Gaussian elimination: recording the transformations

$$
\begin{bmatrix} & & \\ & M_3 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & M_2 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}
=
\begin{bmatrix} & & \\ & U_1 & \\ & & \end{bmatrix}
$$

$$
\begin{bmatrix} & & \\ & M_2 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}
=
\begin{bmatrix} & & \\ & U_2 & \\ & & \end{bmatrix}
$$

$$
\begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix}
\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}
=
\begin{bmatrix} & & \\ & U_3 & \\ & & \end{bmatrix}
$$

$$
\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix}
=
\begin{bmatrix} & & \\ & U_4 & \\ & & \end{bmatrix}
$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_1 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_2 & \\ & & \end{bmatrix}$$

$$\left(\begin{bmatrix} & & \\ & M_2 & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix} \right) \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_3 & \\ & & \end{bmatrix}$$

Gaussian elimination: recording the transformations

$$
\begin{bmatrix} \\ M_3 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ M_2 M_1 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ M_1 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ A \\ \\ \end{bmatrix}
=
\begin{bmatrix} \\ U_1 \\ \\ \end{bmatrix}
$$

$$
\begin{bmatrix} \\ M_2 M_1 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ M_1 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ A \\ \\ \end{bmatrix}
=
\begin{bmatrix} \\ U_2 \\ \\ \end{bmatrix}
$$

$$
\begin{bmatrix} \\ M_1 \\ \\ \end{bmatrix}
\begin{bmatrix} \\ A \\ \\ \end{bmatrix}
=
\begin{bmatrix} \\ U_3 \\ \\ \end{bmatrix}
$$

$$
\begin{bmatrix} \\ A \\ \\ \end{bmatrix}
=
\begin{bmatrix} \\ U_4 \\ \\ \end{bmatrix}
$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} & & \\ & M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_1 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_2 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & M_3 M_2 M_1 & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U_4 & \\ & & \end{bmatrix}$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ .5 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & 0 & 4 & -5 & -2 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

Gaussian elimination: recording the transformations

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ .5 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & 0 & 4 & -5 & -2 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

Failure of Gaussian elimination

But we compute using floating-point numbers!

$$\begin{bmatrix} 10^{-20} & 0 & 1 \\ 1 & 10^{20} & 1 \\ 0 & 1 & -1 \end{bmatrix} \Rightarrow \begin{bmatrix} 10^{-20} & 0 & 1 \\ 0 & 10^{20} & 1-10^{20} \\ 0 & 1 & -1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 10^{-20} & 0 & 1 \\ 0 & 10^{20} & -10^{20} \\ 0 & 1 & -1 \end{bmatrix} \Rightarrow \begin{bmatrix} 10^{-20} & 0 & 1 \\ 0 & 10^{20} & -10^{20} \\ 0 & 0 & 0 \end{bmatrix}$$

Gaussian elimination got the wrong answer due to round-off error.

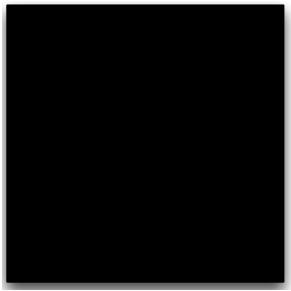
These problems can be mitigated by choosing the pivot element carefully:

- *Partial pivoting:* Among rows with nonzero entries in column c, choose row with entry having *largest* absolute value.
- *Complete pivoting:* Instead of selecting order of columns beforehand, in each iteration choose column to maximize absolute value of pivot element.

In this course, we won't study these techniques in detail.

Instead, we will use Gaussian elimination only for $GF(2)$.

The black box starts to become less opaque

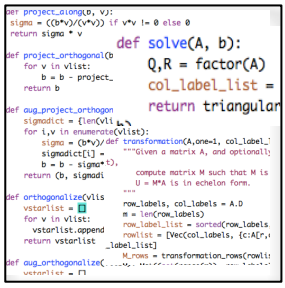


The modules `independence` and `solver` both Gaussian elimination when working over $GF(2)$:

- The procedure `solve(A, b)` computes a matrix M such that MA is in echelon form, and uses M to try to find a solution.
- The procedure `rank(L)` converts to echelon form and counts the nonzero rows to find the rank of L.

