# Computer Security (Sheet #2)

Marius Gavrilescu

2. Any input of size n cannot collide with an input of size not n, because one hash starts with 0 and the other with 1.

   Two inputs of size n cannot collide: if $g(x) = g(y)$ it means $1||x = 1||y$ which means $x = y$.

   Two inputs of size not n hash to $0||f(x)$ and $0||f(y)$, and we know $f$ is collision resistant.

   However, any hash value starting with 1 can trivially be inverted: if we get the hash value $1||x$ we know $h(x) = 1||x$. This means that for half of the possible outputs of the hash we can easily find a preimage, so the function $g$ is not preimage resistant.

3. Let $c$ be a cryptographic hash function. Let $w$ be the function that truncates all inputs longer than $n$ to $n$ characters, and pads all inputs shorter than $n$ with zeroes to $n$ characters. Note that $w$ is not preimage resistant.

   Then define $g(x) = w(x)$, $f(x) = c(x)$ when $x$ has length $n$, and $f(x) = w(x)$ when $x$ does not have length $n$. Note that $h(x) = c(w(x))$ because $f$'s output is always $n$ characters, and since $c$ is preimage-resistant so is $h$. However $f$ is not preimage resistant, and neither is $g$.

   For collision resistance, assume $g$ is not collision resistant. Then it is easy to find $x$ and $y$ such that $g(x) = g(y)$. But then $f(g(x)) = f(g(y))$, so we just found a collision in $h$! So $g$ must be collision resistant, hence the statement is true.

   The statement is also true for 2nd-preimage resistance with the same proof: if $g$ is not collision resistant then given some $x$, we can find $y$ such that $g(x) = g(y)$ and so $h(x) = h(y)$.

4. We need to store $69^7$ inverses, each taking 7 bytes worst-case (if we store each character as one byte). This is $69^7 * 7$ bytes, or about $52TB$. This is a realistic attack.

   For brute-forcing an inverse, we need (on average) to try half of the $69^7$ inputs. If we spend $100ns$ per try, this takes about 372317 seconds, or 4.3 days.

5. Assume there is an easy-to-find collision in $h$, which means $h(PAD(x)) = h(PAD(y))$ for some known $x$ and $y$. Let $u$ be the last block of $PAD(x)$ and $v$ be the last block of $PAD(y)$ such that $PAD(x) = a||u$ and $PAD(y) = b||v$. Then we have $h(a||u) = h(b||v)$ which is equivalent to $g(h(a), u) = g(h(b), v)$. Since we know $a, b, u, v$ and can compute $h(a), h(b)$, this is a conflict in $g$.

6. Assuming the hash outputs are evenly distributed Eve needs to try half of the possible outputs until she succeeds. This means $2^{127}$ tries.

   If we have a timing side-channel, we can differentiate a message where the first byte of the hash is wrong from a message where the first byte of the hash is correct (as in the second case the bank will take slightly longer to reply).

   Hence we can break the bytes one by one: try each option for the first byte until the reply takes slightly longer to arrive, then try each option for the second byte (using the correct first byte) until the reply takes even longer, and so on.

   This requires about 128 tries per byte, which means 2048 tries in total.