Cryptographic hash functions from Expander graphs

Venkatramani Rajgopal

Department of Mathematics
University of Applied Sciences, Mittweida

28 June 2016

Outline

Abstract of the topic

- The Goal of this topic is to construct provable *collision resistant hash functions* from expander graphs.

Abstract of the topic

- The Goal of this topic is to construct provable *collision resistant hash functions* from expander graphs.
- Two specific families of optimal expander graphs are investigated for hash function construction: *families of Ramanujan graphs constructed by Lubotzky-Phillips-Sarnak **(LPS)** and Pizer.*

Abstract of the topic

- The Goal of this topic is to construct provable *collision resistant hash functions* from expander graphs.
- Two specific families of optimal expander graphs are investigated for hash function construction: *families of Ramanujan graphs constructed by Lubotzky-Phillips-Sarnak* ***(LPS)*** *and Pizer.*
- When hash function is constructed from one of Pizer's Ramanujan graphs, then collision resistant follows from hardness of computing isogenies between supersingular elliptic curves.

Abstract of the topic

- The Goal of this topic is to construct provable *collision resistant hash functions* from expander graphs.
- Two specific families of optimal expander graphs are investigated for hash function construction: *families of Ramanujan graphs constructed by Lubotzky-Phillips-Sarnak* ***(LPS)*** *and Pizer.*
- When hash function is constructed from one of Pizer's Ramanujan graphs, then collision resistant follows from hardness of computing isogenies between supersingular elliptic curves.
- Estimating the *cost per bit* to compute these hash functions.

Introduction

The input to the hash is used as directions for walking around the graph and the ending vertex is the output of the hash function.

Introduction

The input to the hash is used as directions for walking around the graph and the ending vertex is the output of the hash function.

- The construction can be applied to any expander graph.
- Ramanujan graphs are *optimal expander graphs* and thus have excellent mixing properties.

Introduction

The input to the hash is used as directions for walking around the graph and the ending vertex is the output of the hash function.

- The construction can be applied to any expander graph.
- Ramanujan graphs are *optimal expander graphs* and thus have excellent mixing properties.
  From a practical viewpoint, these graphs resolve an extremal problem in *communication network theory*.

Introduction

When constructing hash function from Ramanujan graph of elliptic curves over $\mathbb{F}_{p^2}$ with l-isogenies, l a prime different from p, computing collusions is as hard as computing isogenies between supersingular elliptic curves.

Introduction

When constructing hash function from Ramanujan graph of elliptic curves over $\mathbb{F}_{p^2}$ with l-isogenies, l a prime different from p, computing collusions is as hard as computing isogenies between supersingular elliptic curves.
This is belived to be a difficult problem and the best know algorithm to solve this is *square-root time.*

Introduction

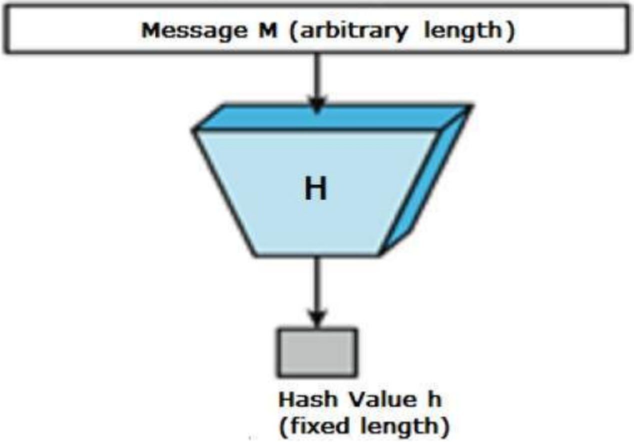
When constructing hash function from Ramanujan graph of elliptic curves over $\mathbb{F}_{p^2}$ with l-isogenies, l a prime different from p, computing collusions is as hard as computing isogenies between supersingular elliptic curves.
This is belived to be a difficult problem and the best know algorithm to solve this is *square-root time*.

Proposal

So it was proposed to set p to be a 256-bit prime, inorder to obtain 128bit of security from the resulting hash function.

Preliminaries

Hash Functions



Preliminaries

Hash Functions

Hash Functions

- A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h = H(M)$. In short, it maps bit strings of some finite length to bit strings of some fixed length.

