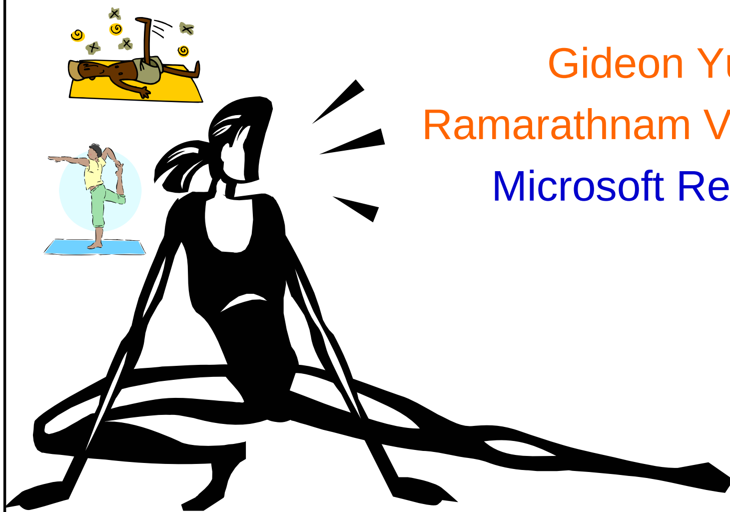


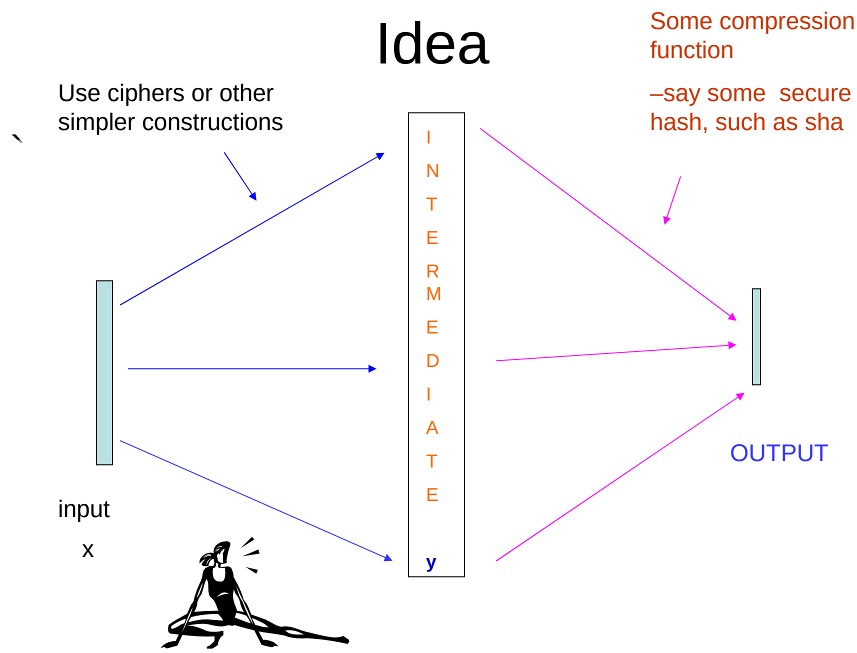
Gideon Yuval
Ramarathnam Venkatesan
Microsoft Research

Stretch before you compress

Secure Hash Constructions

- Secure Hash functions, after a lull of security, are under attack.
  - Differential
  - Renewed interest for new functions
- Some applications do not need so much speed
- All need security
- Can we look at the attacks and thwart them in a simple principled manner?
  - Slower hash functions may be inevitable

Idea

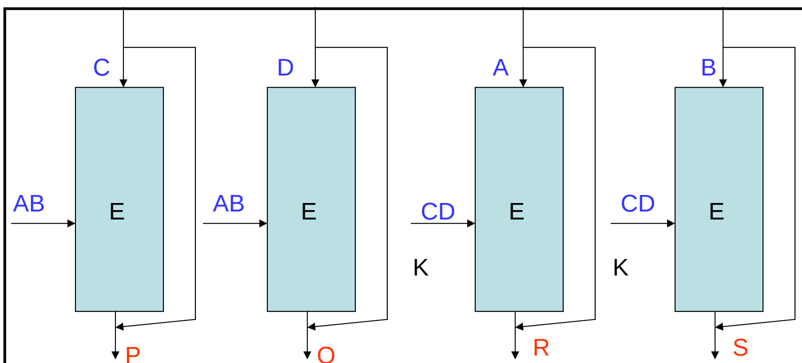


Already stretch happens

- MD4, MD5
  - Simple copying of bits
- Sha,Sha1
  - More randomization of bits
  - For two distinct inputs *x,x'* , Intermediate stage results *y,y'* appear to have some minimum distance between them
- Studied [AHV], [Jutla]

