Producing Collisions for Panama, Instantaneously

Joan Daemen and Gilles Van Assche
gro.noekeon@noekeon.org

STMicroelectronics, Zaventem, Belgium

Abstract. We present a practical attack on the Panama hash function that generates a collision in 2^6 evaluations of the state updating function. Our attack improves that of Rijmen and coworkers that had a complexity 2^{82}, too high to produce a collision in practice. This improvement comes mainly from the use of techniques to transfer conditions on the state to message words instead of trying many message pairs and using the ones for which the conditions are satisfied. Our attack works for any arbitrary prefix message, followed by a pair of suffix messages with a given difference. We give an example of a collision and make the collision-generating program available. Our attack does not affect the Panama stream cipher, that is still unbroken to the best of our knowledge.

Keywords: symmetric cryptography, hash function, collision

1 Introduction

A cryptographic hash function maps a message of arbitrary length to a fixed-size output called a digest. One of the requirements for a cryptographic hash function is collision-resistance: it should be infeasible to to find two different messages that give the same digest.

Panama can be used both as a hash function and as a stream cipher. In the scope of this paper, we will consider only on the hash part. Our attack does not have impact on the security of the Panama stream cipher. Internally, Panama has a state and a buffer, which evolve using a state updating function. For every block of message, the state updating function transforms the state and the buffer. We describe Panama in Sec. 2.

In this article, we describe a method to produce collisions for the Panama hash function that refines the method of Rijmen and coworkers [2] and reduces the workload from 2^{82} to 2^6 applications of the state updating function. We can therefore generate collisions quasi instantaneously. Furthermore, there are many degrees of freedom in the produced messages. The attack works for any initial value of the state. This means that one can find a collision with a pair of messages $(M_1|M, M_1|M^*)$ with an arbitrary prefix M_1. Here, the message parts M and M^* have a fixed difference $M' = M + M^*$. Furthermore, the attacker can append an arbitrary suffix M_2 to both collision messages, independently of M_1, M and M^*. We discuss the structure of the attack in Sec. 3.

Like in [2], we use a differential trail (called differential path in [2]) that leads to a zero difference in state and buffer. A differential trail specifies both the message differences and the differences in the state and in the buffer. For a pair of messages to follow the right differences in the state, a subset of the state bits must satisfy specific conditions. In the attack in [2], part of these conditions were transferred to equations on message words while the remaining ones were satisfied by trying many different message pairs and picking out those for which these conditions happened to be satisfied. In our attack, we transfer *all* conditions to equations on message bits using some simple new techniques explained in Sec. 4. The transfer of equations has negligible workload.

Although very similar, our trail is different from that of [2]. We chose a trail such that the conditions on the state are more easily transferable to equations on the message bits. We describe it in Sec. 5 and all its conditions and their transfer in Sec. 6.

2 Description of Panama

The internal memory of Panama is composed of 273 32-bit words (hereby denoted words) and is organized in two parts: [1]

- the *state*, with 17 words denoted a_0 through a_{16}, and
- the *buffer*, which is an array of 32×8 words, denoted $b_{i,j}$ with $0 \leq i \leq 31$ and $0 \leq j \leq 7$. (Note that b_i indicates a block of 8 words $b_{i,0} \dots b_{i,7}$.)

The $+$ sign applied on bits denotes the exclusive *or* (xor) operation and on words the bitwise xor. In subscripts of the state a, it denotes modulo-17 addition.

The message to hash is padded and divided into blocks of 8 words (i.e., 256 bits) each. It is processed as follows. First, both the state and the buffer are initialized to 0. Then, for each message block $p = (p_0, p_1, \dots, p_7)$ (i.e., for each round), the following operations are applied:

- the state undergoes a non-linear transformation $\theta \circ \pi \circ \gamma$, with

$$\begin{aligned}\gamma &: a_i \leftarrow a_i + (a_{i+1} + \overline{0})a_{i+2} + \overline{0},\\ \pi &: a_i \leftarrow a_{7i \bmod 17} \ggg i(i+1)/2,\\ \theta &: a_i \leftarrow a_i + a_{i+1} + a_{i+4},\end{aligned}$$

  where the invisible multiplication indicates the bitwise *and*, $\overline{0}$ denotes the word with 32 bits 1, and $\ggg$ cyclic right shift of the bits within a word;
- the least significant bit of a_0 is flipped: $a_0 \leftarrow a_0 + 1$;
- the message block is xored into the state:

$$a \leftarrow a + f_{\mathrm{i}\rightarrow\mathrm{s}}(p) \;\Leftrightarrow\; a_{i+1} \leftarrow a_{i+1} + p_i,\; 0 \leq i \leq 7;$$

– eight words of the buffer are xored into the state:

$$a \leftarrow a + f_{\mathrm{b}\to\mathrm{s}}(b_{16}) \;\Leftrightarrow\; a_{i+9} \leftarrow a_{i+9} + b_{16,i},\; 0 \le i \le 7;$$

– the buffer undergoes a linear feedback shift register (LFSR) step:

$$\begin{aligned} b_i &\leftarrow b_{i-1 \bmod 32} \quad (i \neq 25), \\ b_{25} &\leftarrow b_{24} + r(b_{31}), \end{aligned}$$

where the function r is defined as $Y = r(X) \Leftrightarrow Y_j = X_{j+2 \bmod 8}$;

– the message block is xored into the buffer: $b_{0,i} \leftarrow b_{0,i} + p_i$, $0 \le i \le 7$.

After all the message blocks are processed, 33 extra rounds are performed, called blank rounds. These rounds use the state updating function, with the difference that a part of the state (instead of a message block) is input into the buffer: $b_{0,i} \leftarrow b_{0,i} + a_{i+1}$, $0 \le i \le 7$.

Finally, the digest is extracted from the state after the blank rounds.

3 Structure of the attack

The first thing to note is that the presence of the blank rounds makes it hard to produce a collision in the digest if there is a difference in either the state or the buffer after all the message blocks are input. Due to the invertibility of the state updating function such a difference will not cancel out. Moreover, the lack of external input and the propagation properties of the state updating function give the attacker almost no control over the final difference. Therefore, our goal is to produce a collision in both the state and the buffer before the blank rounds.

We produce a collision by following a trail. Two instances of Panama process two different messages (p and $p + dp$), which have a given difference (dp). The trail also specifies the differences in the state (da) and in the buffer (db) between the two instances of Panama, at each round. So, not only the two messages must have the given difference, they must also produce the right difference in the state and in the buffer.

We shall now describe the general structure of the trail used in the scope of this article. We will first talk about the sequence of message differences, then about the differences in the state.

In the sequel, the round numbers are specified between brackets in superscript: $\cdot^{(i)}$. The convention is that $p^{(i)}$ is the message block processed during round i, and $a^{(i)}$ is the value of the state after round i.

