

Specification of Steak2 Cipher

-

**A step towards a provably secure single primitive for
encryption, pseudo random generation and hash
computation**

By

Henrick Hellström

StreamSec HB

2001-may-18

Steak2

Steak2 is a competitively fast error propagating cipher with remarkable security properties and security services. It can be used as a cipher, as a pseudo random generator, and as a hash function.

Security properties

We know of no attack against Steak2 set up with a full sized random key that requires less than a factor of 2^{4000} queries, memory or effort. If Steak2 is set up with a smaller key, that key will be expanded using the Steak2 algorithm itself. It seems safe to assume that the high security bounds of Steak2 will assure that brute force will always be the most effective attack.

Security properties: A single primitive

Because Steak2 is error propagating, it can be used as a hash function. Define $H_k(x) = \text{Steak2_Encrypt}(0^n, k_x)$, where 0^n is a string of n zero bits and k_x is the internal state obtained after encrypting x with the key k . Changing any bit of x will result in a uniformly distributed probability that any bit of $H_k(x)$ will change, practically regardless of the size of n . Since Steak2 is a secure cipher, it is trivially true that $H_k(x)$ will be one-way (because otherwise it would be possible to extract prior plain text directly from the subsequent cipher text).

Speed

Our optimised implementation of Steak2 with an eight-element vector executes with a speed per byte modestly slower than 128-bit block ciphers like Rijndael and TwoFish in CBC-HMAC-mode. Because of the similarities between Steak2 and TwoFish, we estimate that Steak2 should be no less than one quarter as fast as TwoFish-ECB.

Algorithm

In accordance with the reference implementation, Steak2 might be conceptualised as a single iterated function G of the key data and of the previous cipher text byte.

Structure of key stream generator

We define:

$$H: \{0,1\}^{32 \times 3} \times [0..255]^4 \rightarrow \{0,1\}^{32} \text{ as}$$

$H(x, a, v, S) = (M(S(x + a)) + v) \text{ rol } 3$, where S represents the application of four 8-to-8 s-boxes and M represents a 32-to-32 bijective linear transformation.

Each of the four s-boxes is a single cycle permutation. The design purpose of choosing single cycle permutations is that the set S of such permutations is the largest subset S of all permutations that (a) has a linear time bijective map from $[0..|S|)$ into S , and (b) has sufficiently advantageous security properties.

The M linear transformation is an MDS matrix with a rotation left by one applied on the input and a rotation right by one applied on the output. The matrix is generated by sorting the four rows of the 4 times 256 table representation of the TwoFish MDS matrix. Rows 0, 2 and 3 have been sorted using arithmetic comparison in the set of natural numbers, and row 1 has been sorted using arithmetic comparison in $Z_{2^{16}}$. The sorted table T is generated by the elements at position $[i, 2^j]$, and for each i, j , $M_{i*8+j} = T_{i, 2^j}$.

The purpose of the rotate left by 3 is to feed M with different values in both byte 0 and byte 3 of the input, whenever the most significant byte of v changes at the previous round. In combination with the security properties of A , V and S , this will speed up diffusion when H is iterated.

We further define:

$$G_i: \{0,1\}^8 \times \{0,1\}^{32 \times n \times 2} \times [0..255]^4 \rightarrow \{0,1\}^{32} \text{ as}$$

$$G_0(c, A, V, S) = H(v_0, a_{n-1}, v_{n-1}, S), \text{ where } v_{n-1} \text{ shr } 24 <- c, \\ \text{and if } i > 0, G_i(c, A, V, S) = H(G_{i-1}(c, A, V, S), a_{n-i-1}, v_{n-i-1}, S), \text{ where } A \text{ and } V \text{ are two} \\ \text{vectors of eight 32-bit integers respectively.}$$

Lastly, we define:

$$F: \{0,1\}^8 \times \{0,1\}^{32 \times 8 \times 2} \times [0..255]^4 \rightarrow \{0,1\}^8 \text{ as}$$

$F(c, A, V, S) = G_{n-1}(c, A, V, S) \text{ mod } 256$, where c is the previous cipher text byte, A, V, S is the current key data, and $F(c, A, V, S)$ is the key stream output.

Internal state update

In the reference implementation the function H updates the value of v_{n-i-1} that the G function passes to it. More specifically, we have that $v_{n-i-1} <- H(G_{i-1}(c, A, V, S), a_{n-i-1}, v_{n-i-1}, S)$. Lastly, the function F will shift the entire array V eight bits right.

