Homework 2

1 Problem 1 (30 points)

Sometimes, it is useful to assume stronger properties from a one-way function than just one-wayness. For example, we may want the function to be *collision resistant*, which means that it is hard to find two inputs that map to the same output.

Definition 1 *A function* $H : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$*, where* $n(\lambda) < \lambda$*, is* collision resistant *if, for all PPT adversaries* A*, there exists a negligible* ϵ *such that*

$$\Pr[x_0 \neq x_1 \wedge H(x_0) = H(x_1) : (x_0, x_1) \leftarrow A(1^\lambda)] < \epsilon(\lambda)$$

We usually call such functions *hash functions*.

(a) Assuming the existence of a one-way function f, construct a function $f' : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$ such that (1) f' is compressing ($n(\lambda) < \lambda$), (2) f' is one way, but (3) f' is not collision resistant. The domain and range of f' are allowed to be different than the domain and range of f.

(b) Show that any collision resistant hash function is also a one-way function. (Hint: since a collision resistant hash function is compressing, there are guaranteed to be many collisions. Show how to use a one-way function adversary to find such a collision).

(c) Suppose we relax our computational model to allow for *non-uniform* adversaries A. That is, A consists of a PPT algorithm A', as well as an infinite sequence of advice strings $a_1, a_2, \ldots$ where the bit-length of a_i is polynomial in i. On inputs of length λ, A runs A' on whatever input it was given, as well as the advice string a_λ.

Show that collision resistant hash functions are impossible against non-uniform adversaries.

While non-uniform adversaries are perhaps an unrealistic model of computation, the above motivates a slightly different notion of collision resistant hash functions. Here, the hash function is keyed, and an adversary given the key must try to find a collision with respect to that key.

Definition 2 *A keyed function* $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$*, where* $n(\lambda) < m(\lambda)$*, is* collision resistant *if, for all PPT adversaries* A*, there exists a negligible* ϵ *such that*

$$\Pr[x_0 \neq x_1 \wedge H(k, x_0) = H(k, x_1) : (x_0, x_1) \leftarrow A(k), k \xleftarrow{\$} \{0,1\}^\lambda] < \epsilon(\lambda)$$

(d) Explain why your non-uniform attack from part (c) does not apply to keyed hash functions.

(e) Given a keyed collision resistant hash function $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ with $n(\lambda) < m(\lambda)$, show how to construct a new keyed collision resistant hash function $H' : \{0,1\}^\lambda \times \{0,1\}^{m'(\lambda)} \to \{0,1\}^{n(\lambda)}$ where $m'(\lambda)$ is any arbitrary polynomial in λ. In particular, even if H had $m(\lambda) < \lambda$, H' can have $m(\lambda) \gg \lambda$. To do so, each evaluation of H' will run H several times.

2 Problem 2 (15 points)

In class, we tried to build a signature scheme from any one-way function. However, we ran into a roadblock, where we needed a one-time signature scheme whose message space was much larger than its public key. Here, we will use hashing to solve the problem.

Let $(\mathtt{Gen}, \mathtt{Sig}, \mathtt{Ver})$ be a one-time signature scheme with public keys of length $p(\lambda)$ and messages of length $n(\lambda)$, where $n(\lambda)$ may be smaller than $p(\lambda)$. Let $H : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ be a keyed hash function, where $m(\lambda)$ is much larger than $p(\lambda)$. Define $(\mathtt{Gen}', \mathtt{Sig}', \mathtt{Ver}')$ as the following signature scheme for messages of length $m(\lambda)$:

- $\mathtt{Gen}'(1^\lambda)$: run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Gen}(1^\lambda)$. Choose a random $k \leftarrow \{0,1\}^\lambda$. Output $(\mathsf{sk}' = (\mathsf{sk}, k), \mathsf{pk}' = (\mathsf{pk}, k))$.
- $\mathtt{Sig}'(\mathsf{sk}', M) = \mathtt{Sig}(\mathsf{sk}, H(k, M))$. That is, first hash the message with H (using the hash key k), and then sign using $\mathtt{Sig}$.
- $\mathtt{Ver}'(\mathsf{pk}', M, \sigma) = \mathtt{Ver}(pk, H(k, M), \sigma)$.

(a) Show that, if H is collision resistant and $(\mathtt{Gen}, \mathtt{Ver}, \mathtt{Sig})$ is one-time EUF-CMA secure, then so is $(\mathtt{Gen}', \mathtt{Ver}', \mathtt{Sig}')$. Therefore, by combining with with Problem 1, we have a one-time signature scheme from any collision resistant hash function, where the message space is much larger than the public key size. This is sufficient for building a full signature scheme as we saw in class.

(b) Show that the collision resistance of H is also necessary for security. That is, if H is *not* collision resistant (but still compressing), then $(\mathtt{Gen}', \mathtt{Sig}', \mathtt{Ver}')$ cannot possibly be a secure one-time signature scheme.

Collision resistant hash functions are widely believed to exist, and there are many constructions based on number theory. However, it is also widely believed that a generic one-way function is not sufficient to build a collision resistant hash function. Therefore, we are still short of our goal of constructing signatures from arbitrary one-way functions. Fortunately, a slightly weaker notion of collision resistant hashing functions, called *universal one-way hash function (UOWHF)*, is possible from one-way functions, and is sufficient to build signature schemes, albeit with a slight tweak to the construction above.

