Bitwise XOR Operation

To decode the message for this cache, you’ll need to perform a bitwise ‘XOR 21’ operation on each of the numbers in the message.

“Wait! What in the world is a bitwise ‘XOR 21’ operation?”

I’m glad you asked. XOR stands for ‘exclusive or’. The XOR operator compares two true/false statements and returns the value of true if (and only if) exactly one of the statements is true. Frequently, ‘true’ is represented by 1 and ‘false’ is represented by 0. In numeric representation:

\[
\begin{align*}
0 \text{ XOR } 0 &= 0 \\
0 \text{ XOR } 1 &= 1 \\
1 \text{ XOR } 0 &= 1 \\
1 \text{ XOR } 1 &= 0
\end{align*}
\]

Ok – If you’re with me so far, we’ll consider a bitwise XOR operation – in this case ‘XOR 21’. The key is to convert two numbers to binary first - in this case, a number from the message and the number 21. Suppose the number in the message is 7. The binary representation of 7 is 00111 (you can always add zeroes to the front), and the binary representation of 21 is 10101. Now perform the XOR operation successively on each pair of bits:

\[
\begin{align*}
0 & \hspace{0.5cm} 0 \hspace{0.5cm} 1 \hspace{0.5cm} 1 \hspace{0.5cm} 1 \hspace{0.5cm} = \hspace{0.5cm} 7 \\
1 & \hspace{0.5cm} 0 \hspace{0.5cm} 1 \hspace{0.5cm} 0 \hspace{0.5cm} 1 \hspace{0.5cm} = \hspace{0.5cm} 21 \\
1 & \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 1 \hspace{0.5cm} 0 \hspace{0.5cm} = \hspace{0.5cm} 18 \hspace{0.5cm} (\text{result})
\end{align*}
\]

The result is 10010, which when converted from binary back to decimal is 18. The key tells you that 18 equals the letter ‘R’. Therefore, anytime you see a seven in the message, the ‘XOR 21’ operation converts it into an 18, which is then replaced by ‘R’.

All right – so I’m a geek.

(However, if you are also a geek, Wikipedia has some information on the use of the XOR operation in cryptography. See, for example: http://en.wikipedia.org/wiki/One-time_pad)