Truncated Differentials and Skipjack

Halil İbrahim Kaplan

TÜBİTAK BİLGEM KAL

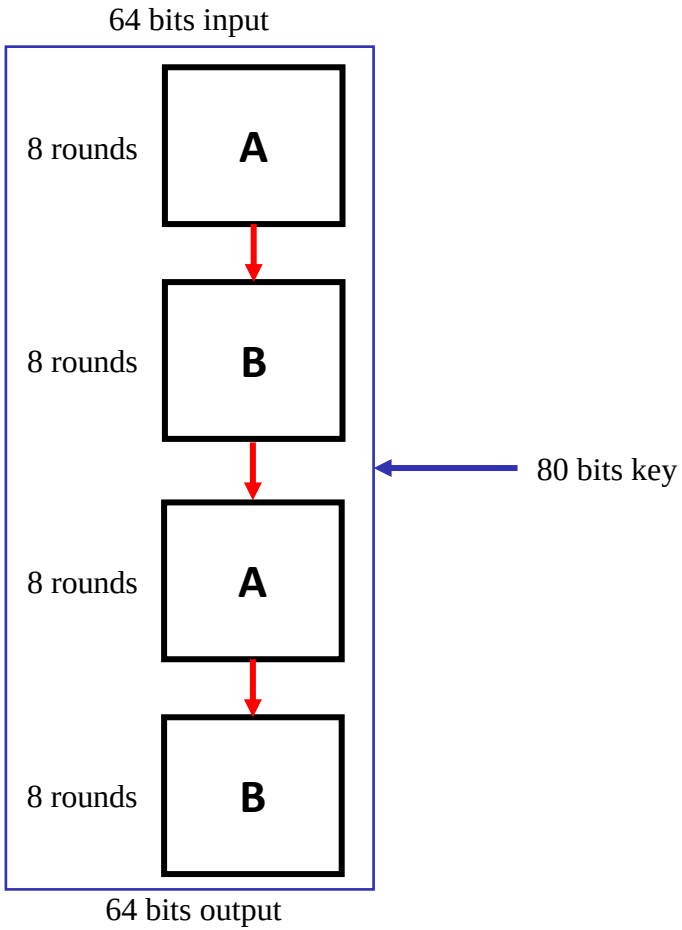
2021

Overview

- Description of Skipjack
- Truncated differentials of Skipjack
- Attacks using truncated differentials

