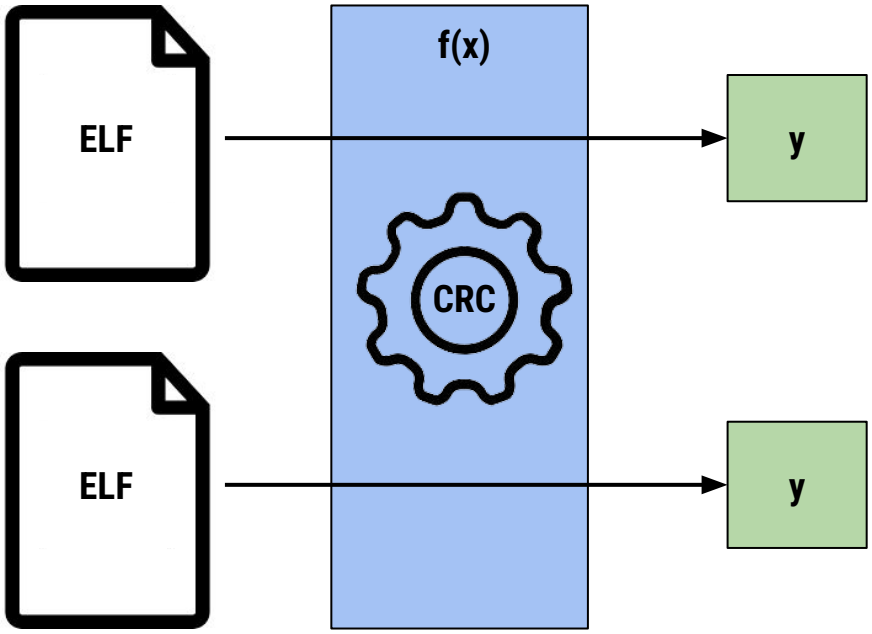
Linear Functions

Tim Nosco

29 FEB 2024

Research Innovations

Our Reference Problem

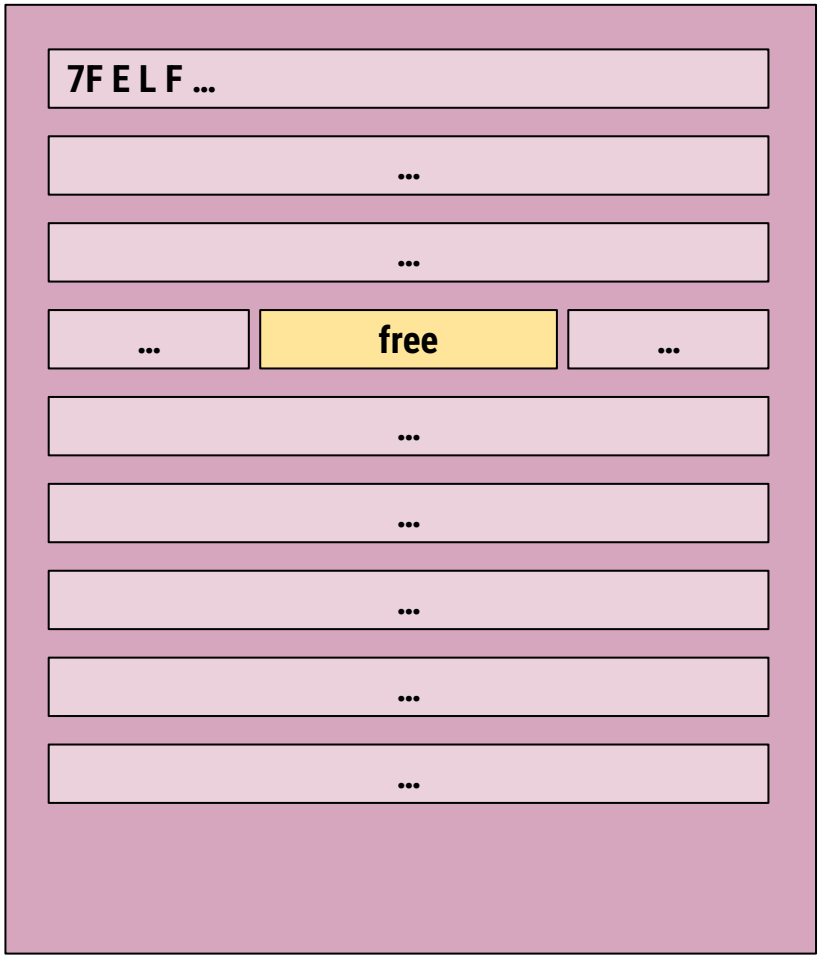


Let's say you have this routine, f(x) that calculates the cyclic redundancy of an input file and outputs an integer.

Our Reference Problem

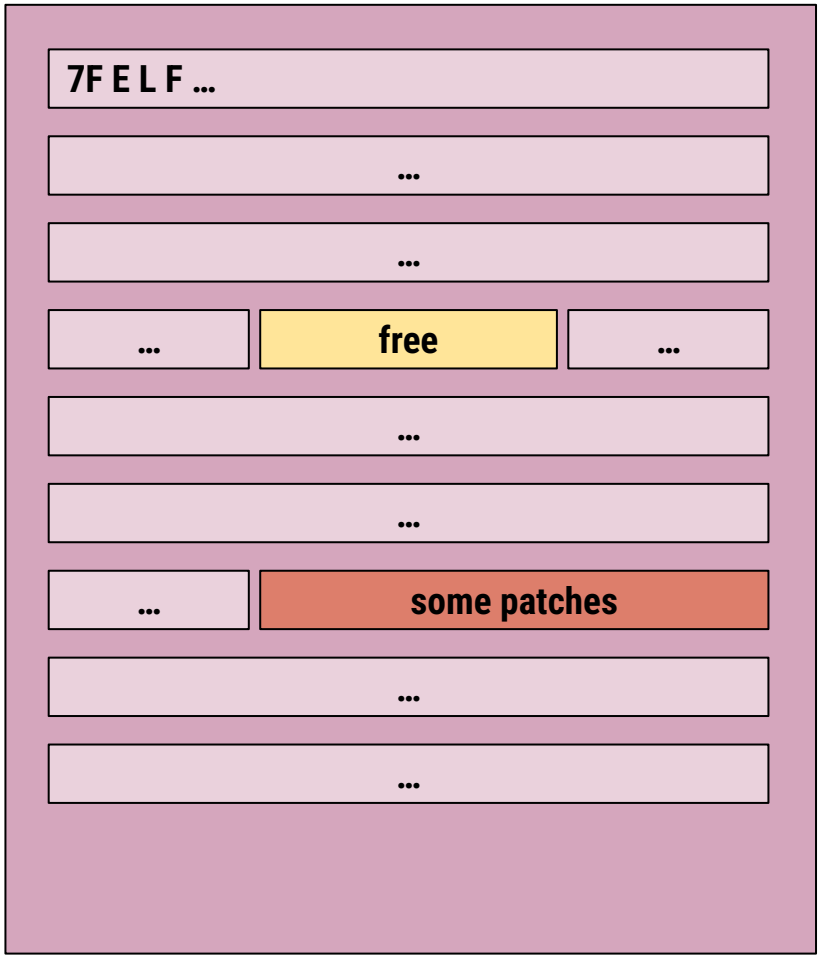
There's this file you'd like to modify.