We saw that Gaussian elimination can be used to find a nonzero vector in the null space of a matrix.... You will use this in an algorithm for factoring integers.

The black box starts to become less opaque



The modules `independence` and `solver` both Gaussian elimination when working over $GF(2)$:

- The procedure `solve(A, b)` computes a matrix M such that MA is in echelon form, and uses M to try to find a solution.
- The procedure `rank(L)` converts to echelon form and counts the nonzero rows to find the rank of L.

We saw that Gaussian elimination can be used to find a nonzero vector in the null space of a matrix.... You will use this in an algorithm for factoring integers.

Other uses of Gaussian elimination over $GF(2)$

Simple examples of other uses of Gaussian elimination over $GF(2)$:

- Solving *Lights Out* puzzles.
- Attacking Python's pseudo-random-number generator:

```
>>> import random
>>> random.getrandbits(32)
1984256916
>>> random.getrandbits(32)
4135536776
>>> random.getrandbits(32)
```

What are the next thirty-two bits to be generated? Using Gaussian elimination, you can predict them accurately.

- Breaking simple authentication scheme (playing the role of Eve)....

Factoring integers

Prime Factorization Theorem: For every integer $N \geq 1$, there is a unique bag of prime numbers whose product is N.

Example:

- 75 is the product of the elements in the bag $\{3, 5, 5\}$
- 126 is the product of the elements in the bag $\{2, 3, 3, 7\}$
- 23 is the product of the elements in the bag $\{23\}$

All the elements in a bag must be prime. If N is itself prime, the bag for N is just $\{N\}$.

"The problem of distinguishing prime numbers from composite numbers and of **resolving the latter into their prime factors** is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length Further, the dignity of the science itself seems to require solution of a problem so elegant and so celebrated."

Carl Friedrich Gauss, Disquisitiones Arithmeticae, 1801

Factoring integers

Prime Factorization Theorem: For every integer $N \geq 1$, there is a unique bag of prime numbers whose product is N.

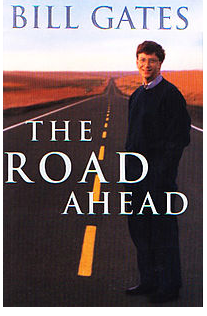
Example:

- 75 is the product of the elements in the bag $\{3, 5, 5\}$
- 126 is the product of the elements in the bag $\{2, 3, 3, 7\}$
- 23 is the product of the elements in the bag $\{23\}$

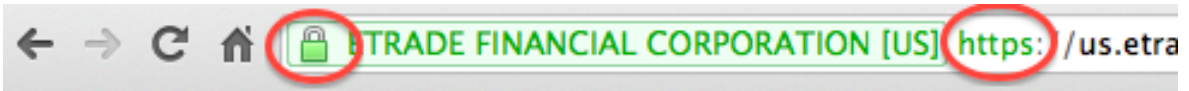
All the elements in a bag must be prime. If N is itself prime, the bag for N is just $\{N\}$.

"Because both the system's privacy and the security of digital money depend on encryption, a breakthrough in mathematics or computer science that defeats the cryptographic system could be a disaster. The obvious mathematical breakthrough would be the development of an easy way to factor large prime numbers."

(Bill Gates, The Road Ahead, *1995).*



Secure Sockets Layer



Secure communication with websites uses HTTPS (Secure HTTP)

which is based on SSL (Secure Sockets Layer)

which is based on the RSA (Rivest-Shamir-Adelman) cryptosystem

which depends on the computational difficulty of factoring integers

Factoring integers

Testing whether a number is prime is now well-understood and easy.

Here's a one-line Python script that gives false positives when input is a Carmichael number (rare) and otherwise with probability $\frac{1}{2^{20}}$:

```
def is_prime(p, n=20): return all([pow(randint(1,p-1),p-1,p) == 1
                                   for i in range(n)])
```

With a few more lines, can get correct answers for Carmichael numbers as well.

The hard part of factoring seems to be this: given an integer N, find any *nontrivial* divisor (divisor other than 1 and N).

If you can do that reliably, you can factor.

Factoring integers the naive way

```
def factor(N):
  for d in range(2, N-1):
    if N % d == 0: return d
```

If d is a divisor of N then so is N/d.