Description of Skipjack

Skipjack

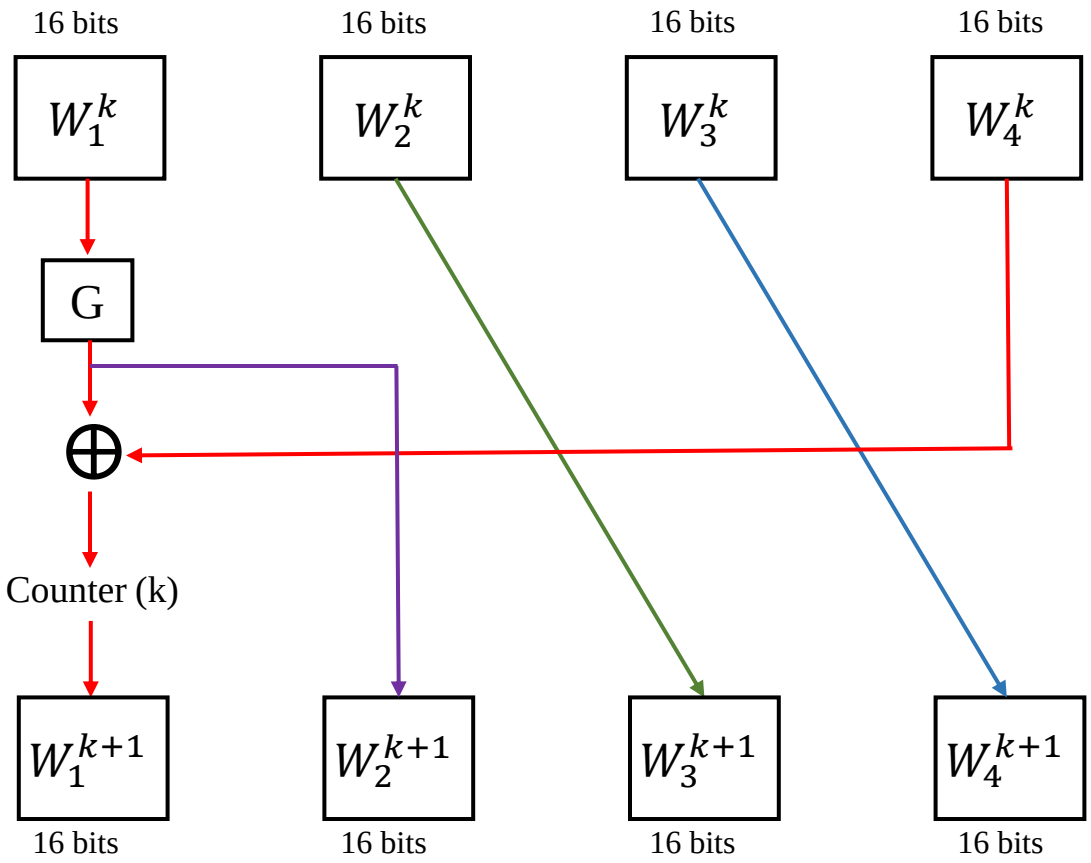


80 bits key partitioning

k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9

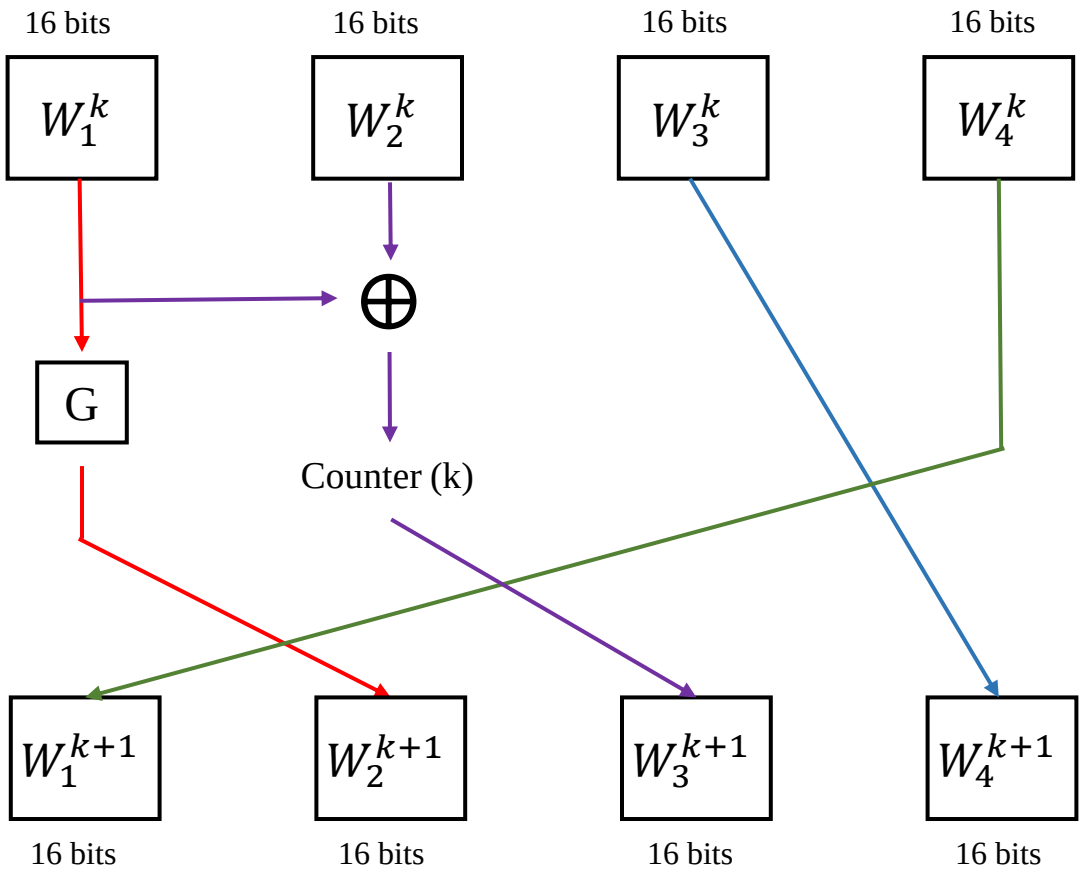
Description of Skipjack

A-Round



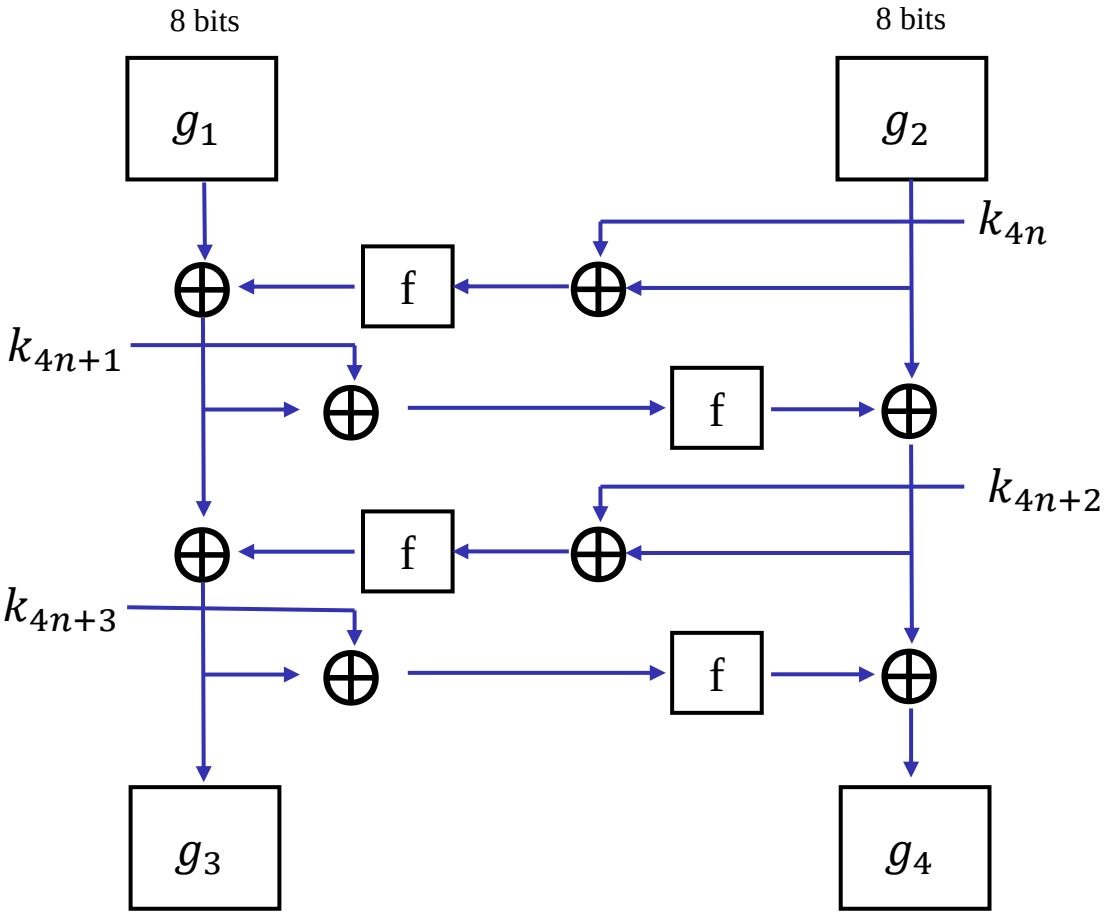
Description of Skipjack

B-Round



Description of Skipjack

G Function



TRUNCATED DIFFERENTIALS OF SKIPJACK

Truncated differentials of Skipjack

- When using truncated differentials, instead of trying to predict the evolution of some difference across an entire block, the cryptanalyst attempts to predict the difference across some fraction of this block.

- With Skipjack it is very natural to consider the difference across the four 16-bit words.

$$(\_ , \_ , \_ , \_)$$

Truncated differentials of Skipjack

- One useful truncated differential characteristic allows us to cover the first 16 rounds of Skipjack :

$$(a, b, 0, c) \xrightarrow{8r_A} (e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0)$$

$$(a, b, 0, c) \xrightarrow{4r_A} (0, d, 0, 0) \longrightarrow \Pr = 2^{-32}$$

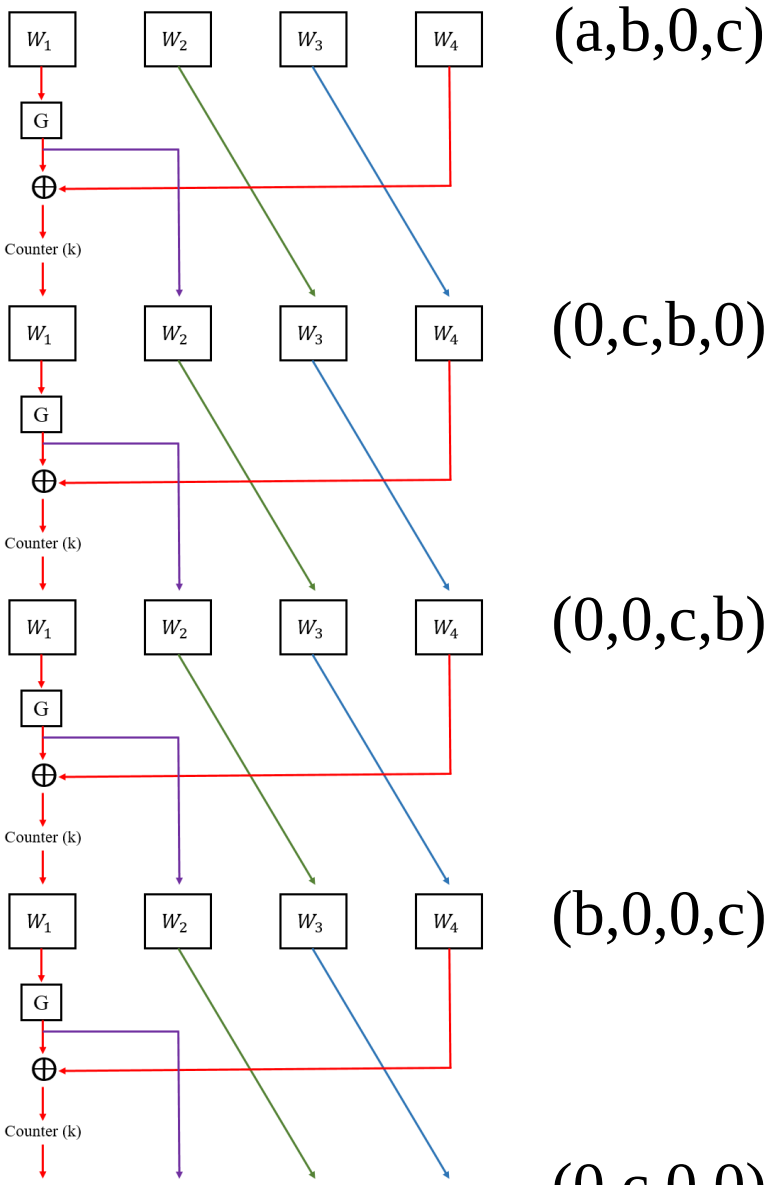
$$(0, d, 0, 0) \xrightarrow{4r_A} (e, e, 0, 0) \longrightarrow \Pr = 1$$

$$(e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0) \longrightarrow \Pr = 1$$

$\longrightarrow$ Total Probability = 2^{-32}

(a,b,…,h denote non-zero 16 bit difference)

$$(a, b, 0, c) \xrightarrow{4r_A} (0, d, 0, 0) \longrightarrow \Pr = 2^{-32}$$



(a,b,0,c)

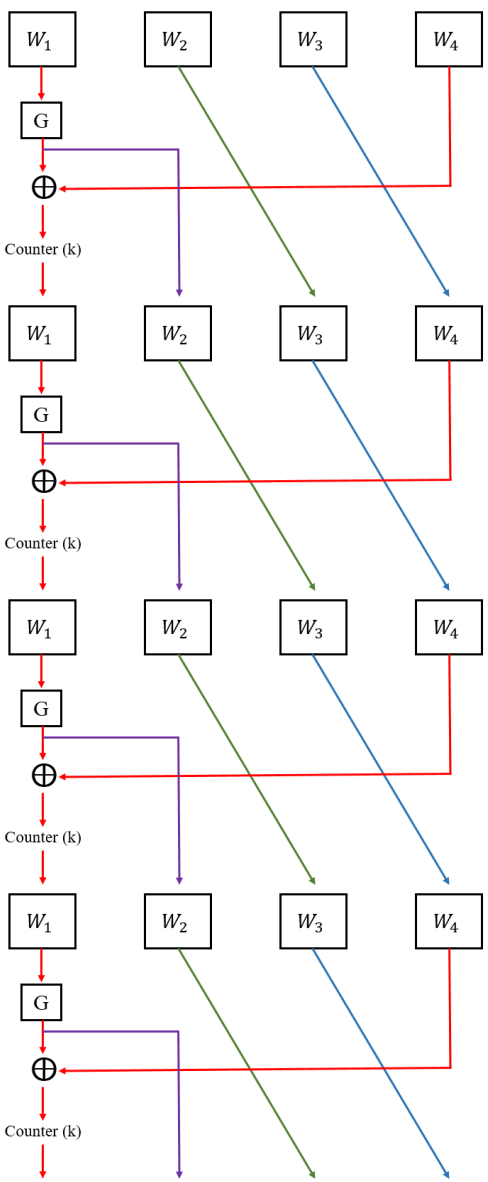
(0,c,b,0)

(0,0,c,b)

(b,0,0,c)

(0,c,0,0)

$$(0, d, 0, 0) \xrightarrow{4r_A} (e, e, 0, 0) \longrightarrow \Pr = 1$$



(0,d,0,0)

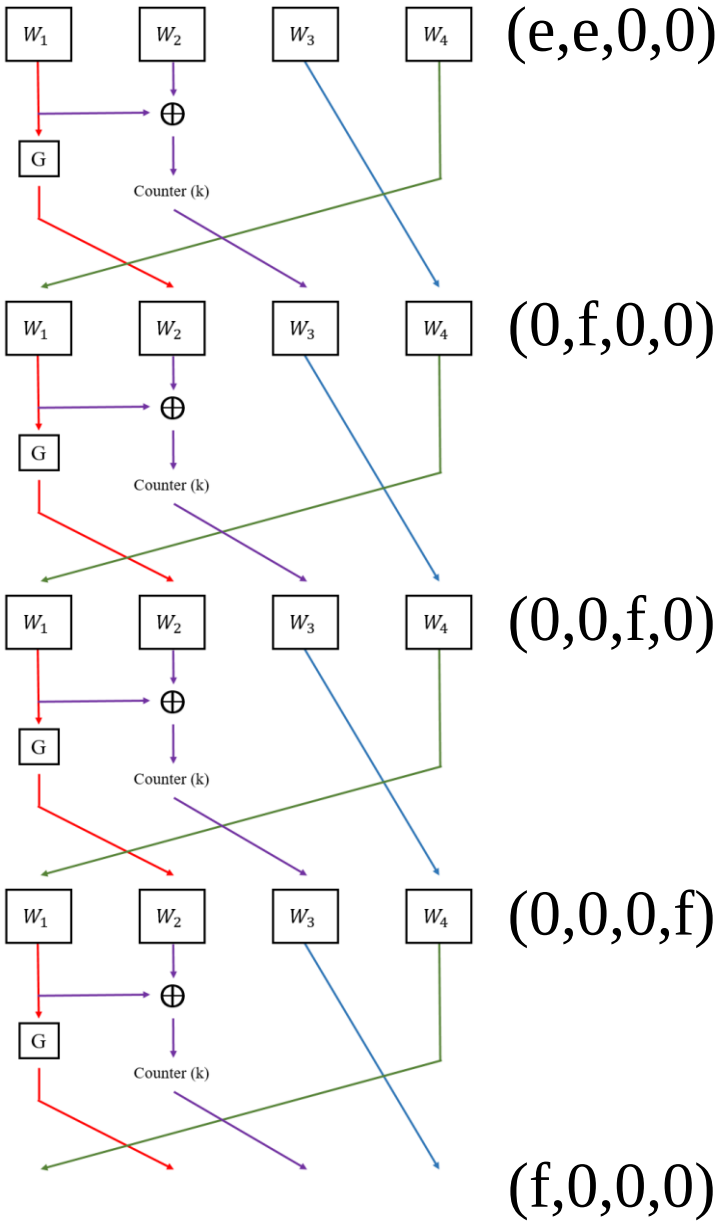
(0,0,d,0)