3.1 Collision in the buffer

The buffer evolves independently from the state and is linear. As noticed in [2], the following message difference sequence gives a collision in the buffer for any x:

$$dp^{(1)} = x,\ dp^{(8)} = r(x),\ dp^{(33)} = x, \text{ all other differences } 0. \tag{1}$$

After 32 rounds, we have $db_{24} = r(x)$ and $db_{31} = x$. After the 33rd round, we get:

$$
\begin{aligned}
db_{25} &\leftarrow db_{24} + r(db_{31}) = r(x) + r(x) = 0, \\
db_0 &\leftarrow db_{31} + dp^{(33)} = x + x = 0.
\end{aligned}
$$

Thanks to the linearity of the buffer, any combination of shifted instances of the sequence (1) results in a collision in the buffer. In [2], two such sequences are used, one distant of two rounds from the other. In this paper, we instead use three such sequences at three consecutive rounds. More precisely, the message sequence is as follows (only non-zero differences are indicated):

$$
\begin{aligned}
(dp^{(1)}, dp^{(2)}, dp^{(3)}) &= (d^{(1)}, d^{(2)}, d^{(3)}), \\
(dp^{(8)}, dp^{(9)}, dp^{(10)}) &= (r(d^{(1)}), r(d^{(2)}), r(d^{(3)}), \\
(dp^{(33)}, dp^{(34)}, dp^{(35)}) &= (d^{(1)}, d^{(2)}, d^{(3)}).
\end{aligned}
$$

3.2 Collision in the state

The state is influenced both by the message blocks and by the buffer words in b_{16}. Let us summarize the sequence of differences that are xored into the state, both from the message block and from b_{16}:

I Rounds $r = i + 0$: State gets difference $dp^{(r)} = d^{(i)}$

II Rounds $r = i + 7$: State gets difference $dp^{(r)} = r(d^{(i)})$

III Rounds $r = i + 17$: State gets difference $db_{16}^{(r)} = dp^{(r-17)} = d^{(i)}$

IV Rounds $r = i + 24$: State gets difference $db_{16}^{(r)} = dp^{(r-17)} = r(d^{(i)})$

V Rounds $r = i + 32$: State gets difference $dp^{(r)} = d^{(i)}$

with $1 \leq i \leq 3$.

After the three rounds in each of the five sequences described above, we will make sure that we have a collision in the state. These are called *subcollisions*. After the last subcollision, we have both a collision in the state and in the buffer, and we are thus guaranteed to obtain the same digest after the blank rounds.

Before we explain how to obtain a subcollision, we need to detail the properties of the difference propagation in γ, the only non-linear operation of the state updating function.

3.3 Difference propagation through γ

Since γ is composed only of bitwise operations, we will only talk about γ as if it operates on 17 bits in this current subsection. The actual γ on words can be seen as 32 such operations in parallel.

Assume that the input of one instance of γ is a, while the input of the other instance is $a + da$. For a given input difference da, not all output differences are possible. The output difference $dc = (dc_0, \ldots, dc_{16})$ is determined by the following equation:

$$dc_i = \gamma_i(da) + da_{i+1}a_{i+2} + da_{i+2}a_{i+1} + 1,$$

where $\gamma_i(a) = a_i + (a_{i+1} + 1)a_{i+2} + 1$ denotes a particular output bit of γ.

Hence, we can obtain an output difference dc from a given input difference da only if a satisfies some conditions. These are as follows:

$$\text{If } da_{i+1} = 1 \text{ and } da_{i+2} = 0, \text{ then } a_{i+2} = dc_i + \gamma_i(da) + 1; \tag{2}$$

$$\text{If } da_{i+1} = 0 \text{ and } da_{i+2} = 1, \text{ then } a_{i+1} = dc_i + \gamma_i(da) + 1; \tag{3}$$

$$\text{If } da_{i+1} = 1 \text{ and } da_{i+2} = 1, \text{ then } a_{i+1} + a_{i+2} = dc_i + \gamma_i(da) + 1. \tag{4}$$

We call conditions of type (2) and (3) *simple* conditions and conditions of type (4) *two-bit parity* conditions. We call a differential (da, dc) for which the set of conditions has a solution a *possible differential.*

Note that the input difference da fully determines the positions of the state bits a_i that are subject to conditions. Assume that we have n consecutive 1s in the pattern da, i.e., we have $da_i = da_{i+n+1} = 0$ and in between $da_{i+l} = 1$ $(1 \leq l \leq n)$. Then there are simple conditions on a_i and on a_{i+n+1}, and $n-1$ two-bit parity conditions on $a_{i+l} + a_{i+l+1}$ $(1 \leq l < n)$. This can be applied to all such patterns in da.

From this follows that the number of conditions is equal to the Hamming weight of da plus the number of 001 patterns in da. (For the particular case of $da = 11111111111111111$, there are 16 independent two-bit parity conditions.) We denote by $w(da)$ the number of conditions due to da.

3.4 Specifying the trail

For our attack to work, we wish to determine equations on the message bits that imply the five subcollisions. In the previous subsection we have shown that given a possible differential (da, dc) over γ, we obtain conditions on input bits of γ.

Consider now subcollision I. Before the first round, there is no difference in the state, hence $da^{(0)} = 0$. At the input of the second round, the message difference appears in the state: $da^{(1)} = f_{\mathrm{i}\to\mathrm{s}}(d^{(1)})$. This determines the input difference of γ in round 2. We now need to specify the output of γ in the second round, but we can equivalently specify $da^{(2)}$, as the other operations are linear. After the third round, the fact that we have a collision in the state imposes that $da^{(3)} = 0$, yielding at the output of the third round a difference equal to $f_{\mathrm{i}\to\mathrm{s}}(d^{(3)})$. Hence a value for $da^{(2)}$ must be chosen such that differentials $(f_{\mathrm{i}\to\mathrm{s}}(d^{(1)}), \pi^{-1} \circ \theta^{-1}(da^{(2)} + f_{\mathrm{i}\to\mathrm{s}}(d^{(2)})))$ and $(da^{(2)}, \pi^{-1} \circ \theta^{-1}(f_{\mathrm{i}\to\mathrm{s}}(d^{(3)})))$ over γ are possible. For a given message difference sequence $d^{(1)}, d^{(2)}, d^{(3)}$ there may be several, one or none such values of $da^{(2)}$. Note that the first differential imposes conditions on $a^{(1)}$ and the second one on $a^{(2)}$.

As θ and π are linear, it follows that a possible differential over the state-updating function imposes conditions on bits of the state $a^{(i)}$. Doing this for differentials over more rounds is more difficult and we avoid it in out attack. Therefore, for each round in which there is non-zero input difference in the state, we need to know the output difference.

