1 Fun with Collision-Resistant Hash Functions

a. We wish to construct an $H_a$ that is collision-resistant such that $H'_a$ (which drops the LSB of $H_a$) is not.

First, we fix two distinct strings $x_1, x_2$. Define $h_1 h_2 \ldots h_k := H(x_1)$, where $h_i$ is the $i$th bit of $H(x_1)$. We will define our first attempt of a hash function $H'_a(x) = h_1 \circ h_2 \circ \cdots \circ h_{k-1} \circ \overline{h_k}$, if $x = x_2$

$H(x)$ otherwise.

In other words, $H'_a$ outputs $H(x_1)$ with its last bit flipped on input $x_2$; otherwise, it simply outputs $H(x)$.

We can see that this is a step in the right direction: $H'_a$ seems intuitively collision-resistant, since we are only changing the output at one point, and it is true that if we drop the LSB then we can find a collision at $x_1, x_2$. But we find that there is a problem – while it isn’t obvious how to find a collision for $H'_a$, it is difficult to prove that we cannot. If our adversary returns a collision $(m_1, x_2)$ for $H'_a$, then we do not know how to find a collision for $H$ since $H'_a(x_2)$ is hardcoded and does not tell us anything about a particular input to $H$. This can break our entire reduction: the adversary knows $x_2$, so it is perfectly allowed to focus exclusively on finding a collision there, and thus $H'_a$ may not be collision-resistant even if $H$ is.

So let us try again. Define

$$H_a(x) = \begin{cases} h_1 \circ h_2 \circ \cdots \circ h_{k-1} \circ \overline{h_k}, & \text{if } x = x_2 \\ H(x) & \text{if } H(x) \neq H_a(x_2) \\ H(x_2) & \text{if } H(x) = H_a(x_2) \end{cases}$$

In other words, $H_a$ outputs $H(x_1)$ with its last bit flipped on input $x_2$. Otherwise, on input $x$, $H_a$ calculates $H(x)$ and checks if it is equal to $H_a(x_2)$: if it is, it outputs $H(x_2)$, otherwise it outputs $H(x)$.

Assume for the sake of contradiction that $H_a$ is not collision-resistant, i.e. there exists an adversary $\mathcal{A}$ which can find a collision with non-negligible probability. Let us define an adversary $\mathcal{B}$ which finds a collision for $H$ with non-negligible probability.
Note first that, by construction, $H_a$ outputs $H_a(x_2)$ at exactly one input, so there does not exist a collision for $H_a$ where one of the inputs is $x_2$. Let $B$ run $A$, finding $a \neq b$ such that $H_a(a) = H_a(b)$. There are then three cases:

**Case 1** $H_a(a) = H_a(b) = H(x_2)$ and $H(a) = H(b) = H_a(x_2)$. $B$ outputs $(a, b)$.

**Case 2** $H_a(a) = H_a(b) = H(x_2)$ and WLOG $H(a) \neq H_a(x_2)$. Then by construction, $H_a(a) = H(a)$, so $B$ outputs $(a, x_2)$.

**Case 3** $H_a(a) \neq H(x_2)$ and $H_a(b) \neq H(x_2)$. Then we know that $H_a(a) = H(a)$ and $H_a(b) = b$, so $B$ outputs $(a, b)$.

Thus $B$ is able to find a collision whenever $A$ can, so $H_a$ is collision-resistant.

$H'_a$ is not collision-resistant. $x_1 \neq x_2$, yet $H'_a(x_1) = H'_a(x_2)$ by construction.

b. Assume for the sake of contradiction that one of $H_1, H_2$ is collision-resistant (WLOG, $H_1$), but $H_b = (H_1(m), H_2(m))$ is not collision resistant. Since $H_b$ is not collision-resistant, there exists an adversary $A$ that can find a collision with non-negligible probability. Let us define an adversary $B$ that finds a collision in $H_1$ with that same probability.

