Cryptographic hash functions from expander graphs

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

2nd NIST Hash Function Workshop
August 24-25, 2006

Status quo

“ WELL WE KNOW WHERE WE'RE GOIN‘
BUT WE DON'T KNOW WHERE WE'VE BEEN

AND WE KNOW WHAT WE'RE KNOWIN'
GIVE US TIME TO WORK IT OUT”

—Talking Heads,
Road to Nowhere

Related work: (provable hashes)

- VSH [Contini, Lenstra, Steinfeld, 2005]
- ECDLP-based [?]
- Zemor-Tillich `94, Hashing with $SL_2(Z)$
- Joye-Quisquater, `97,
- Quisquater 2004, Liardet 2004
- Goldreich, 2000, One-way functions from LPS graphs

Construction of the hash function:

- k-regular graph G
- Each vertex in the graph has a label

Input: a bit string

- Bit string is divided into blocks
- Each block used to determine which edge to follow for the next step in the graph
- No backtracking allowed!

Output: label of the final vertex of the walk

Simple idea

- Random walks on expander graphs are a good source of pseudo-randomness
- Are there graphs such that finding collisions is hard? (i.e. finding distinct paths between vertices is hard)
- Bad idea: hypercube (routing is easy, can be read off from the labels)

What kind of graph to use?

- Random walks on *expander* graphs mix rapidly: log(n) steps to a random vertex
- *Ramanujan* graphs are optimal expanders
- To find a collision: find two distinct walks of the same length which end at same vertex, which you can easily do if you can find cycles

Example: graph of supersingular elliptic curves modulo p (Pizer)

- Vertices: supersingular elliptic curves mod p
- Edges: degree ℓ isogenies between them
- $\ell+1$ – regular
- Graph is Ramanujan
- # vertices ~ p/12
- p ~ 256 bits

Collision resistance

Finding collisions reduces to finding isogenies between elliptic curves:

- Finding a collision→ finding 2 distinct paths between any 2 vertices (or a cycle)
- Finding a pre-image→finding 1 path between 2 given vertices
- $O(\sqrt{p})$ birthday attack to find a collision

One step of the walk:

- $E_1 : y^2 = x^3 + a_4x + a_6$
- $j(E_1) = 1728 * 4a_4^3/(a_4^3 + 27a_6^2)$
- 2-torsion point $Q = (r, 0)$
- $E_2 = E/Q$ (quotient of groups)
- $E_2 : y^2 = x^3 - (4a_4 + 15r^2)x + (8a_6 - 14r^3)$.
- $E_1 \rightarrow E_2$
- $(x, y) \rightarrow (x + (3r^2 + a_4)/(x-r), y - (3r^2 + a_4)y/(x-r)^2)$

Timings

- p **192**-bit prime and $\ell = 2$
- Time per input bit is 3.9×10^{-5} secs.
- Hashing bandwidth: 25.6 Kbps.
- p **256**-bit prime
- Time per input bit is 7.6×10^{-5} secs or
- Hashing bandwidth: 13.1 Kbps.
- 64-bit AMD Opteron 252 2.6Ghz machine.

Other graphs

- Vary the isogeny degree
- 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
  - Ramanujan graph
  - Different problem for finding collisions