## 1 Midterm (MI) Security

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pt</td>
<td>Saying that MI7 =&gt; MI6</td>
</tr>
</tbody>
</table>
| 3 pts | Reduction proving that MI7 => MI6 (not MI6 => not MI7)  
(2 pts for reduction, 1 pt for analysis) |
| 1 pt  | Saying that MI6 does not necessarily => MI7 |
| 3 pts | Counterexample  
(2 pts for counterexample, 1 pt for analysis) |
| 8 pts | |

## 2 More Fun with PRGs

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pt</td>
<td>Saying that Ga is a PRG</td>
</tr>
</tbody>
</table>
| 2 pts | Reduction proving that Ga is a PRG (Ga not PRG => G not PRG)  
(1 pt for reduction, 1 pt for analysis) |
| 1 pt  | Saying that Gb is not necessarily a PRG |
| 2 pts | Argument why, or counterexample for, Gb |
| 6 pts | |

## 2 Even More Fun with OWFs

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pt</td>
<td>Saying that fa is a OWF</td>
</tr>
</tbody>
</table>
| 3 pts | Reduction proving that fa is a OWF (fa not OWF => g not OWP)  
(1 pt for reduction, 2 pts for analysis) |
| 1 pt  | Saying that fb is not necessarily a OWF |
| 3 pts | Counterexample for fb  
(1 pt for counterexample, 2 pts for analysis) |
| 1 pt  | Saying that the “proof” is wrong |
| 1 pt  | Explaining why the “proof” is wrong |
| 10 pts | |
4 Breaking our Assumptions

<table>
<thead>
<tr>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 pt</td>
<td>RSA warmup (extended Euclidean algorithm)</td>
</tr>
</tbody>
</table>
| 3 pts  | Algorithm to compute d’  
(1 pt for algorithm, 2 pts for analysis) |
| 1 pt   | QR warmup (raise to (p-1)/2 mod p and (q-1)/2 mod q) |
| 3 pts  | Algorithm to distinguish QR and QNR  
(1 pt for algorithm, 2 pts for analysis) |
| 2 pts  | Extending scheme to handle even alpha  
(1 pt for scheme, 1 pt for analysis) |
| 10 pts | |

5 Hardcore Bits and OWFs

<table>
<thead>
<tr>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 pts</td>
<td>Reduction (hardcore bit but not OWF =&gt; not hardcore bit)</td>
</tr>
<tr>
<td>2 pts</td>
<td>Analysis of correctness</td>
</tr>
<tr>
<td>1 pt</td>
<td>Analysis of probabilities</td>
</tr>
<tr>
<td>6 pts</td>
<td></td>
</tr>
</tbody>
</table>

Total

40 pts
1 Midterm (MI) Security

MI7 security implies MI6 security, but the converse is not true. To prove this, we show that (1) any MI7-secure symmetric-key cryptosystem is MI6-secure, and (2) that there exists an MI6-secure cryptosystem that is not MI7-secure.

(1) Let $(\text{KeyGen}, \text{Enc}, \text{Dec})$ be an MI7-secure symmetric-key cryptosystem. We prove our claim (1) by reduction. Let $\mathcal{A}$ be a p.p.t. adversary that breaks MI6 security. That is, $\mathcal{A}$ wins the MI6 experiment with nonnegligible advantage $\varepsilon$. Now we construct $\mathcal{B}$ as follows to break MI7 security:

\begin{itemize}
  \item[(i)] If $i = 1^k$, run $\mathcal{A}(1^k)$ to get $(m_0, m_1, s)$ and output this same tuple.
  \item[(ii)] If $i = (1^k, z = (c_a, c_b), s)$, run $\mathcal{A}(1^k, c_a, s)$ to get $b'$. Output $b'$.
\end{itemize}

We claim that if $\mathcal{A}$ succeeds, then $\mathcal{B}$ succeeds. Note that if $\mathcal{A}$ produces $b' = 0$, $\mathcal{A}$ is guessing that $c_a = c_0$. $\mathcal{B}$ accordingly guesses that it was given $(c_0, c_1)$. On the other hand, if $\mathcal{A}$ produces $b' = 1$, $\mathcal{A}$ is guessing that $c_a = c_1$, and $\mathcal{B}$ accordingly guesses that it was given $(c_1, c_0)$.

