A Second-Preimage and Collision Attack on Abacus

David A. Wilson

December 11, 2008

Abstract

A technique for controlling parts of the internal state of the Abacus hash function is described. This technique leads to a second-preimage attack of complexity 2^{344} operations and a collision attack of approximately 2^{172} operations.

The Abacus hash function[1] is a stream-cipher-based hash function and a candidate for the SHA-3 specification.

The internal state of Abacus consists of four arrays ra, rb, rc, and rd of length 1, 5, 37, and 89 bytes respectively. (The state also includes four counters, though these are data-independent.) By selectively choosing message bytes, an attacker can control portions of this internal state.

During one cycle of the "absorb" phase of the hash function, a single message byte affects the state as follows:

- The message byte is XORed with the value rd[0].
- The resulting value is put through a non-linear s-box.
- The s-box output is XORed with the (data-independent) value of ctr4.
- The resulting value is fed into a linear transform over $GF(2^8)$ along with a deterministic function of the internal Abacus state. (This can be viewed as four separate linear equations, each of which outputs a byte.)
- Each of the four transform outputs is put through the s-box (independently, in parallel).
- One of the resulting bytes is fed back into a fixed position in ra, one into rb, one into rc, and one into rd.

The arrays are then rotated by one byte.

Each step of the above sequence is a reversible permutation. Thus, it is possible to specially craft a message byte in order to cause a desired value to appear as one of the bytes in the last step. In particular, if

```
msg[i] = rd[0] ^ sboxinv(ctr4 ^
                         sboxinv(x) ^
                         sboxinv(ra ^ rd[58]) ^ ctr1 ^
                         3 * (sboxinv(rb[0] ^ rc[24]) ^ ctr2) ^
                         2 * (sboxinv(rc[0] ^ rb[3]) ^ ctr3)
                        )
```

then the resulting value stored back into rd will be x.

By applying this technique to 89 consecutive message bytes, one can set the entire rd array to an arbitrary value.

This leads to the following attacks:

Second-preimage attack. Given a message m, determine the state of the rd array after "absorbing" the entire message, but before absorbing any trailing data (the padding or "train"). Let $|m|$ denote the message length in bytes. Choose the first $|m| - 89$ bytes of message m' randomly, then the final 89 according to the above method in order to make the rd array when processing m' equal to the rd array when processing m. The ra, rb, and rc arrays will be equal with probability $2^{-8*(1+5+37)} = 2^{-344}$. If the state is the same, the remainder of the Abacus processing will be the same (since the messages are of equal length). Thus, m' is a second preimage of m with probability 2^{-344}.

Collision attack. Similarly, there is a standard birthday collision attack that also makes use of fixing the rd array. Choose a message length $|m|$ and a target 89-byte value rd*. (The value of rd* does not matter; it could be the zero array.)

- Choose the first $|m| - 89$ bytes of a message randomly.
- Choose the final 89 bytes such that the rd array is equal to rd*.
- Compare the ra, rb, and rc values against those of all previously generated messages.
- Repeat until a collision of the ra, rb, and rc state is found. This will result in a hash collision of the two messages.

Since ra, rb, and rc collectively contain 344 bits, we can expect a collision in about 2^{172} iterations. (Note: The above described attack is memory-inefficient; however, by making each message dependent on the previous one–e.g. by using the previous hash output instead of random bits–one can construct a memory-efficient collision algorithm using the cycle-finding method of Floyd or Nivasch.)

References

[1] Abacus: A Candidate for SHA-3. Neil Sholer. `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Abacus.zip`.