(0,0,0,d)

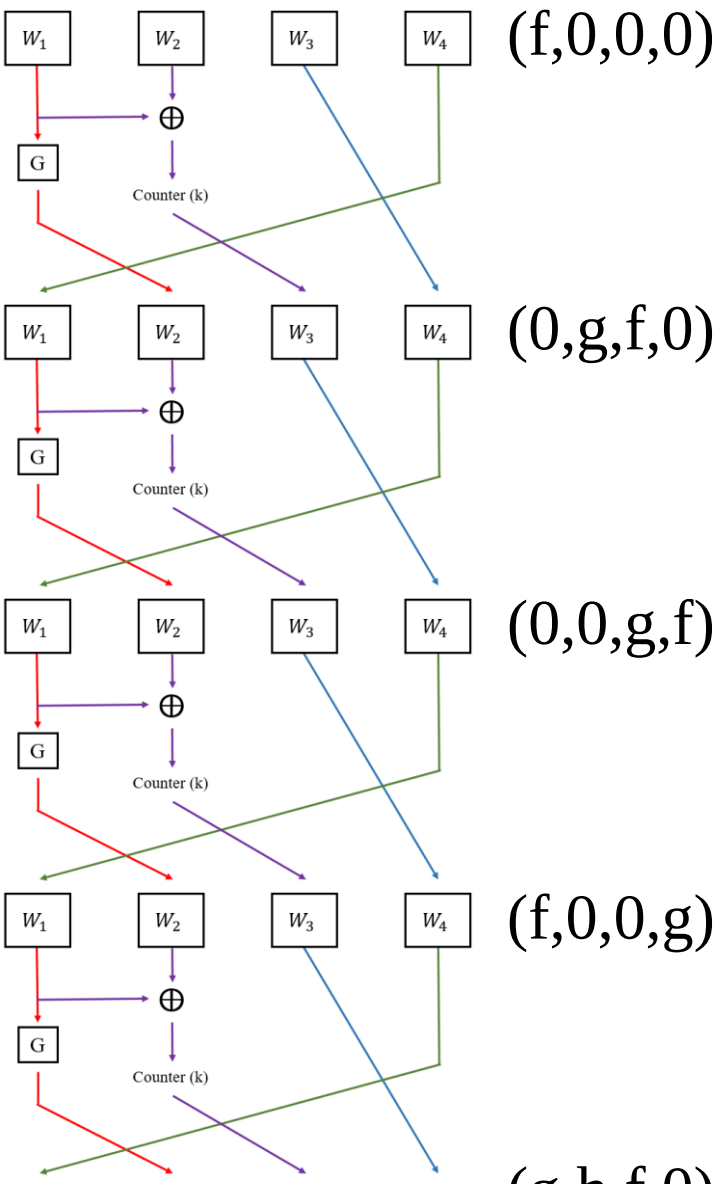
(d,0,0,0)

(e,e,0,0)

$$(e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0) \longrightarrow \Pr = 1$$



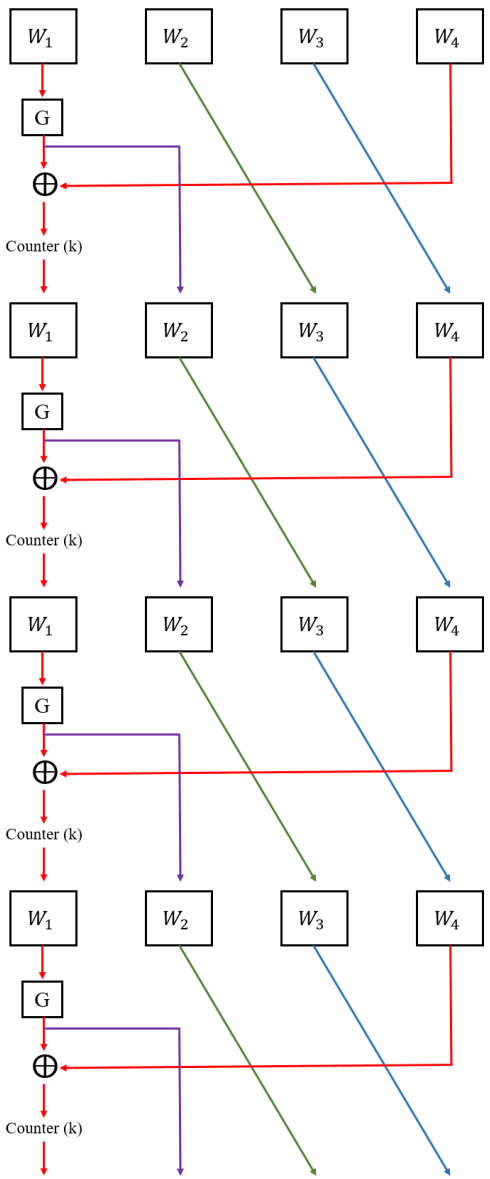
$$(e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0) \longrightarrow \Pr = 1$$



Truncated differentials of Skipjack

The search for truncated differentials:

Round	Difference	Properties
	$(a, b, c, 0)$	a, b, c nonzero
A1:	(d, d, b, c)	b, c, d nonzero
A2:	(f, e, d, b)	e, d, b nonzero, $f \neq e$
A3:	(h, g, e, d)	e, d nonzero, $h \neq g$
A4:	(j, i, g, e)	$e \neq 0$, $(i, g) \neq (0, 0)$, $j \neq i$
A5:	(l, k, i, g)	$(k, i) \neq (0, 0)$, $(i, g) \neq (0, 0)$, $l \neq k$
A6:	(n, m, k, i)	$(m, k) \neq (0, 0)$
A7:	(q, p, m, k)	$(m, k) \neq (0, 0)$
A8:	(s, r, p, m)	$s = k \oplus r$
B1:	(m, t, k, p)	$(m, k) \neq (0, 0)$
B2:	(p, u, α, k)	$\alpha = m \oplus t$
B3:	(k, v, β, α)	$\beta = p \oplus u$
B4:	$(\alpha, w, \gamma, \beta)$	$\gamma = k \oplus v$



(a,b,c,0)

(d,d,b,c)

(f,e,d,b)

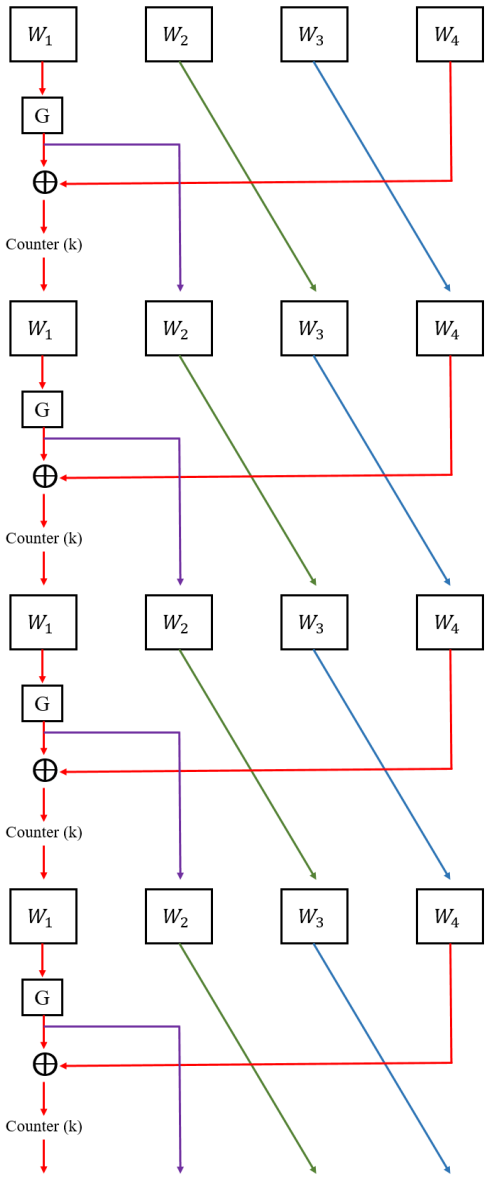
(h,g,e,d)

(j,i,g,e)

Truncated differentials of Skipjack

The search for truncated differentials:

Round	Difference	Properties
	$(a, b, c, 0)$	a, b, c nonzero
A1:	(d, d, b, c)	b, c, d nonzero
A2:	(f, e, d, b)	e, d, b nonzero, $f \neq e$
A3:	(h, g, e, d)	e, d nonzero, $h \neq g$
A4:	(j, i, g, e)	$e \neq 0$, $(i, g) \neq (0, 0)$, $j \neq i$
A5:	(l, k, i, g)	$(k, i) \neq (0, 0)$, $(i, g) \neq (0, 0)$, $l \neq k$
A6:	(n, m, k, i)	$(m, k) \neq (0, 0)$
A7:	(q, p, m, k)	$(m, k) \neq (0, 0)$
A8:	(s, r, p, m)	$s = k \oplus r$
B1:	(m, t, k, p)	$(m, k) \neq (0, 0)$
B2:	(p, u, α, k)	$\alpha = m \oplus t$
B3:	(k, v, β, α)	$\beta = p \oplus u$
B4:	$(\alpha, w, \gamma, \beta)$	$\gamma = k \oplus v$