Some arbitrarily aligned, but contiguous free bits exist that are inconsequential to normal operation of the executable.

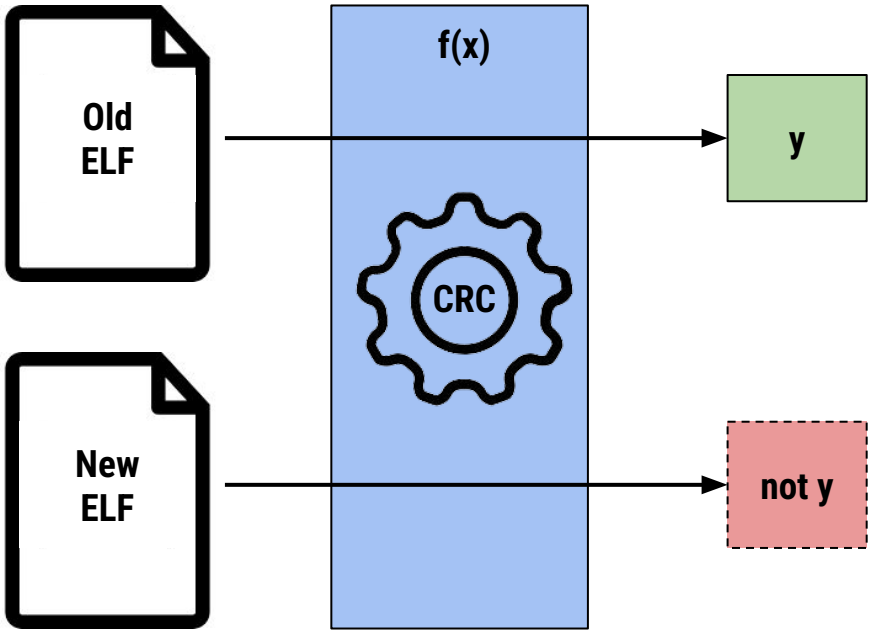


Our Reference Problem

You make some changes to it.



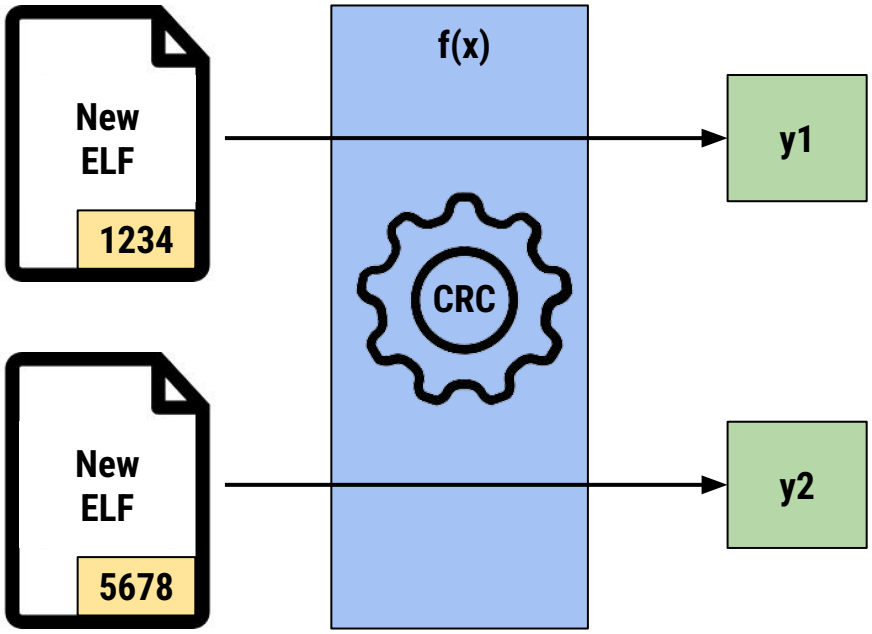
Patching changes the CRC



When you made your patches, you get a different output.

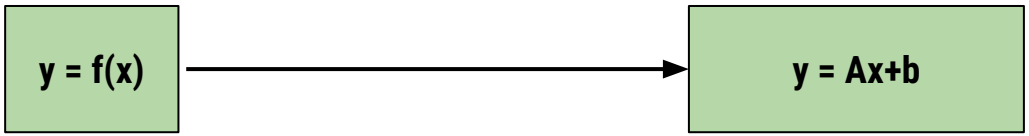
You wish that the new output was the same as the old output.

The meaning of x in f(x)



As we generate inputs, we're only going to manipulate the free bits—the x in f(x)—but still CRC the whole patched ELF.