For subcollisions II to V, applying the same reasoning leads to following round differentials, which we write as differentials over $\theta \circ \pi \circ \gamma$ for compactness:

$$\begin{array}{rllll}
\textbf{I} & \big(f_{\mathrm{i}\to\mathrm{s}}(d^{(1)}), & da^{(2)} + f_{\mathrm{i}\to\mathrm{s}}(d^{(2)})\big) & \big(da^{(2)}, & f_{\mathrm{i}\to\mathrm{s}}(d^{(3)})\big) \\
\textbf{II} & \big(f_{\mathrm{i}\to\mathrm{s}}(r(d^{(1)})), & da^{(9)} + f_{\mathrm{i}\to\mathrm{s}}(r(d^{(2)}))\big) & \big(da^{(9)}, & f_{\mathrm{i}\to\mathrm{s}}(r(d^{(3)}))\big) \\
\textbf{III} & \big(f_{\mathrm{b}\to\mathrm{s}}(d^{(1)}), & da^{(19)} + f_{\mathrm{b}\to\mathrm{s}}(d^{(2)})\big) & \big(da^{(19)}, & f_{\mathrm{b}\to\mathrm{s}}(d^{(3)})\big) \\
\textbf{IV} & \big(f_{\mathrm{b}\to\mathrm{s}}(r(d^{(1)})), & da^{(26)} + f_{\mathrm{b}\to\mathrm{s}}(r(d^{(2)}))\big) & \big(da^{(26)}, & f_{\mathrm{b}\to\mathrm{s}}(r(d^{(3)}))\big) \\
\textbf{V} & \big(f_{\mathrm{i}\to\mathrm{s}}(d^{(1)}), & da^{(34)} + f_{\mathrm{i}\to\mathrm{s}}(d^{(2)})\big) & \big(da^{(34)}, & f_{\mathrm{i}\to\mathrm{s}}(d^{(3)})\big)
\end{array}$$

Hence, the trail is fully determined by the sequence $(d^{(1)}, d^{(2)}, d^{(2)})$, and the 5 state differences $da^{(2)}$, $da^{(9)}$, $da^{(19)}, da^{(26)}$ and $da^{(34)}$. Because the structure of the subcollisions I and V are equal, we can fix $da^{(34)} = da^{(2)}$.

3.5 Symmetric patterns

Like in [2], we use differences with words that are either 0 or $\overline{0}$. This causes the intra-word rotations in π to have no influence on the difference pattern, as all other operations in the state updating function work in a bitwise fashion.

Let us translate this in the case of the word-oriented γ. We can view Equations (2)–(4) as 32 parallel conditions on the bits of the state words. Thanks to the fact that all the difference words da_i are either 0 or $\overline{0}$, the words a_j on which these conditions apply are the same for the 32 bits; either all or none of the 32 bits of a word a_j are affected by a condition. Hence, the equations can be written word-wise. Note however that this does not restrict the value of the state or of the message words to be either 0 or $\overline{0}$, only the differences.

4 Techniques for equation transfer

For a given trail, we have seen in Sec. 3.3 how to express conditions on the state a to get the right output differences. In this section, we explain how to transfer these equations to the message words that the attacker can choose.

We will see that the equations are never transferred to more than two rounds before the start of the subcollision, so there is no overlap between the equations derived from different subcollisions and hence we can satisfy them sequentially.

As the discussion below is generic for all five subcollisions, let us use a common convention. For $j \in \{1, 8, 18, 25, 33\}$, we denote the various stages of the state transformation with the following symbols:

$$\begin{array}{rl}
\xrightarrow{+p^{(j-2)}} & \mathrm{N} \xrightarrow{\gamma} \mathrm{O} \xrightarrow{\theta\circ\pi} \mathrm{P} \xrightarrow{+p^{(j-1)}} \mathrm{Q} \\
& \mathrm{Q} \xrightarrow{\gamma} \mathrm{R} \xrightarrow{\theta\circ\pi} \mathrm{S} \xrightarrow{+p^{(j)}} \mathrm{T} \\
& \mathrm{T} \xrightarrow{\gamma} \mathrm{U} \xrightarrow{\theta\circ\pi} \mathrm{V} \xrightarrow{+p^{(j+1)}} \mathrm{W} \\
& \mathrm{W} \xrightarrow{\gamma} \mathrm{X} \xrightarrow{\theta\circ\pi} \mathrm{Y} \xrightarrow{+p^{(j+2)}} \mathrm{Z}.
\end{array}$$

At a given time, the corresponding italic letter denotes the state value.

Although we will follow a reasoning going backwards from Z down to T, Q or N, the attack works in practice in the forward direction. As the conditions are being satisfied, the state is updated with the known values.

For each subcollision, we wish to have a collision in the state at time Z, hence to have $dZ = 0$. This determines $dY = f_{\mathrm{i}\to\mathrm{s}}(dp^{(j+2)})$ or $dY = f_{\mathrm{b}\to\mathrm{s}}(db_{16}^{(j+1)})$, together with the difference pattern dX via $\pi^{-1} \circ \theta^{-1}$. The trail also specifies the patterns dW and dT (and indirectly dU). The differential (dT, dU) over γ implies conditions on T and (dW, dX) on W. The attacker must satisfy them by choosing appropriate values for $p^{(j-2)}$, $p^{(j-1)}$, $p^{(j)}$ and $p^{(j+1)}$.

The conditions are either simple, i.e., of type $W_i = \mathit{target}$ or two-bit parity conditions $W_i + W_{i+1} = \mathit{target}$ where "*target*" is a known value.

In the sequel, we often speak about the left and right hand sides of an equation. As a convention, the left hand side contains one isolated variable to be solved, while the right hand side contains other variables that are determined from other equations or set to arbitrary values.

4.1 Immediate satisfaction in W

Simple conditions on words W_1 through W_8 can be satisfied by setting the value of $p^{(j+1)}$ accordingly. We call this *immediate satisfaction*:

$$W_i = \mathit{target} \to p_{i-1}^{(j+1)} = V_i + \mathit{target} \text{ (if } 1 \le i \le 8). \tag{5}$$

The value of V_i is determined by the value of T.

A two-bit parity condition $W_i + W_{i+1} = \mathit{target}$ can be satisfied whenever (at least) one of the two words can be modified through p, that is, when $0 \le i \le 8$. For $i = 0$ or $i = 8$, we have:

$$\begin{aligned} W_0 + W_1 = \mathit{target} &\to p_0^{(j+1)} = V_0 + V_1 + \mathit{target}, \text{ and} \\ W_8 + W_9 = \mathit{target} &\to p_7^{(j+1)} = V_8 + V_9 + \mathit{target}. \end{aligned}$$

If $1 \le i \le 7$, however, the value of another message word must be taken into account. This other message word must be treated as known and its value may either be fixed by other conditions or set to an arbitrary value. We have either

$$\begin{aligned} W_i + W_{i+1} = \mathit{target} &\to p_{i-1}^{(j+1)} = V_i + V_{i+1} + p_i^{(j+1)} + \mathit{target}, \text{ or} \\ W_i + W_{i+1} = \mathit{target} &\to p_i^{(j+1)} = V_i + V_{i+1} + p_{i-1}^{(j+1)} + \mathit{target}. \end{aligned}$$

4.2 Bridge from W to T

The conditions on W that cannot be satisfied immediately can be transferred to equations at time U via $\pi^{-1} \circ \theta^{-1}$.