(j,i,g,e)

(l,k,i,g)

(n,m,k,i)

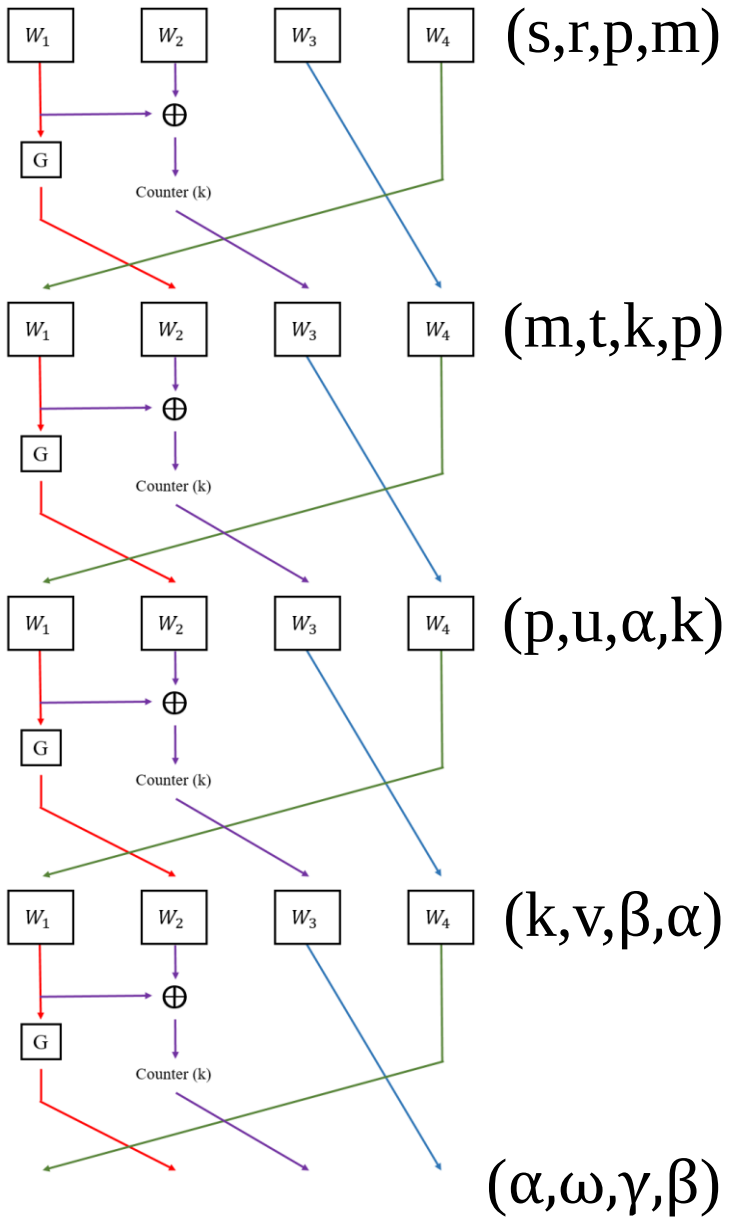
(q,p,m,k)

(s,r,p,m)

Truncated differentials of Skipjack

The search for truncated differentials:

Round	Difference	Properties
	$(a, b, c, 0)$	a, b, c nonzero
A1:	(d, d, b, c)	b, c, d nonzero
A2:	(f, e, d, b)	e, d, b nonzero, $f \neq e$
A3:	(h, g, e, d)	e, d nonzero, $h \neq g$
A4:	(j, i, g, e)	$e \neq 0$, $(i, g) \neq (0, 0)$, $j \neq i$
A5:	(l, k, i, g)	$(k, i) \neq (0, 0)$, $(i, g) \neq (0, 0)$, $l \neq k$
A6:	(n, m, k, i)	$(m, k) \neq (0, 0)$
A7:	(q, p, m, k)	$(m, k) \neq (0, 0)$
A8:	(s, r, p, m)	$s = k \oplus r$
B1:	(m, t, k, p)	$(m, k) \neq (0, 0)$
B2:	(p, u, α, k)	$\alpha = m \oplus t$
B3:	(k, v, β, α)	$\beta = p \oplus u$
B4:	$(\alpha, w, \gamma, \beta)$	$\gamma = k \oplus v$



Truncated differentials of Skipjack

Long truncated differentials:

Truncated differential gives nontrivial information about the ciphertexts after 17 rounds of encryption:

$$(0, a, 0, 0) \xrightarrow{4r_A} (b, b, 0, 0) \xrightarrow{8r_B} (c, d, e, 0) \xrightarrow{5r_A} (f, g, h, i)$$

where f ≠ g and h and i can take any values

We can **distinguish** reduced Skipjack from randomly chosen permutation with following steps:

Truncated differentials of Skipjack

Long truncated differentials:

I. Take $2^{8.5}$ plaintexts with equal values in the first , third and fourth words.

II. Form $\frac{2^{8.5}*(2^{8.5}-1)}{2} \approx 2^{16}$ pairs from plaintexts .

III. Encrypt all pairs.

IV. XOR first two words difference of ciphertexts pairs.

V. If non of the results equal to 0 , then cipher is Skipjack.
(For a randomly chosen permutation a match is found with probability 2^{16})

Truncated differentials of Skipjack

Important practical details:

1) FILTERING

- After accumulating what might be a vast amount of chosen plaintext-ciphertext pairs in an attack, the cryptanalyst needs to throw away as much erroneous data (pairs that do not follow the differential as intended) as possible.

- This is done by filtering. With the truncated differentials we consider, the structure we use for filtering is the presence of a zero difference in some word.

Truncated differentials of Skipjack

Important practical details:

1) FILTERING

$$(a, b, 0, c) \xrightarrow{8r_A} (e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0)$$

Expected ciphertexts difference = (g, h, f, 0)

which means that only pairs of ciphertexts with equal fourth words will be left after filtering.

The more zeros in the expected output difference $\Longrightarrow$ The greater the number of wrong pairs that can be filtered before starting to extract key material.

Important practical details:

2) COUNTING

Some truncated differentials might initially appear to be useful to the cryptanalyst, it is not always possible to extract key information.

One example is the following truncated differential

$$(0, a, b, 0) \xrightarrow{8r_A} (c, d, e, 0) \xrightarrow{8r_B} (0, f, g, h)$$

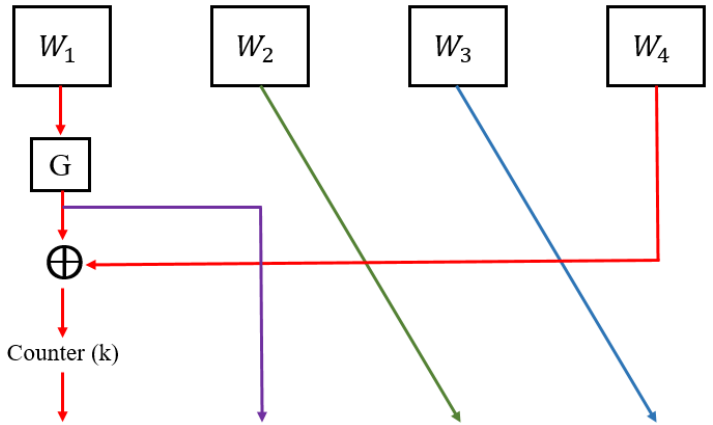
which holds with probability 2^{-32}

Important practical details:

2) COUNTING

When using this differential in an attack it passes over the first round of encryption with probability one and it is not possible to distinguish the correct first-round subkey from the wrong ones.

(Pr = 1)



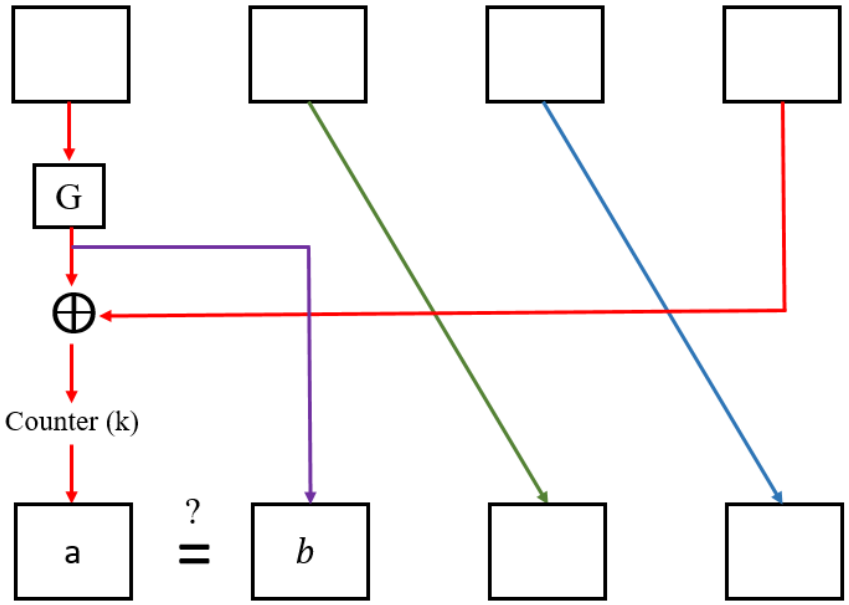
(0,a,b,0)

(0,0,a,b)