Some linear algebra

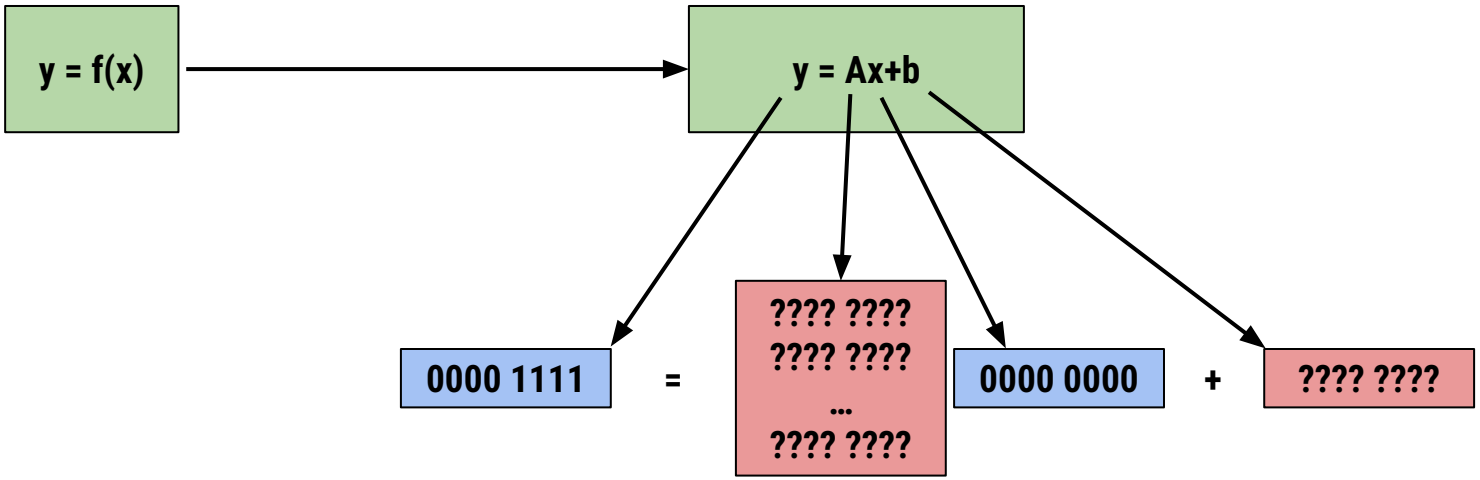


Let’s represent f(x) using linear algebra.

Note: these are matrices.

Some linear algebra

Let's represent our x bytes as a vector of bits, creating a system like shown.

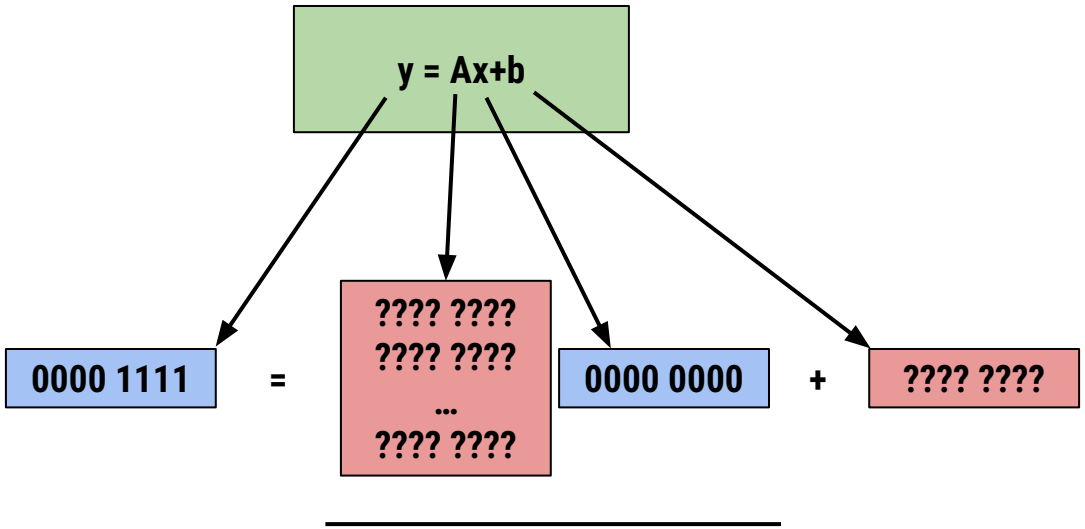


We can measure y and x by manipulating our free bits (x) and calculating the CRC (y).

We don't yet know what A and b are in our new linear algebra model, but we can find out!

The base vector

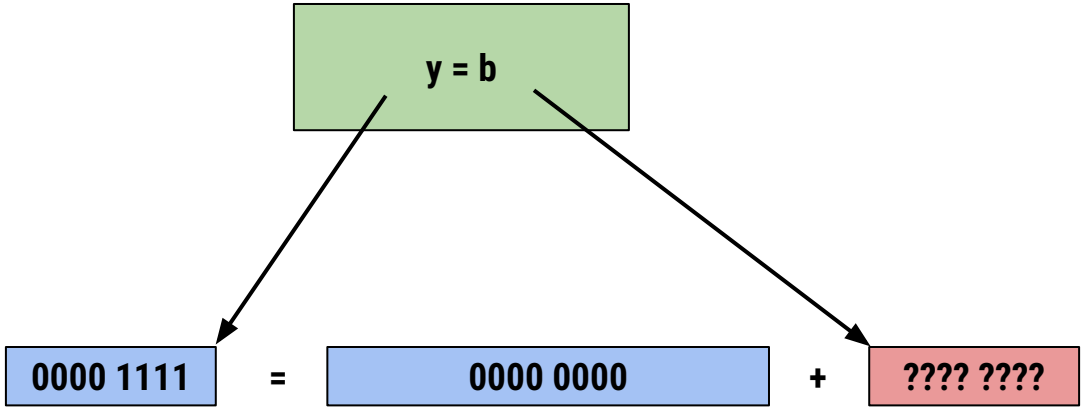
Start by figuring out the value of b.



When x is the zero vector,
the Ax term is also zero!

The base vector

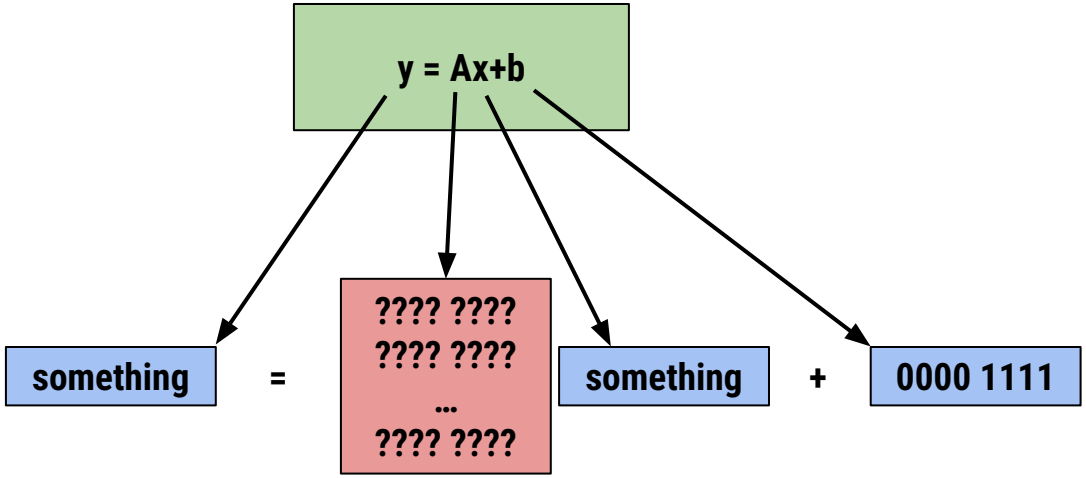
So the value of b is now known to be **0000 1111** .



