is an alternating sequence of vertices and edges

$$W = (v_{i_0}, e_{i_1}, v_{i_1}, e_{i_2}, v_{i_2}, \dots, v_{i_{k-1}}, e_{i_k}, v_{i_k})$$

where  $e_{i_j} = \{v_{i_{j-1}}, v_{i_j}\}$  for  $j = 1, \dots, k$ . We say that the walk is even if  $k$  is even, and closed if  $v_{i_k} = v_{i_0}$ . We can associate a binomial in  $\mathbb{K}[E(G)]$  to  $W$  by  $e_{i_1}e_{i_3} \cdots e_{i_{k-1}} - e_{i_2}e_{i_4} \cdots e_{i_k}$ . In general, all binomials associated to closed even walks of  $G$  are in  $I_G$ . It turns out that these binomials generate  $I_G$ .

**Theorem 5.1.** [27, Proposition 10.1.5] *Let  $G$  be a finite simple graph. Then the toric ideal  $I_G$  of  $G$  is generated by the set of binomials*

$$\{e_{i_1}e_{i_3} \cdots e_{i_{k-1}} - e_{i_2}e_{i_4} \cdots e_{i_k} | (e_{i_1}, \dots, e_{i_k}) \text{ is a closed even walk of } G\}.$$

There are generally infinitely many closed even walks of a graph  $G$ . To achieve a finite generating set, we consider only *primitive closed even walks*.

**Definition 5.2.** Let  $e^A := e_1^{A_1} \cdots e_r^{A_r}$ . A binomial  $e^\alpha - e^\beta \in I_G$  is called *primitive* if there is no other binomial  $e^\gamma - e^\delta \in I_G$  such that  $e^\gamma | e^\alpha$  and  $e^\delta | e^\beta$ .

Not only do the set of primitive closed even walks generate the ideal  $I_G$ , they are also a universal Gröbner basis of  $I_G$ .

**Theorem 5.3.** [27, Proposition 10.1.9] *Let  $G$  be a finite simple graph. Then the set of all primitive binomials of  $I_G$  define a universal Gröbner basis of  $I_G$ , denoted by  $\mathcal{U}(I_G)$ .*

By taking  $I = I_G$  for some  $G$ , we would automatically have a convenient description for  $\mathcal{U}_I = \mathcal{U}(I_G)$ . Even with this description, computing  $\mathcal{U}(I_G)$  directly can be computationally difficult. In fact, the number of elements of  $\mathcal{U}(I_G)$  can grow very quickly. For instance,  $\mathcal{U}(I_{K_8})$  has over 40,000 elements [13]. A description of how to compute this set can be found in [24, Section 7]. A graph-theoretic characterization of primitive closed even walks of a graph can be found in [25].

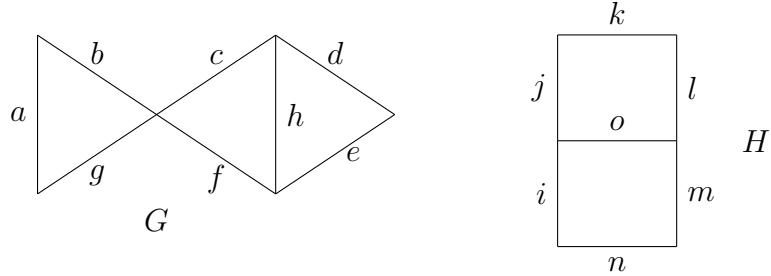
**5.2. Generating large graphs.** In this section, we will show that it is possible to generate graphs for which  $\mathcal{U}(I_G)$  is recursively computable, and for which  $|\mathcal{U}(I_G)|$  is sufficiently large to ensure the security of the protocol  $\mathcal{P}$ . Furthermore, this can be done in polynomial time, depending on the number of constructive steps detailed below. We are going to introduce three operations for this purpose.

5.2.1. *Gluing along a vertex.* Given a graph  $G$ , we can glue a disjoint graph  $H$  to  $G$  along a vertex by selecting some  $v_G \in V(G)$  and  $v_H \in V(H)$  and identifying the two vertices. More specifically, we define a new graph, denoted  $G \star_{v_G, v_H} H$  (or simply  $G \star H$  when  $v_G$  and  $v_H$  are understood), constructed as a disjoint union of the two graphs modulo the relation where  $v_G$  equals  $v_H$ :

$$G \sqcup H /_{v_G \sim v_H}.$$

In general, computing  $\mathcal{U}(I_{G \star H})$  can be difficult given  $\mathcal{U}(I_G)$  and  $\mathcal{U}(I_H)$  since new primitive closed even walks could be formed using odd cycles of  $G$  and  $H$  being linked through  $v_G = v_H$ . Furthermore, these odd cycles are not explicitly recorded in the list of primitive closed even walks, so we can't expect to compute the new list using the previous two lists alone. However, there is a special case where this operation works well. We start with an illustrative example.