A condition on some W_i can be converted into an equation in three words of U. For instance, assume we have to satisfy $W_{10} = 0$. We know that

$$W_{10} = (U_2 \lll 3) + (U_9 \lll 45) + (U_{13} \lll 91).$$

In this case, U_2 will be influenced directly by the message words $p^{(j)}$, and this makes it an ideal candidate for immediate satisfaction in T. So, let us isolate this variable and write:

$$U_2 = ((U_9 \lll 45) + (U_{13} \lll 91)) \ggg 3.$$

Remember that the terms U_9 and U_{13} at the right hand side are treated as known values.

In more general terms, a simple condition on W_i is converted into an equation on $U^\pi_{7i} + U^\pi_{7(i+1)} + U^\pi_{7(i+4)}$, with $U^\pi_i = U_i \lll i(i+1)/2$. One can choose to isolate one of the three variables U_{7i}, $U_{7(i+1)}$ or $U_{7(i+4)}$. Then, the cyclic rotation on the left hand side can be replaced by its inverse on the right hand side. For instance, the isolation of U_{7i} gives the following:

$$W_i = \mathit{target} \to U_{7i} = (U^\pi_{7(i+1)} + U^\pi_{7(i+4)} + \mathit{target}) \ggg 7i(7i+1)/2,\ i \neq 0, \quad (6)$$

while the constant 1 must be taken care of in the case of the word $i = 0$, for instance we can isolate U_0 as $U_0 = U^\pi_7 + U^\pi_{11} + \mathit{target} + 1$.

Similarly, a two-bit parity condition on $W_i + W_{i+1}$ is converted into an equation on $U^\pi_{7i} + U^\pi_{7(i+2)} + U^\pi_{7(i+4)} + U^\pi_{7(i+5)}$. Again, one can choose to isolate either of the four variables.

An equation of the type $U_i = \mathit{target}$ can be written as $T_i + (\bar{0} + T_{i+1})T_{i+2} = \mathit{target} + \bar{0}$. One can transfer the equation on T_i by treating T_{i+1} and T_{i+2} as known values:

$$U_i = \mathit{target} \to T_i = (\bar{0} + T_{i+1})T_{i+2} + \mathit{target} + \bar{0}. \quad (7)$$

Combining the substitutions (6) and (7) is called a *bridge*. Together with the immediate satisfaction, this is the technique we used most often in our collision-generating algorithm.

Of course, all the conditions on T_1 through T_8 are immediately satisfiable by setting the appropriate value in $p^{(j)}$ as in Equation (5) with W and V replaced by T and S, respectively.

4.3 Side bridge

An interesting special case is an equation on $U_0 = \mathit{target}$. Since an equation on T_0 cannot be immediately satisfied via p_j, we can instead create two equations on T_1 and T_2. We can choose from three ways of creating two equations:

$$\begin{array}{lll} U_0 = \mathit{target} \to T_1 = T_0 + \mathit{target} & \text{and} & T_2 = \bar{0} \\ U_0 = \mathit{target} \to T_1 = 0 & \text{and} & T_2 = T_0 + \mathit{target} + \bar{0} \\ U_0 = \mathit{target} \to T_1 = T_0 + \mathit{target} & \text{and} & T_2 = T_0 + \mathit{target} + \bar{0} \end{array}$$

In the sequel, this technique is called a *side bridge*. Note that this technique can also be used on other word positions.

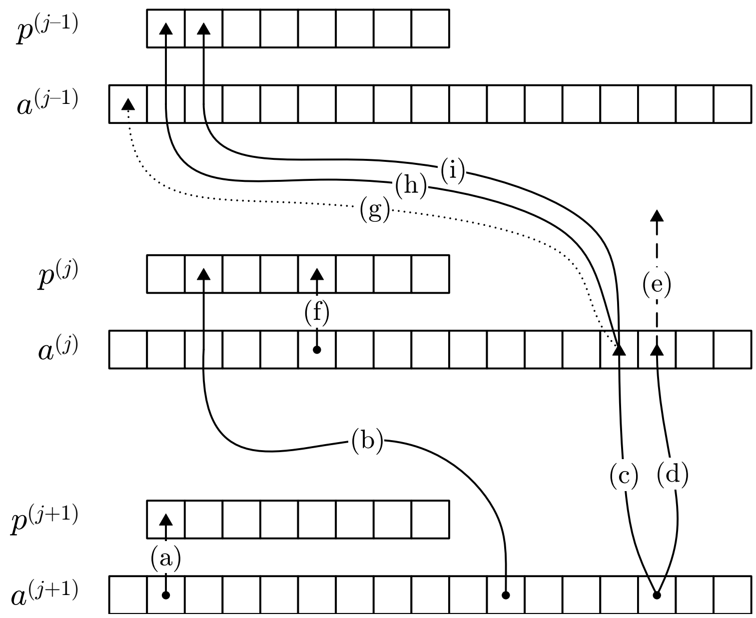

Fig. 1. Schematic illustration of some of the equation transfer techniques. (a) A condition in one of the words $a^{(j+1)}_{1\ldots8}$ can be immediately satisfied via $p^{(j+1)}$. (b) Otherwise, this condition has to be bridged to the previous round. Preferably, it is bridged to one of the words $a^{(j)}_{1\ldots8}$ so that it can be satisfied via $p^{(j)}$. (c)-(d) Sometimes, a bridge (c) must be accompanied by an additional equation (d) to remove circular dependencies. (e) Of course, this additional equation must be also be bridged or satisfied somewhere (not shown explicitly). (f) Conditions can also appear in round j; here is an example of such a condition that can be immediately satisfied via $p^{(j)}$. (g)-(h)-(i) Bridging a condition to the word 0 (g) does not allow immediate satisfaction. Instead, it is possible to side-bridge it to two equations, one on word 1 (h) and one on word 2 (i).

4.4 Dependency removal

As said in the beginning of Sec. 4, when solving an equation, all terms on the right hand side must be known. This imposes constraints on the order in which the equations are solved. In some cases, it may be necessary to remove circular dependencies on the way from U to T. For instance, assume that we have three equations on U:

$$U_4 = \mathit{target}, \tag{8}$$
$$U_6 = \mathit{target}, \tag{9}$$
$$U_8 = U_4 + \mathit{target} \text{ (up to rotation)}. \tag{10}$$

When transferring from U to T, (8) requires T_5 and T_6 to be known, hence that (9) is already solved for T_6. In turn, (9) requires T_7 and T_8 to be known, hence that (10) is solved for T_8. But (10) requires that T_4 is known so that (8) must be solved. To remove this circular set of dependencies, we can force $T_7 = \overline{0}$ so that $U_6 = T_6 + (T_7 + \overline{0})T_8 + \overline{0} = T_6 + \overline{0}$ does not depend on T_8 any more. The dependency of (9) on (10) is removed, and we can solve (9), then (8), then (10).

