The focus in the claim shifts from assuming and showing existence to assuming access of an actual instance and constructing an actual adversary. If you look back at most of the reductions we have seen so far, most of these actually follow this constructive approach. The proofs consist of a recipe that contains every little detail to build an actual adversary against the target scheme lacking only access to an actual instance of some adversary against an underlying building block. For example, we showed how to construct a message authentication code from a pseudorandom function by giving a concrete attack against the pseudorandomness property of the function given a concrete instance of an adversary against the message authentication code.

While in this book we follow the common practice and will also mostly consider the keyed setting, the human ignorance approach yields a sound justification of constructional results as security claims for unkeyed hash functions used in practice.

4.1.2 A Generic Lower Bound for Collision Resistance

While the human ignorance approach argues that the existence of collisions does not imply that it is easy to find collisions, finding collisions via a brute-force search attack can be simpler than is generally assumed. In particular, this is why hash functions need to have sufficiently large outputs and 128 bits, today, should no longer be considered secure.

The Birthday Bound

Given any (compressing) hash function with $n = \mathsf{H.ol}(\lambda)$ bits output, one can always find collisions in roughly $2^{n/2}$ many hash evaluations, with constant probability. This is known as the *birthday bound*, as it resembles the seeming paradox that among only 23 people there is a high chance that two of them will have the same day and month of birth among the 365 possibilities. In our scenario we can envision the hash values of random inputs as their "birthdays" in a set of 2^n possibilities. Finding a collision now requires to find a birthday match among distinct inputs. Let $\mathsf{q} : \mathbb{N} \to \mathbb{N}$ be a polynomial and set $\mathsf{q} := \mathsf{q}(\lambda)$. A collision-finding algorithm $\mathcal{A}$, which evaluates the hash function q times, could now be given as follows: Adversary $\mathcal{A}$ repeatedly samples random inputs $x_i \leftarrow\!\!\$ \{0,1\}^{\mathsf{H.il}(\lambda)}$ and computes $y_i \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x_i)$ to then check whether the latest value y_i collides with any of the previously generated values y_j, in which case it outputs the collision (x_i, x_j). If after q trials it did not find a collision it stops and outputs $\perp$. Following is the collision-finding algorithm given as pseudocode:

```
$\mathcal{A}(1^\lambda, \mathsf{hk})$
1 : for $i = 1..\mathsf{q}$ do
2 :     $x_i \leftarrow\$ \{0,1\}^{\mathsf{H.il}(\lambda)}$
3 :     $y_i \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x_i)$
4 :     // check for collision
5 :     for $j = 1..(i-1)$ do
6 :         if $y_i = y_j$ then
7 :             return $(x_i, x_j)$
8 : return $\perp$
```

Assume that for any hash key hk the function $\mathsf{H.Eval}(\mathsf{hk}, \cdot)$ distributes random inputs uniformly over the output strings $\{0,1\}^n = \{0,1\}^{\mathsf{H.ol}(\lambda)}$, i.e., $\mathsf{H.Eval}(\mathsf{hk}, X)$ is uniform over $\{0,1\}^n$ for $X \leftarrow\$ \{0,1\}^{\mathsf{H.il}(\lambda)}$ sampled uniformly at random. One can show that this is the worst-case scenario for finding collisions, and we will discuss this at the end of this section. We are interested in an upper bound on the probability that *no* collision $y_i = y_j$ occurs in the attack of our adversary $\mathcal{A}$, from which we can derive a lower bound for the complementary event that a collision has been found. More formally, let noColl_i denote the event that no collision occurs up to, and including, the ith sample x_i. Then we are interested in the probability of $\mathsf{noColl}_\mathsf{q}$. It is convenient to rewrite this probability in terms of conditional probabilities

$$\begin{aligned}\Pr[\mathsf{noColl}_\mathsf{q}] &= \Pr[\mathsf{noColl}_\mathsf{q} \ \wedge \ \mathsf{noColl}_{\mathsf{q}-1}] \\ &= \Pr[\mathsf{noColl}_\mathsf{q} \mid \mathsf{noColl}_{\mathsf{q}-1}] \cdot \Pr[\mathsf{noColl}_{\mathsf{q}-1}],\end{aligned}$$

where the conditional probability represents the likelihood that we do not get a collision in the qth attempt, if we have not encountered a collision in the first $q-1$ samples yet. Iterating this process we get

$$\Pr[\mathsf{noColl}_\mathsf{q}] = \prod_{i=1}^{\mathsf{q}} \Pr[\mathsf{noColl}_i \mid \mathsf{noColl}_{i-1}] \cdot \Pr[\mathsf{noColl}_{i-1}],$$

with $\Pr[\mathsf{noColl}_0] := 1$ by definition.