Hash Functions

- A hash function H accepts a variable-length block of data M as input and produces a fixed size hash value $h = H(M)$. In short, it maps bit strings of some finite length to bit strings of some fixed length.
- We are concerned with *Unkeyed hash functions* which are collusion resistant. *Unkeyed hash functions* do not require a secret key to compute the output.
- **Collision resistance:** A hash function H is collision resistant if it is hard to find two inputs that hash to the same output; that is, two inputs x and y such that $H(x) = H(y)$, and $x \neq y$.

Sparse Graph

- A graph with only a few edges is a **sparse graph**.

Preliminaries

Sparse graphs

Sparse Graph

- A graph with only a few edges is a **sparse graph**.
- Graphs are sparse when only a small fraction of the possible number of vertex pairs actually have edges defined between them.

Preliminaries

Sparse graphs

Sparse Graph

- A graph with only a few edges is a **sparse graph**.
- Graphs are sparse when only a small fraction of the possible number of vertex pairs actually have edges defined between them.
- Graphs are usually sparse due to application specific constraints. Eg. Road Networks must be sparse because of road junctions.

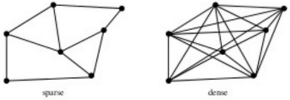


Figure: Sparse vs Dense graph

Expander Graphs

- A *sparse graph* with strong connectivity properties.

Expander Graphs

- A *sparse graph* with strong connectivity properties.
- An Expander is a finite, undirected multigraph in which every subset of the vertices that is not "too large" has a large boundary.

Expander Graphs

- A *sparse graph* with strong connectivity properties.
- An Expander is a finite, undirected multigraph in which every subset of the vertices that is not ”too large” has a large boundary.
- A disconnected graph is not an expander, since the boundary of every connected component is empty. Every connected graph is an expander (but with different parameters).

Preliminaries

Expander Graphs

Expander Graphs

- A *sparse graph* with strong connectivity properties.
- An Expander is a finite, undirected multigraph in which every subset of the vertices that is not "too large" has a large boundary.
- A disconnected graph is not an expander, since the boundary of every connected component is empty. Every connected graph is an expander (but with different parameters).
- A graph is a good expander if it has low degree and high expansion parameter.

So, what is not a good expander ?

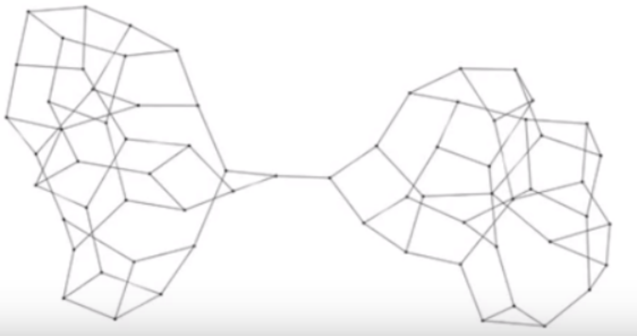
Preliminaries

Expander Graphs

So, what is not a good expander ?

The two graphs below, when taken separately are good expanders, but taken when together has a small ratio of expansion and hence a poor expander.

Figure: A graph X with poor expansion.



Preliminaries

Expander graphs

Definition

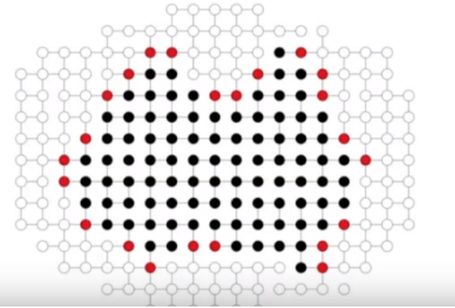
Let $G = (V, E)$ be a graph with vertex set V and edge set E. In undirected graphs, a graph is k regular if the vertex has k edges coming out of it.

Preliminaries

Expander graphs

Definition

Let $G = (V, E)$ be a graph with vertex set V and edge set E. In undirected graphs, a graph is k regular if the vertex has k edges coming out of it.
An expander graph with N vertices has *expansion constant* $c > 0$ if for any subset $U \subset V$ of size $|U| \leq \frac{N}{2}$, the boundary $\tau(U)$ of U has a size $|\tau(U)| \geq c|U|$.



Expansion property

Expansion property

- If a graph has multiple edges, we call it a *multigraph*.

Expansion property

- If a graph has multiple edges, we call it a *multigraph*.
- When two vertices u and v are endpoints of an edge, we say they are adjacent and sometimes write $u \sim v$ to indicate this.

Expansion property

- If a graph has multiple edges, we call it a *multigraph*.
- When two vertices u and v are endpoints of an edge, we say they are adjacent and sometimes write $u \sim v$ to indicate this.
- To any graph, we may associate the *adjacency matrix* A which is an $n \times n$ matrix (where $n = |G|$) with rows and columns indexed by the elements of the vertex set.