When x is the zero vector,
the Ax term is also zero!

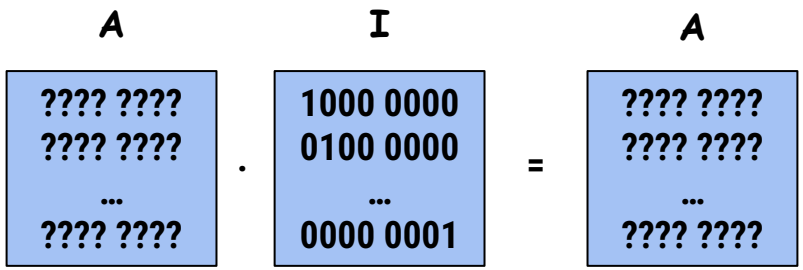
Picking a new value for x

What should we pick next to learn something about A?



Quick Tangent: The Identity Matrix

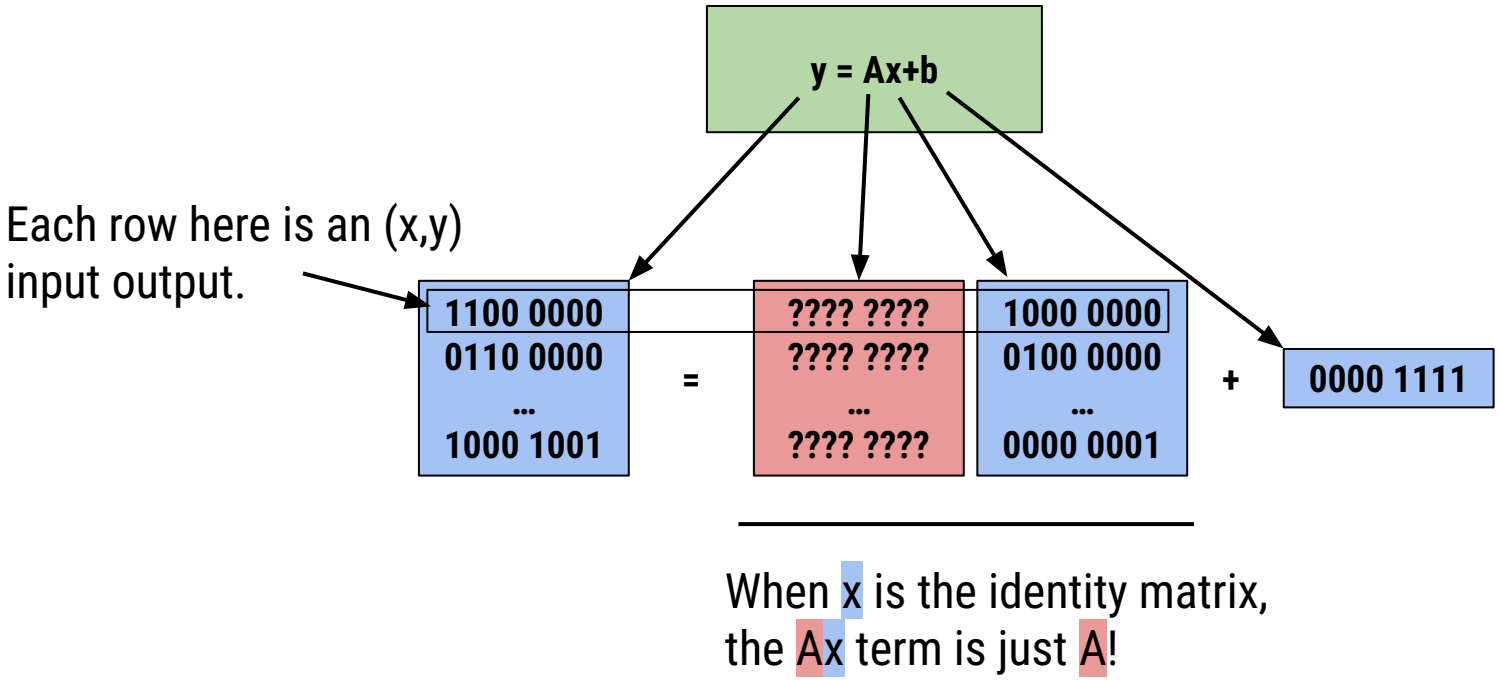
There's this special matrix in linear algebra called the identity matrix and is usually denoted with the letter "I"



When you multiply the A matrix by the identity matrix, you get A again. This is analogous to multiplication by 1 in regular math, where 5*1==5, for example.

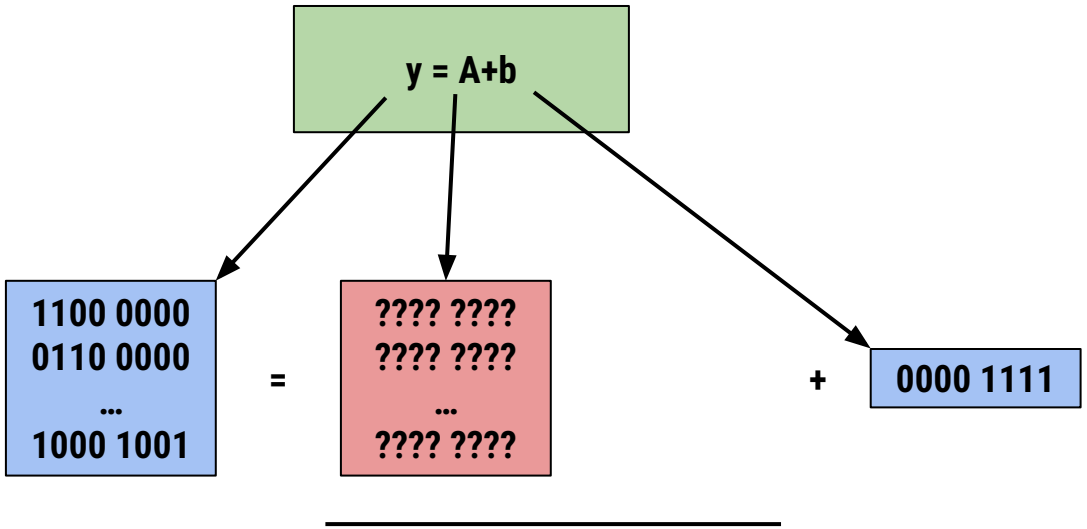
What if?

