1. Extend \( G(X, Y, Z) = XZ \lor Y \overline{Z} \) to a function from bytes to bytes. Fill in for \( A \) and \( B \): If we replace \( A \) by \((1,1,1,1,1,1,1,1)\) then the output of \( G(X, Y, Z) \) is simply given by the single variable \( B \). (Your answer might look like \( A = X \), \( B = Y \), which is wrong.) If we replace \( Z \) by \((1,1,1,1,1,1,1,1)\) then \( G(X, Y, 1) = X \lor 0 = X \). So \( A = Z \), \( B = X \).

2. Alice has uses a timestamping service to put a timestamp on a document. Give an example of who would verify the timestamp and under what circumstances. You should NOT use KERBEROS as an example. There are several examples. A judge might have a timestamp verified if there was an issue as to who created a document first.

3. Alice and Bob are using quantum cryptography. Eve is intercepting the photons along the way and measuring their polarizations. Below I show the polarizations Alice sends and the basis settings for Eve and Bob. Fill in possibilities for what Bob will get. For at least one case, Bob should set the right basis but get the wrong polarization. There are several correct answers.

| Alice sends | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Eve sets   | + | + | + | + | + | + | + | + |
| Bob sets   | + | × | + | × | × | × | + | × |
| Bob gets   | / | / | / | \ | / | / | / | / |

4. Alice and Bob are using quantum cryptography. Eve is intercepting the photons and measuring their polarizations on the two bases. Alice and Bob expose 12 bits (for which they agreed on the basis). On average, how many polarizations to you expect to agree? We expect in half the cases (6), Eve sets the correct basis so Bob will get all of those right. In the other half of the cases, Eve sets the basis wrong so Bob’s measurement is random so that should be correct half of those times (3). So a total of 9.

5. You want your computer to compute \((M!)^N\) in the following way. You have \(1^N, 2^N, 3^N, \ldots\) in storage. First you multiply \(1^N\) by \(2^N\), then multiply that by \(3^N\), etc. Find an upper bound on the running time for computing \((M!)^N\) this way in terms of \(M\) and \(N\). Hints: \(A! \approx (A/e)^A \sqrt{2\pi A} \) (Stirling) and the time to compute \(A \cdot B\) is \(O(\log(A)\log(B))\). There are \(M - 1\) steps. The slowest is computing the product of \((M - 1)!^N\) and \(M^N\). That takes time \(O((M - 1)!^N)\log(M^N) = O(N^2\log((M - 1)!^N)\log(M^N))\). Since \(A! \approx (A/e)^A \sqrt{2\pi A}\), we have \(\log(A!) \approx A(\log(A/e) + \frac{1}{2}\log(2\pi)) = A\log(A) - A\log(e) + \frac{1}{2}\log(2\pi) + \log(A)\) which is dominated by \(A(\log(A))\). So \(\log((M - 1)!) = O((M - 1)\log(M - 1)!) = O(M\log(M))\). Thus \(O(N^2\log((M - 1)!^N)\log(M^N)) = O(N^2M\log^2(M))\). Since there are \(M - 1 \approx M\) steps the entire algorithm takes time \(O(N^2M^2\log^2(M))\).

6. You will reconstruct KERBEROS in this problem. Below is a diagram with four empty circles and arrows. Go look it up!
7. a) Let \( m < n \) be large positive integers of approximately the same size as each other. The time it takes to find \( \gcd(m, n) \) is \( O(\log^3(n)) \). Assume that \( m' < n' \) are also large positive integers of approximately the same size as each other and that the binary representation of \( n' \) is twice as long as that of \( n \). Assume it takes 9 nanoseconds to find \( \gcd(m, n) \). Approximately how long would it take to find \( \gcd(m', n') \)? (Note 1011010010011010 has a binary representation that is twice as long as that of 11000111.) We see that 9 is some constant multiple (depending on the speed of the machine) \( k \) times \( \log^3(n) \). Since the representation of \( n' \) is twice as long as that of \( n \), we have \( n' \approx n^2 \). So the time it takes to compute \( \gcd(m', n') = O(\log^3(n')) = O(\log^3(n^2)) = O((\log(n^2))^3) = O(8\log^3(n)) = k \cdot 8 \cdot 9 \). So it takes 8 times longer or 72 nanoseconds.

b) Let \( n \) be a large positive integer. The time it takes to factor \( n \) using trial division is \( O(\sqrt{n}) \). Assume that \( n'' \) is a large positive integer and that the binary representation of \( n'' \) has 10 more bits than that of \( n \). Assume that it takes 9 nanoseconds to factor \( n \) using trial division. Approximately how long would it take to factor \( n'' \) using trial division? (Note 1011010010011010 has 10 more bits than 110001.) We see that 9 is some constant multiple (depending on the speed of the machine) \( k \) times \( \sqrt{n} \). Since the representation of \( n' \) has 10 more bits than that of \( n \), we have \( n' \approx 2^{10}n \). So the time it takes to factor \( n' \) using trial division is \( O(\sqrt{n'}) = O(\sqrt{2^{10}n}) = O(2^5\sqrt{n}) = k \cdot 32 \cdot 9 \). So it takes 32 times longer, or 288 nanoseconds.
8. Here is the algorithm for RC4 with $n = 2$. To get the array, initialize with $S_0 = 0, \ldots, S_3 = 3$. Then fill another array of four 2-bit strings (which will be thought of as integers from 0 to 3) which is the key array, repeating the key as necessary to fill the entire array $K_0, K_1, \ldots, K_3$.

\[
\begin{align*}
&j := 0 \\
&\text{For } i = 0, \ldots, 3 \text{ do:} \\
&\quad j := j + S_i + K_i \pmod{4}.
\end{align*}
\]

Swap $S_i$ and $S_j$.

End For

\[
\begin{align*}
&j := 0; i := 0 \\
&\text{For } r = 0, \ldots, 3 \text{ do} \\
&\quad i := i + 1 \pmod{4}.
\end{align*}
\]

\[
\begin{align*}
&j := j + S_i \pmod{4}.
\end{align*}
\]

Swap $S_i$ and $S_j$.

\[
\begin{align*}
&t := S_i + S_j \pmod{4}.
\end{align*}
\]

$K_r := S_t$.

End For

The concatenation of the $K_r$’s is the keystream.

Say the key is 10 01 11 or [2, 1, 3]. I have encoded the four letters A = 00, E = 01, M = 10, T = 11. Decrypt the ciphertext 10 01 11 11. (The fact that this starts like the key is coincidence).

\[
\begin{array}{cccc|cccc}
 r & i & j & t & K_S_r & S_0 & S_1 & S_2 & S_3 \\
 0 & 0 & 2 & 2 & 0 & 1 & 2 & 3 \\
 0 & 1 & 0 & 1 & 0 & 1 & 2 & 3 \\
 1 & 2 & 2 & 1 & 1 & 0 & 1 & 2 \\
 3 & 0 & 2 & 3 & 2 & 3 & 1 & 0 \\
 1,1,1,2 \rightarrow 01010110. \text{ We XOR that with } 10011111 \text{ and get } 11001001 = \text{TAME.}
\end{array}
\]