We will augment collision-resistant hashing with a useful property. Consider a collision-resistant hash family (keyed by a key $i$) whose domain $D_i$ can always be written as $M_i \times R_i$ (for example, for the DL-based family studied in class, $D_{p,g,h} = M_{p,g,h} \times R_{p,g,h} = \{1, 2, \ldots, q\}$). We will call this family a trapdoor hash family if it has two additional properties: the algorithm Gen also outputs a trapdoor key $t_i$ in addition to the hash key $i$, and there exists an algorithm $T$ that, on input $(1^k, i, t_i, m_1, r_1, m_2, r_2)$, will output $r_2$ such that $H_i(m_1, r_1) = H_i(m_2, r_2)$ (here, $m_1, m_2 \in M_i$ and $r_1, r_2 \in R_i$) in polynomial time. In other words, given extra information $t_i$, collisions are easy to find in a very strong sense (given one input and half of another, you can find the remaining half).

In the next two problems you will show that some hash families have trapdoors. This means that you may have to trust the person who picked the hash function to not know the trapdoor—because the function is not collision-resistant to anyone who knows the trapdoor. That’s not a problem in the hash-and-sign application of hashing we covered in class, because the signer is the one who picks the hash function. In fact, you will show that the trapdoor can be quite useful.

**Problem 1.** (20 points) Show that the DL-based family studied in class is actually a trapdoor hash family. In other words, demonstrate how to modify Gen, what $t_{p,g,h}$ will be, and construct $T$ that uses $t_{p,g,h}$. (Note that you do not get to modify how the hash function is computed, only how the key is generated. Hint: Gen should no longer be simply selecting $g, h$ blindly at random, but in such a way that an additional piece of information is available to enable collision-finding.)

**Problem 2.** (60 points) Let $p_1 = 2q_1 + 1$ and $p_2 = 2q_2 + 1$ be two distinct safe primes, and $n = p_1 p_2$ be of length $k$ bits.

(a) (10 points) Show by CRT that $QR_n$ is cyclic (the easiest way to do this is to try to construct a generator of $QR_n$ and show that it is indeed a generator by showing that its order is $q_1 q_2$).

(b) (10 points) Show how to efficiently factor $n$ given $q_1 q_2$. In fact, you can factor $n$ given any multiple of $q_1 q_2$, but we won’t show it here.

(c) (15 points) Let $g$ be a generator of $QR_n$, and let $M_{n,g} = R_{n,g} = \{1, 2, \ldots, q_1 q_2\}$. Let $H_{n,g}(m, r) = g^{m2^k + r}$ mod $n$ (note that $m2^k + r$ is just concatenation of $m$ and $r$ as bit strings). Show that this is a collision-resistant hash family under the assumption that such $n$ are hard to factor. Specifically, show how, given $(m_1, r_1)$ and $(m_2, r_2)$ that collide, one can factor $n$. You may use without proof the fact that $n$ can be efficiently factored given any non-zero multiple of $q_1 q_2$.

(d) (15 points) Show also that this is a trapdoor hash family (again, find what trapdoor information Gen should output and what the algorithm $T$ will do).

Trapdoor hash families can be quite useful. For example, if the hash function is chosen by the signer at the same time as she generates her public-secret key pair for the signature scheme (in which case the signer puts $i$ in her public key), then we don’t mind that she knows the trapdoor, since it is in her interests not to reveal it—else, others would be able to find collisions and, therefore, forge signatures on her behalf. Moreover, this enables the signer to perform much of the signature
computaiton ahead of time, before she even knows what message she will be signing, in the following interesting twist of the hash-and-sign paradigm (due to Adi Shamir and Yael Tauman).

Take any signature scheme and modify it as follows. Add the hash key $i$ to your public key, and the trapdoor $t_i$ to your secret key. Before you even know what message you are signing, take a random message $m'$ and value $r'$, and sign the hash $h = H_i(m', r')$ to get $\sigma'$. Then, when the time comes to sign some message $m$, simply run $T$ to find $r$ such that the hash of $(m, r)$ is $h$, and output $(\sigma', r)$ as your signature. Thus, you can precompute most of the signature before you even know the message, and then do only the very quick computation of $T$ once the message is known. This approach is known as “off-line/on-line” signatures and is useful, e.g., when a server has idle cycles to burn some times, and is overloaded at other times.

(e) (10 points) Show that the running time of $T$ for the trapdoor hash family you just built is $O(k^2)$. This makes the on-line part of the signing very efficient — for example, $k$ times faster (i.e., 1,000–2,000 times faster) than RSA.

Problem 3. (20 points) A bit of trivia: a prime $q$, such that $p = 2q + 1$ is also prime, is called a Sophie Germain prime, after a mathematician who lived during 1776–1831 and made significant contributions, even though she was discouraged by her family and denied the opportunity to attend a university because of her gender.

So, let $q$ be a Sophie Germain prime. Show that the following function is a bijection from $QR_p$ to $\{1, 2, \ldots, q\}$ that is efficiently computable and efficiently invertible: $\beta(a) = \{a$ if $a \leq q$, and $p - a$ otherwise}. Hint: use the fact that $p \equiv 3 \pmod{4}$ (but first show why that’s the case).

Show how to use this fact and the discrete-log-based hash function we built in class to get a hash function with domain $\{1, 2, \ldots, q\} \times \{1, 2, \ldots, q\}$ and range $\{1, 2, \ldots, q\}$ (that is, 2 values mod $q$ in the input, 1 value mod $q$ in the output).