What if we carefully crafted our inputs to f(x) so that the x in y=Ax+b was the identity matrix?



What if?

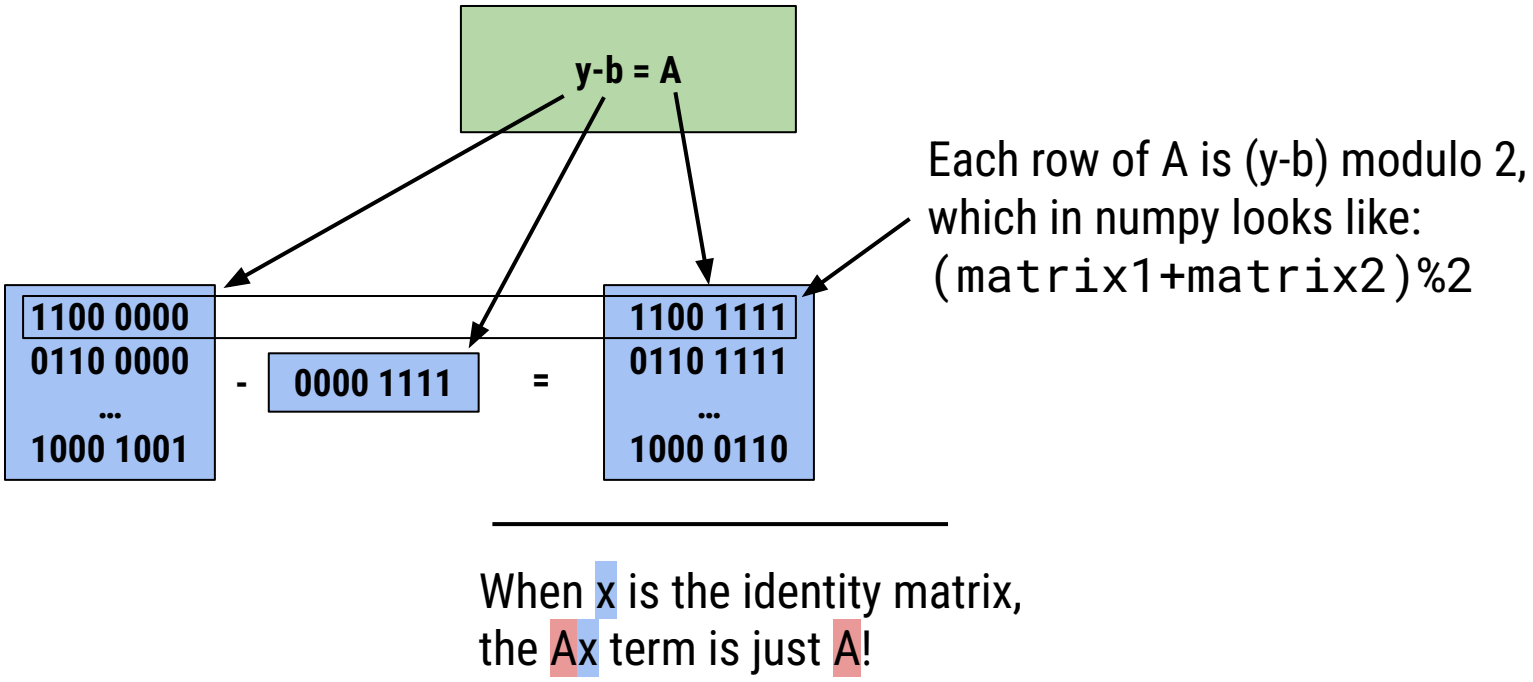
So, when x is the identity matrix, A is equal to y-b.



When x is the identity matrix,
the Ax term is just A!

Solve for M

Reorder the variables, and calculate A.



Quick Tangent: Matrix Operations - Addition

Look at how all three of these operations yield the same result table!

A+B (mod 2)

	0	**1**
0	0	1
1	1	0

A-B (mod 2)

