

Improving the Talos Encryption Scheme via Temporal Seeding

Steven Chiacchira

Here we present a method for mitigating the state collision problem identified in RFC-0. We additionally implement this method in Talos, our official name for the encryption algorithm, and experimentally show that it achieves the desired result of reducing state collisions.

1 Identified Problem

RFC-0 identifies multiple key weaknesses in the Talos algorithm, the most severe being that of state collision. We found that a vast majority (if not all) of the 32 bit keys in our keyspace were liable to collapse to an initial state shared by another key in the early stages of automata evolution. Because the Talos Key Automata Rule has the Markov property, this means that many keys are effectively identical when used in the cipher. This is clearly unacceptable for any cryptographic algorithm, as state collision of this magnitude drastically reduces the keyspace of the cipher.

1.1 Symptoms of State Collision

We noted in RFC-0 that, out of the first 3419 32-bit keys, at least 1581 collide with another key within 2-3 time steps. However, we did not explain that the true number of key collisions within the subset is actually larger due to how the data was collected.

In searching for state collisions, we simulated the first 32,000 states of the automata arising from keys 0-3418, keeping a hashset of all previously seen states to allow for identifying state collisions. A state collision was reported when the currently running automata's state was present in the hashset. However, this scheme does not retroactively report collisions, as the first time a collision is detected with a certain state, it is only reported once (rather than twice). A better method for counting the true number of collisions then would have been to use a hash map to keep track of the number of occurrences of each state, then reporting the number of collisions at the end of the test.

Although we acknowledge that the true number of collisions is necessarily more than the reported 1581 number, this does not affect our analysis or proposed fix, as the truly notable observation was that all collisions were occurring after 2-3 generations of evolution, implying that collisions were occurring due to the automata having relatively few possible second or third states, given an initial state.

Intuitively, the likelihood of a state collision for two keys K_0, K_1 within the first few CA generations should be inversely proportional to the hamming distance between K_0, K_1 . That is, two keys which have similar bits should be more likely to collide, while significantly different keys should be less likely.

1.2 Theoretical Attack

A knowledgeable attacker with enough computational resources would, given this knowledge, know that they only need to search a subset of the keyspace to crack a message by brute force. Such a subspace would consist of representative keys Hamming neighborhoods which minimally cover of the full keyspace. As an example, if our keyspace consisted of two bits (00, 01, 10, and 11), a clever hacker who recognizes that a single bit flip does not change the eventually observed automaton states could check only two keys from the two covering Hamming neighborhoods; keys adjacent to 00 and 00 itself (00, 01, 10), and the final key 11.

2 Mitigation

In order to reduce the number of state collisions within the first few CA generations, we apply a series of 32 transforms to the shift and transpose automata states over an additional 264 generations before encrypting (or decrypting) the first plaintext (or ciphertext) block. We term this series of transforms “Temporal Seeding”, as it spreads the seeding of the initial automaton state through time, in addition to the original spatial seeding. Although 264 generations is quite large compared to the 11 generations between cipher blocks, these generations are only computed for the very first block, meaning that no measurable impact on performance is incurred.

2.1 Temporal Seeding

Givin an initial automaton state S_0 arising from key K and initialization matrix I , we perform temporal seeding by first simulating 8 generations of our automata, yielding state S_7 . We then overwrite cells corresponding to the first bit of key K in I with the first bit of K . We repeat this process sequentially for each bit of K , then simulate 8 final generations to obtain the first block key from the automaton.

In performing this Temporal Seeding process, we allow the automata state many opportunities to diverge from otherwise identical states, decreasing the number of initial collisions.

3 Experimental Results

Using the same method of counting collisions as in RFC-1, we observe only 490 collisions within the first 5 generations of the first 32642 keys. Although this rate of 1.5% is a significant improvement compared to the near 50% rate of collisions in the initial protocol, the ideal rate of 0% has yet to be reached, leaving room for further improvement.