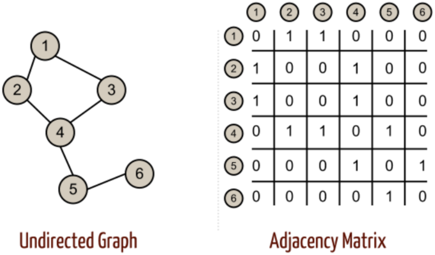
Expansion property

- If a graph has multiple edges, we call it a *multigraph*.
- When two vertices u and v are endpoints of an edge, we say they are adjacent and sometimes write $u \sim v$ to indicate this.
- To any graph, we may associate the *adjacency matrix* A which is an $n \times n$ matrix (where $n = |G|$) with rows and columns indexed by the elements of the vertex set.
- The (x, y)-th entry is the number of edges connecting x and y.

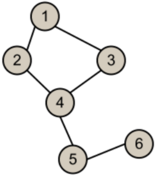
Preliminaries

Expansion property and Ramanujan Graphs

Example of Adjacency matrix.



Example of Adjacency matrix.



Undirected Graph

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	0	0
3	1	0	0	1	0	0
4	0	1	1	0	1	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

Adjacency Matrix

Since our graphs are undirected, the matrix A is symmetric. Consequently, all of its eigenvalues are real.

- So we have for a connected graph, G, the largest eigenvalue is k. We order the eigenvalues as $k > \mu_1 \geq \mu_2 > ... \geq \mu_{N-1}$. Then the expansion constant c can be expressed in terms of the eigenvalues as:

$$c \geq \frac{2(k - \mu_1)}{3k - 2\mu_1}$$

Preliminaries

Expansion property and Ramanujan Graphs

- So we have for a connected graph, G, the largest eigenvalue is k. We order the eigenvalues as $k > \mu_1 \geq \mu_2 > ... \geq \mu_{N-1}$. Then the expansion constant c can be expressed in terms of the eigenvalues as:

$$c \geq \frac{2(k - \mu_1)}{3k - 2\mu_1}$$

- Therefore smaller the eigenvalue μ_1, better the expansion constant.

With this we can now state the theorem of *Alon-Boppana*, leading to the definition of *Ramanujan graph*.

For an infinite family X_m of connected, k-regular graph, with the number of vertices tending to infinity, satisfies the inequality,
$$\lim inf \mu_1(X_m) \geq 2\sqrt{k-1}$$

With this we can now state the theorem of *Alon-Boppana*, leading to the definition of *Ramanujan graph*.

For an infinite family X_m of connected, k-regular graph, with the number of vertices tending to infinity, satisfies the inequality,

$$\lim inf \mu_1(X_m) \geq 2\sqrt{k-1}$$

Ramanujan graph

A *Ramanujan multigraph* is a k-regular graph satisfying the inequality:

$$\mu_1 \leq 2\sqrt{k-1}$$

Construction of Hash Functions

The input to the hash is used as directions for walking around the graph(*without backtracking*). Output is the ending vertex.

Construction of Hash Functions

The input to the hash is used as directions for walking around the graph(*without backtracking*). Output is the ending vertex.
The walk starts at a fixed vertex in the given graph.

Construction of Hash Functions

The input to the hash is used as directions for walking around the graph(*without backtracking*). Output is the ending vertex.
The walk starts at a fixed vertex in the given graph.

Working

- Execute a walk on a k-regular expander graph by breaking the input to the hash function into chunks of size e, so that $2^e = k - 1$.

Construction of Hash Functions

The input to the hash is used as directions for walking around the graph(*without backtracking*). Output is the ending vertex.
The walk starts at a fixed vertex in the given graph.

Working

- Execute a walk on a k-regular expander graph by breaking the input to the hash function into chunks of size e, so that $2^e = k - 1$.
- At each step in the walk, the choice of the edge to follow is determined by the next e bits of the input.

Construction of Hash Functions

The input to the hash is used as directions for walking around the graph(*without backtracking*). Output is the ending vertex.
The walk starts at a fixed vertex in the given graph.

Working

- Execute a walk on a k-regular expander graph by breaking the input to the hash function into chunks of size e, so that $2^e = k - 1$.
- At each step in the walk, the choice of the edge to follow is determined by the next e bits of the input.
- Since backtracking is not allowed, only $k - 1$ choices for the next edge are allowed at each step.

