Cryptographic hash functions from expander graphs

Denis Charles, Microsoft Research
Eyal Goren, McGill University
Kristin Lauter, Microsoft Research

Presented by
Josh Benaloh, Microsoft Research

NIST Hash Function Workshop
November 1, 2005

A new(?) hash paradigm

- Input string is divided into blocks which are used as the “directions” for how to walk around an expander graph
- Output is the final vertex of the hash
- Random walks on expander graphs mix rapidly: log(n) steps to a random vertex
- To find a collision: find two distinct walks of the same length which end at same vertex

Example: graph of supersingular elliptic curves modulo p

- Vertices: supersingular elliptic curves mod p
- Edges: degree 2 isogenies between them
- Graph is Ramanujan (optimal expander)
- Out degree=3, input written in ternary
- Finding collisions reduces to finding pairs of distinct isogenies between elliptic curves
- $O(\sqrt{p})$ birthday attack is best known
- log(p) modular multiplies per edge traversal

Other graphs

- Vary the isogeny degree (e.g. degree 3)
- Ordinary elliptic curves
  - Same efficiency as supersingular graph
  - Finding isogenies: $p^{3/2}\log(p)$ [Galbraith]
  - Isogeny graph with fixed degree not connected
- Lubotzky-Phillips-Sarnak Cayley graph
  - random walk is efficient to implement
  - Optimal expander graph (Ramanujan)
  - Different problem for finding collisions

More information

- Details on steps of walk: *Computing Modular Polynomials*, LMS J. Computational Mathematics, (2005), D. Charles, K. Lauter.
- *Cryptographic hash functions from expander graphs*, D. Charles, E. Goren, K. Lauter, in preparation.
- Complexity of finding collisions: *Constructing isogenies between elliptic curves over finite fields*, LMS J. Computational Mathematics, (1999), S. Galbraith.