We next bound the conditional probability from above for each index i. Suppose that our algorithm is at the ith attempt and that no collision has been found so far, implying that all $i-1$ previous hash values $y_1, \ldots, y_{i-1}$ are distinct. Then we can bound the probability that the ith hash evaluation does *not* hit any of the other $i-1$ previous values $y_1, \ldots, y_{i-1}$ by

$$\Pr[\mathsf{noColl}_i \mid \mathsf{noColl}_{i-1}] \leq \left(1 - \frac{i-1}{2^n}\right).$$

The reason is that $i-1$ slots have already been taken by the $i-1$ values $y_1, \ldots, y_{i-1}$ and now the ith value $\mathsf{H}(x_i)$ is independently and uniformly distributed over the 2^n output strings.

From this we can derive an overall bound, where the second inequality is due to the fact $1 - x \leq e^{-x}$ for any real value x:

$$\begin{aligned}\Pr[\mathsf{noColl}_\mathsf{q}] &\leq \prod_{i=1}^{\mathsf{q}} \left(1 - \frac{i-1}{2^n}\right). \\ &\leq \prod_{i=1}^{\mathsf{q}} e^{-\frac{i-1}{2^n}} = e^{-\frac{1}{2^n}\sum_{i=1}^{\mathsf{q}}(i-1)} = e^{-\frac{1}{2^n}\cdot\binom{\mathsf{q}}{2}}.\end{aligned}$$

Conversely, this means that our algorithm $\mathcal{A}$ finds colliding images with probability at least

$$1 - e^{-\frac{1}{2^n}\cdot\binom{\mathsf{q}}{2}}.$$

Since $\binom{\mathsf{q}}{2} = \frac{\mathsf{q}(\mathsf{q}-1)}{2}$ is on the order of q^2, after roughly $\mathsf{q} \approx \sqrt{2^n}$ attempts we find a collision with constant probability. Specifically, for $\mathsf{q} = \lceil 2^{n/2} + 1 \rceil$ the probability is at least

$$1 - e^{-\frac{1}{2}} \geq 1 - 0.61 \geq 0.39.$$

Accounting for Trivial Collisions

Note that we are not done yet, because we still have to deal with the case that algorithm $\mathcal{A}$ chooses $x_i = x_j$ in line 2, that is, it chooses an input value twice. In case of the birthday paradox, where all "persons" x_i are distinct by definition this can of course not happen and we can thus already apply our intermediate result there. The derived bound applies there to $N = 365$, such that one already encounters a collision among roughly $\sqrt{N} \approx 19$ people with high probability. The reason for often finding the value 23 in the literature is that for this value the collision probability exceeds 0.5 (instead of 0.39 as in our bound).

To formally deal with the case of algorithm $\mathcal{A}$ finding a trivial "collision" $y_i = y_j$ with $x_i = x_j$ we have two options. We might consider adapting the algorithm to deterministically iterate over all inputs in which case, however, functions exist that have their collisions "hidden away" towards the end of the iteration. Instead, we next bound the probability of such trivial collisions.

In order to bound the probability of $\mathcal{A}$ finding a trivial "collision" $y_i = y_j$ with $x_i = x_j$ first note that if the input length was actually smaller than the output length n then there would not be much hope. That is, $\mathcal{A}$ would then almost certainly find such a trivial collision first. However, for a compressing function we have $\mathsf{H.il}(\lambda) \geq n + 1$. With this, we can bound the probability that any distinct random samples x_i, x_j accidentally collide. Since there are at most $\binom{\mathsf{q}}{2} = \frac{\mathsf{q}\cdot(\mathsf{q}-1)}{2}$ such pairs, and each pair collides with probability $2^{-\mathsf{H.il}(\lambda)}$,