In more general terms, we can set $T_{i+1} = \overline{0}$ (resp. $T_{i+2} = 0$) for the equation on $U_i = target$, so that the converted equation on T_i has no dependency on T_{i+2} (resp. T_{i+1}). In the sequel, this technique is called *dependency removal*. To sum up, one can apply either:

$$U_i = target \rightarrow T_i = target + \overline{0} \text{ and } T_{i+1} = \overline{0}, \text{ or}$$
$$U_i = target \rightarrow T_i = target + \overline{0} \text{ and } T_{i+2} = 0.$$

The dependency removal and the other techniques are illustrated in Fig. 1.

4.5 The conditions due to differential (dT, dU)

The differential (dT, dU) fixes some words of T. These words are therefore not available for bridges coming from W. Fortunately, there are some degrees of freedom on the way from W to T to avoid conflicts. For instance, one should choose adequately the component of U that will be isolated on the left hand side. For the attack to work, there must be a way to satisfy and bridge the equations in a non-conflicting way.

The equations on T in turn can be bridged to equations on Q. The conditions on Q_1 through Q_8 can then be satisfied via $p^{(j-1)}$.

However, one must pay careful attention to equation dependencies. When for example choosing $p^{(j)}$ to satisfy a condition in W, it has an impact on the state at time T and subsequent rounds. Indeed, changing the value of T_i influences the values of U_{i-2}, U_{i-1} and U_i. Hence care must be taken, when solving equations that are the result of bridges, that equations solved earlier are not affected. In general, dependency problems become more difficult to manage as the number of bridges grows.

4.6 Solving the equations by correction

When all bridges are determined along with their dependencies, solving the equations is fairly simple and does not involve much more than an evaluation of the state updating function. This is thanks to the fact that the equations are made linear once the right hand side is determined. Let us illustrate this with an example.

Assume that we need to satisfy $W_9 = 0$. Since immediate satisfaction is not possible, we decide to bridge this condition to $T_6 = target$ (via an equation on U_6). At this stage, we do not calculate the actual value of *target*, but we influence T_6 through $p_5^{(j)}$. So, we evaluate the state updating function for round j with $p_5^{(j)} = 0$ and then the state updating function for round $j+1$. After this, we obtain W_9^*, which may not be zero as we wished. However the linearity of the equations implies that we can reuse the value W_9^* (up to a rotation) to determine $p_5^{(j)}$. Since U_6 is rotated by $6(6+1)/2 = 21$ positions, we can simply set $p_5^{(j)} = W_9^* \ggg 21$ and we automatically get $W_9 = 0$.

In short, the linearity allows us to satisfy a condition by correcting the corresponding message word. The right value of the message word is determined (up to a rotation) by the correction to bring to the state word under condition.

The reasoning for satisfying and bridging equations is done backwards in time. In contrast, solving them by correction works forwards in time, and the rounds are evaluated sequentially. The state updating function may be evaluated several times, an extra evaluation being needed for each correction.

The dependencies between equations play an important role in the order in which the corrections are applied. In the same way the right hand side of an equation must be known when solving it, a correction shall not affect equations that have been satisfied earlier.

5 The chosen trail

To choose a suitable trail, an important parameter to consider is the number of conditions on T and on W in the five subcollisions. Actually, this number should be split in the number of conditions that can be immediately satisfied and those that need to be bridged.

To make a trail easy to exploit using the techniques described in Sec. 4, the application of immediate satisfaction and the bridges coming from the later rounds should as much as possible fit within the 8 message words each round. One can chain bridges over several rounds, but as more rounds are bridged dependencies become increasingly difficult to manage. Generally speaking, the higher the number of conditions to bridge, the more difficult it will be to fit them all on a small number of rounds.

The number of conditions in the first subcollision is $w(da^{(1)})$ in T and $w(da^{(2)})$ in W. The number of conditions for the other four subcollisions are $(w(da^{(i)}), w(da^{(i+1)}))$ with $i = 8, 18, 25$ and 33. Note that the fifth subcollision is identical to the first one. We split each of these numbers $w(da)$ as $w(da) = w_{\text{is}}(da) + w_{\text{b}}(da)$, where $w_{\text{is}}(da)$ is the number of conditions that can be immediately satisfied and $w_{\text{b}}(da)$ that must be bridged.

As a heuristic criterion, we minimize the maximum number of conditions to bridge:

$$W_{\text{b}} = \max_i \{w_{\text{b}}(da^{(i)})\}.$$

We have searched exhaustively through all $255^2 256$ patterns $(d^{(1)}, d^{(2)}, d^{(3)})$ and selected the one that has a collision trail with a minimal W_{b}. This resulted in the trail determined by following values:

- $d^{(1)} = 00000101$, $d^{(2)} = 11010000$, $d^{(3)} = 01111011$,
- $da^{(2)} = 10111110000000101$,
- $da^{(9)} = 00100011101011000$,

- $da^{(19)} = 00010111001100000$ and
- $da^{(26)} = 11111100111010010$,

where each digit represents a word either all-zero (0) or all-one (1); the word positions increase from left to right. The number of conditions that can be immediately satisfied or bridged is given in Table 1 below. As a comparison, the trail used in [2] has $W_b = 7$, while our trail has $W_b = 5$.

	I		II		III		IV	
	$da^{(1)}$	$da^{(2)}$	$da^{(8)}$	$da^{(9)}$	$da^{(18)}$	$da^{(19)}$	$da^{(25)}$	$da^{(26)}$
w_{is}	2	6	3	5	0	5	0	8
w_{b}	1	3	0	4	3	3	3	5

Table 1. Number of conditions in the trail used in our attack.

6 Equation transfer in the chosen trail

In this section, we describe in detail how the equations are transferred to the message words using the techniques in Sec. 4 for the trail specified in Sec. 5.

6.1 Subcollisions I and V

For round 1, we have the following differences:

$db^{(0)}_{16} = 00000000$	$da^{(0)} = 0000000000000000 = dQ$
$dp^{(1)} = 00000101$	**Round 1**
$da^{(1)} = 0000001010000000 = dT$	$dU = 0000100110000000$

From the pattern in dT, we can see that we have three conditions on T_5, T_7 and T_9, namely

$$T_5 = dU_4 + \gamma_4(dT) + \overline{0} = 0, \tag{11}$$
$$T_7 = dU_6 + \gamma_6(dT) + \overline{0} = dU_5 + \gamma_5(dT) + \overline{0} = 0, \tag{12}$$
$$T_9 = dU_8 + \gamma_8(dT) + \overline{0} = \overline{0}. \tag{13}$$

The equations (11) and (12) can be immediately satisfied in round 1 via $p^{(1)}_4$ and $p^{(1)}_6$, whereas (13) must be bridged to Q:

$$T_9 = \overline{0} \to Q_2 = ((\overline{0} + Q_3)Q_4 + \overline{0}) + (R^{\pi}_6 + R^{\pi}_{12} + \overline{0}) \ggg 3, \tag{14}$$

which can be immediately satisfied in round 0 via $p^{(0)}_1$. Note that this implies that Q_3, Q_4, …, Q_8, Q_{12}, Q_{13} and Q_{14} are known when solving for Q_2.