Thus, if $\mathcal{A}$’s advantage in the MI6 experiment is $\varepsilon$, then $\mathcal{B}$’s advantage in the MI7 experiment is also at least $\varepsilon$. Since $\varepsilon$ is assumed to be nonnegligible, this implies that $\mathcal{B}$ breaks MI7 security.

\hfill $\blacksquare$
Consider the following cryptosystem $Q = (\text{KeyGen}, \text{Enc}, \text{Dec})$:

- **Key generation**: $\text{KeyGen}(1^k)$ outputs $sk \leftarrow \{0,1\}^k$ chosen uniformly at random.
- **Encryption**: Given $m \in \{0,1\}^k$, $\text{Enc}(sk, m)$ outputs $c \leftarrow m + sk \pmod{2^k}$.
- **Decryption**: $\text{Dec}(sk, c)$ outputs $m \leftarrow c - sk \pmod{2^k}$.

We claim that $Q$ is MI6-secure but not MI7-secure\(^1\). In the following analysis, the $(\mod 2^k)$ is implicitly included in any arithmetic, but is not explicitly written.

$Q$ is MI6-secure because given any $m_0$, $m_1$, and $c \leftarrow \text{Enc}(sk, m_b)$ for $b \leftarrow \{0,1\}$, a p.p.t. adversary $A$ cannot do better than guessing at random (and having a success probability of $\frac{1}{2}$). This is because if $m_0 \neq m_1$, then either $c = m_0 + sk_0$ or $c = m_1 + sk_1$, where $sk_0$ and $sk_1$ are equally valid and equally likely to have been chosen by $\text{KeyGen}(1^k)$. If $m_0 = m_1$, $c$ is totally independent of $b$, and $A$ certainly cannot do better than guessing at random.

However, $Q$ is not MI7-secure. Consider a p.p.t. adversary $A$ defined as follows:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>If $i = 1^k$, choose $m_0 \leftarrow {0,1}^k$ and $m_1 \leftarrow {0,1}^k \setminus {m_0}$. Set $s = m_0 \circ m_1$.</td>
</tr>
<tr>
<td>(ii)</td>
<td>If $i = (1^k, z = (c_a, c_b), s)$, recover $m_0$ and $m_1$ by parsing $s$. If $c_a - m_0 = c_b - m_1$, output 0. Else if $c_a - m_1 = c_b - m_0$, output 1.</td>
</tr>
</tbody>
</table>

First note that $A$ runs in probabilistic polynomial time. In step (ii), $A$ first checks if $c_a - m_0 = c_b - m_1$. Observe that if $(c_a, c_b) = (c_0, c_1)$, then $c_a = m_0 + sk$ and $c_b = m_1 + sk$. Then $c_a - m_0 = sk = c_b - m_1$, and the first check passes, so $A$ correctly outputs 0. In the other case, if $(c_a, c_b) = (c_1, c_0)$, then $c_a = m_1 + sk$ and $c_b = m_0 + sk$. Then $c_a - m_1 = sk = c_b - m_0$ and the second check passes, so $A$ correctly outputs 1. $A$ succeeds with nonnegligible probability, so $Q$ is not MI7-secure.

Thus, we have exhibited a cryptosystem $Q$ that is MI6-secure but not MI7-secure. MI6 security does not imply MI7 security.\(\blacksquare\)

---

\(^1\)Much like the real $Q$. 