*Example 5.4.* Consider the graphs  $G$  and  $H$  pictured below.

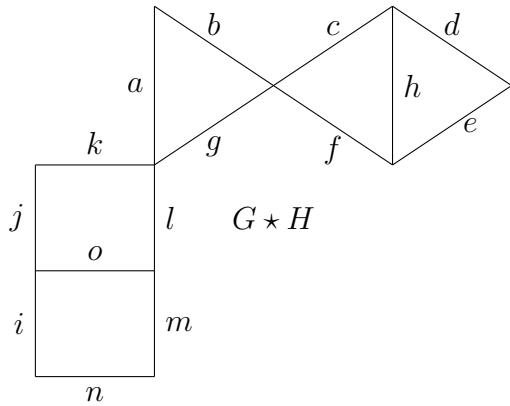


The set of primitive closed even walks for each can be directly computed as

$$\mathcal{U}(I_G) = \{ce - df, acf - bgh, ac^2e - bdgh, adf^2 - begh\}$$

$$\mathcal{U}(I_H) = \{im - no, jl - ok, ikm - jln\}.$$

We can create a new graph  $G \star H$  by gluing on  $H$  at a vertex of  $G$ , say at the vertex incident to  $a$  and  $g$  in  $G$ , and  $k$  and  $\ell$  in  $H$ :



We can check that the set of primitive closed even walks for the resulting graph is the union of both lists

$$\mathcal{U}(I_{G \star H}) = \{ce - df, acf - bgh, ac^2e - bdgh, adf^2 - begh, im - no, jl - ok, ikm - jln\}.$$

In fact, we would have arrived at the same result if we chose any other pair of vertices to identify.  $\square$

In general,  $\mathcal{U}(I_{G \star H})$  contains the union of  $\mathcal{U}(I_G)$  and  $\mathcal{U}(I_H)$ . When  $H$  is a bipartite graph however (i.e. contains no odd-length cycles), we get the reverse containment too. The next proposition is motivated by [21, Section 2.0.3].

**Proposition 5.5.** *Let  $G$  and  $B$  be finite simple graphs such that  $B$  is bipartite and  $V(G) \cap V(B) = \emptyset$ . Let  $v_G \in V(G)$  and  $v_B \in V(B)$ , and form a new graph  $G \star B$  by identifying  $v_G$  and  $v_B$ . Then*

$$\mathcal{U}(I_{G \star B}) = \mathcal{U}(I_G) \sqcup \mathcal{U}(I_B).$$

*Proof.* One direction is clear, since any primitive closed even walk of  $G$  or  $B$  must remain primitive in  $G \star B$ . Therefore  $\mathcal{U}(I_G) \sqcup \mathcal{U}(I_B) \subseteq \mathcal{U}(I_{G \star B})$ .

For the other direction, note that by [25] (and rephrased in [11, Theorem 1.7]), a primitive closed even walk is either an even cycle, or contains at least two odd cycles. If  $G \star B$  has a primitive closed even walk  $\Gamma$  involving odd cycles, then these odd cycles must be in  $G$  since  $B$  is bipartite. If  $\Gamma$  includes an edge of  $B$ , then the walk must pass through  $v_B = v_G$  at least twice (in order to start and end in  $G$ ). The edges between the first instance of  $v_B$  and the second instance will define an even cycle  $\Gamma_B$  of  $B$ , which is not possible by [17, Lemma 2.2 (ii)].

Similarly, if there is some even cycle of  $G \star B$  that is not contained in  $G$  or  $B$  exclusively, then we can write it as

$$v_{i_1}, e_{i_1}, v_{i_2}, e_{i_2}, \dots, e_{i_k}, v_{i_{k+1}}$$

where all  $e_i$  and  $v_i$  are distinct except for  $v_{i_1} = v_{i_{k+1}}$ . If the cycle uses edges in  $B$ , then  $v_B$  would appear twice in the list, unless  $v_{i_1} = v_{i_{k+1}} = v_B$ , which would mean that all of the edges are either entirely in  $G$  or entirely in  $B$ , a contradiction.  $\square$

**5.2.2. Star contractions and subdivisions.** Next, we will consider a graph operation called a star contraction. Its use in the context of toric ideals of graphs was first introduced in [21].