$B$ runs $A$, getting a collision $m_1 \neq m_2$ such that $H_b(m_1) = (x, y) = H_b(m_2)$. By definition of $H_b$,

$$x = H_1(m_1) = H_2(m_2)$$
$$y = H_2(m_1) = H_2(m_2),$$

so $B$ outputs $(m_1, m_2)$ as well, and is correct just as often as $A$. Thus, $H_b$ is collision resistant when either of $H_1, H_2$ is.

c. Let $H_1$ be a CRHF but $H_2$ be the constant function $H_2(x) = 0^k$. Then $H_c$ is not collision resistant, as it is the constant function $H_c(x) = H_1(0^k)$. If we switch the definitions of $H_1, H_2$, we get that $H_c(x) = 0^k$. So even if one of $H_1, H_2$ is a CRHF, there exist choices for the other function that make $H_c$ not a CRHF.

d. **Proof 1** Assume for the sake of contradiction that $F_d$ is not a PRF. In other words, there exists an adversary $A$ which can distinguish between $F_d$ and a random function with non-negligible probability $\epsilon(k)$. Let us define an adversary $B$ which can distinguish between $F$ and a random $R$ with non-negligible probability.

Our adversary $B$ is given access to oracle $O_B$, which is either $F(k, \cdot)$ or a truly random function $R$. $B(1^k)$ runs $A(1^k)$. If $A$ queries $x_i$, then $B$ responds with $O_B(H(x_i))$. $B$ then outputs whatever $A$ outputs.

If $O_B$ was $F$, then $B$ will act just like $F_d$ on $A$’s oracle queries, so $B$ will be correct with the same probability as $A$. We now need to show that, if $O_B$ is $R: \{0,1\}^k \mapsto \{0,1\}^k$, then $A$ will act as if it was given oracle access to a random function $\{0,1\}^k \mapsto \{0,1\}^k$.  

Homework 9 Solutions
Since \( H \) is collision-resistant, we know that \( A \) will only select \( x_i \neq x_j \) such that 
\[
H(x_i) = H(x_j)
\]
with negligible probability. Since \( R \) is random, \( A \) gains no information about \( H(x_i) \) from \( R(H(x_i)) \), so \( A \) only has an additional advantage when it does happen to find a collision for \( H \).

Thus, \( B \) is correct with probability \( \epsilon(k) - \nu(k) \) for negligible probability \( \nu \), and thus \( F \) is not a PRF. But this contradicts our definition of \( F \), so \( F_d \) must be a PRF.

## 2 Adaptive Security

a. Define \((G^*, E^*, D^*)\) by:\(^1^2\)
\[
G^*(1^k) = G(1^k)
\]
\[
E^*(pk, m) = \begin{cases} 
0^k & \text{if the first } k \text{ bits of } pk \text{ are equal to } m \\
E(pk, m) & \text{otherwise}
\end{cases}
\]
\[
D^*(sk, c) = \begin{cases} 
\text{first } k \text{ bits of } pk & \text{if } c = 0^k \\
D(sk, c) & \text{otherwise}
\end{cases}
\]

This cryptosystem satisfies Definition 2, but not Definition 1. First, we note that any secure cryptosystem must satisfy that the range of \( G(1^k) \) has larger-than-polynomial size in \( k \), as otherwise an adversary could enumerate them all and decrypt any message in polynomial time. This means that a non-adaptive adversary Eve would have only a negligible chance of seeing \( E^*(pk, pk) \), and thus our contrived cryptosystem is still non-adaptively semantically secure.

On the other hand, an adaptive adversary can output the first \( k \) bits of \( pk \) as their chosen message, ensuring that they will always see \( 0^k \) when looking at a real ciphertext. Thus, our contrived cryptosystem is not adaptively semantically secure.

b. Definition 1 does imply Definition 2.

**Proof 2 (Proof by contradiction.)** Assume for the sake of contradiction that there exists a cryptosystem \((G, E, D)\) that satisfies Definition 1, but does not satisfy Definition 2. Thus, for every Sim there must exist some poly-length message sequence \( M \) and some adversary \( A \) which can succeed in the experiment in Definition 3 with non-negligible advantage. Let us build an adversary \( B \) which breaks the adaptive semantic security of the cryptosystem.

\(^1\)In the case that \( E(pk, m) = 0^k \) for some other \( m \), we can have \( E^*(pk, m) \) output \( E(pk, pk) \) instead. We assume for simplicity that no such \( m \) exists.

