Related-key Attacks Against Full Hummingbird-2

Markku-Juhani O. Saarinen
mjos@iki.fi

Research (and my travel!) sponsored by current Intellectual Property owners of Hummingbird-2.

Fast Software Encryption 2013
Singapore, Singapore
13 March 2013

Hummingbird-2

Hummingbird-2 [RFIDSec 2011] is a lightweight authenticated encryption algorithm with a **128-bit secret key** and a **64-bit IV**.

Developed largely in response to my attacks [FSE 2011] against Hummingbird-1, which recovered its 256-bit secret key with 2^{64} effort. That was a single-key attack.

I was involved in the design of cipher number two; we tried to only make minimal changes necessary to counter that attack and some other attacks we found during design phase.

<u>Prior art</u>: I am not aware of any other *(correct)* attacks against the full cipher.

Architecture

All data paths are 16-bit as Hummingbird is intended for really low-end MCUs. State size is 128 bits.

Hummingbird-2 has high "key agility". The secret key is used *as it is* during operation (no real key schedule!). The 128-bit key is split into eight 16-bit words:

$$K = (K_1 \mid K_2 \mid K_3 \mid K_4 \mid K_5 \mid K_6 \mid K_7 \mid K_8).$$

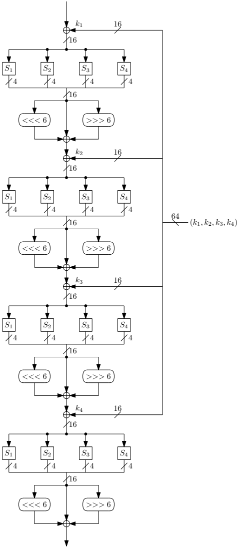
There is only one nonlinear component, called WD16. This is a 16-bit permutation keyed by four subkeys (64 bits total):

$$c = \mathrm{WD16}(p,\ k_1,\ k_2,\ k_3,\ k_4).$$

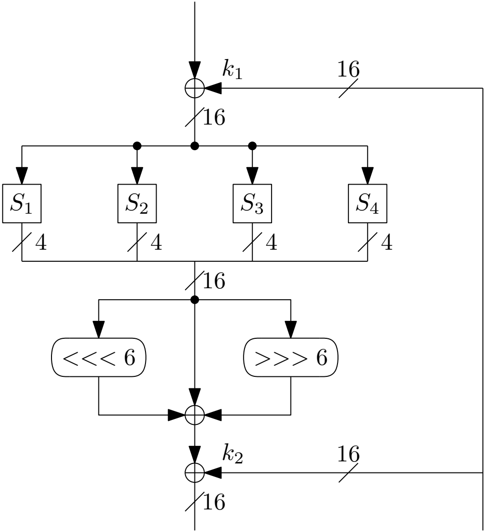
The subkeys are either $(K_1,\ K_2,\ K_3,\ K_4)$ or $(K_5,\ K_6,\ K_7,\ K_8)$.

1: A simple WD16 related-key observation

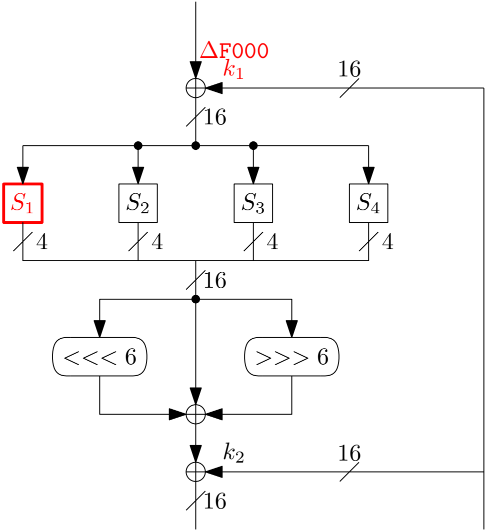
WD16 – High Level View



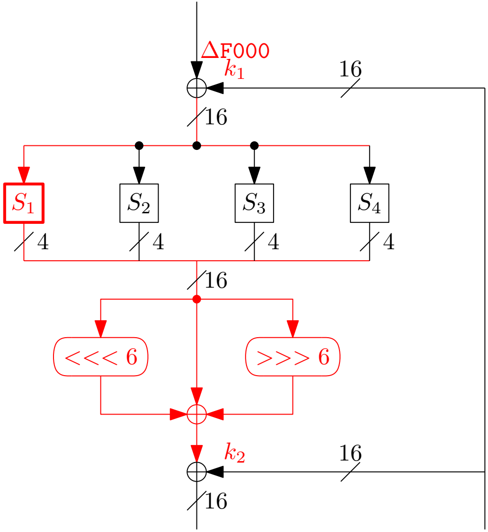
WD16 – Zoom ..



