Hash collision calculator

$$\left(\frac{256^{\textit{file size in bytes}}}{2^{\textit{hash strenght in bits}}}\right)$$

different files of that size produce the same hash of that strength

If the answer is 1, you have 1:1 correspondence; there are as many files as there are hash.
If the answer is <1, only that fraction of hash are required for 1:1 correspondence.

...now that you have collision for 1 file size,
sum collision for any range of files

Samples:

1 **32**-Byte file produces only 1 unique 256-bit hash. (256^32) / (2^256) = 1
That's because the number of possible files is equal to the number of possible hash;
a good hash attempts to mirror the file in distortion but conservation. It ensures that
collisions don't look too plausible, and that plausible collisions are extremely far apart.

256 different **33**-Byte files produce the same 256-bit hash. (256^33) / (2^256) = 256
(By adding 1 Byte, there are **256** times as many possible files as there are hash.)

65,536 different **34**-Byte files produce the same 256-bit hash. (256^34) / (2^256) = 65,536

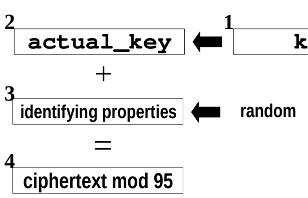
16,777,216 different **35**-Byte files produce the same 256-bit hash. (256^35) / (2^256) = 16,777,216

10^2331 different **1kB** files produce the same 256-bit hash. (256^1,000) / (2^256) = 10^2331

10^2,408,162 different **1MB** files produce the same 256-bit hash. (256^1,000,000) / (2^256) = 10^2,408,162
The formula actually divides (number of possible n-Byte files) by (number of possible n-bit hash.)
If you plug in a 31-Byte file, 1/256th of possible 256-bit hash are needed for 1:1 correspondence.

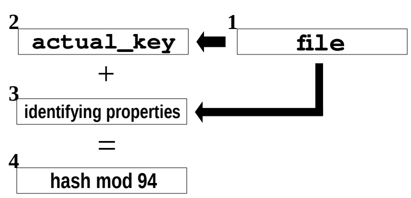
Hash without collision or reversal shortcuts—a variable Authorship function:

Authorship multi-way function (of type step-down)



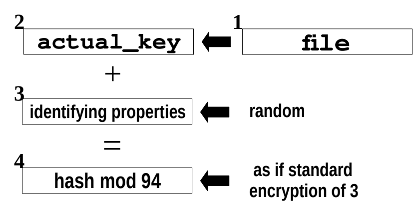
If 1 and 4 are known, 3 is deduced and tested against search priorities. (In Multiway, actual_key is 1MB, transformed for each use as paired with file bytes, 3 is input files, and 4 undergoes mod 256. There, 3 is not publicly-verifiable; decrypting parties must build personal search priorities to sift through output files for plausible artifacts.)

Bad hash



If 1 is known, 4 is known because 1 seeds for 3. This makes 3 inherent to this function, but no collision or reversal shortcuts means no inherent solutions. (Both Multiway and Authorship begin with perfect secrecy of the One-time pad then step down from there by transforming actual_key. Here and in Authorship, 3 is publicly-verifiable.)

Good hash



When creating hash, enter randomness to generate 3. This means unique hash even for the same file. When checking hash, PROVIDE 1 AND 4 as if standard Authorship decryption; this utility will have 2 options: create, verify. FILE AND HASH MUST BE PROVIDED TO VERIFY so that 3 is deduced and tested against search priorities (the hash is not for comparison.)