The benefit of using this particular update function is that it requires no additional code, except a single assignment at each round. There are other update functions that have better security properties (e.g. s-box updates, or encryption of V using a second cipher), but since there is no known attack against Steak that requires less than a $O(2^{4000})$ effort, using such update functions would probably be an overkill.

Security proof

The insecurity level of a keyed function is by convention defined as the probability that you might distinguish it from a random function after a certain number of queries to it using any distinguisher. (cf Goldwasser & Bellare 1999) The task at hand is to (1) find an appropriate way to model Steak as a structure of keyed functions, (2) model the way these functions are promoted, and (3) find the most effective distinguisher for each of these functions.

Firstly, let $F_k: \{0,1\}^8 \Rightarrow \{0,1\}^8$ be the function such that k is the key data of Steak, and $F_k(x)$ is the key

stream output of Steak given the key data k and the preceding cipher text byte x . If the vector size is large enough, this function has a very low insecurity level. In order to prove so, the following observations should be made:

Proposition 1: Let h be a selector of any combination of 256 distinct elements out of a set of at least 65536 of the least natural numbers. Let the injective function $G_h: \{0,1\}^8 \Rightarrow \{0,1\}^{32}$ be such a combination. Then, for any random function $g: \{0,1\}^8 \Rightarrow \{0,1\}^8$, there is an h such that for all x we have that $g(x) = G_h(x) \bmod 256$.

Now, let's structure the function F_k in the following manner: Let $G_k^i: \{0,1\}^8 \Rightarrow \{0,1\}^{32}$ be the dynamic key data generator of F_k at round i . Then for all x we have that $F_k(x) = L_k(G_k^{n-1}(x)) \bmod 256$, where n is the vector size, and the function L_k simply undoes the key data feedback of the last round. Furthermore, due to the PV-pipe of the feedback process, we have that the 256-element combination corresponding to the function G_k^i has a selector corresponding to the preceding round function G_k^{i-1} . In other words, since the main function of Steak is bijective, the probability increases with the vector size of Steak that for any random function g there is an equivalent function F_k . The bijectivity of the main function also ensures that the number of collisions for each F_k in the set $\{F_k\}$ converges to a constant function of k as the vector size increases. This means that for all practical purposes the function F_k might be modeled as a random function.

Secondly, we ought to prove that the process of selecting the next function $F_{x^o k}$ also might be modeled as a random function.

However, one ought to notice that there would be a potential weakness to any such proof. For each of the 2^{2048} possible g there is only a limited number of keys k such that F_k is equivalent to g . If this set is known, then for each cipher text x it would be trivial to compute the set of possible functions $F_{x^o k}$. For all we know it would be an immensely complex task to find the set $\{k\}$ corresponding to a random function g since the set of all keys is larger than 2^{6000} , but we can't prove that it can't be done.

One should also note that there are more than 2^{4000} full sized keys corresponding to each function g . Since all S-boxes are single cycle permutations, the ratio of keys actually inconsistent with the extension of the promotion from F_k to $F_{x^o k}$ cannot be larger than 3 divided by 256 (given that full sized random keys are used), because that's the trivial upper bound of the probability that, for a vector size of one, you will not find two keys k, k' , that differs in at most three of the S-boxes and are such that both F_k and $F_{k'}$ are equivalent to g_0 and for all possible values of x both $F_{x^o k}$ and $F_{x^o k'}$ are equivalent to g_1 . (Note that this is an approximate model. The MTT is not MDS, so the relation between the sboxes would have to be more complex, but since the MTT is bijective the probability of finding them is roughly bounded by the same number, in particular if the vector size is larger than one.) If one does not take the static and dynamic vectors into account, this entails that the unicity distance of Steak cannot be less than 3 bytes and key retrieval requires an effort of at least $O(2^{24})$, where O is bounded downwards by the effort of finding the set $\{k\}$ corresponding to the value pair $(x, g(x))$ for a random function g . These bounds are fully independent of the security of the promotion function.

This said, the proof is more or less trivial.

