In Section 4.6.3 we described a "birthday attack" for finding a collision in an arbitrary hash function. If the output length of the hash function is ℓ bits, the attack finds a collision with constant probability using $\Theta(2^{\ell/2})$ hash-function evaluations. A drawback of that approach, however, is that the attack also requires storage of $\Theta(2^{\ell/2})$ hash outputs. Here, we describe an improved attack using roughly the same time as before, but storing only a *constant* number of hash outputs.

The basic idea is similar to one used also (in a different context) in Section 8.1.2. The attack starts by choosing a random initial value $x_0 \in \{0,1\}^{\ell+1}$ and then, for $i = 1, \ldots$, computing $x_i := H(x_{i-1})$ and $x_{2i} := H(H(x_{2(i-1)}))$. (Note that $x_i = H^i(x_0)$ for all i, where H^i here refers to i-fold iteration of H.) In each step the values x_i and x_{2i} are compared; if they are equal, there is a collision somewhere in the sequence $x_0, x_1, \ldots, x_{2i-1}$ and the algorithm runs a sub-routine to find it. (This step is described in more detail below.) The key point is that this algorithm only requires storage of the two hash values x_i and x_{2i} in each iteration.

We formally describe the algorithm and then give a complete analysis.

ALGORITHM 0.1
A small-space birthday attack

Input: A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$
Output: Distinct x, x' with $H(x) = H(x')$

```
x_0 ← {0,1}^{ℓ+1}
x' := x := x_0
for i = 1 to 2^{ℓ/2} + 1:
    x := H(x)
    x' := H(H(x'))
    // Now x = H^i(x_0) and x' = H^{2i}(x_0)
    if x = x' break
if x ≠ x' return fail
x' := x,  x := x_0
for j = 1 to i:
    if H(x) = H(x') return x, x' and halt
    else x := H(x),  x' := H(x')
    // Now x = H^j(x_0) and x' = H^{j+i}(x_0)
```

Let $q = 2^{\ell/2} + 1$ be the upper bound on the number of iterations run by the algorithm. Consider the sequence of values $x_1, \ldots, x_q$, where $x_i = H^i(x_0)$. If we model H as a random function, each of these values is uniformly and independently distributed in $\{0,1\}^\ell$ until the first repeat occurs. (If $x_i = x_j$ then we must have $x_{j+1} = x_{i+1}$.) Using the same analysis as in Lemma A.10, with probability greater than $1/4$ there is some repeat in this sequence; we show that whenever such a repeat is present, our algorithm finds a collision.

Assume there is some repeated value in the sequence $x_1, \ldots, x_q$. The following holds (cf. Claim 8.2):

CLAIM 0.2 *Let $x_1, \ldots, x_q$ be a sequence of values with $x_m = H(x_{m-1})$. If $x_I = x_J$ with $I < J$, then there exists an $i < J$ such that $x_i = x_{2i}$.*

PROOF If $x_I = x_J$, then the sequence $x_I, x_{I+1}, \ldots$ repeats with period $J - I$. (That is, for all $i \geq I$ and integers $\delta \geq 0$ it holds that $x_i = x_{i+\delta(J-I)}$.) Take i to be the smallest multiple of $J - I$ that is greater than or equal to I; that is, $i \stackrel{\text{def}}{=} (J - I) \cdot \lceil I/(J - I) \rceil$. We must have $i < J$ since the sequence $I, I + 1, \ldots I + (J - I - 1)$ contains a multiple of $J - I$. Since $2i - i = i$ is a multiple of the period and $i \geq I$, it follows that $x_i = x_{2i}$. ■

By the claim above, if there is a repeated value in the sequence $x_1, \ldots, x_q$ then there is some $i < q$ for which $x_i = x_{2i}$. But that means that in iteration i of our algorithm, we have $x = x'$ and the algorithm breaks. Next, the algorithm sets $x' := x$ ($= x_i$) and $x := x_0$. The algorithm then proceeds until it finds the *smallest* $j \geq 0$ for which $x_j = x_{j+i}$, and outputs x_{j-1}, x_{j+i-1} as a collision. (Note $j \neq 0$ because $|x_0| = \ell + 1$ and $|x_i| = \ell$ and hence $x_0 \neq x_i$.) Such a j exists because taking $j = i$ works. By construction $H(x_{j-1}) = H(x_{j+i-1})$, and $x_{j-1} \neq x_{j+i-1}$ by minimality of j.