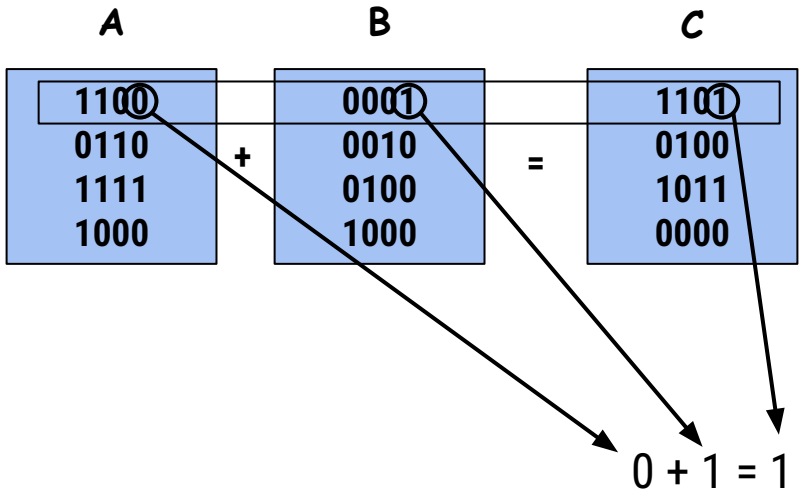
	0	**1**
0	0	1
1	1	0

A xor B

	0	**1**
0	0	1
1	1	0

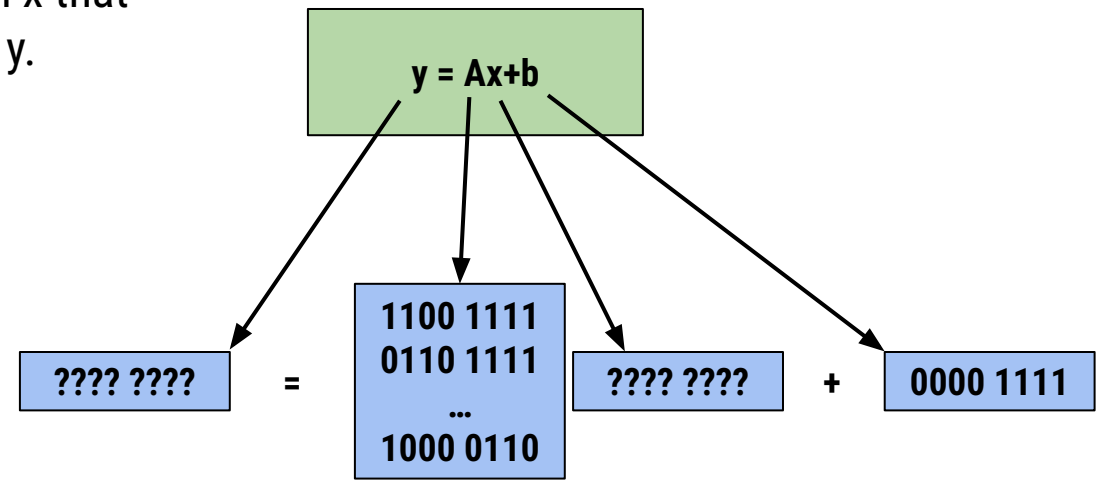
Quick Tangent: Matrix Operations - Addition

To do the operation, just treat each row and column independently.



Back to the problem

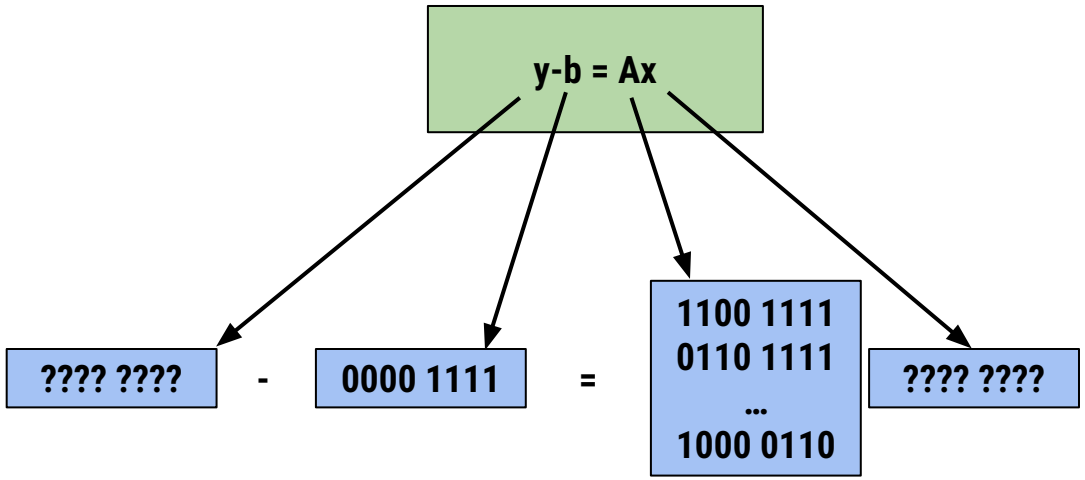
We want to pick an x that gives us a desired y.



Restated: what bits do I put in the free space so my CRC result is the same as before.

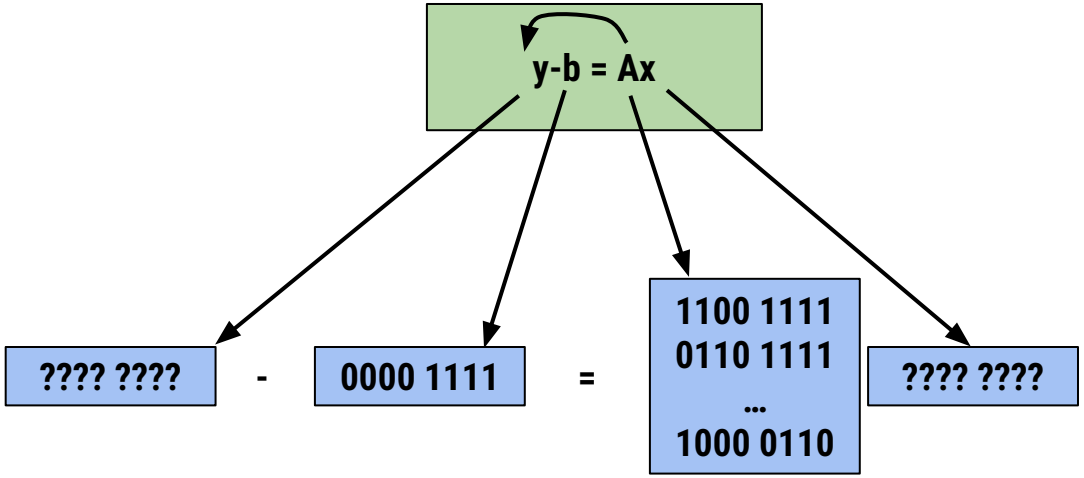
Solving for x

We subtract b from both sides.



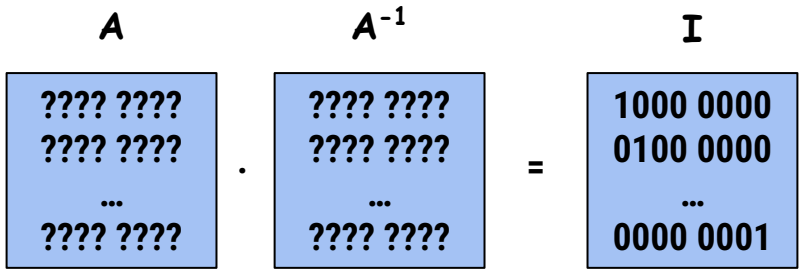
Solving for x

How do I get M onto the other side?



Quick Tangent: The Inverse

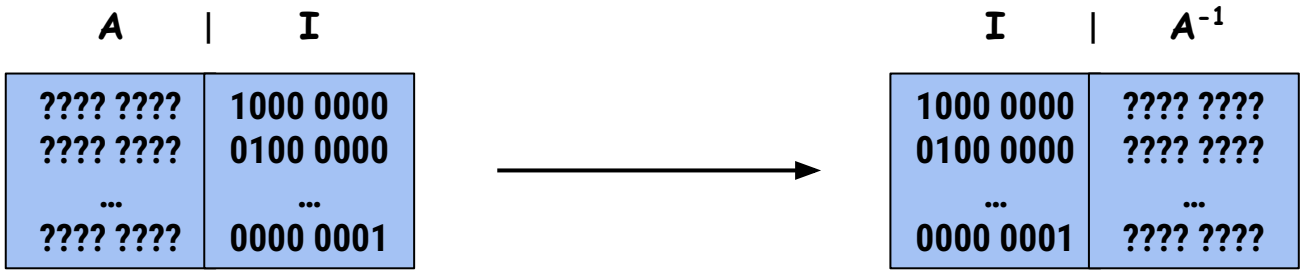
There's this special operation in linear algebra called an inverse usually denoted with a superscript -1 as in A^{-1}.