Stretching

- We only look at the compression function
- Let the input be A,B,C,D
- Let $E_K(x)$ be a block cipher so that *2.*|*x*|=|*k*|
- *Example parameters*
  - A,B,C,D are 128 bits
  - K=256
- E.g. AES/Rijndael cipher



Input

A,B,C,D

Output

P,Q,A,B,C,D,R,S

Tag=P,Q,R,S

PQ=front

RS=tail

Stretch function computes a tag; prepends the front, appends the tail tag

Stretch function

A,B,C,D → R,S,A,B,C, D, P,Q

1. This mapping is invertible
   - NO collision in this stage
2. We only count on the pseudo-randomness of the tags: P,Q,R,S.
   - need not be perfectly random
3. The format may not be important
   - Chosen with the structure of the recent attacks in mind

Attacks to bias the tags

- Attacks that can use the fact that keys of the cipher are now at attackers choice
  - Can bias the distribution of the tag P,Q,R,S
- Attacks become feasible if one can perform some inversion tasks

  Invert F: F(AB,C)=E_{AB}(C)+C= P. That is, given P find some AB and C.

  Invert G: Fix AB: G(D)=E_{AB}(D)+D= Q. That is, given Q find some AB and C.
  - We need that the inversions of F and G are infeasible
  - But we need to know exactly what is the effort to cause a given bias in some quantitative way

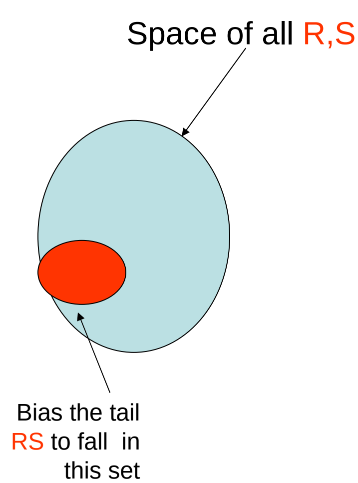
Biassing the tags: Making the front tag PQ arbitrary

- The obvious but expensive attack to make PQ arbitrary
  1. Set P to be arbitrary, find AB and C by inverting F

     $F(AB,C)=E_{AB}(C)+C= P$
  2. But then one loses control of the key AB in
     - $Q=F(AB,D)=E_{AB}(D)+D$
     - To make Q arbitrary invert $G(D)= E_{AB}(D)+D =Q$ for a fixed AB
  - Now A,B,C,D are all fixed.
  - Thus one can expect the tail tag R,S to be "random".

Biassing the tags

- A more complicated attack can try to do bias the tail RS to fall in some set:
- simultaneous birthday attacks to find
  - Many 4 tuples A,B,C,D yielding same P,Q
  - And then compute R,S, hope that they fall in some set
- Bias P,Q,



Unbiassability

- We need the tags to be unbiassable
  - formalization
- One would expect the entropy in the output using arbitrary (e.g., not-inverting F or G) attacks that run in time t

$$\approx |\log(\frac{t}{T_F})| + |\log(\frac{t}{T_G})| + (2-\varepsilon).(\text{BlockCipherLength})$$

where T_F = time required to invert F

T_G = time required to invert G

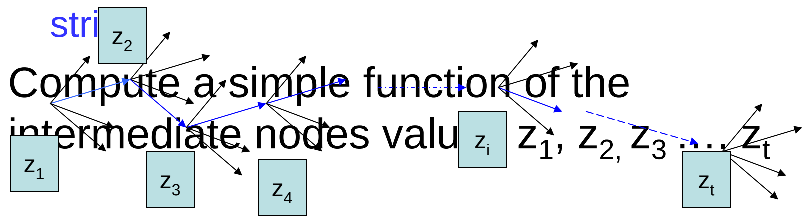
ε = some very small constant

Other waysto stretch

- We can add one more round
  - Use RSAB as input
  - compute a new front tag UV
    - No need to compute the tail tag
  - Output UV ABCD RS.
  - Slower
- Alternate designs to mimic the properties of the above tags.

Expander graph based tags

- Take a suitable expander graph
- Take a walk
  - Start at a node based on ABCD
  - Perform a walk based on a fixed random stri
- Compute a simple function of the intermediate nodes valu $z_1, z_2, z_3 \ldots z_t$



Parameters

- Tag length: We showed a simple scheme for length doubling
  - Smaller than double.
  - increasing by 50% may suffice for some applications
- Performance: (somewhat less than) half the speed of the cipher.