**Definition 5.6.** [21, Definition 3.4] Let  $G$  be a graph with  $v \in V(G)$ , and  $N_E(v)$  be the list of edges in  $E(G)$  which are incident to  $v$ . The *star contraction* of  $G$  at  $v$  is the graph  $G_v$  formed by performing an edge contraction on all of the edges in  $N_E(v)$  simultaneously. That is,  $G_v$  is constructed by first deleting all edges in  $N_E(v)$ , and then identify all vertices in the neighborhood of  $v$ .

**Example 5.7.** [21, Example 3.0.6] Consider the star contraction of the graph  $G$  below along the vertex  $v$  incident to  $e$  and  $f$ . The list of primitive closed even walks of  $G$  and  $G_v$  have also been listed. Notice that we can get the list of elements in  $\mathcal{U}(I_{G_v})$  from  $\mathcal{U}(I_G)$  by setting  $e = f = 1$ .

$$\begin{array}{ccc}
 \begin{array}{c} \text{a} & & \text{b} \\ \text{f} & \text{---} & \text{g} \\ \text{e} & & \text{d} \end{array} & \longrightarrow & 
 \begin{array}{c} \text{b} \\ \text{g} \\ \text{a} \\ \text{d} \\ \text{c} \end{array} \\
 \langle ace - bdf, ae - fg, bd - cg \rangle & \longrightarrow & \langle ac - bd, a - g, bd - cg \rangle
 \end{array}$$

□

To simplify notation, we will define the ring homomorphism

$$\pi_v : \mathbb{K}[E(G)] \rightarrow \mathbb{K}[E(G) \setminus N_E(v)]$$

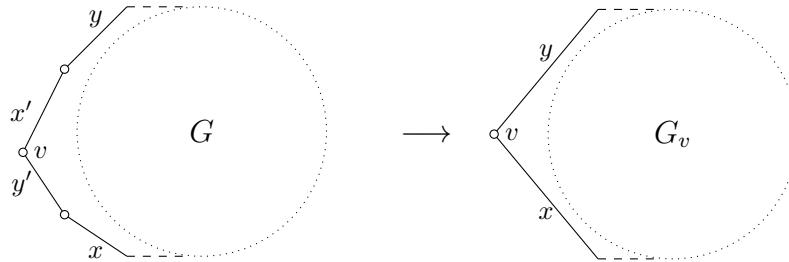
on generators by  $e \mapsto 1$  if  $e \in N_E(v)$ , and  $e \mapsto e$  otherwise. To avoid any issues with defining primitive walks for multigraphs (like in the previous example), we will restrict to the case when the star contraction results in a simple graph.

**Lemma 5.8.** [21, Theorem 3.10] *Let  $G$  be a finite simple graph. Suppose that  $v \in V(G)$  is such that  $G_v$  is a simple graph. Then*

$$\mathcal{U}(I_{G_v}) \subseteq \pi_v(\mathcal{U}(I_G))$$

These results allow us to produce new graphs through star contractions while still having control over the enumeration of primitive closed even walks. Note that even though the containment in Lemma 5.8 is generally proper, the set  $\pi_v(\mathcal{U}(I_G))$  still defines a universal Gröbner basis of  $I_{G_v}$  (although not a reduced basis).

This operation can also be undone to produce larger graphs, a process called a *star subdivision*, generally discussed in [4] for toric ideals of graphs. We will show that in the special case when the subdivision is done along a vertex of degree 2, the list of primitive closed even walks has an explicit description. To do this, consider a graph with the following structure:



An important feature of such a graph is that the star contraction along the vertex  $v$  incident to  $x'$  and  $y'$  results in another degree 2 vertex (which we also call  $v$  by an abuse of notation). In this case, we will say that  $G$  is the (unique) star subdivision of  $G_v$  along  $v$ . More generally, there are usually multiple star subdivisions of a graph (see [4, Definition 3.0.3]) if the degree of  $v$  is greater than two.

To demonstrate the effect on the list of primitive closed even walks after the star subdivision, consider the following map on polynomial rings,

$$\psi_v : \mathbb{K}[\mathbf{e}, x, y] \rightarrow \mathbb{K}[\mathbf{e}, x, y, x', y']$$

defined by  $x \mapsto xx'$ ,  $y \mapsto yy'$ , and  $f \mapsto f$  for  $f \in \mathbf{e} = E(G) \setminus \{x, x', y, y'\}$ . Notice that if  $m_1x - m_2y$  is a closed even walk of  $G_v$  (where  $m_1, m_2$  are monomials with support in  $\mathbf{e}$ ), then  $\psi_v(m_1x - m_2y) = m_1xx' - m_2yy'$  is a closed even walk of  $G$ . The next result shows that the same is true for primitive walks.