$\min\{d, N/d\} \leq \sqrt{N}$

This shows that it suffices to search among $2, 3, \ldots, \text{int}(\sqrt{N})$

```
def factor(N):
  for d in range(2, intsqrt(N)):
    if N % d == 0: return d
```

where `intsqrt(N)` is a procedure I provide

Useful subroutine: `gcd(m,n)`

`gcd(m,n)` return the greatest common divisor of positive integers m and n. This algorithm is attributed to Euclid, and it is very fast. Here's the code:

```
def gcd(x,y): return x if y == 0 else gcd(y, x % y)
```

Example:

- gcd(12, 16) is 4
- gcd(276534813447635747652, 333070702552660863114) is 18172055646

Using square roots to factor N

Find integers a and b such that

$$a^2 - b^2 = N$$

for then

$$(a - b)(a + b) = N$$

so $a - b$ and $a + b$ are divisors (ideally nontrivial)

How to find such integers? Naive approach...

- Choose integer a slightly more than $\sqrt{N}$
- Check if $\sqrt{a^2 - N}$ is an integer.
- If so, let $b = \sqrt{a^2 - N}$ Success! ☺
  Now $a - b$ is a divisor of N
- If not, repeat with another value for a

Example: $N = 77$

$a = 9$
$\sqrt{a^2 - N} = \sqrt{4} = 2$
so let $b = 2$
$a - b = 7$ is a divisor of N

Example: $N = 23 \cdot 41$
$a = 31 \Rightarrow a^2 - N = 18$ ☹
$a = 32 \Rightarrow a^2 - N = 81$ ☺

For large N, it takes too long to find a good integer a. ☹
We will show how **linear algebra** ☺ helps us synthesize a good integer a.

Using square roots to factor N

Find a and b such that

$$a^2 - b^2 = kN$$

for some integer k. Then

$$(a-b)(a+b) = kN$$

$\{\text{prime factors of } a-b\} \cup \{\text{prime factors of } a+b\} = \{\text{prime factors of } k\} \cup \{\text{prime factors of } N\}$

Suppose $\{\text{prime factors of } N\} = \{p, q\}$.

If

- p and q are both factors of $a-b$, or
- p and q are both factors of $a+b$

then $\gcd(a-b, N)$ will not find a nontrivial divisor.
However, if

- p is a factor of $a-b$ and q is a factor of $a+b$, or
- p is a factor of $a+b$ and q is a factor of $a-b$

then $\gcd(a-b, N)$ **will** find a nontrivial divisor.

Example:: $N = 7 \cdot 11$
$k = 2 \cdot 3 \cdot 5 \cdot 13$

if $a-b = 2 \cdot 7 \cdot 11$ and $a+b = 3 \cdot 5 \cdot 13$
then $\gcd(a-b, N) = N$ ☹

if $a-b = 2 \cdot 5 \cdot 11$ and
$a+b = 3 \cdot 7 \cdot 13$

then $\gcd(a-b, N) = 11$ ☺

How to find integers a, b such that $a^2 - b^2 = kN$

Idea: Start by finding the first thousand prime numbers $p_1, \ldots, p_{1000}$.

- Choose a
- Compute $a^2 - N$.
- See if $a^2 - N$ can be factored using only $p_1, \ldots, p_{1000}$
- If not, throw it away.
- If so, record a and the factorization of $a^2 - N$

Repeat a thousand and one times

a	$a * a - N$	factorization
51	182	$2 \cdot 7 \cdot 13$
52	285	$3 \cdot 5 \cdot 19$
53	390	$2 \cdot 3 \cdot 5 \cdot 13$
58	945	$3^3 \cdot 5 \cdot 7$
61	1302	$2 \cdot 3 \cdot 7 \cdot 31$
62	1425	$3 \cdot 5^2 \cdot 19$
63	1550	$2 \cdot 5^2 \cdot 31$
67	2070	$2 \cdot 3^2 \cdot 5 \cdot 23$

Now we want to find a subset $\{a_1, \ldots, a_k\}$ such that $(a_1^2 - N) \cdots (a_k^2 - N)$ is a perfect square.