Midterm Solutions Page 2 / 8
2 More Fun with PRGs

Solution written by Giovanni Pittalis.

a. Let us define the following distributions:

\[ D(k) = \{ x \leftarrow \{0,1\}^k : G(x) \} \]
\[ D_a(k) = \{ x \leftarrow \{0,1\}^k : G_a(x) \} \]
\[ R(k) = \{ x \leftarrow \{0,1\}^k : x \} \]

By definition, since \( G \) is a length-doubling PRG, we know that \( D(k) \approx R(2k) \). We will then use this in a reduction to show that \( G_a \), where \( G_a(x) = G(G(x)) \), is also a PRG. Assume for the sake of contradiction that \( G_a \) is not a PRG, and let ppt \( A \) distinguish between \( D_a(k) \) and \( R(4k) \) with non-negligible advantage. Consider then ppt \( B \):

<table>
<thead>
<tr>
<th>B, on input ( y ):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let ( y' = G(y) ).</td>
</tr>
<tr>
<td>2. Run ( A(y') ):</td>
</tr>
<tr>
<td>i. If ( A ) outputs 1 (choosing ( D_a(k) )), output 1 (choosing ( D(k) ))</td>
</tr>
<tr>
<td>ii. If ( A ) outputs 0 (choosing ( R(4k) )), output 0 (choosing ( R(2k) ))</td>
</tr>
</tbody>
</table>

Whenever \( y \leftarrow D(k) \), then \( y = G(x) \) for random \( x \). In this case the input to \( A \) is \( y' = G(G(x)) \) for random \( x \), which is identically distributed to \( D_a(k) \). Furthermore, whenever \( y \leftarrow R(2k) \), then \( y = x \) for random \( x \). In this case the input to \( A \) is \( y' = G(x) \) for random \( x \in \{0,1\}^{2k} \). Since \( G \) is a PRG, this distribution is computationally indistinguishable from \( R(4k) \). Therefore, it follows that if \( A \) has a non-negligible advantage in distinguishing between \( D_a(k) \) and \( R(4k) \), then \( B \) must have the same non-negligible advantage in distinguishing between \( D(k) \) and \( R(2k) \). We note that \( B \) is a ppt because running \( G \) to generate \( y' \) takes polynomial time, and \( A \) is ppt. This means that \( D(k) \approx R(2k) \), which contradicts the fact that \( G \) is a PRG, thus we conclude that \( G_a \) is also a PRG.
b. In addition to the distributions defined in part a., let us define the following:

\[ D_b(k) = \{ x \leftarrow \{0,1\}^k : G_b(x) \} \]

Consider the definition of \( G_b(x) \), which outputs the number of 0’s in \( G(x) \) written as a poly\((k)\)-bit string. Let \( n_0 \) be the number of 0’s in \( G(x) \). Since \( G : \{0,1\}^k \rightarrow \{0,1\}^{2k} \), we have that \( 0 \leq n_0 \leq 2k \), which means \( n_0 \) can be fully represented in \( \log(2k) \) bits. If then \( n_0 \) is written as a poly\((k)\)-bit string, it follows that all the bits except the rightmost \( \log(2k) \) bits will always be zero, i.e. the leftmost \( \text{poly}(k) - \log(2k) \) bits will always be zero. We can use this fact to build a ppt \( A \) that distinguishes between \( D_b(k) \) and \( R(\text{poly}(k)) \):

\[
A, \text{ on input } y:
\begin{align*}
1. & \text{ Output 1 (choosing } D_b(k)) \text{ if the leftmost } \text{poly}(k) - \log(2k) \text{ bits of } y \text{ are all 0.} \\
2. & \text{ Else, output 0 (choosing } R(\text{poly}(k)).)
\end{align*}
\]

Note that \( A \) is ppt because checking the leftmost \( \text{poly}(k) - \log(2k) \) bits of \( y \) takes polynomial time in the length of \( y \).