3 Problem 3 (35 points)

Shamir secret sharing solves the t-out-of-n secret sharing problem, where we wish to divide a secret amongst n people, such that any t of them can reconstruct the secret, but no $t-1$ of them can. Here, we will investigate generalizations to more complex access structures.

(a) In a weighted threshold secret sharing scheme, each user i has a non-negative integer weight w_i, and there is a threshold $t \geq 0$. The goal is to secret share a message to n users, such that a set of users S can reconstruct the secret if and only if the total weight of the users in S, $w_S = \sum_{i \in S} w_i$, is at least t.

Show how to modify Shamir secret sharing so that it works for thresholds. In the case where all weights are a polynomially bounded, your scheme should be efficient.

More general structures are also possible. In general, a secret sharing scheme is specified by a collection $\mathcal{C}$ of *allowable* subsets S. $\mathcal{C}$ is called an *access structure*. We want that any set $S \in \mathcal{C}$ can reconstruct the secret, but any $S \notin \mathcal{C}$ cannot reconstruct the secret. For this to make sense, $\mathcal{C}$ must be *monotone*, meaning if $S \in \mathcal{C}$ and $S \subseteq S'$, then $S' \in \mathcal{C}$. This is because S' could just pretend it was actually S, and ignore the shares for users in $S' \setminus S$.

A monotone boolean formula is a circuit consisting of AND and OR (no NOT gates) where the fanout of each gate is one (so that the output wire of each gate feeds into exactly one gate).

Associate subsets $S \subseteq [n]$ with bit strings $x \in \{0,1\}^n$, where $x_i = 1$ if and only if $i \in S$. Let F be a monotone boolean formula on n-bit inputs. Let $\mathcal{C}$ be the access structure defined as the set of S such that $F(S) = 1$. Since F is monotone, so is $\mathcal{C}$.

(b) Construct a secret sharing scheme for the access structure $\mathcal{C}$. The size of shares, and the running times for sharing and reconstruction, should be polynomial in the size of F.

(Hint: first devise a secret sharing scheme for the special cases where F is a single gate. Your overall scheme will consist of many secret sharings, one for each gate, appropriately stitched together. The property of the scheme should be that a set of users S can reconstruct the secret for a particular gate if and only if that gate outputs 1 when F is applied to S.)

Your answer to part (b) can be used to construct secret sharing schemes for arbitrary access structures, since for any monotone access structure $\mathcal{C}$, there is a monotone formula F that decides $\mathcal{C}$. Unfortunately, however, F may be exponentially large and the scheme will therefore not be efficient and will require exponentially large shares.

However, if we rely on computational assumptions, we can do better. A monotone circuit is a circuit consisting of AND and OR gates (no NOT gates). The difference from a formula is that each output wire can feed into many other gates. Let C be a monotone circuit, and $\mathcal{C}$ be the access structure defined by C: $S \in \mathcal{C}$ if and only if $C(S) = 1$.

(c) Construct a secret sharing scheme for the access structure $\mathcal{C}$. For concreteness, you may consider the secret to be a single bit b. We no longer ask for perfect secrecy of b, but instead only computational. That is, for any set $S \notin \mathcal{C}$, given the shares for users in S, it should be computationally infeasible to distinguish a sharing of $b = 0$ from a sharing of $b = 1$.

(Hint: take the circuit C, and write it as a sequence of AND, OR, and FANOUT gates. AND and OR are exactly as you'd expect, except they only have a single output wire that can feed into only a single gate. To allow for more fanout, a FANOUT gate takes as input a single wire, and duplicates it on several output wires.

Your scheme will treat AND and OR gates as in part (b), but will have to do something different for FANOUT gates. To handle FANOUT gates, you may assume one-way functions, or anything implied by one-way functions (PRGs, PRFs, secret key encryption, etc).)

4 Problem 4 (15 points)

In class, we discussed fairness in MPC protocols, where either all parties learn the output, or none of them do. We also discussed how in general, this seems impossible: the receiver of the second-to-last message must know the output (since she will not

receive any more messages). Therefore, if that user is corrupt, she can abort and not send the last message, preventing the recipient from learning the message.

One possible way around this difficulty is to slightly relax fairness. Suppose the output is a single bit. Instead of saying either all parties learn the output or none of them do, instead we will say that at any point in the protocol, all the parties have a guess of what the output would be, but they will have only limited confidence in the guess. At the beginning, the guess will be completely uncorrelated with the actual output, and at the end, the guess will be perfectly (or at least statistically) correct. In each round, the guess is gradually refined. If a malicious user aborts prematurely, she will only have her imperfect guess, and the other users will have a guess with similar accuracy.

Now our last round adversary might learn the bit with certainty, but then all other users will have a pretty good idea of what the output is supposed to be.

Suggest how this idea might be implemented. You may assume a MPC protocol for arbitrary functionalities. You do not need to formally prove that the protocol achieves the desired fairness goal, but you should explain informally why it should.