**Proposition 5.9.** *Let  $G$  be a finite simple graph and suppose that  $v \in V(G)$  has degree 2 in  $G$  and degree 2 in the star contraction  $G_v$ . Let the edges incident to  $v$  be labeled as above. Then*

$$\mathcal{U}(I_{G_v}) = \pi_v(\mathcal{U}(I_G))$$

and

$$\mathcal{U}(I_G) = \psi_v(\mathcal{U}(I_{G_v})).$$

*Proof.* First note that for  $\Gamma \in I_G$  and  $\gamma \in I_{G_v}$ :

$$\psi_v(\pi_v(\Gamma)) = \Gamma \quad \text{and} \quad \pi_v(\psi_v(\gamma)) = \gamma,$$

so it suffices to prove the first equality to show that the second claim is also true.

By the structure of primitive closed even walks (see [25]), any primitive closed even walk of  $G$  that passes through  $v$  must also pass through the edges  $x, y, x'$  and  $y'$ . Furthermore, it would either pass through all 4 edges exactly once or twice. Therefore, all binomials in  $\mathcal{U}(I_G)$  which correspond to a primitive walk passing through  $v$  must be of the form:

$$m_1xx' - m_2yy' \quad \text{or} \quad m_1(xx')^2 - m_2(yy')^2$$

where  $m_1$  and  $m_2$  are monomials with support in  $\mathbf{e} = E(G) \setminus \{x, x', y, y'\}$ .

Let  $\Gamma = m_1xx' - m_2yy'$  be a primitive closed even walk of  $G$ . Assume that  $\pi_v(\Gamma) = m_1x - m_2y$  is not primitive in  $G_v$ . Then there would be some other binomial  $\gamma = m_3 - m_4 \in I_{G_v}$  such that  $m_3|m_1x$  and  $m_4|m_2y$ . If the support of  $m_3$  and  $m_4$  does not include  $x$  or  $y$ , then  $\gamma$  is unaffected by the star subdivision of  $G_v$  and corresponds to a closed even walk of both  $G$  and  $G_v$ , contradicting that  $\Gamma$  is primitive.

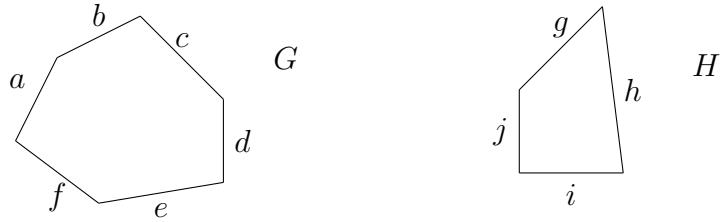
The only other case is when  $x|m_3$  and  $y|m_4$  (since  $\gamma$  passing through  $v$  must also pass through both  $x$  and  $y$ ). In this case,  $\psi_v(\gamma) = m_3x' - m_4y'$  defines a closed even walk of  $G$  such that  $m_3x'|m_1xx'$  and  $m_4y'|m_2yy'$ , also contradicting the fact that  $\Gamma$  is primitive. The case  $\Gamma = m_1(xx')^2 - m_2(yy')^2$  is similar. Since walks that do not pass through  $v$  are unaffected by the star contraction, we have shown that  $\pi_v(\mathcal{U}(I_G)) \subseteq \mathcal{U}(I_{G_v})$ . Together with Lemma 5.8, we have shown that  $\mathcal{U}(I_G) = \psi_v(\mathcal{U}(I_{G_v}))$ , as required.  $\square$

**5.2.3. Gluing even cycles.** Finally, we can obtain new graphs for which we can recursively generate the list of primitive closed even walks using cycle gluing. We can do this similarly to the vertex gluing defined earlier, except that we identify two edges instead of two vertices. More specifically, given disjoint graphs  $G$  and  $H$ , and edges  $e_G \in E(G)$  and  $e_H \in E(H)$ , we can produce a new graph of the form

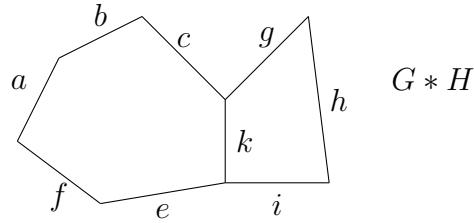
$$G \sqcup H / e_G \sim e_H$$

which we denote by  $G *_{e_G, e_H} H$  (or simply  $G * H$  when  $e_G$  and  $e_H$  have already been specified). The use of cycle gluing in the context of toric ideals of graphs and geometric vertex decomposition was introduced in [10, Theorem 3.11]. In the proof of that result, the structure of  $\mathcal{U}(I_{G*H})$  was described, which we demonstrate in the next example.