Let \( p_{D_b,0}(k) \) and \( p_{R,0}(k) \) be the probabilities that \( A(y) \) outputs 0 given that \( y \leftarrow D_b(k) \) and \( y \leftarrow R(\text{poly}(k)) \) respectively. Whenever \( y \leftarrow D_b(k) \), we argued that the leftmost \( \text{poly}(k) - \log(2k) \) bits of \( y \) will always be all 0’s, therefore \( p_{D_b,0}(k) = 0 \). On the other hand, if \( y \leftarrow R(\text{poly}(k)) \), we have the following:

\[
p_{R,0}(k) = \Pr[\text{At least 1 bit in the leftmost } \text{poly}(k) - \log(2k) \text{ bits of } y \text{ is 1}| y \leftarrow R(\text{poly}(k))]
= 1 - \Pr[\text{All leftmost } \text{poly}(k) - \log(2k) \text{ bits of } y \text{ are 0’s}| y \leftarrow R(\text{poly}(k))]
= 1 - \frac{2^{\log(2k)}}{2^{\text{poly}(k)}}
= 1 - \frac{2k}{2^{\text{poly}(k)}}
\]

It then follows that \( |p_{D_b,0}(k) - p_{R,0}(k)| = \left| 0 - 1 + \frac{2k}{2^{\text{poly}(k)}} \right| = 1 - \frac{2k}{2^{\text{poly}(k)}} \), which is non-negligible. This means that ppt \( A \) can distinguish \( D_b(k) \) and \( R(\text{poly}(k)) \) with non-negligible advantage, from which we conclude that \( G_b \) is not a PRG.
3 Even More Fun with OWFs

Solution written by Peter Eastwood.

a. Yes, it follows. Suppose \( f_a \) not an OWF, with a p.p.t. inverter \( \mathcal{A} \) with nonnegligible success chance \( \eta(k) \). Then we can simply construct an inverter \( \mathcal{B} \) for \( g \) which takes a string \( z \), overwrites the first bit with a 0, and gives it to \( \mathcal{A} \), returning \( \mathcal{A} \)'s output (clearly p.p.t.). \( \mathcal{B} \)'s probability of success is

\[
2^{-k} \sum_{y \in \{0,1\}^k} \alpha_k(y) \cdot p_0(h(y)) + \alpha_1(y) \cdot p_1(h(y)) = 2^{-k} \sum_{y' \in \{0,1\}^{k-1}} p_0(0y') + p_1(0y') = 2^{-k} \cdot 2^{-k+1} \eta(k) = \frac{1}{2} \eta(k)
\]

where \( \alpha_i(x) \) denotes the indicator function for \( x \)'s first bit being \( i \); \( p_i(z) \) is the probability that \( \mathcal{A} \) will find correct inverse \( w \) for \( z \), and the first bit of \( g(w) \) is \( i \); and \( h(y) \) is \( y \) mod \( 2^{k-1} \), such that \( f_a = h \circ g \).

In the first equality, we observe that for every possible last \( k - 1 \) bits of an element in the range of \( g \), there are precisely two range elements \( y_1, y_2 \) with this suffix (by \( g \) OWP): one whose first bit is 1 and one whose first bit is 0. In the second equality, we note that \( p_0(z) + p_1(z) \) is simply the probability of correctly inverting \( z \), so the sum over all elements of the range of \( f_a \) of this function is the size of the range \( (2^{k-1}) \) multiplied by the probability of correctly inverting \( (\eta(k)) \). The last equality is self-explanatory.

Thus, \( \mathcal{B} \)'s probability of success is non-neligible as well, giving a contradiction of \( g \) one-way.

b. No, it does not follow. Consider a LPOWF \( g : \{0,1\}^k \to \{0,1\}^k \), and construct the function \( f : \{0,1\}^{2k} \to \{0,1\}^{2k} \) as follows: \( f(x_1 \circ x_2) = 0^k \circ x_2 \) if \( x_1 = 0^k \), and \( f(x_1 \circ x_2) = 1 \circ 1^{k-1} \circ g(x_2) \) otherwise. This \( f \) is a one-way function; if it were not, with p.p.t. inverter \( \mathcal{A} \), we could construct \( \mathcal{B} \) which took \( k \)-bit \( y \) as input, gave \( 1 \circ 0^{k-1} \circ y \) to \( \mathcal{A} \), and received \( x_1 \circ x_2 \), outputting \( x_2 \).