Truncated differentials of Skipjack

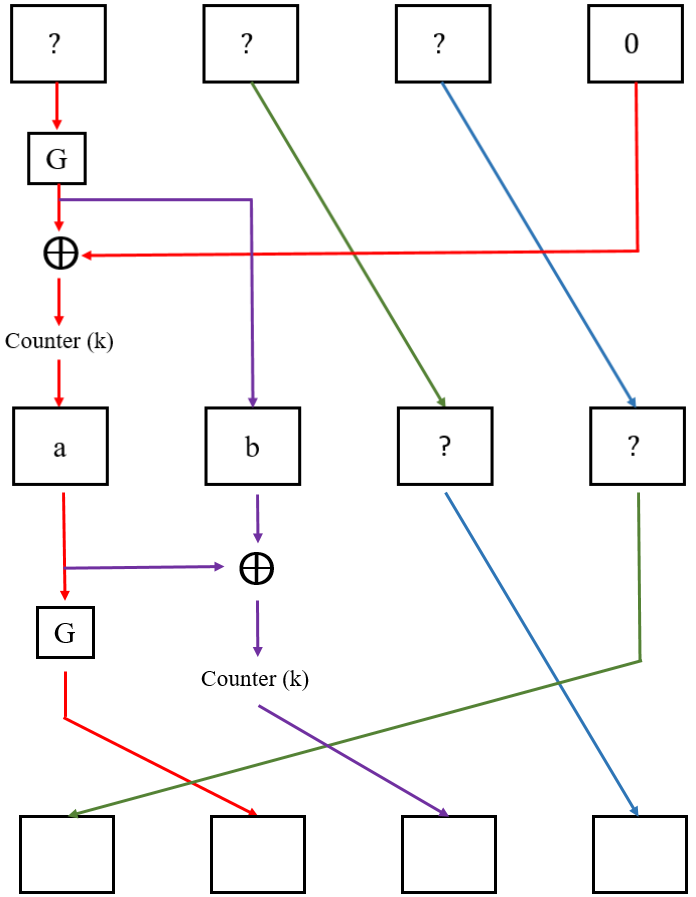
The search for truncated differentials:

- When combining A-rounds with B-rounds as required in Skipjack an extra “rule” is required.
- In the case where an A-round is followed by a B-round and where the output difference of the A-round has nonzero values assigned to the first two words, one needs to know if the difference in the first word is equal to the difference in the second word.



The search for truncated differentials:

- This is of vital importance in the calculation of the probability of the differential in the B-round.
- However it is also easy to incorporate this consideration as a part of the search, since the differences in the two first output words from an A-round will be equal if, and only if, the fourth words of the inputs to the A-round are equal.
- Since no extra “rules” are needed in the transition from a B-round to an A-round, one can find truncated differentials for any number of A- and B-rounds.



Truncated differentials of Skipjack

The search for truncated differentials:

The search revealed several truncated differentials for the full Skipjack with probability 2^{-64}. One example is the following differential where the words $a, \ldots, m$ can take any nonzero values.

$$(0, a, b, c) \xrightarrow{8r_A} (0, 0, 0, d) \xrightarrow{8r_B} (0, e, f, g) \xrightarrow{8r_A} (h, h, i, j) \xrightarrow{8r_B} (0, k, l, m),$$

This differential allows only for a very limited amount of filtering since only the form of the leftmost word of the ciphertext is restricted.

Furthermore, the leftmost word of the plaintext difference in all cases is zero, which means that key material cannot be extracted from analysis of the first round since all possible subkeys are equally likely. Thus, these differentials do not seem to be useful in an attack.

ATTACKS USING TRUNCATED DIFFERENTIALS

Attacks using truncated differentials

The first sixteen rounds :

$$(a, b, 0, c) \xrightarrow{8r_A} (e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0) \qquad (\Pr = 2^{-32})$$

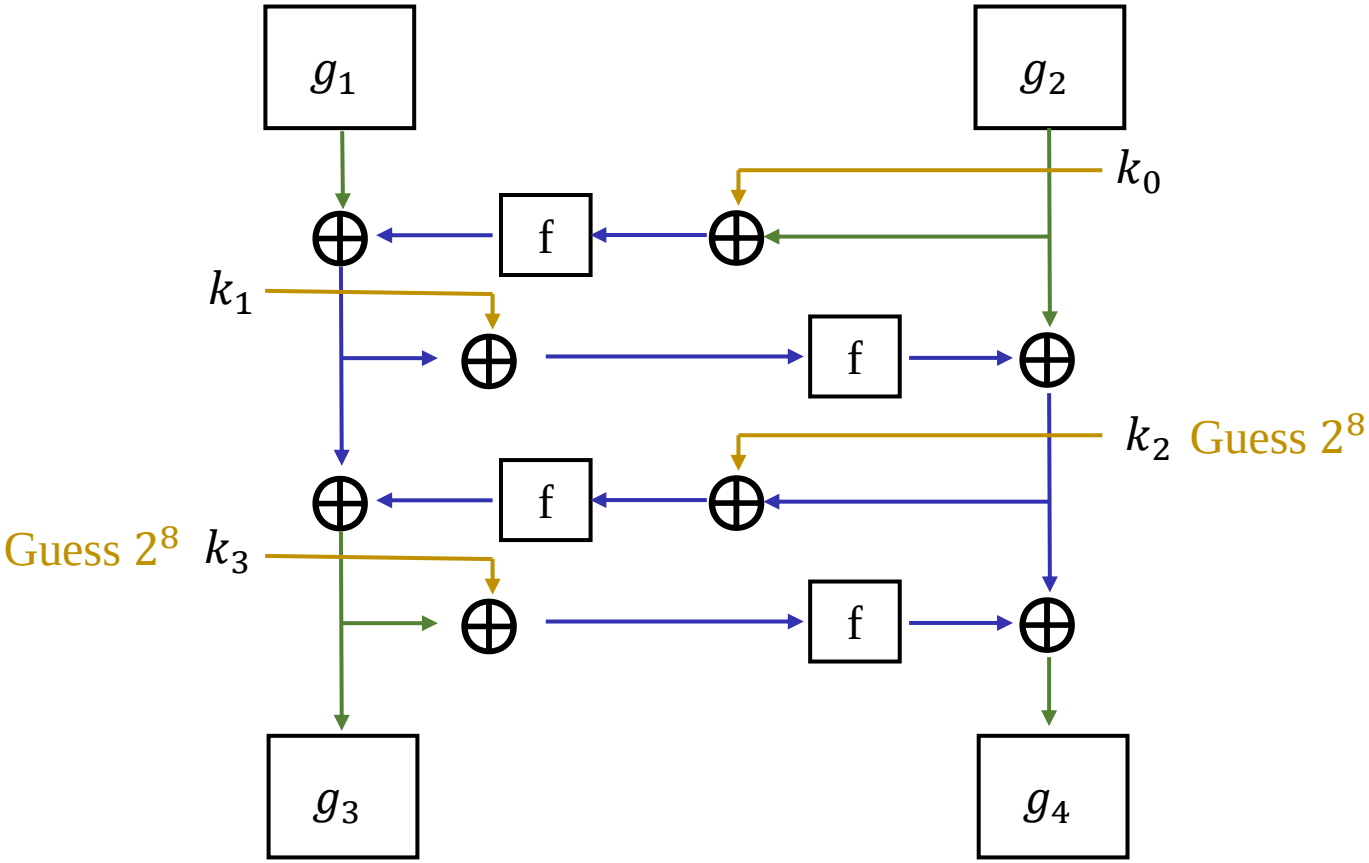
- Choose 2^{17} plaintexts where the third words are fixed and obtain the corresponding ciphertexts.
- From these plaintexts one can form about $\frac{2^{17}*(2^{17}-1)}{2} \approx 2^{33}$ pairs with the desired starting difference.
- With a high probability,$2^{33} * 2^{-32} = 2$ right pairs will follow the truncated differential.
- Observing that the rightmost word has zero difference, we can immediately filter out many wrong pairs before moving on to the next stage of the analysis with $2^{33} * 2^{-16} = 2^{17}$ pairs of data.

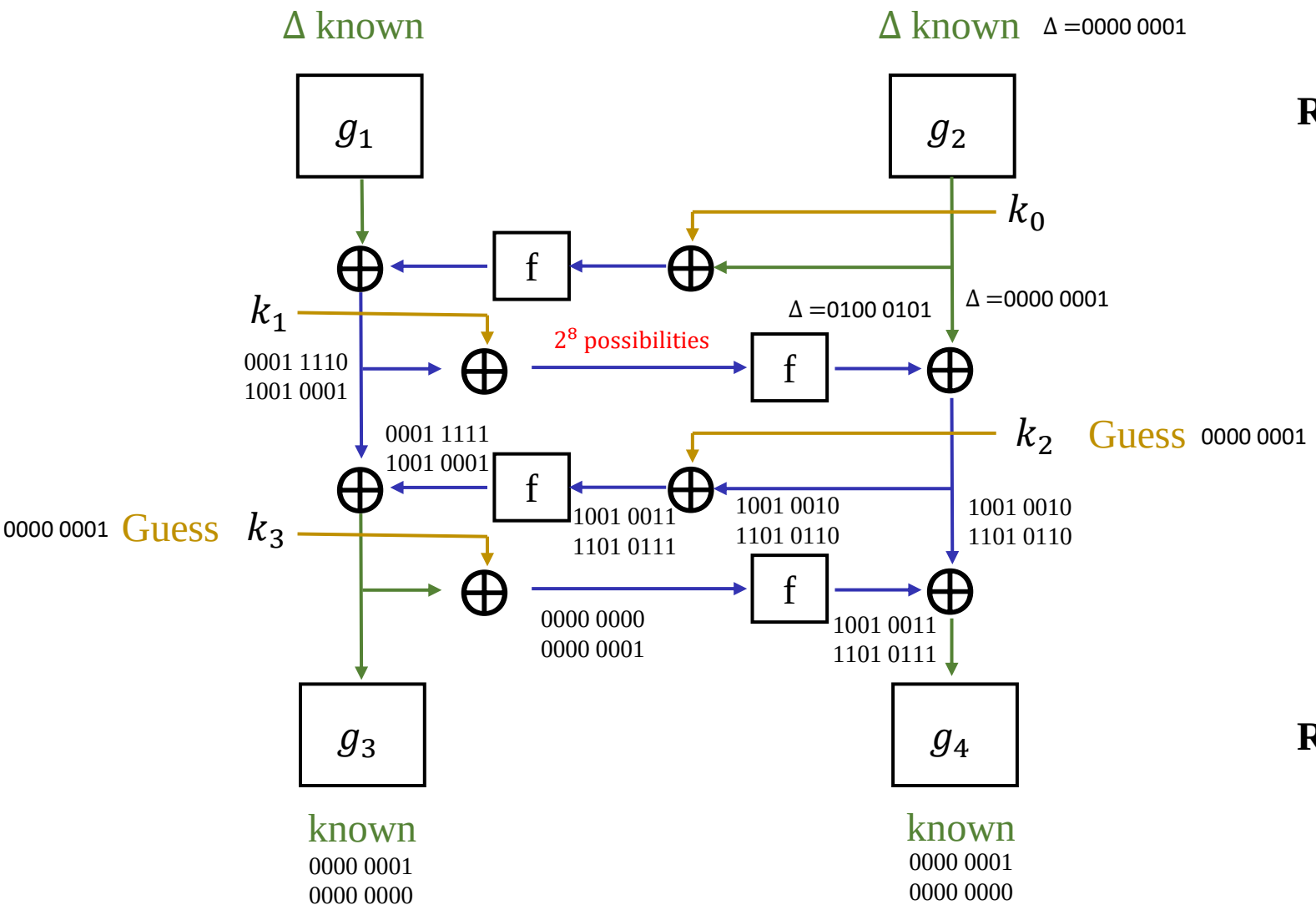
The first sixteen rounds :

$$(a, b, 0, c) \xrightarrow{8r_A} (e, e, 0, 0) \xrightarrow{8r_B} (g, h, f, 0) \qquad (\Pr = 2^{-32})$$

