Equivalent keys for HPC

David Wagner
University of California, Berkeley

Results

Key-expansion is lossy.

$\Rightarrow$ Equivalent keys easy to find for all HPC's.

$\Rightarrow$ Can break some HPC-based hash functions.

Key schedule

1. Initialize KX[0...255] with some constants.
2. XOR the $64n$-bit key into KX[0... n-1].
3. Set $(s0, \ldots, s7)$ to (KX[248],...,KX[255]).
4. For j=0 to 2, do
5. For i=0 to 255, do
6. XOR (KX[i] $\oplus$ KX[i+83]) + KX[$s0$] into $s0$. (Lossy!)
7. Replace $(s0, \ldots, s7)$ by $f(s0, \ldots, s7, i, j)$. (Doesn't depend on KX.)
8. Overwrite KX[i] with $s2$ + $s6$.

The idea

Suppose $s0 \equiv 0 \bmod 256$ the first time step #6 is run:

- Then $s0$ is independent of the high bit of KX[0] after step #6.
- And, at step #8, we clobber KX[0], overwriting with a value that does not depend on the old high bit.
- No trace of the high bit of the first key word will remain.

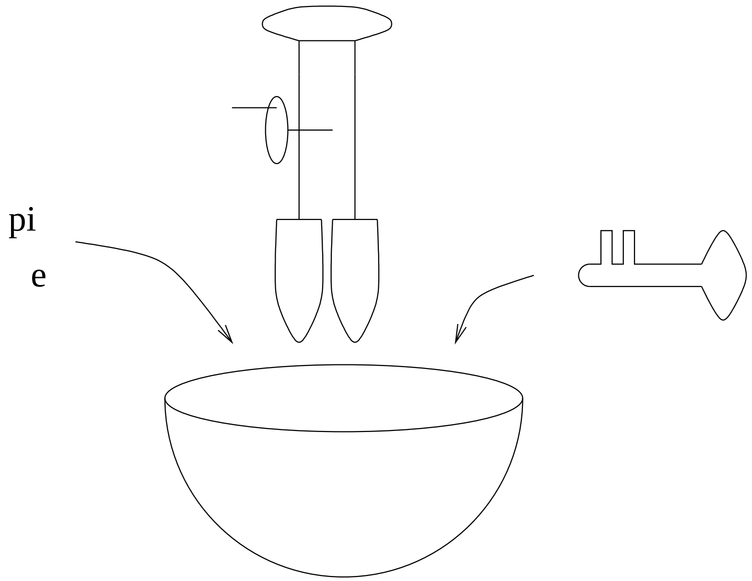
$\Rightarrow$ To find equivalent keys, try flipping one bit of the key.

In other words…

Key-scheduling algorithm:

Mix lots of constants with one copy of the key.

Stir vigorously.



But… If you stir too vigorously, some of the mixture might spill out!

Empirical measurements

Flipping a random key bit gives an equivalent key with prob. 2^{-8}–2^{-10}, for many bit positions.

$\Rightarrow$ Equivalent keys are easy to find.

$\Rightarrow$ Can break Davies-Meyer hash mode.

(can find collisions, second pre-images)

Summary

WE HAVE FOUND ONE FISHY FEATURE OF HPC.