Abelian Square-Free Dithering and Recoding for Iterated Hash Functions

Ronald L. Rivest

MIT CSAIL

ECRYPT Hash Function Conference

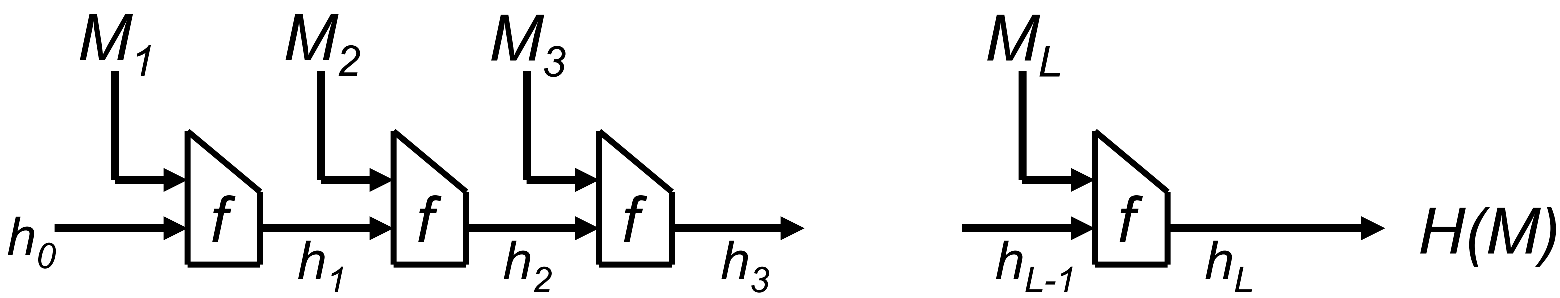
June 23, 2005

CSAIL

Outline

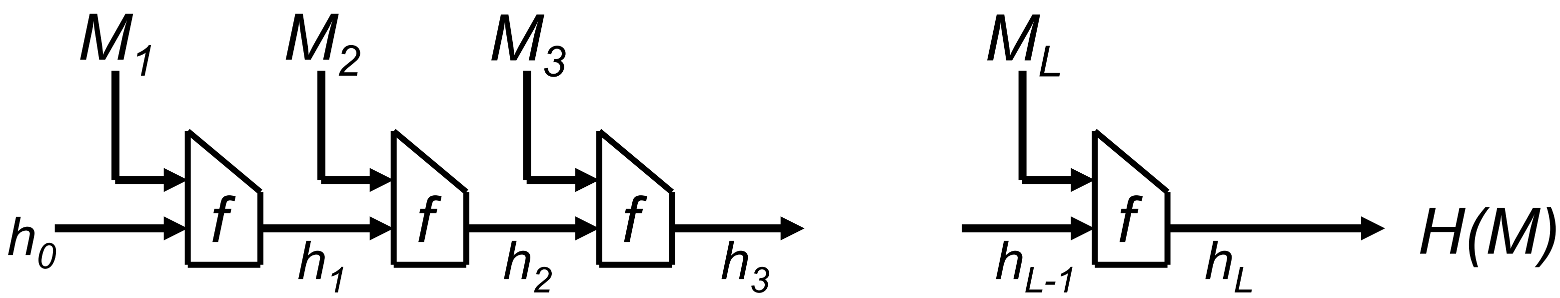
- Dean/Kelsey/Schneier Attacks
- Square-Free Sequences
  - Prouhet-Thue-Morse Sequences
  - Towers of Hanoi
- Abelian Square-Free Sequences
  - Keränen's Sequence
- Dithering and Recoding
- Open Questions
- Conclusions

Typical Iterated hashing



- Message extended with 10* & length (MD)
- f is *compression function.*
- h_0 is *initialization vector (IV)*
- h_i is i-th *chaining variable*
- Last chaining variable h_L is hash output $H(M)$

Dean/Kelsey/Schneier Attacks



- Assumes one can find *fixpoint h* for f, M_0:
  $$h = f(h, M_0)$$
- Can then have *message expansion attacks* that find *second preimage* by
  - Finding many fixpoint pairs (h,M)
  - Finding a fixpoint h in actual chain for given message
  - Finding another shorter path from h_0 to some chaining variable
  - Creating second preimage with this new starting path using message expansion to handle Merkle-Damgard strengthening