- For each surviving pair(2^{17}) we check which keys in the last round result in a difference that follows the differential after decryption by one round. About 2^{16} values for the 32-bit key will be suggested by this test for each pair.
- Similarly, for each surviving pair we check which keys(2^{16}) result in differences that follow the differential after encryption by one round.
- It is possible to find these 2^{17} suggested values with offline work comparable to about 2^{17} G-box computations.

The trick is to use a precomputed table which, given differences y, z, allows us to find input x such that F (x) ⊕ F (x⊕y) = z with one table lookup. We guess k_2, k_3, decrypt up by two layers of the G-box, and use the precomputed table to recover k_0, k_1, noting that z is known from the G-box input difference and y is known as a result of decrypting up two layers.




Δ known
Δ known
Δ =0000 0001
g_1
g_2
Round 15
k_0
f
k_1
Δ =0100 0101
Δ =0000 0001
2^8 possibilities
0001 1110
1001 0001
f
0001 1111
1001 0001
k_2
Guess
0000 0001
f
1001 0011
1101 0111
1001 0010
1101 0110
1001 0010
1101 0110
0000 0001
Guess
k_3
f
0000 0000
0000 0001
1001 0011
1101 0111
g_3
g_4
Round 16
known
0000 0001
0000 0000
known
0000 0001
0000 0000

Attacks using truncated differentials

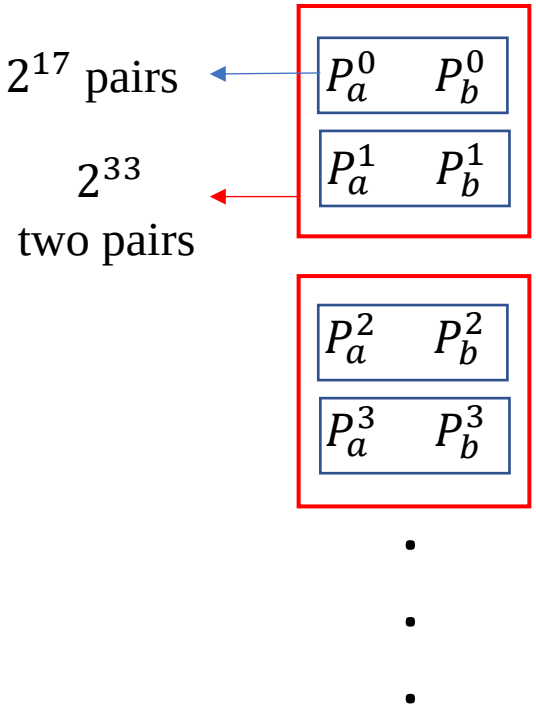
The first sixteen rounds(Approach 1) :

- We have $2^{32} * 2^{17} = 2^{49}$ key suggestion for 64 bits.
- Count on those 64 key bits and look for a counter whose value exceeds one.
- By the birthday paradox, only about $\frac{2^{2*49}}{2^{64+1}} = 2^{33}$ wrong key values would remain, and each suggested value for the eight key bytes could be tested by exhaustive search over the remaining two unknown bytes.
- We could recover the key with about $(2^{17} \times 2^{32}) + (2^{33} \times 2^{16}) = 2^{50}$ work

BUT the need for 2^{64} counters equal to 2^{64} space for the array in the memory makes this approach totally impractical.

The first sixteen rounds (Approach 2) :

- Examine the plaintext pairs two at a time.
- For each two plaintext pairs use the calculation of G in the 16th round to recover a list of possible values for the subkey. On average we expect to find about **one** possible subkey value.
- Similarly, the G computation in the first round is used to recover a possible value for another four key bytes.
- The suggested value for these eight key bytes can then be tested by exhaustive search over the remaining two unknown key bytes.



The first sixteen rounds (Approach 2) :

- There are about $\frac{2^{17}*(2^{17}-1)}{2} \approx 2^{33}$ ways to choose two plaintext pairs.
- Each one suggests 1 unique key. So we have 2^{33} key suggestion for 64 key bits.
- 2 bytes of master key remains unknown.
- To recover full key, we make exhaustive search for $2^{33} * 2^{16} = 2^{49}$ keys.
- We made $2^{33} * 2^{16} = 2^{49}$ G computation for finding 2^{33} key suggestion.

2^{49} G computation $= 2^{45}$ 16 round skipjack encryption

TOTAL WORK = $2^{45} + 2^{49} \approx 2^{49}$ (16 round skipjack encryption)

Attacks using truncated differentials

The first sixteen rounds :

- Complexity could be reduced using alternative techniques if more texts are available.

The first sixteen rounds (Approach 3) :

- We can form 2^{37} plaintext pairs from 2^{19} chosen plaintexts, and count on the last-round subkey.
- About $2^{37-16} = 2^{21}$ pairs survive filtering.
- Construct an array of 2^{32} which consist of all possible values of 32 bits.
- Each surviving pair(2^{21}) suggest 2^{16} values for 32 bit key.
- Increase the counter of array for corresponding value in 2^{16} suggestion.

The first sixteen rounds (Approach 3) :

- Incrementing the counters requires work equivalent to about $2^{16} * 2^{21} = 2^{37}$ computations of G.

- The right counter will be suggested about $2^5 + 2^5$ times, whereas the wrong counter will be suggested 2^5 times on average.

- Only about 32 wrong counters will exceed their mean value by $2^{2.5} \approx 5.66$ standard deviations or more, so only about 33 values for the last-round subkey will survive.

The first sixteen rounds (Approach 3) :

- Similarly, we can find 33 possibilities for the first-round subkey with another 2^{37} computations of G.
- So time equivalent to $\frac{2^{37}+2^{37}}{16} = 2^{34}$ 16 round skipjack encryptions, we can recover 33^2 possibilities for 64 key bits.
- Finally, those 33^2 possibilities can be tested with an exhaustive search over the remaining two unknown key bytes.

COMPLEXITY = $2^{34} + 33^2 \times 2^{16} \approx 2^{34}$ 16 round skipjack encryptions

with 2^{19} chosen plaintexts and 2^{32} space.

The middle sixteen rounds :

We will use the differential:

$$(0, a, 0, 0) \xrightarrow{8r_B} (0, b, c, d) \xrightarrow{4r_A} (h, h, f, g)$$

Probability = 1

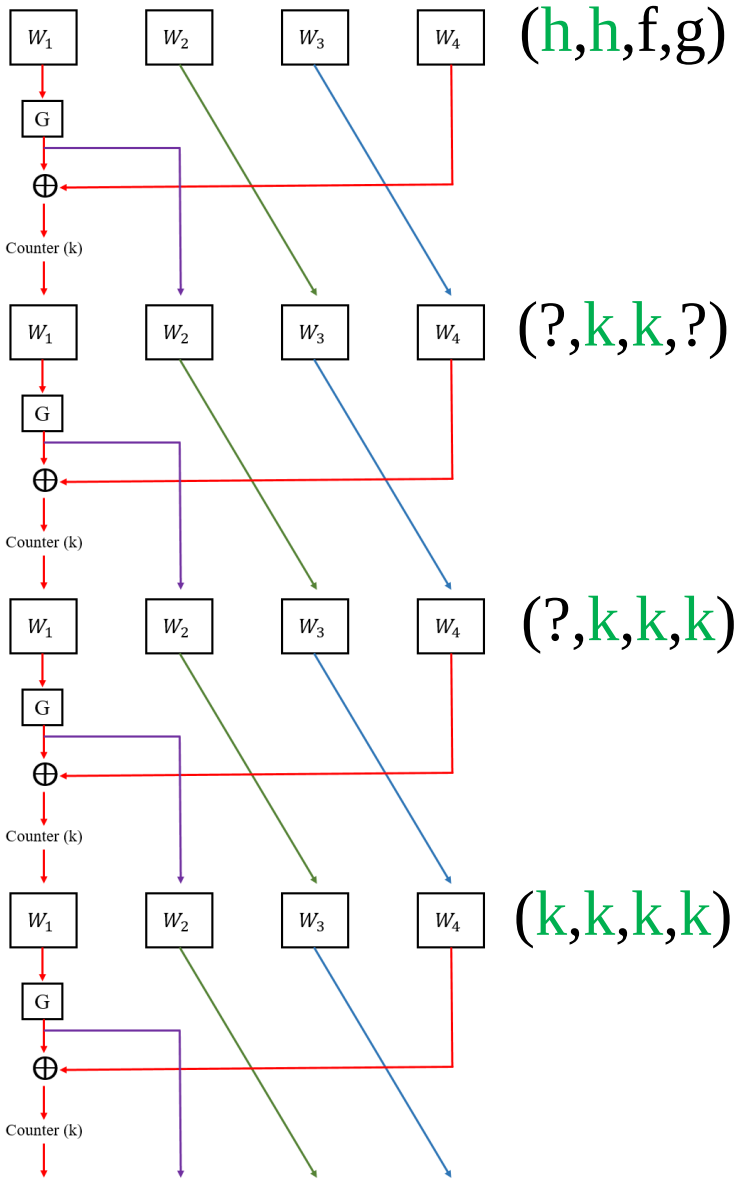
Obtain n independent pairs with the first, third, and fourth words of the input fixed.

