

Re: My nonreversible keyrescheduler with full entropy 2^256

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2006-07/msg00542.html>

- *From:* jt64@xxxxxxxx
 - *Date:* 16 Jul 2006 08:40:43 -0700
-

John E. Hadstate skrev:

<jt64@xxxxxxxx> wrote in message
news:1153014838.634366.66970@xx

This password rescheduler makes a password morph into the full 2^256 states, it is none reversible and to find output rescheduled key z you must know t, y and x.

t and y is dependent upon bitpermutations.
x is the savestat created by XOR t and y
so to figure out z you must know all three, or "THE ORIGINAL PASSWORD"
to create t,y,x and roll ->forward in keyscheduling scheme.

You can not recreate t,y,x by knowing/solving one rescheduled key.
The algorithm that creates z is nonreversible you can not find t,y,x even if you know z.

[init]->
password^invpassword=savestatepw
permute password at bitlevel
permute invpassword at bitlevel

So:

$X = T^Y$
 $T = \text{AES_Encrypt}(K, T)$
 $Y = \text{AES_Encrypt}(K, Y)$

[while more passes/byteblocks to reschedule]->
password[^]invpassword[^]savestatepw=keysheduled password
password[^]invpassword=savestatepw
permutate password at bitlevel
permutate invpassword at bitlevel

While more blocks:

$R = T^Y X$
 $X = T^Y$
 $T = \text{AES_Encrypt}(K, T)$
 $Y = \text{AES_Encrypt}(K, Y)$

Questions:

(1) Initial values for T, Y, and K? Where are they?

Let us call initial password T just to make it real stupid we take
scale it down to 16 bit

T=0011 0100 0001 0110

Y=0110 1000 0010 1100 [reversed password]

I have no idea if one is to conclude that the bitdistribution "always
is balanced" within a 256 bit password. Maybe you could tell, but if it
is not i propose just somkind of balancing schedule for password.

You ask for K i am not sure what it is because you proposed it, is it
the permutation algorithm?

If you clarify i give you answer

(2) How does the plaintext get converted to ciphertext?

This is not the actual cipher it is the method to synchronus renew a
key without any keyexchange, the cipher though work just similar, the
session key Z

is expanded into a 256 ->byte permutation that is treated in the manner
below

[init]->

Permutation[^]invPermutation=savestatePermutation

Re: My nonreversible keyrescheduler with full entropy 2^{256}

permute Permutation at bytelevel
permute invPermutation at bytelevel

[while more 256byteblocks to encrypt]->
PRNGbytes=Permutation^invPermutation^savestatPermutation
savestatePermutation=Permutation^invPermutation
permute Permutation at bytelevel
permute invPermutation at bytelevel
cipherbytes=PRNGbytes^plaintext

It is quite similar to the keyrescheduler/rekey mechanism but on byte level

(3) What's the point of this?

To bring ambiguity into the deciphering where you isolate the session key from the actual original key. One session key can have many combinations of [internal] pseudokeys so if you have a permutation blocksize of 256 byte and someone bruteforce the session key of it you could rekey to make the bruteforced password useless.

->You can change the session key every 3 KB without actually exchange any new keys.

Since session key is only oneway communicative with pseudokeys you can not recreate them by the session key. So the attacker only manage to bruteforce 3 KB but after that the bruteforce is useless once rekeyed, and if you do it every 3 KB there is not much other option than bruteforce.

JT

.