To begin with, it is important that the dynamic key data V_m significantly contributes to the promotion of F_k to $F_{x^o k}$. If it doesn't, then it might be possible to extract the s-boxes and the vectors separately from the sequence (or rather 256-branched tree) of (F_k^m) , and it would make less difference whether the dynamic vector is pseudo random or not. More specifically, we would prefer that for any random function g and fixed A_k and D_k , there is a constant epistemic probability that one might find a vector V_m such that $F_k^m = g$. This seems to be the case: Each G_k^m depends (close to) linearly on the value of $V_m[n-i-1]$, but a linear change to G_k^m will, after it has passed through $A_k[n-i-2]$ and D_k , result in a non-linear change to $G_k^{i+1^o m}$. This is largely due to the fact that the s-boxes are single cycle, and hence never can be fully linear. Consequently, the probability is extremely low that there are non-zero values x and y such that $(V_m[n-i-1]+x, V_m[n-i-2])$ and $(V_m[n-i-1], V_m[n-i-2]+y)$ will result in the same function $G_k^{i+1^o m}$. The degree to which this property holds for large chunks of V_m ought to be high. The s-boxes will always be non-linear to some degree, and this non-linearity will show in the way G_k^m selects $G_k^{i+1^o m}$.

The pseudo randomness of V_m might be established as follows: Let $H_k^{i^o m^j}$ be such that for any i, j, m, k , we have that $H_k^{i^o m^j}(PV_{n-i+j-1}) = PV_{n-i-1}$, where $PV_i = V[i]$ according to the notation of the algorithm. Define the value $PV_{n-i+j-1}$ such that if $n-i+j-1$ is greater than n , then this is the corresponding internal state of F_k^{m-1} at round $i-j-n$, etc. Note that the four most and least significant bits of $H_k^{i^o m^j}$ will always depend (close to linearly) on the cipher text byte C_{m-1} . This byte will be pseudo random if the least significant byte of $H_k^{m-1^o j}$ is pseudo random. It isn't material in this context that this byte is public. Now, according to the assessments about the impact of V_m on the selection of each G_k^m , it is correspondingly reasonable to assess that there is a value e such that if $j > e$, then $H_k^{i^o m^j}$ is a pseudo random permutation. (Our statistical tests indicate that $e = 3$ is sufficient.) This entails that if e.g. $n = 8$ and $j = 4$, then the string of $V_1[4], V_1[0], V_2[4], V_2[0], \dots$ will be pseudo random up to a length corresponding to at least the strength of each $H_k^{4^o m^2}$, $H_k^{8^o m^4}$. The entropy of the promotion from F_k^m to F_k^{m+1} cannot be less than the entropy of this string, because the strings $V_1[7], V_1[3], \dots, V_1[6], V_1[2], \dots, V_1[5], V_1[1], \dots$ will all have the same security bound, and if the conjunction of them would decrease the strength of the promotion from F_k^m to F_k^{m+1} , then this must be due to the way some $H_k^{i^o m^j}$ selects the next $H_k^{i+1^o m^j}$, and that would be contrary to our assumptions that this step cause a high probability security increment.

It is important to note that there is a close relationship between the function F_k^m and the promotion of F_k^m to F_k^{m+1} , since each G_k^m is constituted by the same steps in the algorithm as $H_k^{i^o m^j}$. There is however a "but": Although the set $\{\dots, (G_k^{i-1^o m}(x), G_k^m(x)), \dots\}$ is a subset of $H_k^{i^o m^j}$, knowing G_k^m extensively, but nothing about $G_k^{i-1^o m}$, will not leak any information at all about $H_k^{i^o m^j}$ given that $i > 1$. In the informal sense, a cipher like Steak will never be cracked by an adversary who first learns the value of one vector element, solves an equation to learn another value, etc, but rather as the result of an statistical attack of some kind. Primarily because the only thing known to attacker in the first place is the cipher text and possibly the plain text - secondarily because unequivocal knowledge of isolated elements of key data will still leave too many unknown variables for the equations to be solved. The point of this exercise is foremost to prove that it is impossible that

knowledge of one function F_k^m could enable the adversary to make more than a lucky guess about any internal key data, but also to justify the assessment that knowledge of several functions in the tree (F_k^m) could not enable the adversary to do significantly better, because the relationship between these functions is too complex.