*Example 5.10.* Consider the graphs  $G$  and  $H$



where  $\mathcal{U}(I_G) = \{ace - bdf\}$  and  $\mathcal{U}(I_H) = \{hj - ig\}$ . We can define a new graph  $G * H$  by gluing along two edges, say  $d$  and  $j$  (which we call  $k$  after the identification).



The list of primitive closed even walks for  $G * H$  becomes

$$\mathcal{U}(G * H) = \langle ace - bfk, hk - ig, aceh - bfgi \rangle$$

where one additional walk is produced by extending  $ace - bdf$  to bypass  $d = k$  and traverse the even cycle instead.  $\square$

The “extended” walks from the example are formed by taking any walk through the edge used for gluing and extending the walk to traverse the even cycle. We make this more precise as follows. Let  $G$  be a finite simple graph with  $e_G \in E(G)$ , and  $C_{2k}$  be some disjoint cycle with  $e_C \in E(C_{2k})$ . We will glue  $G$  to  $C_{2k}$  along  $e_G$  and  $e_C$  to produce a new graph  $G * C_{2k}$ . Suppose that  $\gamma = u_1 e_G^\ell - v_1 \in \mathcal{U}(I_G)$  where  $\ell = 1, 2$ , and  $u_2 e_C - v_2 \in \mathcal{U}(I_{C_{2k}})$  is the binomial defined by the cycle  $C_{2k}$ . Here  $u_1, u_2, v_1, v_2$  are monomials with support not including  $e_G$  or  $e_C$ . If  $e_G$  is glued to  $e_C$  and relabeled as  $e$ , then the extension of  $\gamma$ , denoted by  $\bar{\gamma}$ , is the binomial  $u_1 v_2 - v_1 u_2$  if  $\ell = 1$  and  $u_1 v_2^2 - v_1 u_2^2$  if  $\ell = 2$ . It is not difficult to see that both of these define primitive closed even walks contained in  $\mathcal{U}(I_{G*C_{2k}})$ .

**Proposition 5.11.** *Let  $G$  be a finite simple graph and  $C_{2k}$  be a disjoint cycle of length  $2k$ ,  $k > 1$ . Let  $e_G \in E(G)$  and  $e_C \in E(C_{2k})$ , and form a new graph  $G * C_{2k}$  by identifying  $e_G$  and  $e_C$  as the edge  $e$ . If  $\Gamma \in \mathcal{U}(I_{G*C_{2k}})$ , then either:*

- $\Gamma \in \mathcal{U}(I_G)$
- $\Gamma$  is the binomial defining  $C_{2k}$
- $\Gamma = \bar{\gamma}$  for some  $\gamma \in \mathcal{U}(I_G)$  which passes through  $e_G$

*Proof.* We will abuse notation and write  $G$  and  $C_{2k}$  for the subgraphs of  $G * C_{2k}$  used to construct the gluing. Suppose that some edge of  $C_{2k}$  appears in a primitive closed even walk  $\Gamma$  of  $G * C_{2k}$ . Then  $\Gamma$  is either an even cycle or a primitive walk containing at least two odd cycles [11, Theorem 1.7]. In the first case, we follow the argument of the proof of Theorem 3.11 in [10] to conclude that the even cycle is either  $C_{2k}$  itself, is an even cycle of  $G$ , or has the form  $\bar{\gamma}$  where  $\gamma$  is an even cycle of  $G$  which passes through  $e$ .

If  $\Gamma$  includes at least two odd cycles and is not exclusively in  $G$ , then it must be of the form  $\bar{\gamma}$  for some primitive walk  $\gamma$  of  $G$  which passes through  $e$ . Indeed, if there is a walk  $\Gamma$  that includes the edges of  $C_{2k}$ , then it must pass through the endpoints of  $e$ , so let  $\gamma$  be the walk where the sequence of edges of  $C_{2k} \setminus e$  in  $\Gamma$  are replaced by  $e$ . Let  $u_2e - v_2$  be the binomial of  $C_{2k}$  in  $G * C_{2k}$ , where  $u_2, v_2$  are monomials with support in  $E(C_{2k} \setminus e)$ . There are now two cases to consider:

Case 1: If  $e$  appears exactly once in the walk  $\gamma$  so that  $\gamma = u_1e - v_1$  (where  $u_1, v_1$  are monomials with support in  $E(G \setminus e)$ ), then the proof of Theorem 3.11 in [10] shows that  $\bar{\gamma} = u_1v_2 - v_1u_2$ . To show that it is primitive, observe that any  $u_3 - v_3 \in I_{G*C_{2k}}$  which doesn't pass through  $e$  will either use all variables in  $E(C_{2k} \setminus e)$ , or will not use any of the variables of the cycle. Assume that  $u_3|u_1v_2$  and  $v_3|v_1u_2$ . Then we can write  $u_3 = c_3g_3$  and  $v_3 = c_4g_4$  where  $c_3|v_2$ ,  $c_4|u_2$ ,  $g_3|u_1$  and  $g_4|v_1$ . Then either  $\gamma$  is not primitive because of  $g_3e - g_4$ , or the binomial for  $C_{2k}$  is not primitive because of  $c_4e - c_3$ , which is a contradiction.

Case 2: If  $e$  appears twice in  $\gamma$ , then we can write  $\gamma = u_1e^2 - v_1$ . As above, we can show that  $\bar{\gamma} = u_1v_2^2 - v_1u_2^2$ , by tracing out  $\gamma$  in the following way. Let  $e = \{a, b\}$ . Start at vertex  $a$ , and trace out the portion of  $\gamma$  that starts at  $a$ , stays in  $G \setminus e$ , and returns to vertex  $a$ . Then cross through all edges in  $C_{2k} \setminus e$  to get to vertex  $b$ . Then trace out the portion of  $\gamma$  that starts at vertex  $b$  and stays in  $G \setminus e$ , returning to vertex  $b$ . Finally, cross the edges of  $C_{2k} \setminus e$  again to get back to  $a$ . Note that the intermediate vertices in a primitive walk can only be visited twice (since every cut vertex only belongs to two blocks by [25, Theorem 2.2]). We can show that  $\bar{\gamma}$  is primitive using a similar argument as above.  $\square$

**5.2.4. Main Theorem.** Using the previously mentioned operations, we are now ready to show that arbitrarily large universal Gröbner bases can be produced in polynomial time. Starting with a small graph where  $\mathcal{U}(I_G)$  can be computed directly, and through random applications of each operation, a sufficiently large (and asymmetric) graph  $H$  with computable  $\mathcal{U}(I_H)$  can be constructed to secure the system  $\mathcal{P}$ . By asymmetric, we mean that repetitive iterations of the same operation should be avoided (such as simply gluing on a 4-cycle successively).

**Theorem 5.12.** *Let  $G$  be a finite simple graph such that  $\mathcal{U}(I_G)$  is known. Then by using one of the following operations*

- (1) *Gluing a disjoint bipartite graph to  $G$  along some  $v \in V(G)$  (as in Section 5.2.1)*
- (2) *Gluing a disjoint even cycle to  $G$  along some  $e \in E(G)$  (as in Section 5.2.3)*
- (3) *Star subdividing along a degree two vertex of  $G$  (as in Proposition 5.9)*
- (4) *Performing a star contraction along a vertex  $v$  such that  $G_v$  is a simple graph (as in Definition 5.6)*

we can produce a graph  $G'$  such that the number of operations to compute  $\mathcal{U}(I_{G'})$  is linear in  $N = |\mathcal{U}(I_G)|$ . Furthermore, by using any combination of  $k$  operations (1) to (4), and choosing  $k$  sufficiently large, we can produce a graph  $H$  such that  $|\mathcal{U}(I_H)|$  is as large as desired, with computational complexity  $O(N^k)$ .

*Proof.* For the first three operations, the explicit method in which  $\mathcal{U}(I_{G'})$  is obtained from the  $\mathcal{U}(I_G)$  is described in Propositions 5.5 and 5.11 and also Proposition 5.9. Here we would produce a larger list  $\mathcal{U}(I_{G'})$  given  $\mathcal{U}(I_G)$  for the first two operations, while the third operation would maintain the cardinality of the sets but increase the degree.

The fourth operation maintains the same cardinality, although the new list of closed even walks may not all be primitive (this is still okay in the context of Gröbner bases since we are simply adding generators which may be unnecessary for the Gröbner computation).