When you multiply A by its inverse, A^{-1}, you get the identity matrix, I.

Quick Tangent: Finding The Inverse

To find A^{-1}, you concatenate the identity matrix, I, and then perform row operations on the new matrix until the left becomes the identity matrix, I.



For every row r1, you can xor another row r2 so that r1^=r2.

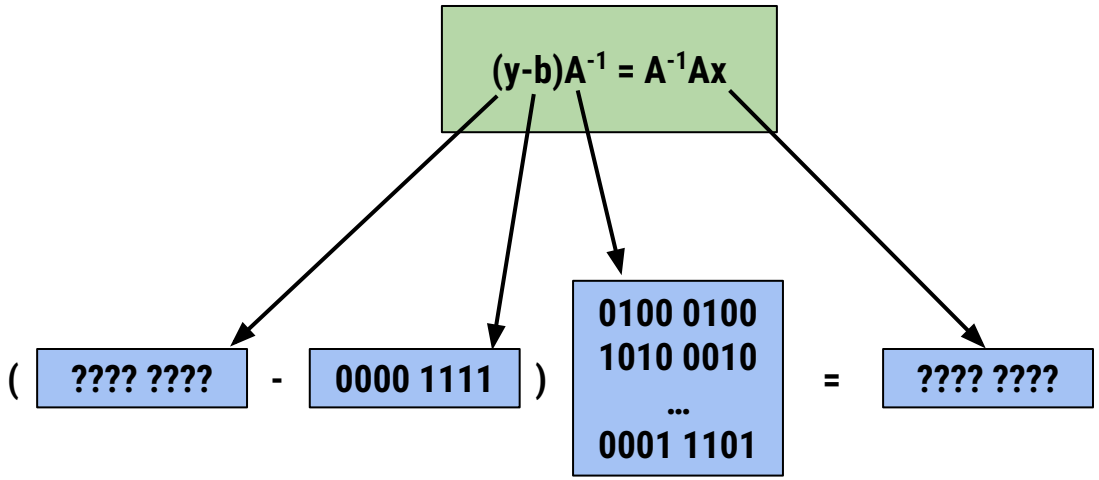
A good strategy is to set the 1's of the new identity matrix first, then use those rows to clear the ones of other rows in the same column.

Note: not every matrix has an inverse.

It's a little complicated for now...