\(^2\)WLOG we assume that \(|pk| \geq k\); otherwise, we could simply pad \( pk \) with 0’s.
Let us fix a particular simulator Sim. Assume that B has \( m_k \) hardcoded into its description when run on \( 1^k \). B, in the first round, return \( m_k \) as its chosen message. Once it gets the encryption \( c \), B runs A with input \( c \) and output whatever it outputs. Since we are attempting to distinguish from the same simulator as A, and the ciphertext we give to A is the same distribution as that given in the Real distribution of Definition 1, we will be correct with the same probability as A. This breaks Definition 2, and thus we must have that Definition 1 implies Definition 2.

3 Nested Encryption

**Proof 3** Recall that semantic security is equivalent to indistinguishability-security. Assume for the sake of contradiction that our nested encryption scheme \((\text{KeyGen}', \text{Enc}', \text{Dec}')\) is not ind-secure, and thus there exists some PPT adversary \( A' \) that can distinguish between encryptions of messages for \( m_0, m_1 \) with non-negligible advantage. That is, \( C'_0(1^k) \approx C'_1(1^k) \), where

\[
C'_b(1^k) = \{(PK', SK') \leftarrow \text{KeyGen}'(1^k); c' \leftarrow \text{Enc}'(m_b, PK'); (1^k, PK', c', \ell(k))\}.
\]

We will show via a reduction that we can use such a PPT to create an adversary A that breaks the ind-security of \((\text{KeyGen}, \text{Enc}, \text{Dec})\) with non-negligible advantage.

On input \((1^k, PK, c, \ell(k))\), A runs KeyGen\((1^k)\) to get a keypair \((PK_2, SK_2)\). A then encrypts \( c \) again with this public key, finding \( z = \text{Enc}(c, PK_2) \). A runs \( A'(1^k, (PK, PK_2), z, \ell(k)) \) and returns whatever it returns.

If A was given an encryption \( c \) of \( m_b \), then \( z \) is the encryption \( \text{Enc}'(m_b, (PK, PK_2)) \); thus, \( A' \) is given something from the distribution \( C'_b \) (and, due to the correctness of the encryption scheme, assuredly not from the distribution \( C'_b \neq 1 \)) and will be correct with nonnegligible advantage. But the original cryptosystem was ind-secure, and this is a contradiction; thus, \( A' \) cannot exist, and so \((\text{KeyGen}', \text{Enc}', \text{Dec}')\) must be ind-secure. By the equivalence of indistinguishability security and semantic security, the nested encryption scheme is also semantically secure.

4 Cramer-Shoup Cryptosystem

a. We can construct this adversary as follows: on input \( PK \), A is allowed to spend its first interactive phase make as many queries as it wants, although none are necessary. In the end, A will output two messages \( m_0, m_1 \) chosen at random from the message space (as well as the necessary state information \( s \)). Upon receiving \( m_0, m_1, s \) and a ciphertext

\[
C = (g^r, h^r, A^r \cdot m_b, B^r)
\]
for $b \in \{0, 1\}$, $A$ will pick a random value $r' \leftarrow \mathbb{Z}_q$ and compute

$$C' = (g^{r} \cdot g^{r'}, h^{r} \cdot h^{r'}, A^{r} \cdot m_b \cdot A^{r'}, B^{r} \cdot B^{r'}).$$

Since this is re-randomized, it will be a different ciphertext and so the decryption oracle will decrypt it properly and output the underlying plaintext $m_b$. $A$ can then simply check if $m_0 = m_b$ and output 0 if and only if this equality holds. Because the ciphertext $C'$ is a valid ciphertext and distinct from $C$, the decryption oracle will always decrypt it correctly and return the message $m_b$. This means $A$ succeeds with probability 1 and thus breaks the CCA-2 security of the cryptosystem.

b. A well-formed ciphertext will be of the form

$$C = (g^{r}, h^{r}, A^{r} \cdot m, (B C^β)^r)$$

where

$$β = H(g^{r}, h^{r}, A^{r} \cdot m).$$

If we write this as a tuple of the form $(R, S, P, T)$, we see that

$$T = (B C^β)^r$$

$$= ((a^b h^b)(g^w h^z)^β)^r$$

$$= ((a^b h^b)(g^{βw} h^{βz})^r$$

$$= (g^{a+βw} h^{b+βz})^r$$

$$= (g^r)^{a+βw} (h^r)^{b+βz}$$

$$= R^{a+βw} S^{b+βz},$$

so that the cryptosystem really is correct.