Since \( x_1 = 0^k \) with negligible probability when \( U_{2k} \) is the distribution on inputs to \( f \), observe that the distribution \( D = U_{2k} \setminus \{ x_1 \circ x_2 \mid x_1 = 0^k \} \) is statistically indistinguishable from \( U_{2k} \), so \( f(D) = 1 \circ 0^{k-1} \circ g(U_k) \) is statistically indistinguishable from \( f(U_{2k}) \). So because \( \mathcal{A} \) correctly inverts with non-negligible probability on input from \( f(U_{2k}) \), it does so on input from \( 1 \circ 0^{k-1} \circ g(U_k) \), which is what \( \mathcal{B} \) gives it. But a correct inverse to such an input \( 1 \circ 0^{k-1} \circ y \) is necessarily of the form \( x_1 \circ x_2 \), where \( f(x_2) = y \), so \( \mathcal{B} \) does indeed invert with non-negligible probability. Contradiction, so \( f \) is one-way.

Now, given any output \( 0^k \circ z \) (all elements in the range of \( f_a \) look like this), an inverter for \( f_a \) need only return \( 0^k \circ z \) right back, since this is a correct inverse. So \( f_a \) is not one-way. (For the sake of completeness, we can define behavior for \( f \) on odd-length inputs – perhaps it ignores the last bit – but this does not interfere meaningfully with the mechanics of the proof.)
c. No, it doesn’t hold. Yes, some particular $f_i$ might be easy to invert, but when we talk about a function being an OWF, we’re really by necessity talking about a family of functions, parametrized by $k$, so it’s incoherent to talk about one particular function being one-way or not. This is the crux of why this ‘sorites’-style argument doesn’t hold up. “For what $i$ does the function $f_i$ cease to be one-way?” is meaningless for any fixed $k$; if we do not fix $k$, the answer is “for no $i$”, because for any finite $i$, $f_i$ is still a one-way family of functions parametrized by $k$ (I claim this without proof, but the construction proceeds similarly to (b)).

Alternately, one can say that the proof simply falls apart because it invokes the hybrid argument when the hybrid argument does not apply. The proof is not arguing two distributions are computationally indistinguishable, so this is an unfounded hand-wave at best.
4 Breaking our Assumptions

a. Since we know \( p \) and \( q \), we can compute \( \varphi(N) = (p - 1)(q - 1) \). We know that \( \gcd(e, \varphi(N)) = 1 \), so we can use the extended Euclidean algorithm to find \( K \) and \( L \) such that \( Ke + L\varphi(N) = 1 \), which implies that \( Ke \equiv 1 \pmod{\varphi(N)} \). We then return \( d \equiv K \pmod{\varphi(N)} \).

b. Run the extended Euclidean algorithm to find \( K \) and \( L \) such that
\[ Ke + L\lambda(N) = \gcd(e, \lambda(N)). \]
If \( \gcd(e, \lambda(N)) = 1 \), then \( Ke \equiv 1 \pmod{\lambda(N)} \). But since \( \lambda(N) = \alpha \varphi(N) \), we also have that \( Ke \equiv 1 \pmod{\varphi(N)} \). Return \( d' \equiv K \pmod{\varphi(N)} \).

Otherwise, let \( D_1 = \gcd(e, \lambda(N)) \). Since \( e \) and \( \varphi(N) \) are relatively prime, we must have that \( D_1 \mid \alpha \). Let \( \lambda'(N) = \frac{\lambda(N)}{D_1} = \frac{\alpha}{D_1} \varphi(N) \) be our new multiple of \( \varphi(N) \). Note that it is not necessarily the case that \( \gcd(e, \lambda'(N)) = 1 \). Let \( D_2 = \gcd(e, \lambda'(N)) \). Again, we have that \( D_2 \mid \alpha \), so consider \( \lambda''(N) = \frac{\alpha}{D_1D_2} \varphi(N) \). Continue this process until \( \gcd(e, \frac{\alpha}{D_1D_2D_3\cdots D_k}\varphi(N)) = 1 \). Then use the extended Euclidean algorithm to find \( \tilde{K} \) such that \( \tilde{K} e \equiv 1 \pmod{\frac{\alpha}{D_1D_2D_3\cdots D_k}\varphi(N)} \), which gives \( \tilde{K} e \equiv 1 \pmod{\varphi(N)} \). Return \( d' \equiv \tilde{K} \pmod{\varphi(N)} \).