The middle sixteen rounds :

1. Guess k_0, k_1, k_2, k_3. For each pair, peel off the 16^{th} round to learn the value of h that this key guess suggests.

Subkey used	Round
0,1,2,3	1,6,11,16
4,5,6,7	2,7,12
8,9,0,1	3,8,13
2,3,4,5	4,9,14
6,7,8,9	5,10,15

k = known value



Round 13

(h,h,f,g)

Round 14

(?,k,k,?)

Round 15

(?,k,k,k)

Round 16

(k,k,k,k)

KNOWN CIPHERTEXT

Learn the value of h that this key guess suggests

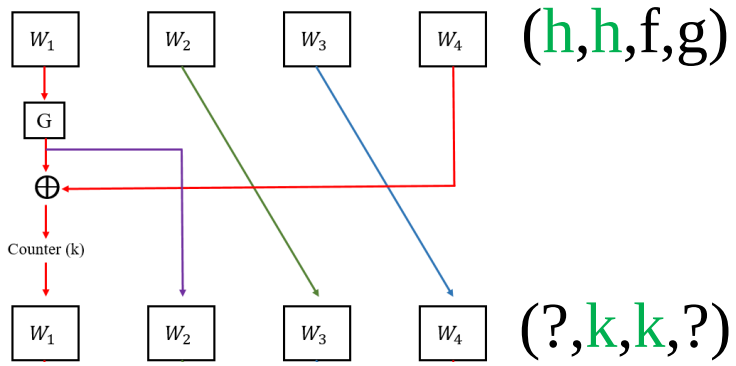
Guess k_0, k_1, k_2, k_3. For each pair, peel off the 16^{th} round

The middle sixteen rounds :

2. Recover k_9. A naive approach is to simply guess k_9; reversing three layers of the computation of G in round 13 (using k_0, k_1, k_9) will give the right half (low byte) of h in each pair if our guess for $k_{9\ldots3}$ was correct. This gives a filtering condition on 8n bits.

The work factor of this phase will be about 2^{32}, and we expect 2^{40-8n} values of $k_{9\ldots3}$to remain.

Round 13



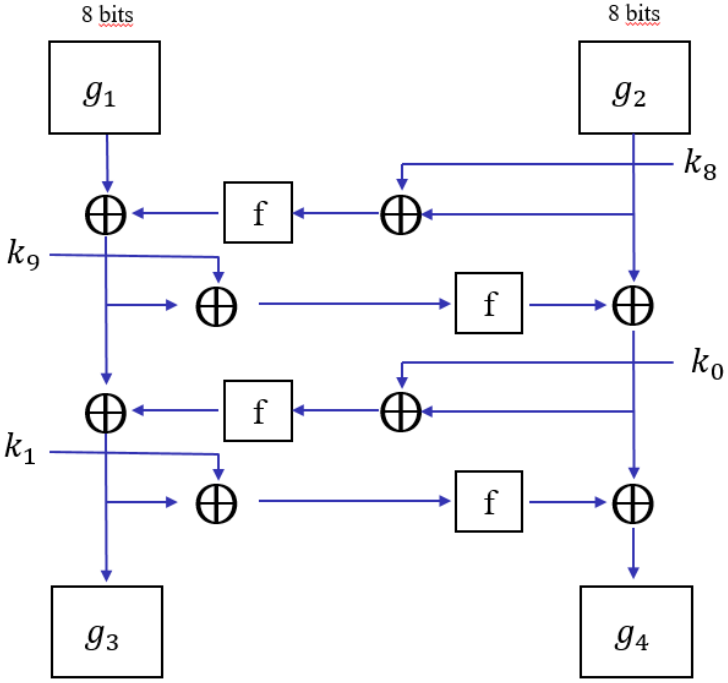
Key used	Round
0,1,2,3	1,6,11,16
4,5,6,7	2,7,12
8,9,0,1	3,8,13
2,3,4,5	4,9,14
6,7,8,9	5,10,15

The middle sixteen rounds :

2. Recover k_9. A naive approach is to simply guess k_9; reversing three layers of the computation of G in round 13 (using k_0, k_1, k_9) will give the right half (low byte) of h in each pair if our guess for $k_{9\ldots3}$ was correct. This gives a filtering condition on 8n bits.

The work factor of this phase will be about 2^{32}, and we expect 2^{40-8n} values of $k_{9\ldots3}$to remain.

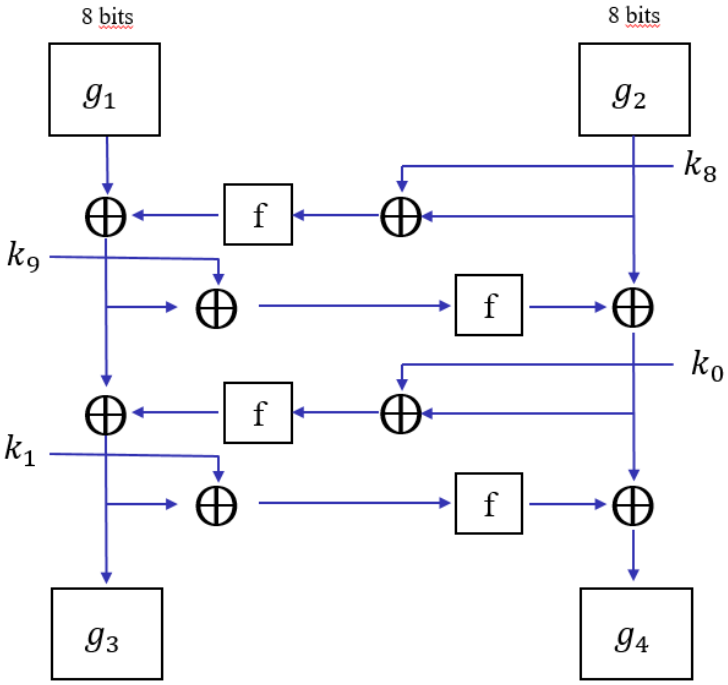
G in round 13



The middle sixteen rounds :

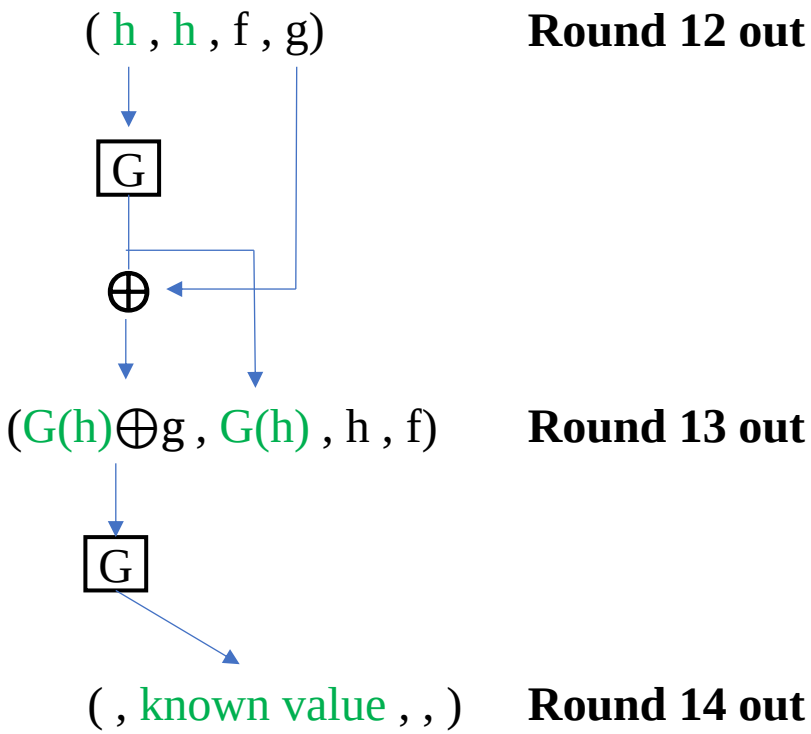
3. Recover k_8. We can use the same technique as in the second phase, this time reversing a fourth layer of the G transformation in round 13. We predict that the exclusive-or of the values obtained should be the same as the left half (high byte) of h in each pair if our guess was correct. This gives a filtering condition on 8n bits, so 2^{48-16n}possibilities for $k_{8\ldots3}$ will remain. With proper implementation, this phase takes about 2^{40-8n}work.

G in round 13



The middle sixteen rounds :

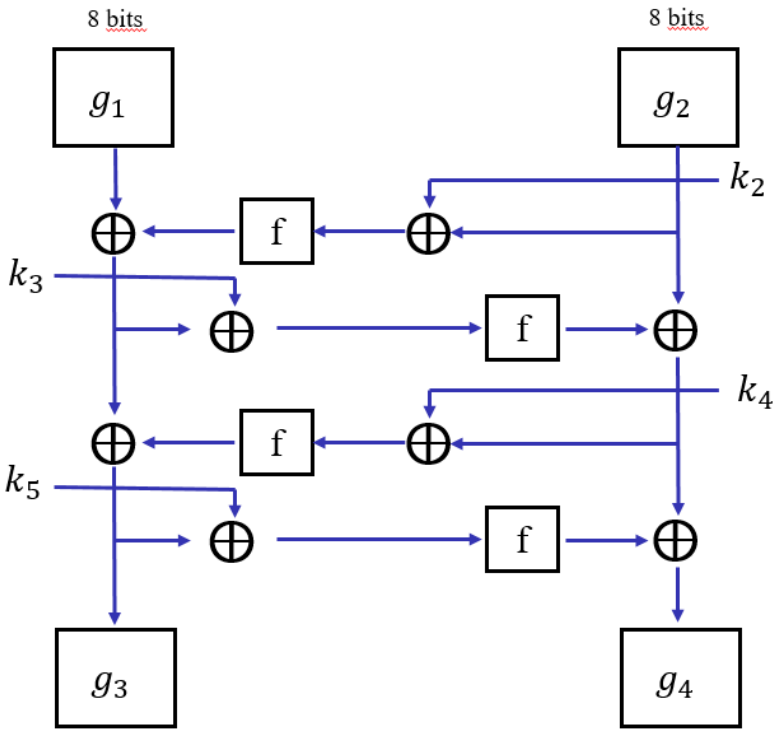
4. Guess k_4 and k_5. Now decrypt through the computation of G in round 14 (using $k_2, \ldots, k_5$) to learn g. This will suggest 2^{64-16n} values for $k_8, \ldots, k_5$, with a similar work factor.



