Secret Transmissions

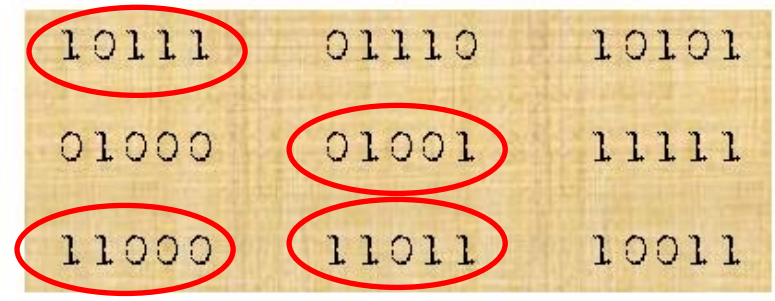
By Michael Slack

Agent X's First System

With a single check digit to sum the whole message to an even number, Agent X can always tell an error has occurred, given that a maximum of one digit can be transmitted falsely. This is due to the message being binary. If you let the correct message equal $2m$ (where m is any natural number), then any error will result in an odd-sum message:

- If a "0" is transmitted falsely, it must be sent as a "1", so $2m$ becomes $2m + 1$, which must be odd
- If a "1" is transmitted falsely, it must be sent as a "0", so $2m$ becomes $2m - 1$, which also must be odd.
- So, provided only one error can occur, Agent X can always tell if an error has occurred, since a single error will always generate an odd total.

Using this check digit, Agent X would know that the circled messages are error-free.:



As for the codes containing errors, the true message cannot be retrieved, since the error could have occurred to any one of the digits, and therefore there are five possible true messages for each false one.

Agent X's Second System

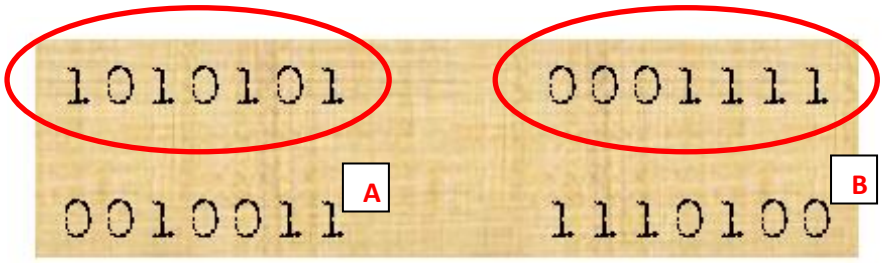
If you were to receive Agent X's message, and you wanted to check that it had been sent correctly, you would take the string $xyazbcd$ and from this find the substrings $xabd$, $yacd$ and $zbcd$. Each of these substrings should sum to an even number, so if a substring doesn't, you have four candidates for the source of the error. Equally, if you do have an even substring, then you know that you've got four innocent digits (as it were…).

	x	y	a	z	b	c	d
$xabd$	Odd	Even	Odd	Even	Odd	Even	Odd
$yacd$	Even	Odd	Odd	Even	Even	Odd	Odd
$zbcd$	Even	Even	Even	Odd	Odd	Odd	Odd

In the table above, you can see how an error in any of the digits creates its own unique pattern; x is the only digit in the table to create an “Odd, Even, Even” pattern, for example. So, provided only one error occurs in the message, the error can always be identified, by finding the pattern of odd and even substrings then and finding its corresponding false digit. This also allows you to correct the message, since if you know that, say, a “1” is a false digit, then it must be a “0”, and you now have a correct message. This is true no matter the false digit, and no matter the false digit’s true value.

If you had more check digits and more message digits, you wouldn’t have to draw up a whole table to like above to find the guilty digit. You could write the general substrings (xabd, etc.) in a list; for any substring that is even, you find every occurrence of each digit in the substring occurring in all the substrings, and cross them out. You then look at all of the odd substrings, and find the one digit (x,a,d, etc.) that is a member of each of them and hasn’t been crossed out. This is your guilty digit.

Below, the circled codes are error free.



Code A above is false because xabd and yacd are odd. This means that a is the error digit, so the true message would be 1000101.

Code B above is false because xabd and zbcd are odd. This means that b is the error digit, so the true message would be 1110000.