Construction of Hash Functions

Random walks on expander graphs

A random walk on an expander graph mixes very fast, so the output of the hash function will be uniform provided the input was uniformly random.

Construction of Hash Functions

Random walks on expander graphs

A random walk on an expander graph mixes very fast, so the output of the hash function will be uniform provided the input was uniformly random.

The output of a random walk on an expander graph with N vertices tends to the uniform distribution after $O(log(N))$ steps.

Pizer's Ramanujan graphs

We define the graph $G(p, l)$ with vertex set V, a set of supersingular elliptic curves over the finite field $\mathbb{F}_{p^2}$. Here p and l are two distinct prime numbers.

Pizer's Ramanujan graphs

We define the graph $G(p, l)$ with vertex set V, a set of supersingular elliptic curves over the finite field $\mathbb{F}_{p^2}$. Here p and l are two distinct prime numbers.

Elliptic curve

An *elliptic curve* is the set of solutions to an equation of the form $y^2 = x^3 + ax + b$ where a and b are rational numbers s.t $4a^3 + 27b^2 \neq 0$.

Pizer's Ramanujan graphs

An elliptic curve can take the below forms.

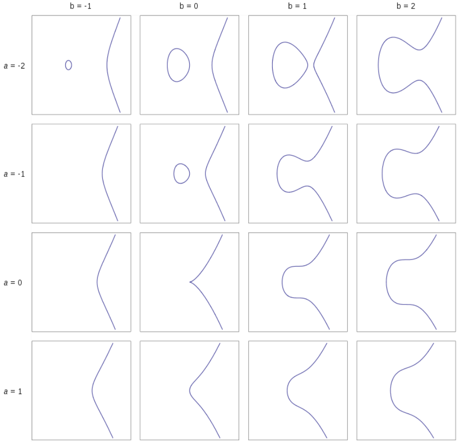


Figure: Elliptic Curve

Pizer's Ramanujan graphs

Supersingular Elliptic curves

An elliptic curve over a finite field of characteristic p is *supersingular* if it has no p-torsion over any extension field.
Elliptic curves which are not supersingular are called *ordinary*.

Pizer's Ramanujan graphs

From the results given by Prof. Silverman, Joseph; "The Arithmetic of Elliptic Curves" we have,

Pizer's Ramanujan graphs

From the results given by Prof. Silverman, Joseph; "The Arithmetic of Elliptic Curves" we have,

- The vertices are labelled with $j-$invariants of the curve. The number of vertices of $G(p, l)$ is $\lfloor \frac{p}{12} \rfloor + \epsilon$, where $\epsilon \in [0,1,2]$
- In this construction of the hash function, a further restriction is imposed on $p : p \equiv 1 \pmod{12}$.

Pizer's Ramanujan graphs

From the results given by Prof. Silverman, Joseph; "The Arithmetic of Elliptic Curves" we have,

- The vertices are labelled with $j-$invariants of the curve. The number of vertices of $G(p, l)$ is $\lfloor \frac{p}{12} \rfloor + \epsilon$, where $\epsilon \in [0,1,2]$
- In this construction of the hash function, a further restriction is imposed on $p : p \equiv 1 \pmod{12}$.
- Since there are approximately $p/12$ distinct $j-$invariants, we choose a linear congruential function to map $j-$invariants from $\mathbb{F}_{p^2}$ into $\mathbb{F}_p$ for the output of the hash.

Pizer's Ramanujan graphs

From the results given by Prof. Silverman, Joseph; "The Arithmetic of Elliptic Curves" we have,

- The vertices are labelled with $j-$invariants of the curve. The number of vertices of $G(p, l)$ is $\lfloor \frac{p}{12} \rfloor + \epsilon$, where $\epsilon \in [0,1,2]$
- In this construction of the hash function, a further restriction is imposed on $p : p \equiv 1 \pmod{12}$.
- Since there are approximately $p/12$ distinct $j-$invariants, we choose a linear congruential function to map $j-$invariants from $\mathbb{F}_{p^2}$ into $\mathbb{F}_p$ for the output of the hash.
- Thus the output of the hash will be $log(p)$ bits. Typically p is of cryptographic size $\approx 2^{256}$.

Pizer's Ramanujan graphs

Hash Computation

- The input is first transformed to the base-l number.

Pizer's Ramanujan graphs

Hash Computation

- The input is first transformed to the base-l number.
- The walk in the $l+1$ regular graph is started from a fixed vertex E_0. From each vertex there are l choices to go to the next vertex since backtracking is not allowed.

