hashcrypt

Modular block encryption based upon a secure hash function. Very experimental, not suitable for production. Latexed version of this README here.

Round Description

Given a key k, for each 256-bit block in the plaintext, $p_1, p_2, p_3 \ . \ . \ .$

the ciphertext $c_1 := H(k) \oplus p_1$ and each encrypted block is defined recursively:

$$c_n := H(n|H(n|c_{n-1})|H(n|k)) \oplus p_n$$

where H(x) is a secure hash function.

Note: $c_n = R(p_n)$.

Iterating rounds

Rounds can be successively applied like so, up until the nth round where n less than or equal to the number of blocks needed to encrypt:

Round 1: $R(p_1), R(p_2), R(p_3) \ . \ . \ .$

Round 2: $R(p_1), R(R(p_2)), R(R(p_3)) \ . \ . \ .$

Round n: $R(p_1), R(R(R(p_2))) \ . \ . \ .$

Security

The security of this cipher scheme depends on the preimage resistance of $H(x)$ in known-plaintext attacks. Additionally, it assumes that from $H(x)$ or $H(k|x)$ for any $k \neq n$, it is impossible to predict $H(n|x)$.

To construct an ideal H(x), you can combine multiple hash functions from various families like so:

$$H(x) = keccak(x) \oplus BLAKE(BLAKE2(BLAKE3(x))) \oplus SHA256(x)$$

Rationale and considerations for constructing H(x): - provides security, even if one of the families of hash functions is thoroughly broken - various hashes have a limited output space, xoring multiple different families helps to increase the output space to be closer to 256-bit - makes it extremely difficult to undo.

Implementation

Coming soon.