Then for round 2, we have the following differences:

$db_{16}^{(1)} = 00000000$	
$dp^{(2)} = 11010000$	**Round 2**
$da^{(2)} = 10111110000000101 = dW$	$dX = 01110010000001111$

From the pattern in dW, this imposes nine conditions on $W_0 + W_{16}$, W_1, $W_2 + W_3$, $W_3 + W_4$, $W_4 + W_5$, $W_5 + W_6$, W_7, W_{13} and W_{15}. We will not detail the right hand sides of the corresponding equations, as they can easily be found as explained in Sec. 3.3. The equations on the words W_1 through W_7 can be immediately satisfied via $p^{(2)}$. The other equations are solved as follows:

- The condition on $W_0 + W_{16}$ can be transferred to an equation on $U_4^\pi + U_7^\pi + U_{10}^\pi + U_{11}^\pi$. We isolate U_4 on the left hand side and transfer it to an equation on T_4, which can be immediately satisfied via $p_3^{(1)}$. (Here, T_5, T_6, $\ldots$, T_{13} must be known.)
- The condition on W_{15} is transferred to an equation on $U_3^\pi + U_{10}^\pi + U_{14}^\pi$, from which we isolate U_3, transfer it T_3 and satisfy it via $p_2^{(1)}$. Notice that when isolating T_3, this means putting T_4 and T_5 on the right hand side. The value of T_4 must thus be determined before that of T_3, hence the condition on W_{15} may only be solved after the one on $W_0 + W_{16}$. (Here, T_4, T_5, T_{10}, T_{11}, T_{12}, T_{14}, T_{15} and T_{16} must be known.)
- Similarly, the condition on W_{14} becomes an equation on $U_0^\pi + U_6^\pi + U_{13}^\pi$, then on U_6, then on T_6, then on $p_5^{(1)}$. (Here, T_0, T_1, T_2, T_7, T_8, T_{13}, T_{14} and T_{15} must be known.)

Finally in round 3, the differences in the state cancel. We process subcollision V in the same way, using the message blocks 32 rounds later.

6.2 Subcollision II

For round 8, we have the following differences:

$db_{16}^{(7)} = 00000000$	$da^{(7)} = 00000000000000000 = dQ$
$dp^{(8)} = 00010100$	**Round 8**
$da^{(8)} = 00001010000000000 = dT$	$dU = 00100010000000000$

Hence, we have three conditions on T_3, T_5 and T_7, which can be immediately satisfied via $p_2^{(8)}$, $p_4^{(8)}$ and $p_6^{(8)}$. This is summarized in the table below.

Condition on	**via**	**then on**	**satisfied via**
T_3, T_5, T_7			$p_2^{(8)}$, $p_4^{(8)}$, $p_6^{(8)}$

Then for round 9, we have the following differences:

$db_{16}^{(8)} = 00000000$	
$dp^{(9)} = 01000011$	**Round 9**
$da^{(9)} = 00100011101011000 = dW$	$dX = 10101001000011000$

Here we have nine conditions on W_1, W_3, W_5, $W_6 + W_7$, $W_7 + W_8$, W_9, W_{11}, $W_{12} + W_{13}$ and W_{14}. As usual, the conditions on W_1 through W_8 can be immediately satisfied via $p^{(9)}$.

Condition on	**via**	**then on**	**satisfied via**
W_1, W_3, W_5, $W_6 + W_7$, $W_7 + W_8$			$p_0^{(9)}$, $p_2^{(9)}$, $p_{4\ldots6}^{(9)}$
$W_{12} + W_{13}$	$U_0^\pi + U_{10}^\pi + U_{13}^\pi + U_{16}^\pi$	T_1 and T_2, using side bridge	$p_0^{(8)}$ and $p_1^{(8)}$
W_9	$U_2^\pi + U_6^\pi + U_{12}^\pi$	T_6	$p_5^{(8)}$
W_{14}	$U_3^\pi + U_7^\pi + U_{13}^\pi$	T_{13}, with $T_8 = \overline{0}$	$p_7^{(8)}$
T_{13}	$R_0^\pi + R_6^\pi + R_{14}^\pi$	Q_6	$p_5^{(7)}$
W_{11}	$U_3^\pi + U_9^\pi + U_{16}^\pi$	T_9, with $T_{11} = 0$	
T_9	$R_2^\pi + R_6^\pi + R_{12}^\pi$	Q_2, with $Q_1 = \overline{0}$	$p_0^{(7)}$ and $p_1^{(7)}$
T_{11}	$R_3^\pi + R_9^\pi + R_{16}^\pi$	Q_3	$p_2^{(7)}$

For the condition on W_{14}, we transfer it to an equation on $U_3^\pi + U_7^\pi + U_{13}^\pi$. We cannot isolate U_3 or U_7 and transfer it to T_3 or T_7 since we already have an equation on both. Instead, we isolate U_{13} and transfer the condition to T_{13}. We also remove the dependency of U_7 on T_9 by setting $T_8 = \overline{0}$, since T_9 will be needed below.

The condition on W_{11} can be transferred to an equation on $U_3^\pi + U_9^\pi + U_{16}^\pi$. Since T_3 is already busy, we instead isolate U_9 and transfer the condition to T_9. However, U_9 also depends on T_{10}, which is influenced by Q_2; but Q_2 is needed to satisfy the condition on T_9. To remove this circular dependency, we force $T_{11} = 0$ so that $U_9 = T_9 + \overline{0}$ does not depend on T_{10} any more.

The condition on T_9 is bridged to a condition on Q_2. However, Q_2 influences R_0, whose value is needed to solve for Q_6; but Q_6 influences R_6, whose value is needed to solve for Q_2. We force $Q_1 = \overline{0}$ so that $R_0 = Q_0 + \overline{0}$ no longer depends on Q_2.

Finally in round 10, the differences in the state cancel.

6.3 Subcollision III

For round 18, we have the following differences:

$db_{16}^{(17)} = 00000101$	$da^{(17)} = 0000000000000000 = dQ$
$dp^{(18)} = 00000000$	**Round 18**
$da^{(18)} = 0000000000000101 = dT$	$dU = 0000000000010011$

We have three conditions on T_0, T_{13} and T_{16}, which cannot be immediately satisfied in round 18.

