**Subject:** OFFICIAL COMMENT: WaMM is Withdrawn
**From:** "John Washburn" <crypto@washburnresearch.org>
**Date:** Sat, 20 Dec 2008 23:22:58 -0500
**To:** Multiple recipients of list <hash-forum@nist.gov>

WaMM is withdrawn from the competition.

While I have a relatively simple fix for the boneheaded XOR mixing I introduced at the last minute, it is not sufficient to continue with WaMM.

My July version used full vector multiplication of the message vector with the state matrix. This created excellent blending of every bit in the incoming message block with every bit of the state matrix, but this was an order of magnitude slower than the SHA-512 implementation running on the same machines.

I have used the analogy to my family and non-technical friends, that this is similar to showing up to a NASCAR race with a car that tops out at 20 to 22 miles per hour. Yes, you get around the track, but a reasonable question is why are you on the track in the first place?

The simple XOR-ing of the message bits with a row of the state matrix brought the speed to parity with SHA-512, but as David Wilson pointed out, the change gutted the second pre-image resistance of WaMM to nothing.

Substituting:
  WaMM(MessageByte[j], StateMatrix[iRow][j]) + WaMM(StateMatrix[iRow][j], MessageByte[j])
for
  MessageByte[j] XOR StateMatrix[iRow][j]
blocks Wilson's second pre-image attack with only a slight performance hit. (Still slower than SHA-512 though which is not good)

This small change fixes the pre-image attack because there is 35% chance there is no MessageByte which solves
WaMM(MessageByte[j], StateMatrix[iRow][j]) + WaMM(StateMatrix[iRow][j], MessageByte[j]) = C
For a fixed C and fixed StateMatrix[iRow][j].

This "solution" though skews the possible values which could be in any byte of the State Matrix. Any given byte of the state matrix is no longer uniformly distributed from 0x00 to 0xFF. This effect is similar to the distribution of quadratic residues when multiplying mod 256. My "residue" issue is not as severe as that of quadratic residues, but similar. The distribution of possible values calculated for the State Matrix are not uniformly distributed.

Unfortunately, anything which makes the state matrix more like a random pool slows the performance dramatically. I am between Scylla and Charibdis; Speed or Security.

The only way out of the box is to drop my use of matrix multiplication as the primary mixing operator.

That change though means WaMM is not (Wa)shburn (M)atrix (M)ultiplication. I don't see how any "tweak" small enough to be allowed by the NIST will overcome this design decision and still produce a hash of reasonable speed.

Please consider WaMM as withdrawn.