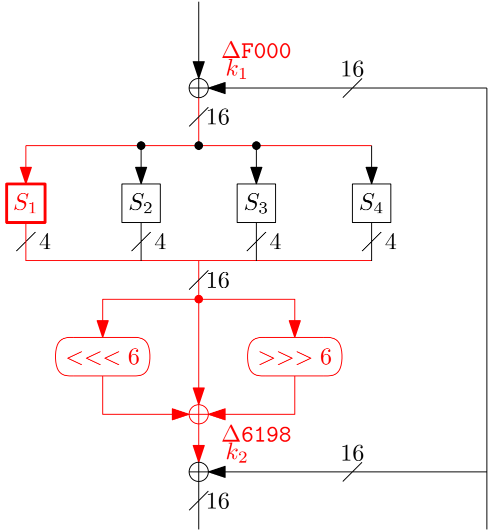
Say there's a related key word $k_1 \oplus k_1' = \mathtt{F000}$



Mixed into a 16-bit difference.. you guessed it



Cancels it out when $k_2 \oplus k_2' = 6198$ with $p = 1/4$.



Observation 1

WD16 has 64-bit related keys that (with $p = 1/4$) produce equivalent output for any given input word !

- - - - -

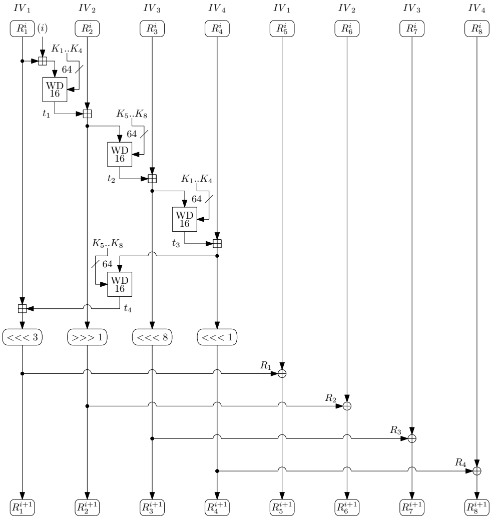
Note that for such related keys there are also *unequal* input word pairs that produce equivalent output with a significant probability.

These observations of WD16 allow us to construct an effective attack – strengthening WD16 appears to make these attacks unfeasible.

(The FSE 2010 attack on Hummingbird-1 would have worked on any WD16 function.)

2: Observations on the Hummingbird-2 structure

4 init rounds turn the 64-bit IV into a 128-bit state



Observation 2

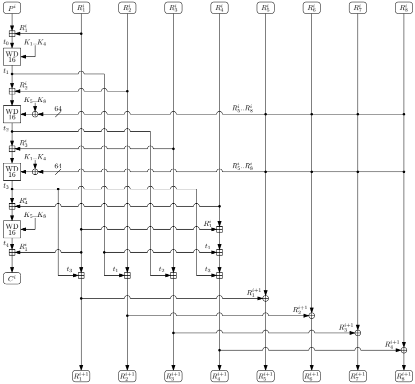
Stated as: *"For each key K, there is a family of 432 related keys K' that yield the same state R after four initialization rounds with probability $P = 2^{-16}$ over all IV values."*

In other words: A state collision for these related keys is really easy to find. The number $432 = 6 \times 72$ is simply the total number of $p = 1/4$ key relations for full 128-bit keys.

Birthday implication: Since the number of usable relations (XOR differences) is large, the set of randomly keyed "encryptors" such as RFID tokens required to find a related pair is significantly smaller than would generally be expected.

Now think about "export grade" instances...

HB2 encrypts data one 16-bit word at a time



Observation 3: If the state is undisturbed, $(1/4)^2 = 1/16$ probability of matching ciphertexts with these related keys!

3: A key recovery method

Attack model

We have two “black box” encryption / decryption oracles, one with key K and an another with key K'.

We arbitrarily pick one of the easier relations for sake of presentation:

$$K \oplus K' = (\texttt{F000 6198 0000 0000 0000 0000 0000 0000}).$$

We are allowed to make a reasonable number of chosen plaintext / ciphertext / IV queries to these black boxes. The goal is to try to figure out K.

I should mention that I’ve fully implemented this attack. There has been some incorrect attacks on eprint, now withdrawn.

