Cluster Algebra Cryptography

Yuguang (Roger) Bai

January 2021

1 Introduction

Asymmetric encryption algorithms are based on the idea of a "one-way function", which is a function that is easy to compute, but difficult to reverse. For example, it is easy to multiply two numbers together but hard to factor a given number. This project looks to create use such a one-way function based on cluster algebras and see how useful it is as a cryptographic algorithm.

2 Background

2.1 Cluster Algebras

Let $n \in \mathbb{Z}_{\geq 1}$ and $\mathcal{F}$ be the field of rational functions over $\mathbb{Q}$ in n variables. A **seed** of $\mathcal{F}$ is a pair $\Sigma = (\mathbf{x}, Q)$ where $\mathbf{x} = \{x_1, ..., x_n\}$ generates $\mathcal{F}$ freely and Q is a finite quiver with n vertices. We assume Q has no loops and no 2-cycles. Instead of Q, we will often use a matrix $B = (b_{ij})$ to display the same information as Q where

$$b_{ij} = \#\{\text{arrows } i \to j\} - \#\{\text{arrows } j \to i\}.$$

The quiver Q and matrix B are called the **exchange quiver/matrix**.

For $1 \leq k \leq n$, we define a new seed $\mu_k(\Sigma) = (\mu_k(\Sigma), \mu_k(Q))$ called the **mutation of Σ in direction** k. We have $\mu_k(\mathbf{x}) = (x_1, ..., x_k^*, ..., x_n)$ where

$$x_k^* = \frac{\prod_{i \to k} x_i + \prod_{k \to i} x_i}{x_k} = \frac{\prod_{b_{ik}>0} x_i + \prod_{b_{ik<0}} x_i}{x_k}.$$

These mutation equations are called **exchange relations**. The quiver $\mu_k(Q)$ is constructed from Q by

1. adding a new arrow $i \to j$ for each pair of arrows $i \to k$ and $k \to j$;
2. reversing the orientation of every arrow with target or source k;
3. and erasing every pair of opposite arrows.

In terms of matrices, we obtain a new matrix $\mu_k(B) = (b'_{ij})$ where

$$b'_{ij} = \begin{cases} -b_{ij} & \text{if } i = k \text{ or } j = k \\ b_{ij} + \text{sgn}(b_{ik}) \max\{0, b_{ik}b_{kj}\} & \text{otherwise} \end{cases}$$

Note that $\mu_k(\mu_k(\Sigma)) = \Sigma$.

If we have a seed $\Sigma = (\mathbf{x}, Q)$, then $\mathbf{x} = \{x_1, ..., x_n\}$ is called a **cluster** and its elements are called **cluster variables**. The **cluster monomials** are the elements that are a product of cluster variables in a fixed cluster. The **cluster algebra** is defined as the subalgebra generated by all cluster variables.

It is proved in [FZ03] that a cluster algebra has finitely many seeds if and only if there exists a seed whose quiver is of Dynkin type.

We see that mutation a seed is quite simple, either graphically using a quiver or algebraically with matrices, but given two seeds, it is in general difficult to see if they are mutation equivalent. This will be the basis of how we will create a cryptographic algorithm.

2.2 Asymmetric Encryption

A lot of popular asymmetric encryption algorithms such as RSA and ECC are based on the idea of a "one-way function", but these use commutative properties, such as the product of numbers in RSA and an abelian group in ECC. However, the mutation of seeds is not commutative, that is in general, $\mu_i\mu_j\Sigma \neq \mu_j\mu_i\Sigma$, even up to reordering.

The following is a key exchange protocol using a non-abelian group G created by Anshel-Anshel-Goldfeld (for $a, b \in G$, we denote $b^a := a^{-1}ba$):

1. Select and publish $a_1, ..., a_k, b_1, ..., b_m \in G$.
2. A picks a secret word x in $a_1, ..., a_k$. We view $x = x(a_1, ..., a_k)$ as a function in the a_i's.
3. A sends $b_1^x, ..., b_m^x$ to B.
4. B picks a secret word y in $b_1, ..., b_m$. We view $y = y(b_1, ..., b_m)$ as a function in the b_i's.
5. B sends $a_1^y, a_2^y, ..., a_k^y$ to A.
6. A computes $x(a_1^y, ..., a_k^y) = y^{-1}xy$, and then $x^{-1}y^{-1}xy$.
7. B computes $y(b_1^x, ..., b_m^x) = x^{-1}yx$, and then $y^{-1}x^{-1}yx = (x^{-1}y^{-1}xy)^{-1}$.
8. A and B use the shared key $K = x^{-1}y^{-1}xy$.

Fix a seed Σ of n variables. We can view mutation operations on Σ as a group $G = \{\mu_i : 1 \leq i \leq n\} \cup \{\mu_0\}$ where μ_0 is the identity element representing no mutation. We will denote $\mu_i\mu_j$ to be mutation Σ first in direction j and then in direction i. Since $\mu_i\mu_i\Sigma = \Sigma$, then $\mu_i^{-1} = \mu_i$ so G is a group. Instead of using the entire seed Σ, we can just use the mutation quiver or matrix. If the mutation quiver is chosen so that it is not mutation equivalent to a Dynkin quiver, then we are guaranteed G to be infinite.

References

[FZ03] Sergey Fomin and Andrei Zelevinsky. Cluster algebras II: Finite type classification. Invent. Math., 154:63–121, 2003. 1