Condition on	via	then on	satisfied via
T_0	$R_0^\pi + R_7^\pi + R_{11}^\pi$	Q_7	$p_6^{(17)}$
T_{13}	$R_0^\pi + R_6^\pi + R_{13}^\pi$	Q_6	$p_5^{(17)}$
T_{14}	$R_3^\pi + R_{10}^\pi + R_{14}^\pi$	Q_3	$p_2^{(17)}$

Then for round 19, we have the following differences:

$db_{16}^{(18)} = 11010000$	
$dp^{(19)} = 00000000$	**Round 19**
$da^{(19)} = 0001011100110000 = dW$	$dX = 0111101110010000$

This implies conditions on W_2, W_4, $W_5 + W_6$, $W_6 + W_7$, W_8, W_9, $W_{10} + W_{11}$, W_{12}.

Condition on	via	then on	satisfied via
W_2, W_4, $W_5 + W_6$, $W_6 + W_7$, W_8			$p_1^{(19)}$, $p_{3...5}^{(19)}$, $p_7^{(19)}$
W_{12}	$U_6^\pi + U_{10}^\pi + U_{16}^\pi$	T_6	$p_5^{(18)}$
W_9	$U_2^\pi + U_6^\pi + U_{12}^\pi$	T_2, with $T_4 = 0$	$p_1^{(18)}$ and $p_3^{(18)}$
$W_{10} + W_{11}$	$U_2^\pi + U_3^\pi + U_{13}^\pi + U_{16}^\pi$	T_3	$p_2^{(18)}$

To solve the conditions on W_9 and W_{10}, we need to remove some dependencies. On the one hand, the condition on W_9 is transferred to an equation on $U_2^\pi + U_6^\pi + U_{12}^\pi$, of which we want to isolate U_2 and transfer to T_2; this requires to know T_3 and T_4. On the other hand, the condition on $W_{10} + W_{11}$ becomes an equation on $U_2^\pi + U_3^\pi + U_{13}^\pi + U_{16}^\pi$; to isolate U_3 and transfer the equation to T_3, we need to know U_2. This dependency can be removed by forcing $T_4 = 0$ so that U_2 does not depend on T_3.

Finally in round 20, the differences in the state cancel.

6.4 Subcollision IV

For round 25, we have the following differences:

$db_{16}^{(24)} = 00010100$	$da^{(24)} = 0000000000000000 = dQ$
$dp^{(25)} = 00000000$	**Round 25**
$da^{(25)} = 0000000000010100 = dT$	$dU = 0000000001110100$

We have three conditions on T_{11}, T_{13} and T_{15}, which cannot be immediately satisfied in round 25.

Condition on	via	then on	satisfied via
T_{13}	$R_0^\pi + R_6^\pi + R_{13}^\pi$	Q_6, with $Q_2 = 0$	$p_5^{(24)}$ and $p_1^{(24)}$
T_{15}	$R_3^\pi + R_{10}^\pi + R_{14}^\pi$	Q_3	$p_2^{(24)}$
T_{11}	$R_3^\pi + R_9^\pi + R_{16}^\pi$	side bridge to $Q_0 = 0$ and to Q_1	$p_0^{(24)}$ for Q_1
Q_0	$O_0^\pi + O_7^\pi + O_{11}^\pi$	N_7	$p_6^{(23)}$

The condition on T_{11} cannot be satisfied neither in round 25 nor in round 24. As we transfer it to an equation on $R_3^\pi + R_9^\pi + R_{16}^\pi$, we cannot isolate R_3 as it would conflict with the condition on R_{15}. We instead isolate R_{16} and side-bridge it to $Q_0 = 0$ and to an equation on Q_1. The equation on Q_1 can be satisfied in round 24 via $p_0^{(24)}$. Consequently, $Q_0 = 0$ must be bridged to round 23.

To be able to solve for T_{13}, we must prevent Q_1 from influencing R_0. Hence, we force $Q_2 = 0$.

Then for round 26, we have the following differences:

$db_{16}^{(25)} = 01000011$	
$dp^{(26)} = 00000000$	**Round 26**
$da^{(26)} = 11111100111010010 = dW$	$dX = 11000101010010000$

This implies conditions on W_0+W_1, W_1+W_2, W_2+W_3, W_3+W_4, W_4+W_5, W_6, W_7, W_8+W_9, W_9+W_{10}, W_{11}, W_{13}, W_{14} and W_{16}. Among them, the conditions on W_1 through W_8 can be immediately satisfied via $p^{(26)}$.

Condition on	**via**	**then on**	**satisfied via**
W_0+W_1, W_1+W_2, W_2+W_3, W_3+W_4, W_4+W_5, W_6, W_7, W_8+W_9			$p^{(26)}$
$W_9 + W_{10}$	$U_6^\pi + U_9^\pi + U_{12}^\pi + U_{13}^\pi$	T_6, with $T_8 = 0$	$p_5^{(25)}$ and $p_7^{(25)}$
W_{13}	$U_0^\pi + U_6^\pi + U_{13}^\pi$	side bridge to T_1 and T_2	$p_0^{(25)}$ and $p_1^{(25)}$
W_{16}	$U_0^\pi + U_4^\pi + U_{10}^\pi$	T_4	$p_3^{(25)}$
W_{11}	$U_3^\pi + U_9^\pi + U_{16}^\pi$	T_3	$p_2^{(25)}$
W_{14}	$U_3^\pi + U_7^\pi + U_{13}^\pi$	T_7	$p_6^{(25)}$

In the condition on $W_9 + W_{10}$, we set $T_8 = 0$ to remove the dependency of U_6 on T_7. This way, the value of T_7 can be determined after that of T_6.

Note that the condition on W_{13} becomes an equation on $U_0^\pi + U_6^\pi + U_{13}^\pi$. Since U_6 is already taken, we isolate U_0 and side-bridge it to T_1 and T_2.

Finally in round 27, the differences in the state cancel.

7 Example of collision and workload

We wrote a program that produces collisions based on the trail described in Sec. 5 and using the equation transfer described in Sec. 6 [4]. The workload of the collision-generating function is about 65 applications of the state-updating function: 35 for the 35 message inputs and 30 additional ones for the bridges and some XORs.

An example of pair of collision messages is given in Table 2, which was obtained using our program. Each line represents a message block; for each block, the words in hexadecimal must be read from left to right. The first message is given by the hexadecimal digits in Table 2, while the second message is obtained by xoring with $\overline{0} = \texttt{ffffffff}$ all the underlined words.