Find a state collision

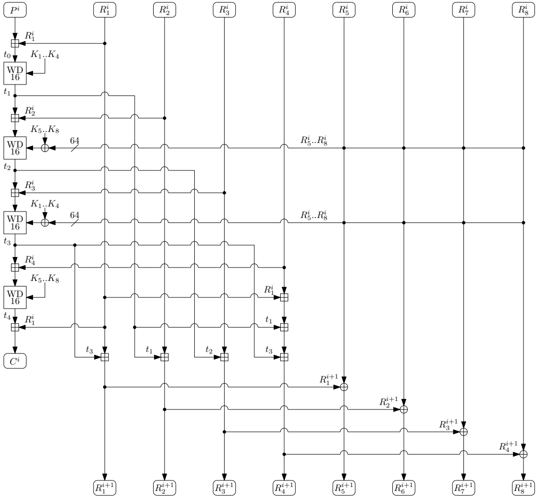
First we want to find an IV value that produces matching state R after the four-round initialization procedure for both K and K'

As shown by Observation 2, we can brute force such a collision with 2^{16} effort.

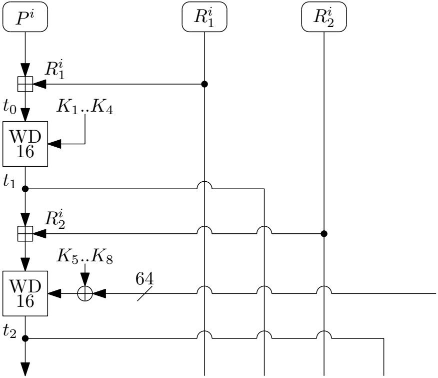
Detection of a matching state can be made by trial encryptions as shown by Observation 3.

The attack requires only a single IV value..

Remember the encryption routine..



Zoom to upper left corner: R_1^i recovery.



We then attack R_1^i, the first word of the internal state in the encryption stage. This is done by analyzing carry overflow in the very first addition (Section 3.3).

Lots of bit twiddling trickery required..

Table: *(No 2 in the paper)* High nibbles of intermediate values $N = ((P^i \boxplus R_1^i) \oplus K_1)) \gg 12$ and $N' = ((P'^i \boxplus R_1^i) \oplus K_1') \gg 12$ in WD16 that will provide a collision. These are the pairs for which $S_1(N) \oplus S_1(N' \oplus \texttt{0xF}) = \texttt{0x6}$. Note that in the diagonal there are four entries as expected; if $N = N'$ there is a 1/4 probability of a collision.

$N \backslash N'$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-
1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-
4	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	F
6	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	C	-	-	-
8	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-
C	-	-	-	-	-	-	-	-	-	-	-	B	-	-	-	-
D	-	-	-	-	-	-	-	-	-	-	-	-	-	D	-	-
E	-	-	-	-	-	-	-	-	-	-	-	-	-	-	E	-
F	-	-	-	-	-	-	-	-	-	9	-	-	-	-	-	-

Armed with R_1^i, we have a 2^{64} attack

We do all kinds of queries and derive more quantities..

$$t_3^i = R_1^{i+1} \boxminus R_1^i.$$

$$t_4^i = C^i \boxminus R_1^i.$$

$$t_3^i \boxplus R_4^i = t_3^{i+1} \boxplus R_4^{i+1}.$$

$$R_4^{i+1} = R_4^i \boxplus R_1^i \boxplus t_3^i \boxplus t_1^i$$

$$t_1^i = \boxminus R_1^i \boxminus t_3^{i+1}.$$

In the end we have sufficient information to brute force the first half of the key without having to worry about the second:

$$t_1^i = \mathrm{WD16}(t_0^i, K_1, K_2, K_3, K_4).$$

Conclusions

Complexity of related-key attack

I turned the search for the first half of the key into a time-memory trade-off. This shrunk the complexity for finding the first 64 key bits (only) to around 2^{36}.

However we also need to know the second half. I haven't found a trade-off for this half; 2^{64} ops are required.

Since the latter half dominates $2^{36} \ll 2^{64}$, the overall complexity of attack against a random 128-bit key K is 2^{64}.

I wouldn't be very surprised if someone found a $2^{\approx 32}$ attack against some specific key relation even in a 2-key attack.

Hummingbird-2ν

The appendix of the paper has a description of an experimental S-Boxless variant. Hummingbird-2ν replaces the WD16 function with $c = \chi_\nu(p, k_1, k_2, k_3, k_4)$, which is based on χ functions that we have grown to respect while doing cryptanalysis on KECCAK.

Everything else is exactly as in Hummingbird-2 (this was a design restriction to this particular variant).

The basic building blocks of χ_ν are the two involutions

$$f(x) = \big((x \lll 2) \wedge \neg(x \lll 1) \wedge (x \ggg 1)\big) \oplus x$$
$$g(x) = \big(\neg x \wedge (x \lll 4) \wedge \neg(x \lll 12)\big) \oplus (x \lll 8)$$

Check it out and tell us what you find.

Thank You...

**“Hummingbirds are like regular birds.
They just can’t remember the lyrics.”**