1 Error-Correcting Codes

(a) Recall from class the error-correcting code for erasure errors, which protects against up to $k$ lost packets by sending a total of $n + k$ packets (where $n$ is the number of packets in the original message). Often the number of packets lost is not some fixed number $k$, but rather a fraction of the number of packets sent. Suppose we wish to protect against a fraction $\alpha$ of lost packets (where $0 < \alpha < 1$). At least how many packets do we need to send (as a function of $n$ and $\alpha$)?

(b) Repeat part (a) for the case of general errors.

Solution:

(a) Suppose we send a total of $m$ packets (where $m$ is to be determined). Since at most a fraction $\alpha$ of these are lost, the number of packets received is at least $(1 - \alpha)m$. But in order to reconstruct the polynomial used in transmission, we need at least $n$ packets. Hence it is sufficient to have $(1 - \alpha)m \geq n$, which can be rearranged to give $m \geq n/(1 - \alpha)$.

(b) Suppose we send a total of $m = n + 2k$ packets, where $k$ is the number of errors we can guard against. The number of corrupted packets is at most $\alpha m$, so we need $k \geq \alpha m$. Hence $m \geq n + 2\alpha m$. Rearranging gives $m \geq n/(1 - 2\alpha)$.

Note: Recovery in this case is impossible if $\alpha \geq 1/2$.

2 Alice and Bob

(a) Alice decides that instead of encoding her message as the values of a polynomial, she will encode her message as the coefficients of a degree 2 polynomial $P(x)$. For her message $[m_1, m_2, m_3]$, she creates the polynomial $P(x) = m_1 x^2 + m_2 x + m_3$ and sends the five packets $(0, P(0)), (1, P(1)), (2, P(2)), (3, P(3))$, and $(4, P(4))$ to Bob. However, one of the packet $y$-values is changed by Eve before it reaches Bob. If Bob receives $(0,1), (1,3), (2,0), (3,1), (4,0)$

and knows Alice’s encoding scheme and that Eve changed one of the packets, can he recover the original message? If so, find it as well as the $x$-value of the packet that Eve changed. If he can’t, explain why. Work in mod 7.
(b) Bob gets tired of decoding degree 2 polynomials. He convinces Alice to encode her messages on a degree 1 polynomial. Alice, just to be safe, continues to send 5 points on her polynomial even though it is only degree 1. She makes sure to choose her message so that it can be encoded on a degree 1 polynomial. However, Eve changes two of the packets. Bob receives $(0,5)$, $(1,7)$, $(2,x)$, $(3,5)$, $(4,0)$. If Alice sent $(0,5)$, $(1,7)$, $(2,9)$, $(3,−2)$, $(4,0)$, for what values of $x$ will Bob not uniquely be able to determine Alice’s message? Assume that Bob knows Eve changed two packets. Work in mod 13.

(c) Alice wants to send a length 9 message to Bob. There are two communication channels available to her: Channel A and Channel B. When $n$ packets are fed through Channel A, only 6 packets, picked arbitrarily, are delivered. Similarly, Channel B will only deliver 6 packets, picked arbitrarily, but it will also corrupt (change the value) of one of the delivered packets. Each channel will only work if at least 10 packets are sent through it. Using each of the two channels once, provide a way for Alice to send her message to Bob so that he can always reconstruct it.

**Solution:**

(a) We can use Berlekamp and Welch. We have: $Q(x) = P(x)E(x)$. $E(x)$ has degree 1 since we know we have at most 1 error. $Q(x)$ is degree 3 since $P(x)$ is degree 2. We can write a system of linear equations and solve:

\[
\begin{align*}
    d &= 1(0 − e) \\
    a + b + c + d &= 3(1 − e) \\
    8a + 4b + 2c + d &= 0(2 − e) \\
    27a + 9b + 3c + d &= 1(3 − e) \\
    64a + 16b + 4c + d &= 0(4 − e)
\end{align*}
\]

Since we are working in mod 7, this is equivalent to:

\[
\begin{align*}
    d &= −e \\
    a + b + c + d &= 3 − 3e \\
    a + 4b + 2c + d &= 0 \\
    6a + 2b + 3c + d &= 3 − e \\
    a + 2b + 4c + d &= 0
\end{align*}
\]

Solving yields:

\[
Q(x) = x^3 + 5x^2 + 5x + 4, E(x) = x - 3
\]

To find $P(x)$ we divide $Q(x)$ by $E(x)$ and get $P(x) = x^2 + x + 1$. So Alice’s message is $m_1 = 1, m_2 = 1, m_3 = 1$. The $x$-value of the packet Eve changed is 3.

