AES-hash

Principal Submitter: Bram Cohen
(415)775-6963
bram@gawth.com
1166 Pine Street #11
San Francisco, CA 94109

Auxiliary Submitter: Ben Laurie
ben@algroup.co.uk

May 2, 2001

1 Properties

AES-hash is a secure hash function, meaning it takes an arbitrary bit string as input and returns a fixed length (in this case, 256 bit) string as output.

Any alteration of the input should completely garble the output. Finding two files which hash to the same value should require on average approximately 2^{128} operations. Finding a file which hashes to a specific value should require on average 2^{255} operations. It should be impossible to deduce anything about a file from its hash in a way faster than guessing at the entire original file.

AES-hash parallelizes to the limited extent that its key setups can be done in parallel with its encryptions, but a file as a whole must be hashed serially. It does, however, only require a single pass.

Secure hash modes require no keying material, but it would be straightforward to make keyed variants.

AES-hash requires a small fixed amount of memory to keep its H_i values in but only needs to keep a single block of the hashed file in memory at once.

AES-hash works on arbitrary bitstrings, to make it applicable in as wide a variety of applications as possible.

All secure hash functions act as a sort of perverse compression mechanism, boiling everything down to a small fixed size, in this case 256 bits.

2 Intellectual Property

AES-hash is largely derivative of Davies-Meyer and its padding is derived from other hashing functions. We know of no intellectual property restrictions on it's use, and make no claim of ownership of our own.

3 Description

AES-hash is a secure hash mode for AES, with the same properties and key length as SHA-256. Its advantage is greater performance.

Rijndael is used in 256-bit key, 256-bit block mode.

First, the file to be hashed is padded to make it's length be an even multiple of the block size and include a length encoding. This is done by padding with zeros to the next size which is an odd multiple of 128 bits and then appending a 128-bit big-endian encoding of the number of bits in the original file.

The resulting file is a multiple of 256 bits long, and has the property that no two different files pad to the same final file. It also works on arbitrary bitstrings, not ones which happen to be an even number of bytes. It is assumed that byte strings are big-endian, so, for example, a 'one followed by seven zeros' corresponds to 128.

Once that is done, the file is hashed using a slight variant of Davies-Meyer.

For Davies-Meyer, the file is broken into 256-bit blocks $x_1, x_2 \cdots x_n$.

$$H_0 = 2^{256} - 1 \tag{1}$$

$$H_i = E_{x_i}(H_{i-1}) \oplus H_{i-1} \tag{2}$$

H_n is the Davies-Meyer hash. It is appended to the end of the file as x_{n+1} and H_{n+1} is computed. H_{n+1} is the AES-hash of the original file. That is

$$H = H_{n+1} = E_{H_n}(H_n) \oplus H_n \tag{3}$$

alternatively, the final value can be computed using the exclusive or of H_n and x_n as the key and H_n as the block to be encrypted. This has the advantage of keeping the input size at 2^{512} instead of reducing it to 2^{256}. This dramatically increases the cost of an attack on the final block.

$$H = H_{n+1} = E_{H_n \oplus x_n} \oplus H_n \tag{4}$$

Without the last step, an attacker could take a hash without knowing the corresponding file, and use it to generate the hash of a file which is the original file with an appended bitstring of arbitrary content.

Since this algorithm returns the Davies-Meyer hash of a file that is an extension of the input file, it should be at least as strong as Davies-Meyer.

Davies-Meyer was selected because it is unbroken and, unlike Matyas-Myer-Oseas, it allows the next key setup to be done in parallel with the previous encryption.

The length encoding at the end provides some resistance to attacks which find a cycle in the round function. It is consistent with the same operation in SHA-1.

AES-hash allows for single-pass hashing almost as fast as rijndael in 256-bit key and block mode, with the same hash size as SHA-256.

4 Characteristics

Security Function	hashing
Error Propagation	infinite
Synchronization	none
Parallelizability	interleaved
Keying Material Requirements	none
Counter/IV/Nonce Requirements	none
Memory Requirements	256 bits of hash + 256 bits of input bitstream at a time + AES requirements for two key schedules
Pre-processing Capability	N/A
Message Length Requirements	padding necessary (to an odd multiple of 128 bits + 128 bits)
Ciphertext Expansion	fixed reduction to 256 bits
Other Characteristics	none

5 Performance

If the AES key schedule setup takes time T_k, a single block encryption takes time T_e, a 256-bit exclusive or takes time T_x and the input file plus padding consists of n 256 bit blocks, then the total processing time T_P is

$$T_P = (n+1)(T_k + T_e + T_x) \tag{5}$$

and the total elapsed time T_E is

$$T_E = T_k + T_e + T_x + n \min(T_e + T_x, T_k) \tag{6}$$