Pizer's Ramanujan graphs

Hash Computation

- The input is first transformed to the base-l number.
- The walk in the $l+1$ regular graph is started from a fixed vertex E_0. From each vertex there are l choices to go to the next vertex since backtracking is not allowed.
- The edges corresponding to $l+1$ subgroups at each vertex needs to be ordered in a consistent way. At each step we need to compute these $l+1$ subgroups and follow the edge corresponding to the input digest.

Pizer's Ramanujan graphs

Hash Computation

- The input is first transformed to the base-l number.
- The walk in the $l+1$ regular graph is started from a fixed vertex E_0. From each vertex there are l choices to go to the next vertex since backtracking is not allowed.
- The edges corresponding to $l+1$ subgroups at each vertex needs to be ordered in a consistent way. At each step we need to compute these $l+1$ subgroups and follow the edge corresponding to the input digest.
- The output is the j invariant of the curve corresponding to the last vertex reached.

Isogenies

Isogenies

Study of the maps between curves.

Isogenies

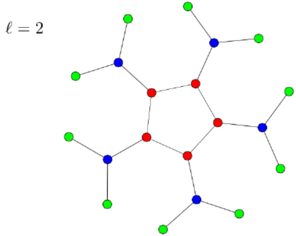
Study of the maps between curves.

- Since an elliptic curve has a distinguished zero point, it is natural to single out the maps that respect this property.
- An isogeny of elliptic curves is a surjective morphism (i.e. a *map* that preserves addition and the point at infinity) between elliptic curves.

Isogenies

Study of the maps between curves.

- Since an elliptic curve has a distinguished zero point, it is natural to single out the maps that respect this property.
- An isogeny of elliptic curves is a surjective morphism (i.e. a *map* that preserves addition and the point at infinity) between elliptic curves.
- Isogeny graphs.
  The edges of the graph are isogenies of degree l, where l is a prime different from p.



Collision resistance

Collision resistance

- There is a collision in the hash function when two distinct inputs hash to the same output.

Collision resistance

- There is a collision in the hash function when two distinct inputs hash to the same output.
- Finding a collision in this hash function is equivalent to finding two isogenies of the same l-power degree between a pair of supersingular elliptic curves.

Collision resistance

- There is a collision in the hash function when two distinct inputs hash to the same output.
- Finding a collision in this hash function is equivalent to finding two isogenies of the same l-power degree between a pair of supersingular elliptic curves.
- If the graph $G(p, l)$ does not have small cycles then this problem is very hard to compute. Which is why we had introduced restriction on congruence class of prime p. *[* $p \equiv 1 \ (mod\ 12)$ *]*
- Distinct inputs result in distinct paths in the graph and distinct paths lead to distinct isogenies.

Findings

- To compute the hash function from Pizer's graph when $l = 2$ requires roughly $2log(p)$ field multiplications per bit of input to the hash function.

Findings

- To compute the hash function from Pizer's graph when $l = 2$ requires roughly $2log(p)$ field multiplications per bit of input to the hash function.
- Advantage in this construction is that the output of hash function is $log(2)$bits and the efficiency may be improved with optimization.

Findings

- To compute the hash function from Pizer's graph when $l = 2$ requires roughly $2log(p)$ field multiplications per bit of input to the hash function.
- Advantage in this construction is that the output of hash function is $log(2)$bits and the efficiency may be improved with optimization.
- Also, hash functions from LPS graphs where more efficient to compute than from Pizer's graph.

Findings

- To compute the hash function from Pizer's graph when $l = 2$ requires roughly $2log(p)$ field multiplications per bit of input to the hash function.
- Advantage in this construction is that the output of hash function is $log(2)$bits and the efficiency may be improved with optimization.
- Also, hash functions from LPS graphs where more efficient to compute than from Pizer's graph.
- This Hash function for primes of varying size where implemented.

References I

Denis X. Charles, Eyal Z. Goren and Kristin E Lauter
Cryptographic hash functions from expander graphs.

M.Ram Murthy
Ramanujan Graphs
Department of Mathematics, Queen's University, Kingston, Ontario,Canada

Vaibhav Rajan
Cryptographic hash functions from expander graphs.
Ecole Polytechnique Federale de Lausanne, Switzerland

Joseph H. Silverman
The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics
Springer-Verlag, 1986.

References II

Wikipedia
Expander Graphs.
https://en.wikipedia.org/wiki/Expander_graph

Wikipedia
Elliptic Curve.
https://en.wikipedia.org/wiki/Elliptic_curve