c. To determine if some \( x \in \text{QR}_N \cup \text{QNR}_N \) is a quadratic residue or not, we can apply Euler’s criterion using \( p \) and \( q \). In other words, we know that \( x \in \text{QR}_N \) if and only if:
\[ x^{\frac{p-1}{2}} \equiv 1 \pmod{p} \]
and
\[ x^{\frac{q-1}{2}} \equiv 1 \pmod{q} \]

d. As in part (c), we want some way of computing Euler’s criterion. We can achieve this by noticing that raising \( x^{\frac{p-1}{2}} \) (mod \( p \)) to an odd power will not change its value (1 or -1). Now, consider \( \lambda'(N) = \frac{\lambda(N)}{2} = \frac{\alpha}{2} \varphi(N) \). Since \( q \equiv 3 \pmod{4} \), \( \frac{q-1}{2} \) is odd, so \( \lambda'(N) \) is an odd multiple of \( \varphi(N) \). The same holds when we swap \( p \) and \( q \). Thus, we can compute \( x^{\lambda'(N)} \pmod{N} \), which will be 1 for \( x \in \text{QR}_N \) and -1 for \( x \in \text{QNR}_N \).

e. We can extend our result from part (d) to handle even \( \alpha \) by dividing \( \lambda(N) \) by two until it is no longer even. This will divide out the factor of four from \( \varphi(N) \) as before and then proceed to reduce \( \alpha \) to an odd number by eliminating its even factors. Since we are dividing by two each time, there can be no more iterations than the number of bits in \( \lambda(N) \), making this process polynomial in the size of the input.
5 Hardcore Bits and OWFs

For the sake of contradiction, assume that there exists an injective function $f_{pk}$ with a hardcore bit such that $f_{pk}$ is not one-way. Then there exists some p.p.t. $A$ which on input $y$, outputs $x'$ such that $f_{pk}(x') = y$ with nonnegligible probability $\varepsilon$. Now we construct p.p.t. $B$ that distinguishes $D_0(1^k)$ and $D_1(1^k)$. $B$ is defined as follows:

$B$, on input $(y, b)$:

1. Run $A$ on input $y$ to get output $x'$.
2. If $y = f_{pk}(x')$, compute $B_{pk}(x')$ and compare it to $b$. If $B_{pk}(x') = b$, output 0. Else output 1. (Equivalently, output $B_{pk}(x') \oplus b$.)
3. If $y \neq f_{pk}(x')$, randomly pick 0 or 1 to output.

We analyze the reduction in two cases:

- If $A$ outputs $x'$ such that $f_{pk}(x') = y$, then because $f_{pk}$ is injective, $B_{pk}(x)$ will evaluate to $b$ or $\overline{b}$ corresponding to whether $B$ was given a sample from $D_0(1^k)$ or $D_1(1^k)$, respectively. As such, $B$ will correctly distinguish between the two distributions.

- Alternatively, if $A$ outputs $x'$ such that $f_{pk}(x') \neq y$, then $B$ will correctly distinguish between $D_0(1^k)$ and $D_1(1^k)$ exactly half of the time.

Since the former will occur with nonnegligible probability $\varepsilon(k)$ and the latter will occur with probability $1 - \varepsilon(k)$, $B$ will output the correct answer with probability $\frac{1}{2} + \frac{\varepsilon(k)}{2}$, which is a nonnegligible advantage.