**Alternative solution:** Since we have 5 points, we have to find a polynomial of degree 2 that goes through 4 of those points. The point that the polynomial does not go through will be the packet that Eve changed. Since 3 points uniquely determine a polynomial of degree 2, we can
pick 3 points and check if a 4th point goes through it. (It may be the case that we need to try all sets of 3 points.) We pick the points \((1,3), (2,0), (4,0)\). Lagrange interpolation can be used to create the polynomial but we can see that for the polynomial that goes through these 3 points, it has 0s at \(x = 2\) and \(x = 4\). Thus the polynomial is 

\[
k(x - 2)(x - 4) = k(x^2 - 6x + 8) \pmod{7} \equiv k(x^2 + x + 1) \pmod{7}.
\]

We find \(k \equiv 1\) by plugging in the point \((1,3)\), so our polynomial is 

\[
x^2 + x + 1.
\]

We then check to see if the this polynomial goes through one of the 2 points that we didn’t use. Plugging in 0 for \(x\), we get 1. The packet that Eve changed is the point that our polynomial does not go through which has \(x\)-value 3. Alice’s original message was 

\[
m_1 = 1, m_2 = 1, m_3 = 1.
\]

(b) Since Bob knows that Eve changed 2 of the points, the 3 remaining points will still be on the degree 1 polynomial that Alice encoded her message on. Thus if Bob can find a degree 1 polynomial that passes through at least 3 of the points that he receives, he will be able to uniquely recover Eve’s message. The only time that Bob cannot uniquely determine Alice’s message is if there are 2 polynomials with degree 1 that pass through 3 of the 5 points that he receives. Since we are working with degree 1 polynomials, we can plot the points that Bob receives and then see which values of \(x\) will cause 2 sets of 3 points to fall on a line. 

\[
(0,5), (1,7), (4,0)\] already fall on a line. If \(x = 6\), \((1,7), (2,6), (3,5)\) also falls on a line. If \(x = 5\), \((0,5), (2,5), (3,5)\) also falls on a line. If \(x = 9\), \((0,5), (2,9), (4,0)\) falls on the original line, so here Bob can decode the message. If \(x = 10\), \((2,10), (3,5), (4,0)\) also falls on a line. So if \(x = 6, 5, 10\), Bob will not be able to uniquely determine Alice’s message.

(c) Channel A will deliver 6 packets so we can send a message of length 6 encoded on a polynomial of degree 5 though it. If we send 10 points though channel A, it doesn’t matter which 6 points Bob gets, he will still be able to reconstruct our degree 5 polynomial. Since the channel B has 1 general error, we can only send a message of length 4 encoded on a degree 3 polynomial through it. If we send 10 points, Bob will get 6 points to calculate a degree 4 polynomial with 1 general error, which he is able to do. Thus to send our length 8 message, we can send the character 1 - 6 through a channel A and the characters 7 - 9 through channel B.

**Alternative Solution:** Alice can interpolate a polynomial of degree 8 encoding the message of length 9. She sends 10 points from that polynomial through channel A and another 10 points from the same polynomial through channel B. Bob will receive 6 points from channel A and 6 points from channel B, with one of them corrupted. He can use Berlekamp-Welch with \(n = 9\) and \(k = 1\) to recover the original polynomial. He retrieves the message by evaluating the polynomial on relevant points.

### 3 Error-Detecting Codes

Suppose Alice wants to transmit a message of \(n\) symbols, so that Bob is able to detect rather than correct any errors that have occurred on the way. That is, Alice wants to find an encoding so that Bob, upon receiving the code, is able to either

(I) tell that there are no errors and decode the message, or
(II) realize that the transmitted code contains at least one error, and throw away the message.

Assuming that we are guaranteed a maximum of \( k \) errors, how should Alice extend her message (i.e. by how many symbols should she extend the message, and how should she choose these symbols)? You may assume that we work in \( \text{GF}(p) \) for very large prime \( p \). Show that your scheme works, and that adding any lesser number of symbols is not good enough.

Solution:
Since \( k \) bits can break, it seems reasonable to extend our message by \( k \) symbols for a total of \( n+k \). And indeed, we show that this works: Let Alice generate her message \( y_0, \ldots, y_{n-1} \) of length \( n \) by constructing the unique polynomial \( f \) of degree \( \leq n-1 \) that passes through \((i, y_i)\) for \( i \in \{0, \ldots, n-1\} \), and add the \( k \) extra symbols \( y_j = f(j) \), where \( j \in \{n, \ldots, n+k-1\} \). Now Bob receives the message \( r_i, i \in \{0, \ldots, n+k-1\} \), upon which he interpolates the unique degree \( \leq n-1 \) polynomial \( g \) that passes through the points \((0, r_0), \ldots, (n-1, r_{n-1})\). We claim that the message is corrupted if and only if \( g(i) \neq r_i \) for some \( i \in \{n, \ldots, n+k-1\} \).