Dithering and Recoding

- Make hash function round *dependent on round index i* as well as h_{i-1} and M_i
- *Dithering:* include dither input d_i to compression function:

  $$h_i = f(h_{i-1}, M_i, d_i)$$

- *Recoding:* Include dither input as part of i-th message block

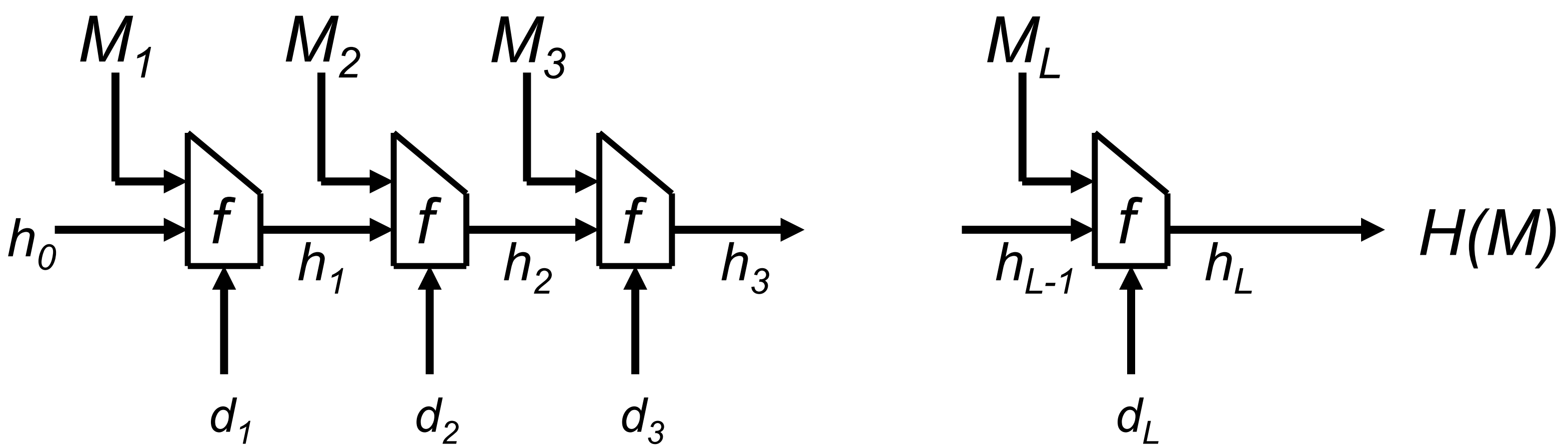
  $$h_i = f(h_{i-1}, M'_i)$$

  where

  $$M'_i = (M_i, d_i)$$

- (These are equivalent, of course…)

Iterated hashing with dithering



- How to choose dither input d_i?
  - Could choose $d_i = i$
  - Could choose $d_i = r_i$ (pseudo-random)
  - Use *square-free sequence* d_i (repetition-free sequence; no repeated symbols or subwords.)

Square-Free Sequence

- A sequence is *square-free* if it contains no two equal adjacent subwords.
- Examples:

  `abracadabra` is square-free
  `hobbit` is not (repeated "`b`")
  `banana` is not (repeated "`an`")

- Dithering with a square-free sequence prevents message expansion attacks. (Would need fixpoint that works for all dither inputs.)

Infinite square-free sequences

- There exists infinite square-free sequences over 3-letter alphabet.
- Start with parity sequence:

  0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0…

  i-th element is parity of integer *i*. This (Prouhet-Thue-Morse, or PTM) sequence is only *cube-free,* but…
- Sequence of inter-zero gap lengths in PTM is square-free:

  2 1 0 2 0 1 2 1 0 1 2 0 2 1 0 2 0 1 2 0 2 1…

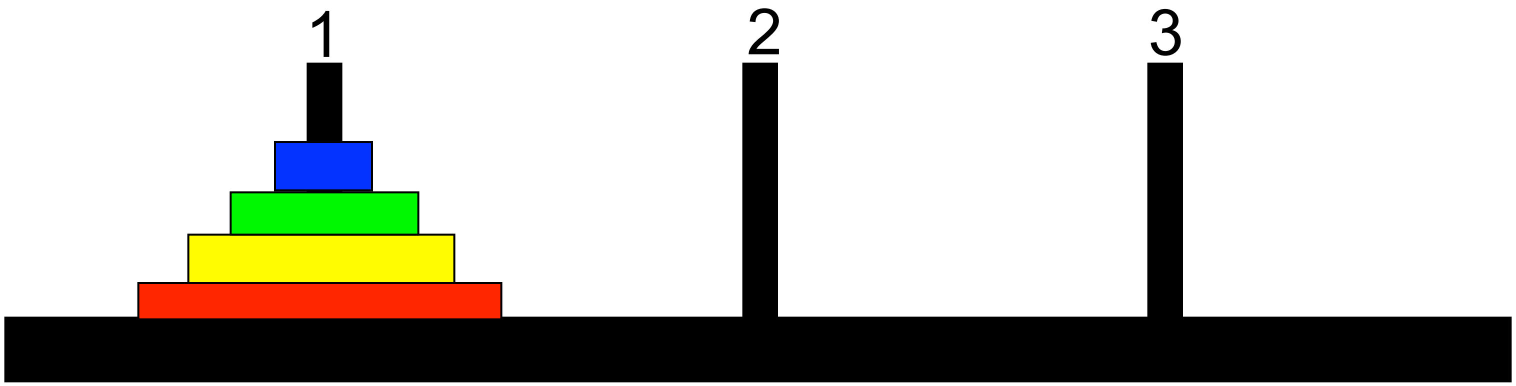
Generating infinite sf sequences

- Or:
  - Take two copies of PTM sequence; shift second one over by one, then code vertical pairs:

```
A = 00, B = 01, C = 10, D = 11:
  0 1 1 0 1 0 0 1 1 0 0 1 0 1 …
  - 0 1 1 0 1 0 0 1 1 0 0 1 0 …
  ----------------------------
  - C D B C B A C D B A C B C …
```

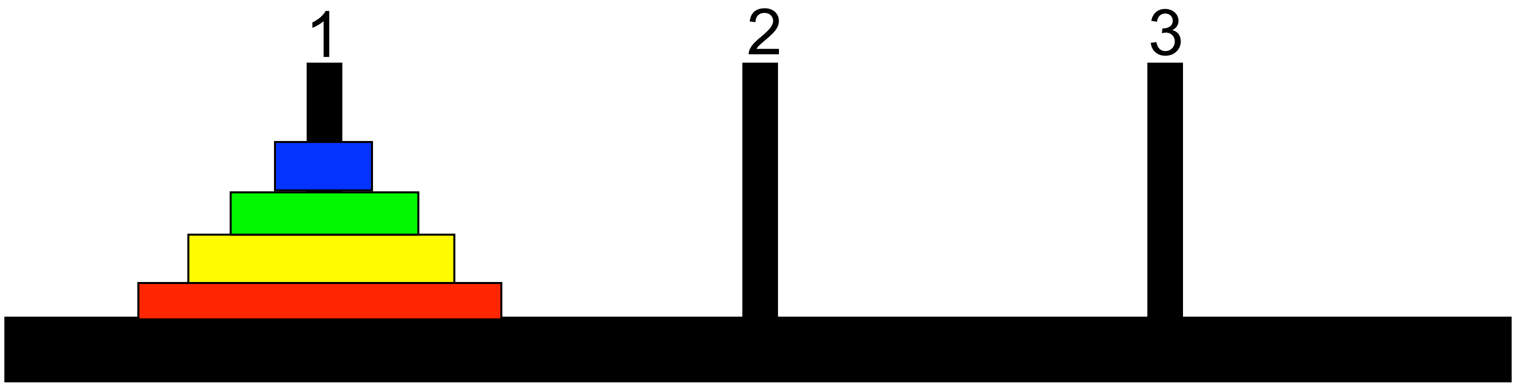
- Result is also square-free.

Towers of Hanoi Sequence



- Optimal play moves small disk on odd moves cyclically 1->2->3->1->2->3…; even moves are then forced.
- Code moves with six letters as A[1->2], B[1->3],C[2->1],D[2->3],E[3->1],F[3->2]
- Optimal sequence is square-free! (Shallit &c)

Towers of Hanoi Sequence



- Code moves with six letters as A[1->2], B[1->3],C[2->1],D[2->3],E[3->1],F[3->2]
- Optimal play:

  A B D A E F A B D C...

- Easy to generate sequence for infinitely many disks...

Abelian square-free sequences

- An even stronger notion of "repetition-free" than (ordinary) square-free.
- A sequence is *abelian square-free* if it contains no two adjacent subwords *yy'* where *y' is a permutation* of *y* (possibly identity permutation).
- Example:

  `abelianalien`

  is square-free but not abelian square-free, since "`alien`" is a permutation of "`elian`".