The middle sixteen rounds :

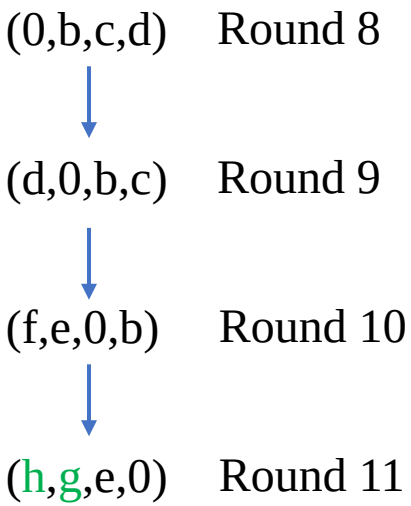
4. Guess k_4 and k_5. Now decrypt through the computation of G in round 14 (using $k_2, \ldots, k_5$) to learn g. This will suggest 2^{64-16n} values for $k_8, \ldots, k_5$, with a similar work factor.

G in round 14



The middle sixteen rounds :

5. Recover k_7. The outputs of the G transformation in round 10 are now known, and the inputs have known difference a. We can decrypt two layers of G in round 10 (using k_8 and k_9), and then derive k_7 (which is used in the next layer) with a precomputed lookup table, as above. With proper implementation, this phase takes 2^{64-16n} work, and we expect that about 2^{72-24n} possibilities for $k_7, \ldots, k_5$ will remain at the conclusion of this phase.



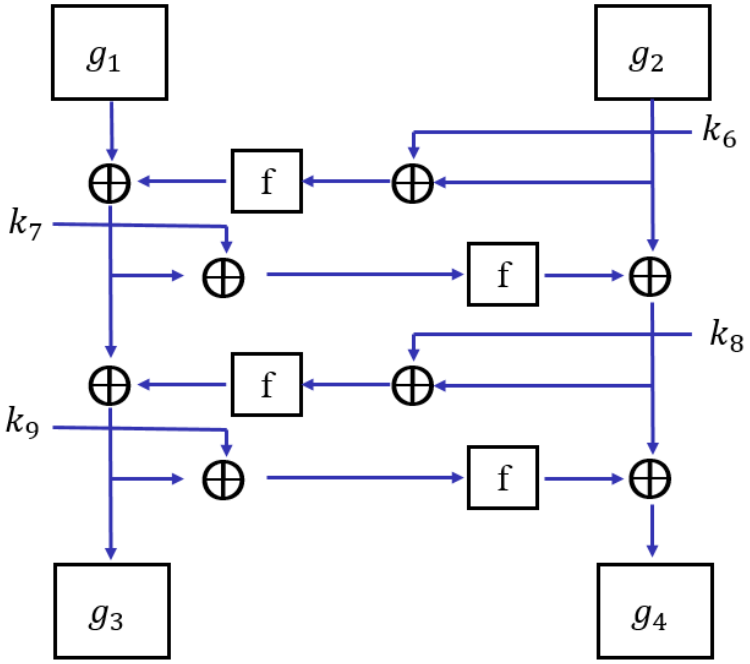
$$f = G_{k_{2,3,4,5}}(d) \oplus c$$

.
.
.

The middle sixteen rounds :

5. Recover k_7. The outputs of the G transformation in round 10 are now known, and the inputs have known difference a. We can decrypt two layers of G in round 10 (using k_8 and k_9), and then derive k_7 (which is used in the next layer) with a precomputed lookup table, as above. With proper implementation, this phase takes 2^{64-16n} work, and we expect that about 2^{72-24n} possibilities for $k_7, \ldots, k_5$ will remain at the conclusion of this phase.

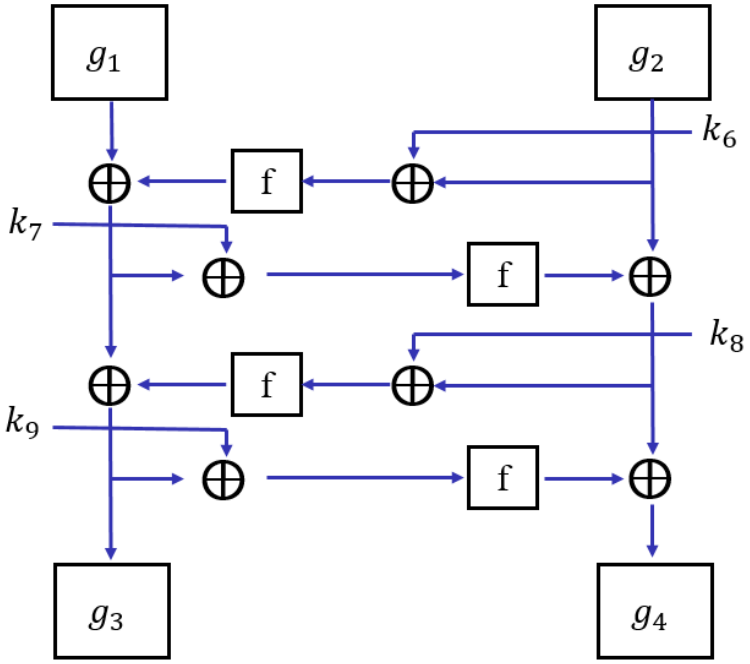
G in round 10



The middle sixteen rounds :

6. Recover k_6. Complete the analysis of the computation of G in round 10 by deriving k_6 from its known outputs and its known input difference a. With proper implementation, this phase takes 2^{72-24n} simple operations, and about 2^{80-32n} suggested values for the entire key $k_0, \ldots, k_9$ will be left.

G in round 10



The middle sixteen rounds :

7. Check suggested values. We can check each suggested value for the key in any of a number of ways.

   One simple way is to do a full trial decryption on a few of the texts.

   Alternately, one could encrypt through the G transformations in rounds two,three, and six to check the result against the known input to G in round 10. This will require only four G computations and thus can be quite a bit faster than a full trial decryption.

   We expect that this final phase will quickly eliminate all incorrect keys.

Key used	Round
0,1,2,3	1,6,11,16
4,5,6,7	2,7,12
8,9,0,1	3,8,13
2,3,4,5	4,9,14
6,7,8,9	5,10,15

The middle sixteen rounds :

The work required is about

$$2^{32} + 2^{32} + 2^{40-8n} + 2^{64-16n} + 2^{64-16n} + 2^{72-24n} + 4 \times 2^{80-32n}$$

simple operations. For $n = 1$ this gives 2^{51} steps and 2^{34} steps for $n = 2$. Of course, each step requires just a single G computation (often quite a bit less), so this is equivalent to about 2^{47} (respectively 2^{30}) trial encryptions. The result is a very sharp attack against the middle 16 rounds of Skipjack.

The last twenty-eight rounds :

We use 28-round differential :

$$(a,b,0,c) \xrightarrow{4r_A} (d,e,0,0) \xrightarrow{8r_B} (f,g,0,h) \xrightarrow{8r_A} (i,i,0,0) \xrightarrow{8r_B} (j,k,l,0)$$

$$pr = 2^{-16} \qquad pr = 2^{-16} \qquad pr = 2^{-32} \qquad pr = 1$$

where $(a,b,0,c) \xrightarrow{4r_A} (d,e,0,0)$ starts in the fifth round.

Attacks using truncated differentials

The last twenty-eight rounds :

- Choose 2^{41} plaintexts where the values of the third words are fixed to some arbitrary value.
- Form $\frac{2^{41}*(2^{41}-1)}{2} \approx 2^{81}$ pairs.
- $2^{81} * 2^{-64} = 2^{17}$ right pair expected.
- Using the rightmost word of the ciphertexts, we can filter out wrong pairs, leaving $2^{81} * 2^{-16} = 2^{65}$ pairs.

Attacks using truncated differentials

The last twenty-eight rounds :

- For each surviving pair(2^{65}) we check which keys in the last round result in a difference that follows the differential after decryption by one round. About 2^{16} values for the 32-bit key will be suggested by this test for each pair.
- Similarly, for each surviving pair we check which keys result in differences that follow the differential after encryption by one round.
- So, each pair suggest $2^{16+16} = 2^{32}$ values for 8 key bytes.
- Overall we will find $2^{65+32} = 2^{97}$ values for 64 bit keys.

Attacks using truncated differentials

The last twenty-eight rounds :

- The expected value of the counter for a wrong value of the key is 2^{33},whereas the expected value of the counter for the correct value of the key will be $2^{33} + 2^{17}$since each of the 2^{17} right pairs will include the correct key value among the set of 2^{33} values suggested.

- This would mean that with a high probability the correct value of the key is among the 16% most suggested values.

- The total time for the analysis stage of this attack amounts to $2^{65+17} = 2^{82}$ G computations, a work effort that is equivalent to about 2^{77} encryptions. Thus, this attack is just faster than an exhaustive search for the key but the work effort required and the need for 2^{64} counters makes the attack totally impractical.

References

1. Lars R. Knudsen, Matthew J. B. Robshaw, and David Wagner. “Truncated Differentials and Skipjack.” In: CRYPTO’99. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 165–180.

2. E. Biham, A. Biryukov, O. Dunkelmann, E. Richardson, A. Shamir. Initial Observations on the Skipjack Encryption Algorithm. June 25, 1998. Available at http://www.cs.technion.ac.il/~biham/Reports/SkipJack/.