The backward direction becomes clear when stated as its contrapositive: If the message contains no error, then \( g(i) \) and \( f(i) \) coincide on all of \( n \) points \( \{0, \ldots, n-1\} \). Since they are both of degree \( n-1 \), they must be the same polynomial and hence \( g(i) = f(i) = r_i \) for all \( i \).

Let us prove the forward direction: Since we know that at most \( k \) errors occurred, there must exist a subset \( A \subseteq \{0, \ldots, n+k-1\} \) of size \( n \) on which \( r_i = y_i \). Now either

1. \( A = \{0, \ldots, n-1\} \), in which case \( g = f \) and at least one error must have occured for some \( j_0 \in \{n, \ldots, n+k-1\} \). But then \( r_{j_0} \neq y_{j_0} = f(j_0) = g(j_0) \), which is what we wanted to show.

2. Or at least one error occured for an index \( i \in \{0, \ldots, n-1\} \) in which case \( g \neq f \). But since \( g \) and \( f \) are of degree \( n-1 \) and \( |A| = n \), \( f \) and \( g \) cannot take the same values on \( A \), so there must be some element \( j_0 \in A, j_0 \in \{n, \ldots, n+k-1\} \) for which \( g(j_0) \neq f(j_0) = y_{j_0} = r_{j_0} \).

Lastly, we need to show that our algorithm doesn’t work if Alice extends her message by less than \( k \) symbols, which we can do by crafting a counterexample: Assume Alice sends \( m < n+k-1 \) symbols in the same fashion as above, then we may corrupt \( y_{n-1}, \ldots, y_m \) by setting \( r_{n-1} \neq y_{n-1} \) and \( r_j = h(j) \) for \( j \in \{n, \ldots, m-1\} \), where \( h \) is the unique polynomial of degree \( \leq n-1 \) passing through \((0, y_0), \ldots, (n-2, y_{n-2}), (n-1, r_{n-1})\). Since Bob is going to reconstruct \( g = h \), \( g(j) = r_j \) for all \( j \in \{n, \ldots, m-1\} \) and he will not notice the corruption.

4 Secret Sharing with Spies

An officer stored an important letter in her safe. In case she becomes unreachable in battle, she decides to share the password (which is a number) with her troops. However, everyone knows that there are 3 spies among the troops, but no one knows who they are except for the three spies themselves. The 3 spies can coordinate with each other and they will either lie and make people not able to open the safe, or will open the safe themselves if they can. Therefore, the officer would like a scheme to share the password that satisfies the following conditions:
• When $M$ of them get together, they are guaranteed to be able to open the safe even if they have spies among them.

• The 3 spies must not be able to open the safe all by themselves.

Please help the officer to design a scheme to share her password. What is the scheme? What is the smallest $M$? Show your work and argue why your scheme works and any smaller $M$ couldn’t work. (The troops only have one chance to open the safe; if they fail the safe will self-destruct.)

**Solution:**

The key insight is to realize that both polynomial-based secret-sharing and polynomial-based error correction work on the basis of evaluating an underlying polynomial at many points and then trying to recover that polynomial. Hence they can be easily combined.

Suppose the password is $s$. The officer can construct a polynomial $P(x)$ such that $s = P(0)$ and share $(i, P(i))$ to the $i$-th person in her troops. Then the problem is: what should the degree of $P(x)$ be and what is the smallest $M$?

First, the degree of polynomial $d$ should not be less than 3. It is because when $d < 3$, the 3 spies can decide the polynomial $P(x)$ uniquely. Thus, $n$ will be at least 4 symbols.

Let’s choose a polynomial $P(x)$ of degree 3 such that $s = P(0)$. We now view the 3 spies as 3 general errors. Then the smallest $M = 10$ since $n$ is at least 4 symbols and we have $k = 3$ general errors, leading us to a “codeword” of $4 + 2 \cdot 3 = 10$ symbols (or people in our case). Even though the 3 spies are among the 10 people and try to lie on their numbers, the 10 people can still be able to correct the $k = 3$ general errors by the Berlekamp-Welch algorithm and find the correct $P(x)$.

**Alternative solution:**

Another valid approach is making $P(x)$ of degree $M - 1$ and adding 6 public points to deal with 3 general errors from the spies. In other words, in addition to their own point $(i, P(i))$, everyone also knows the values of 6 more points, $(t + 1, P(t + 1)), (t + 2, P(t + 2)), \ldots, (t + 6, P(t + 6))$, where $t$ is the number of the troops. The spies have access to total of $3 + 6 = 9$ points so the degree $M - 1$ must be at least 9 to prevent the spies from opening the safe by themselves. Therefore, the minimum $M$ is 10.