If  $|\mathcal{U}(I_G)| = N$ , then the first operation simply merges two sets, which is done in linear time. The second operation requires at most  $N + 1$  new elements to be added to the list of primitive closed even walks (one instance of  $C_{2k}$ , and at most one  $\bar{\gamma}$  computation for each  $\gamma \in \mathcal{U}(I_G)$ ), so has complexity  $O(N)$ . The third operation increases the degree of at most  $N$  walks, which again has complexity  $O(N)$ . Finally, the star contraction requires a substitution of at most  $N$  polynomials, which is again  $O(N)$ . Iterating these operations would result in the product of the complexity bounds, proving the  $O(N^k)$  bound.  $\square$

## 6. CONCLUSIONS AND ALTERNATE PROTOCOLS

We conclude with some brief observations about the use of universal Gröbner bases for securing data. The protocol  $\mathcal{P}$  is just one possible vision of how universal Gröbner bases could be used in cryptography. We hope that this article will spur interest in other possible uses of the construction of  $\mathcal{U}_I$  proposed in Section 5, especially by those better versed with the practical issues concerning cryptographic implementations.

We offer several remarks on alternate approaches:

- Choosing  $\mathbb{K}$  to be a finite field would increase the difficulty of the Gröbner computations and would likely improve the security of  $\mathcal{P}$ .
- When  $\tau = \emptyset$ , one shortfall of the system is the amount of time that Party  $A$  needs to decrypt the message. This may make the  $\tau = \emptyset$  protocol useful for blockchain applications where rewards are used to incentivize the completion of brute-force verifications.
- Symmetric Diffie-Hellman type initializations of  $\mathcal{P}$  may be possible by Party  $A$  providing a common monomial ideal, followed by  $A$  and  $B$  each choosing their own initial ideals and combining it with this common ideal. Sending such “combined” ideals (using unions, intersections, etc.) may reveal too much information about degree bounds of generators in the choices of  $A$  and  $B$ . Masking the choices using hash functions would yield a similar security to the  $\tau \neq \emptyset$  case.
- Party  $B$  only sending partial information about  $K_B$  would reduce the number of keys that  $A$  needs to check, offering a middle ground between the  $\tau = \emptyset$  and  $\tau \neq \emptyset$  initializations of  $\mathcal{P}$ . Choosing  $\mathcal{E}$  to be a symmetric encryption algorithm would also

reduce  $A$ 's decryption time, since such schemes are usually less computationally intensive compared to their asymmetric counterparts.

As a final note, universal Gröbner bases for toric ideals have been better studied and are generally faster to compute. Using some  $\mathcal{U}_I$  associated to the toric ideal of a graph may introduce a weakness to the system if chosen poorly. Generally, the complexity of computing universal Gröbner bases for a toric ideal of a graph still remains exponential in the number of edges [23, Section 4].

An alternate approach is to build a large enough  $\mathcal{U}_I$  using the techniques in Section 5, and then add one (carefully selected) non-toric generator to the list, followed by a recomputation of a universal Gröbner basis for the new list. If an attacker does not know the graph  $G$ , then the toric universal Gröbner basis algorithms from [9, 24] would be difficult to implement. Furthermore, even if  $G$  were known, choosing it large enough would make those computations difficult.

## REFERENCES

- [1] M. Bardet, J.C. Faugére and B. Salvy. On the complexity of the  $F_5$  Gröbner basis algorithm. *Journal of Symbolic Computation* 70, (2015), 49-70.
- [2] B. Barkee, D.C Can, J. Ecks, T. Moriarty, and R.F Ree. Why you cannot even hope to use Gröbner Bases in Public Key Cryptography. *J. Symb. Comp.* 18, (1994) 497–501.
- [3] B. Barkee, M. Ceria, T. Moriarty and A. Visconti. Why you cannot even hope to use Gröbner bases in cryptography: an eternal golden braid of failures. *Applicable Algebra in Engineering, Communication and Computing* 31, (2020), 235-252.
- [4] J. Bell-Colley, Hamiltonian Cycles and Primitive Closed Even Walks of Graphs, *Virginia State University Master's Thesis*, ProQuest, (2023).
- [5] D. Bernstein. *Introduction to post-quantum cryptography*. Post-Quantum Cryptography, Springer, (2009).
- [6] M. Caboara, F. Caruso, and C. Traverso. Gröbner bases for public key cryptography *Conference Proceedings: Symbolic and Algebraic Computation*, ISSAC 2008, Linz/Hagenberg, Austria, (2008).
- [7] A. Couvreur, R. Mora, and J.P. Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, (2023), 3-38.
- [8] D.A. Cox, J. Little, and D. O'shea. *Ideals, Varieties, and Algorithms*. Vol. 4. New York, Springer, (2015).
- [9] D. Cox, J. Little, and H. Schenck. *Toric Varieties*. Graduate Studies in Mathematics Vol. 124, American Mathematical Society (2011).
- [10] M. Cummings and S. Da Silva and J. Rajchgot and A. Van Tuyl. Geometric vertex decomposition and liaison for toric ideals of graphs. *Algebraic Combinatorics* 6(4), (2023), 965–997.
- [11] S. Da Silva, E. Naguit and J. Rajchgot. A note on toric ideals of graphs and Knutson-Miller-Yong decompositions. *arXiv: 2502.08069*, (2025).
- [12] J. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. Effective lattice point counting in rational convex polytopes. *J. of Symbolic Computation* 38, (2004), 1273–1302.
- [13] J. De Loera, B. Sturmfels and R. Thomas. Gröbner bases and triangulations of the second hypersimplex. *Combinatorica* 15, (1995), 409–424.
- [14] T. W. Dubé. The structure of polynomials ideals and Gröbner bases. *SIAM Journal on Computing*, 19(4), (1990), 750–773.
- [15] D. Eisenbud, *Commutative algebra with a view towards algebraic geometry*, Springer Graduate Texts, 150, (1995).