Combine $a_1 = 52$, $a_2 = 67$, $a_3 = 71$

$(a_1^2 - N)(a_2^2 - N)(a_3^2 - N) = (3 \cdot 5 \cdot 19)(2 \cdot 3^2 \cdot 5 \cdot 23)(2 \cdot 3 \cdot 19 \cdot 23)$
$= 2^2 \cdot 3^4 \cdot 5^2 \cdot 19^2 \cdot 23^2 = (2 \cdot 3^2 \cdot 5 \cdot 19 \cdot 23)^2$

How to find a subset that works?

Finding a subset that works

Represent each factorization as a vector over $GF(2)$:

Represent $p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ by $\{p_1 : (a_1 \ \% \ 2), p_2 : (a_2 \ \% \ 2) \ldots, p_k : (a_k \ \% \ 2)\}$

Let $A =$ matrix whose rows are these vectors.

A subset of factorizations whose product is a perfect square $=$ a subset of A's rows whose sum is the zero vector

Therefore need to find a nonzero vector in $\{\mathbf{u} \ : \ \mathbf{u} * A = \mathbf{0}\}$

If number of rows $>$ rank of matrix then there exists such a nonzero vector.

a	$a * a - N$	factorization	vector.f
51	182	$2 \cdot 7 \cdot 13$	$\{2 : \text{one}, 13 : \text{one}, 7 : \text{one}\}$
52	285	$3 \cdot 5 \cdot 19$	$\{19 : \text{one}, 3 : \text{one}, 5 : \text{one}\}$
53	390	$2 \cdot 3 \cdot 5 \cdot 13$	$\{2 : \text{one}, 3 : \text{one}, 5 : \text{one}, 13 : \text{one}\}$
58	945	$3^3 \cdot 5 \cdot 7$	$\{3 : \text{one}, 5 : \text{one}, 7 : \text{one}\}$
61	1302	$2 \cdot 3 \cdot 7 \cdot 13$	$\{31 : \text{one}, 2 : \text{one}, 3 : \text{one}, 7 : \text{one}\}$
62	1425	$3 \cdot 5^2 \cdot 19$	$\{19 : \text{one}, 3 : \text{one}, 5 : 0\}$
63	1550	$2 \cdot 5^2 \cdot 31$	$\{2 : \text{one}, 5 : 0, 31 : \text{one}\}$
67	2070	$2 \cdot 3^2 \cdot 5 \cdot 23$	$\{2 : \text{one}, 3 : 0, 5 : \text{one}, 23 : \text{one}\}$
68	2205	$3^2 \cdot 5 \cdot 7^2$	$\{3 : 0, 5 : \text{one}, 7 : 0\}$

Improving on the simple authentication scheme

- Password is an n-vector $\hat{\mathbf{x}}$ over $GF(2)$
- **Challenge:** Computer sends random n-vector $\mathbf{a}$
- **Response:** Human sends back $\mathbf{a} \cdot \hat{\mathbf{x}}$.

Repeat until Computer is convinced that Human knows password $\hat{\mathbf{x}}$.

Eve eavesdrops on communication,
learns m pairs $\mathbf{a}_1, b_1, \ldots, \mathbf{a}_m, b_m$
such that b_i is right response to challenge $\mathbf{a}_i$

The password $\hat{\mathbf{x}}$ is a solution to

$$\underbrace{\begin{bmatrix} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix}}_{A} \begin{bmatrix} \mathbf{x} \end{bmatrix} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}}_{\mathbf{b}}$$

Once rank A reaches n, the solution is unique, and Eve can use Gaussian elimination to find it.

Making the scheme more secure:
The way to make the scheme more secure is to introduce mistakes.

- In about $1/6$ of the rounds, randomly, Human sends the *wrong* dot-product.
- Computer is convinced if Human gets the right answers 75% of the time.

Even if Eve knows that Human is making mistakes, she doesn't know which rounds involve mistakes.

Gaussian elimination does not find the solution when some of the right-hand side values b_i are wrong.

In fact, we don't know *any* efficient algorithm Eve can use to find the solution.

Threshold secret-sharing

All-or-nothing secret-sharing is a method to split the secret into two pieces so that both are required to recover the secret.

We could generalize to split the secret among four teaching assistants (TAs), so that jointly they could recover the secret but any three cannot.

However, it is risky to rely on all four TAs showing up for a meeting.