For example: Knowing both G_k^{i-1} and G_k^i extensively will reveal no more than a 2^{-24} fraction of the extension of H_k^{i-1} . Furthermore, knowing G_k^{i-1} and G_k^i extensively would make it theoretically possible to compute the set $\{B\}$ of all possible values of the sum $B = A[n-i-1] + V[n-i-2]$ through exhaustive key search, but only B would be revealed. If the set $\{B\}$ is sufficiently large, extensive knowledge of G_k^{i-1} and G_k^{i+1} would not add a significant amount of information about either $V_m[n-i-2]$ or $V_{m+1}[n-i-2]$. A conservative estimation is that $|\{B\}| > 2^{31.9}$ (based on the number of ways to modify the s-boxes, and obtain the same functions G_k^{i-1} and G_k^i , that would not be possible because the s-boxes are single cycle and the MTT is not MDS), implying that one would need at least 10 value pairs ($G_k^{i-1}(x), G_k^i(x)$) for different m before the intersection of the sets $\{B\}$ was less than 2^{31} . Given that the adversary's knowledge of G_k^{i-1} and G_k^i was likely to actually only be probabilistic to begin with, the increment of knowledge obtained by making more queries is more or less negligible.

Thirdly, we prove that the advantage of any distinguisher for Steak has a polynomial bound, and base this on few assumptions:

1. The relation between the key stream output and the least byte $V[0][0]$ of the internal state vector has no distinguisher with an advantage greater than any other distinguisher for Steak. This is a justified assumption: $V[0][0]$ is a bijective function of $V_m[0][0]$ with the key stream output x as the key parameter, and $V_m[0]$ will have an impact on x only after being distorted by nearly all other key data. This assumption is clearly similar to the assessment that F_k^m might be modeled as a random function.
2. The probability is negligible that, for any pair of key data k and k' , there is a more efficient way to tell if $F_k = F_{k'}$ other than computation of at least one element in the each respective codebook.

The second assumption is justified by the properties of G_k^i and implies that it would be infeasible to find an advantageous trade-off between time, memory and probability, for any method that would narrow the set $\{k\}$ corresponding to F_k^m down to the subset corresponding to F_k^{m+1} . Suppose that we have found a way of representing a subset M_m of the set $\{k\}$ corresponding to F_k^m . (For the sake of simplicity, I will assume that M_m is represented either extensionally, or by the combination of formulae in the vectors and s-boxes separately. There might be ways to represent M_m that put the s-boxes and the vectors into the same formula, but I will assume that since the s-boxes belong to the entire set of single cycle permutations, it will be infeasible to parameterize the entire key space using such formulae, and that it in particular will be infeasible to "do calculus" on such formulae, e.g. to take two arbitrary formulae and in linear time calculate the intersection of the two sets.)

Suppose that the probability that m, k is actually in M_m equals $p = |M_m|/|\{k\}|$, and that we also have found the next subset M_{m+1} and that $|M_m| = |M_{m+1}|$. Then,

by the second assumption, we would have to have at least $2|M_m|$ bytes of memory or perform $O(|M_m|^2)$ computations in order to find the intersection of M_m and M_{m+1} . If we instead have a method of telling if two sets M_m and $M_{m'}$ are intersected, and do so with a constant probability of q , then the advantage of the distinguisher would be an exponential function of the number of queries, but the effort would be a polynomial function of that same number of queries (we would have to test each pair M_m and $M_{m'}$, or else the advantage would at most increase polynomially). If we don't have that kind of resources, and we won't have that if p is significantly larger than 2^{-4000} , and if the effort is a linear function of the number of queries, then the advantage of the distinguisher we are working with cannot increase exponentially depending on the number of queries (as it would if we were able to unequivocally represent the entire sets $\{k\}$ and were able to find and represent their intersections). Conversely, if M_m has a brute forceable size, then p cannot be significantly larger than 2^{-4000} , and the attack would most likely fail for the first sequence of queries. The attack would not be successful unless it was possible to learn something from the failures, so that the set M'_m selected for the next attempt had the same cardinality, but the probability was larger that k actually belonged to it. It is a provable fact that the set M_m would be too small for such an attack to successfully eliminate possible values of the vector elements (except $V_m[n-1][3]$, which would already be known), because too many s-boxes in the complement set of M_m would make each of these vector element values possible. The converse is also true, because it is, by assumption, not possible to unequivocally rule out that a particular set of s-boxes is inconsistent with a sequence (F_k^m) with substantially less effort than brute force on the vectors. So the elimination would have to be probabilistic, and hence the argument of the preceding paragraph applies. The advantage of any distinguisher must hence be bounded by a polynomial function of the number of queries. This concludes this rather informal proof. Q.E.D.