- [16] K. Fukuda, A. Jensen and R. Thomas. Computing gröbner fans. *Mathematics of Computation* 76(260), (2007), 2189-2212.
- [17] F. Galetto, J. Hofscheier, G. Keiper, C. Kohne, M.E.U. Paczka, A. Van Tuyl, Betti numbers of toric ideals of graphs: a case study. *Journal of Algebra and its Applications* 18, (2019).
- [18] A.N. Jensen. *Computing Gröbner Fans and Tropical Varieties in Gfan*. In: M. Stillman, J. Verschelde, and N. Takayama. (eds) Software for Algebraic Geometry. The IMA Volumes in Mathematics and its Applications Vol. 148, Springer, (2008).
- [19] A.N. Jensen. Gfan, a software system for Grobner fans. *Macaulay2 Software Package*, (2006). (<http://www.math.tu-berlin.de/jensen/software/gfan/gfan.html>)
- [20] Z.J. Lou, R. Liu, A. Mehta, and M.L. Ali. Demystifying the RSA algorithm: an intuitive introduction for novices in cybersecurity, *Journal of Computing Sciences in Colleges* 40(3), (2024), 85-99.
- [21] A. Nachman. Exploring graph-theoretic properties using geometric vertex decomposition. *Virginia State University Master's Thesis*, ProQuest, (2023).
- [22] B. Stackpole. Quantum Computing: What Leaders Need to Know Now. *MIT Sloan*, 11 Jan. (2024).
- [23] Y. Stamatiou and C. Tatakis. An algorithm for computing the universal Gröbner Basis of graph ideals. *International Journal of Computer Mathematics*, (2019).
- [24] B. Sturmfels. *Gröbner bases and convex polytopes*. American Mathematical Soc. Vol. 8., (1996).
- [25] C. Tatakis, A. Thoma. On the universal Gröbner bases of toric ideals of graphs. *Journal of Combinatorial Theory*, Ser. A 118, (2011), 1540–1548.
- [26] T. Theobald. On the frontiers of polynomial computations in tropical geometry. *J. Symbolic Comput.* 41, (2006), pp. 1360-1375.
- [27] R.H. Villarreal. *Monomial Algebras*. Second Edition. Monographs and Research Notes in Mathematics, CRC Press, Boca Raton, FL, (2015).
- [28] D. Willsch, P. Hanussek, G. Hoever, M. Willsch, F. Jin, H. De Raedt, and K. Michielsen. The State of Factoring on Quantum Computers. *arXiv:2410.14397*, (2024).
- [29] D. Willsch, M. Willsch, F. Jin, H. De Raedt, and K. Michielsen. Large-scale simulation of Shor's Quantum Factoring Algorithm. *Mathematics* 11, no. 19, (2023).

(Sergio Da Silva) DEPT. OF MATHEMATICS AND ECONOMICS, VIRGINIA STATE UNIVERSITY, 1 HAYDEN DRIVE, PETERSBURG, VIRGINIA 23806, USA

*Email address:* `sdaasilva@vsu.edu, smd322@cornell.edu`

(Aniya Stewart) DEPT. OF MATHEMATICS AND ECONOMICS, VIRGINIA STATE UNIVERSITY, 1 HAYDEN DRIVE, PETERSBURG, VIRGINIA 23806, USA

*Email address:* `aste9039@students.vsu.edu, stewart.aniya@gmail.com`