$p^{(0)}$	002911b8	f4046c0d	18be4673	67847de2	4ae13b51	3d6c1b7e	2cd6267d	72ae641d
$p^{(1)}$	69522bd8	5f903d84	25558553	c194e805	1f7427d8	37edf3e4	bc922535	01eb3a6b
$p^{(2)}$	0e8257d3	2ea67fd6	0682df75	c21387fe	caa1b829	ccc994ba	9d03bd1c	00992518
$p^{(3)}$	01244898	305e252b	440d462c	491c5b2e	4d061f8b	4db745f9	15473f0e	54de79dc
$p^{(4)}$	39b355bc	2d1261f0	074d4fca	4dc8390e	6443663d	66bb5f6d	428b7e94	26a61a31
$p^{(5)}$	701f5092	5d037474	7a5a4baf	767d758d	450940b5	12383ea4	3b253990	1e1f71d5
$p^{(6)}$	6e5d785e	1ad4176a	63cb2040	6bfc19fc	7f965d80	7ff57876	4e455002	323b054b
$p^{(7)}$	24168c78	d6646fb1	9a2ac8f2	030a45b1	301c3921	e58d996a	56ae7f7d	0732105a
$p^{(8)}$	69bd59fc	6e3b4bdf	1adc0aac	22ee5482	4062e4cf	85f91c0a	45b21fe0	f25f2094
$p^{(9)}$	d7992b2c	1a491c5e	8dc2afaf	3bf6154e	a8ab7031	797d40fa	475d1ef4	e842e121
$p^{(10)}$	4cad0094	314f2b74	5e14301d	4df21075	494469e5	2e405ddc	13667210	1cd05258
$p^{(11)}$	366b5346	66c441da	42305df2	7eb75e5b	60327a81	2c3b3ba0	15a12e7f	54220e5c
$p^{(12)}$	3ef673cb	0822691d	59913a36	409d0de9	12e16f49	798b6174	121f0502	73da3555
$p^{(13)}$	58b077d2	26ca08ac	3699151a	09021b0b	7bb90ef7	57724ba9	139d0f26	70494f23
$p^{(14)}$	692c4a40	4a80585b	187e5da3	16c57533	689955b9	3cd52635	13e96788	40803068
$p^{(15)}$	5db27fad	33ea62e1	23c91a2a	48cc15d5	575331b2	60bf1732	5c674a5d	3cd6190a
$p^{(16)}$	0fbf7ae5	2f14185a	6ad630dc	047e26e9	422d0f77	54dc195d	368e05eb	0d6662b5
$p^{(17)}$	79836169	75ef70c5	2cae43f4	2c49396c	3c613693	e13226d5	5bc5e69d	288f3f57
$p^{(18)}$	3a615feb	58d1d14c	00795183	c49baa76	5e9d7604	79f7f59b	19166db2	617207a2
$p^{(19)}$	6b723ed5	f5fe7f4e	401d5fa4	9acbcbfe	038420bc	e3aac878	202e7da1	5b28d301
$p^{(20)}$	440246c3	18d7068f	6be842d6	5039652a	542c5a21	19530314	6bcb5da9	0fc946d4
$p^{(21)}$	0e127504	5f1e7011	28336601	78742718	249e328d	2b0c3dae	11f406bb	5dd55373
$p^{(22)}$	6ad4001c	5a9f6260	4cd460ca	5fa41c20	205913cf	127e075d	003555d6	07cf042f
$p^{(23)}$	67322c45	6d225953	1af46629	0ecc37d7	46cf7da8	01d35159	b428c608	3a2d3d0d
$p^{(24)}$	4fe67839	304d058a	9ffce0a5	09751255	37e6124e	24851e01	591d2784	252a2fd9
$p^{(25)}$	23c3189f	3362c465	d6437d3f	d4bccbbe	507872ed	f78a65dd	aaa618d1	556224c8
$p^{(26)}$	581ecd2f	305c16ce	83fde1d9	6b9f1da2	7a1f06d4	efbfe9b6	5fdcde8c	018136fc
$p^{(27)}$	0c7b1785	50052d8e	0c153981	380717b0	773b727d	06334fbf	728245ee	251f74b1
$p^{(28)}$	1d1842e4	62705d85	34927fe9	19da7c12	50646f07	4d546b8b	39ce12c6	3bb12fec
$p^{(29)}$	4c852466	513e15e2	6d697ca3	6a155ef3	4ff85bb9	5c4603f4	486a5290	30046806
$p^{(30)}$	17965e32	5e7368b9	470e0ff4	73d95c04	1f163002	182f532d	4d67752c	596871e0
$p^{(31)}$	4ad4041e	2cf76301	3f4a3974	0a4a00f8	5ed01d43	4e573590	4f68140b	587675a2
$p^{(32)}$	66fa4037	aa864c4d	49bb6092	6f117408	74ad1b53	4eae041c	5d2462b5	05881d37
$p^{(33)}$	5579473e	7cfe6737	ebed6b2a	912f3f6a	dd8bfb4b	329eae68	96076905	6f3c52cc
$p^{(34)}$	06e8849c	5f456809	102bfd9d	527ab906	a1d33100	72aa5ea1	8ab21c2b	68f50f55
$p^{(35)}$	45c52997	39607312	345919ca	263d7857	3b971002	40276cb6	138a726c	29593908

hash result:

$h(p)$	45d93522	0168bdcd	e830f65a	6e46f3e9	1bb0bbd6	3d37a576	718f4032	0c65079f

Table 2. Example of pair of messages that produce a collision.

8 Conclusions

In this paper, we have explained how to refine the attack [2] in order to produce collisions in PANAMA using only about 2^6 evaluations of the state updating function. As noted in [2], PANAMA gives too many degrees of freedom per round to the attacker.

One could consider to fix PANAMA to be resistant against this type of attack. We have actually done this in [3] and the result is RADIOGATÚN. Its design was based on the insight obtained from the attack in [2] and the possibility of the attack in this paper. This lead us to reduce the number of message words injected each round from 8 to 3, giving an attacker much less freedom per round and requiring more bridges with accompanying dependency problems for a trail with similar complexity. More importantly, we have added feedback from the state to the buffer, making the buffer evolution during hashing become nonlinear. This makes the split in nicely separated subcollisions no longer possible. For more explanations on the evolution from PANAMA to RADIOGATÚN we refer to Appendix A of [3].

Interestingly, our attack on PANAMA can be seen as an application of *trail backtracking* [3]. In this context, we have defined a metric of a trail called its *backtracking depth*. As we explain in [3], the backtracking depth gives a good idea of the number of rounds that must be bridged at the worst point in the trail. The backtracking depth of the trail we used in this paper turns out to be only 2. This suggests that the conditions can be satisfied at any round in the trail by the message words injected immediately before it and those before the previous round and hence that the number of bridges is rather limited. In the design of RADIOGATÚN one of the main criteria is exactly the non-existence of collision trails with low backtracking depth.

References

1. J. Daemen and C.S.K. Clapp, "Fast hashing and stream encryption with PANAMA," *Fast Software Encryption 1998, LNCS 1372*, S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 60-74.
2. V. Rijmen, B. Van Rompay, B. Preneel, J. Vandewalle; "Producing Collisions for PANAMA," *Fast Software Encryption 2001, LNCS 2355*, M. Matsui, Ed., Springer-Verlag, 2002, pp. 37-51.
3. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "RADIOGATÚN, a Belt-and-Mill Hash Function", presented at the NIST Second cryptographic hash workshop, August 2006, available from `http://radiogatun.noekeon.org/`
4. Program to generate collisions for PANAMA, available from `http://radiogatun.noekeon.org/panama`