Infinite ASF sequences exist

- Thm (Keränen). There exists infinite ASF sequences on four letters.
- Keränen's sequence based on "magic sequence" S of length 85:
  abcacdcbcdcadcdbdabacabadbabc
  bdbcbacbcdcacbabdabacadcbcdca
  cdbcbacbcdcacdcbdcdadbdcbca
- Let $\sigma(w)$ denote word *w* with all letters shifted one letter cyclically:
  σ(abcacd) = bcdbda

Generating infinite asf sequence(I)

- Start with Keränen's magic sequence
  S = abcac...dcbca (length 85)
- Apply morphism:
  a → S = abcac...dcbca
  b → $\sigma(S)$ = bcdbd...adcdb
  c → $\sigma^2(S)$ = cdaca...badac
  d → $\sigma^3(S)$ = dabdb...cbabd
  simultaneously to all letters.
- Repeat to taste (each sequence is prefix of next, and of infinite limit sequence).

Generating infinite asf sequence(II)

- Count *i = 0* to infinity in base 85
- Apply simple four-state machine to base-85 representation of *i* (high-order digit processed first).
- Output `a/b/c/d` is last state.
- Requires *constant (amortized) time per output symbol.*

Dithering with ASF sequence

- Since Keränen's ASF sequence on four letters is so easy to generate efficiently, we propose using it to dither an iterated hash function.
- This add negligible computational overhead, and only two new bits of input to compression function.

Recoding with ASF sequence

- Can also recode message using given ASF sequence. (This is essentially equivalent to dithering, just viewed another way…)

Open Questions

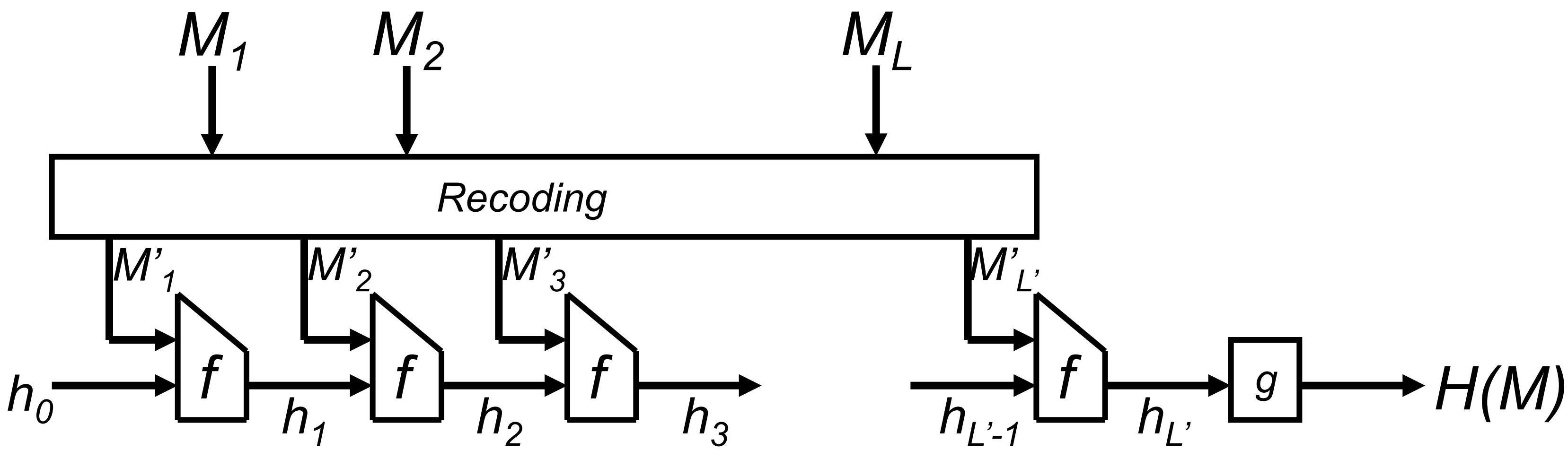
- Can Dean/Kelsey/Schneier attacks be adapted to defeat use of ASF sequences in hash function?
- Does ASF really add anything over SF?
- Are there generalizations of ASF that could be used? ("Even more" pattern-free?)
- Where else in cryptography can ASF sequences be used?

Conclusions

- Abelian square-free sequences seem to be a very inexpensive way to prevent repetitive inputs from causing vulnerabilities in hash functions.
- (Thanks to Jeff Shallit and Veikko Keränen for teaching me about square-free and abelian square-free sequences.)

(The End)

Iterated hashing

- 

Iterated hashing with dithering

-