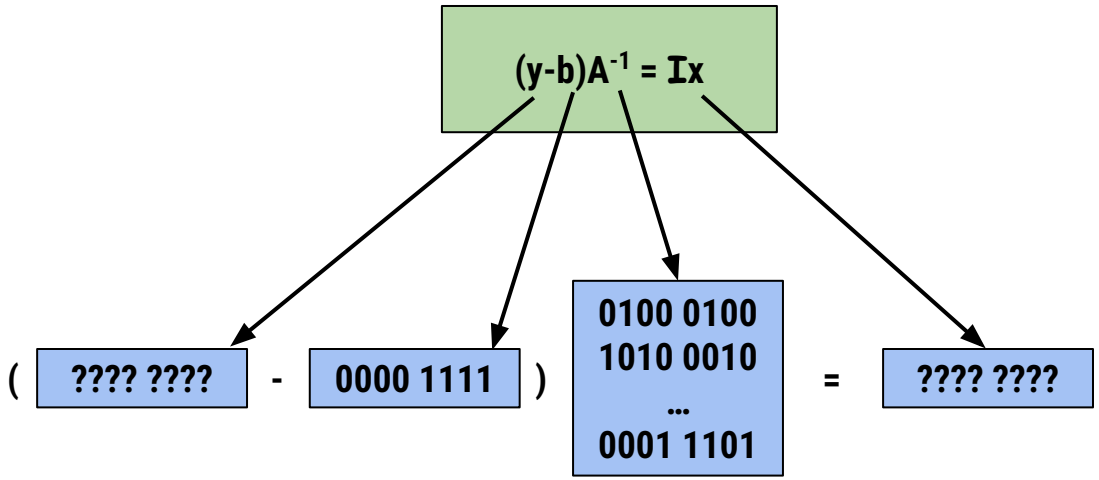
Solving for x

We multiply both sides by A^{-1}



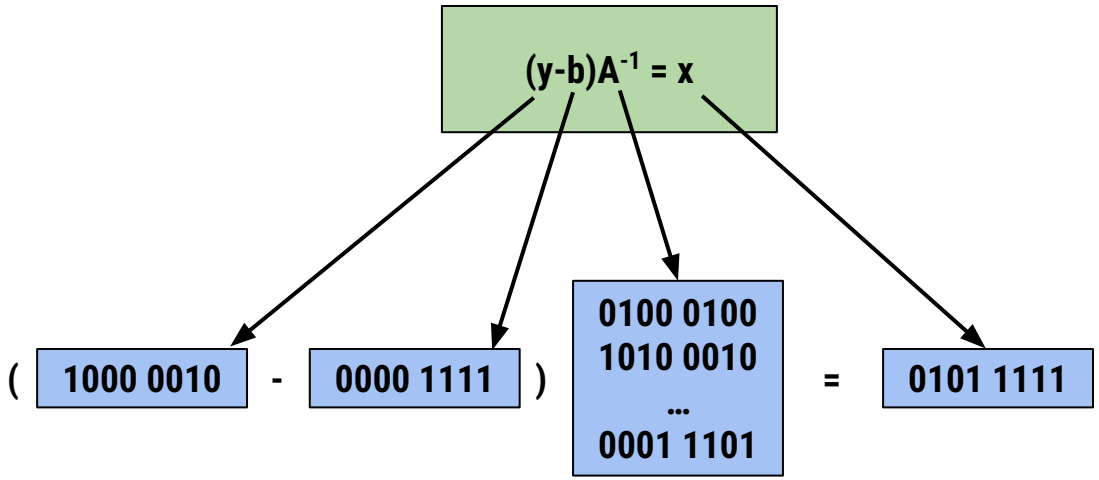
Solving for x

$A^{-1}A = I$



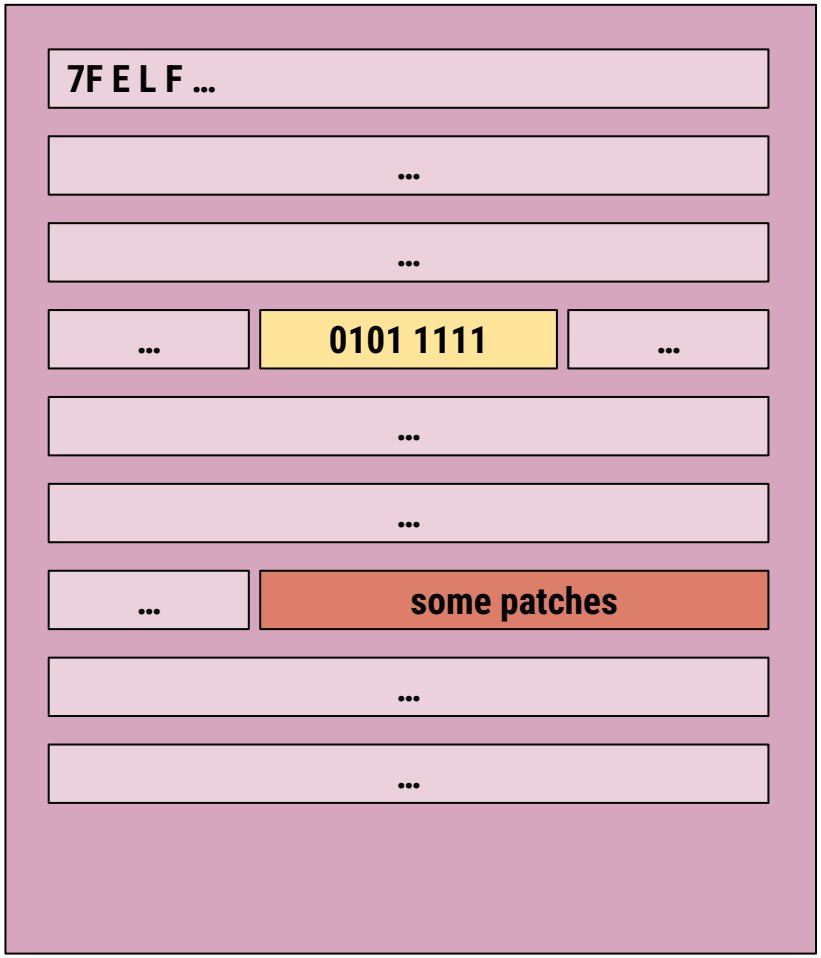
You're done.

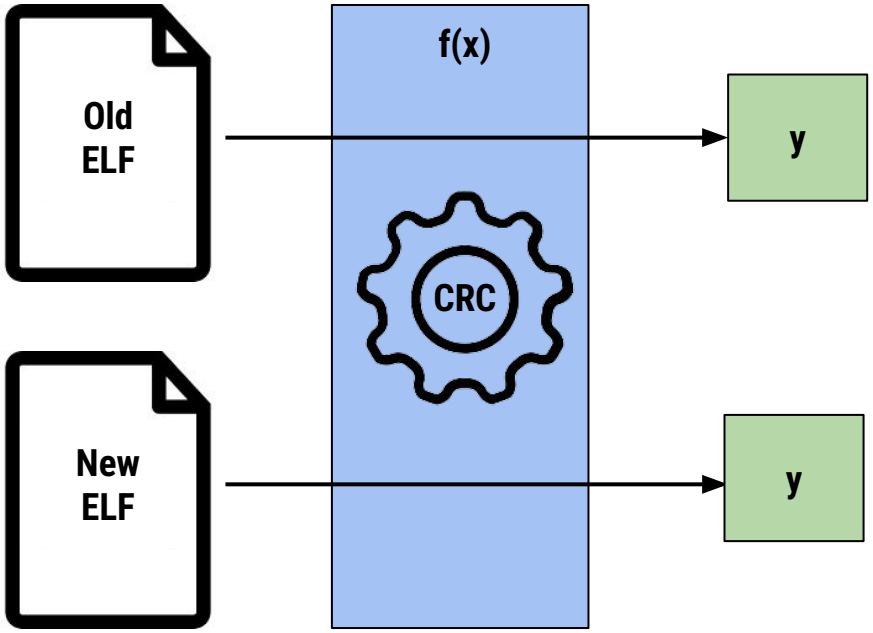
Plug in the desired y, and calculate x.



To do matrix multiplication in numpy, you can do:
`numpy.matmul(matrix1, matrix2)%2`

Our Reference Problem





Put in your value.
Profit.

Demo

Exercise

Code

```
def append_identity(matrix):
    return numpy.concatenate((
        matrix,
        numpy.identity(matrix.shape[0], dtype=int)
    ), axis=1)

def matadd(matrix1, matrix2):
    return (matrix1+matrix2)%2

def xor_rows(matrix, dst, src):
    # dst = dst ^ src
    matrix[dst] = matadd(matrix[dst], matrix[src])

def clear_over_one(matrix, start_row):
    for lower_row in range(0, start_row):
        if matrix[lower_row][start_row] == 1:
            xor_rows(matrix, lower_row, start_row)
```

Code

```
def matmul(matrix1, matrix2):
    return numpy.matmul(matrix1,matrix2)%2

def set_identity_one(matrix, start_row):
    is_set = (matrix[start_row][start_row] == 1)

    for lower_row in range(start_row+1, matrix.shape[0]):
        if matrix[lower_row][start_row] == 1:
            if not is_set:
                xor_rows(matrix, start_row, lower_row)
            is_set = True
            xor_rows(matrix, lower_row, start_row)

    if not is_set:
        print(matrix)
        raise Exception(f"no one in column {start_row} available")
```

Code

```
def gaussian_elimination_inverse(matrix):
    M = append_identity(matrix.copy())

    for row in range(matrix.shape[0]):
        set_identity_one(M, row)
        clear_over_one(M, row)

    I, Minv = numpy.hsplit(M, 2)
    assert(numpy.array_equal(
        I,
        numpy.identity(matrix.shape[0], dtype=int)
    ))

    return Minv
```