Instead we want a *threshold* secret-sharing scheme: share a secret among four TAs so that

- any three TAs could jointly recover the secret, but
- any two TAs could not.

There are such schemes that use fields other than $GF(2)$, but let's see if we can do it using $GF(2)$.

Threshold secret-sharing using five 3-vectors over $GF(2)$

Failing attempt: Here's an idea: select five 3-vectors over $GF(2)$ $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$.

These vectors are supposed to satisfy the following requirement:

Requirement: every set of three are linearly independent.

To share a one-bit secret s among the TAs, I randomly select a 3-vector $\mathbf{u}$ such that $\mathbf{a}_0 \cdot \mathbf{u} = s$. I keep $\mathbf{u}$ secret, but I compute the other dot-products:

$$\begin{aligned}\beta_1 &= \mathbf{a}_1 \cdot \mathbf{u}\\ \beta_2 &= \mathbf{a}_2 \cdot \mathbf{u}\\ \beta_3 &= \mathbf{a}_3 \cdot \mathbf{u}\\ \beta_4 &= \mathbf{a}_4 \cdot \mathbf{u}\end{aligned}$$

I give the bit β_1 to TA 1, I give β_2 to TA 2, I give β_3 to TA 3, and I give β_4 to TA 4.

Can any three TAs recover the secret?
For example, suppose TAs 1, 2, and 3 want to recover the secret. They solve the matrix-vector equation

$$\left[\begin{array}{c}\mathbf{a}_1\\ \hline \mathbf{a}_2\\ \hline \mathbf{a}_3\end{array}\right]\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} = \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}$$

Since the matrix is square and the rows $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ are linearly independent, the matrix is invertible, so $\mathbf{u}$ is the only solution. The TAs use `solve` to recover $\mathbf{u}$, and take the

Is the secret safe?

Now suppose two rogue TAs, TA 1 and TA 2, decide they want to obtain the secret without involving either of the other TAs. They know β_1 and β_2. Can they use these to get the secret s? The answer is no: their information is consistent with both $s = 0$ and $s = 1$. Since the matrix

$$\left[\begin{array}{c} \mathbf{a}_0 \\ \hline \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right]$$

is invertible, each of the two matrix equations

$$\left[\begin{array}{c} \mathbf{a}_0 \\ \hline \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$\left[\begin{array}{c} \mathbf{a}_0 \\ \hline \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

has a solution. The solution to the first equation is a vector $\mathbf{v}$ such that $\mathbf{a}_0 \cdot \mathbf{v} = 0$, and the solution to the second equation is a vector $\mathbf{v}$ such that $\mathbf{a}_0 \cdot \mathbf{v} = 1$.

Threshold secret-sharing with five pairs of 6-vectors

The proposed scheme seems to work. The catch is that first step:

- Select five 3-vectors over $GF(2)$ $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$ satisfying

Requirement: every set of three are linearly independent.

Unfortunately, there are no such five vectors.

Instead, we seek ten 6-vectors $\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2, \mathbf{a}_3, \mathbf{b}_3, \mathbf{a}_4, \mathbf{b}_4$ over $GF(2)$.

We think of them as forming five pairs:

- Pair 0 consists of $\mathbf{a}_0$ and $\mathbf{b}_0$,
- Pair 1 consists of $\mathbf{a}_1$ and $\mathbf{b}_1$,
- Pair 2 consists of $\mathbf{a}_2$ and $\mathbf{b}_2$, and
- Pair 3 consists of $\mathbf{a}_3$ and $\mathbf{b}_3$.
- Pair 4 consists of $\mathbf{a}_4$ and $\mathbf{b}_4$.

The requirement is as follows:

Requirement: For any three pairs, corresponding six vectors are linearly indep.

To share two bits s and t:

- I choose a secret 6-vector $\mathbf{u}$ such that $\mathbf{a}_0 \cdot \mathbf{u} = s$ and $\mathbf{b}_0 \cdot \mathbf{u} = t$.
- I give TA 1 the two bits $\beta_1 = \mathbf{a}_1 \cdot \mathbf{u}$ and $\gamma_1 = \mathbf{b}_1 \cdot \mathbf{u}$, I give TA 2 the two bits $\beta_2 = \mathbf{a}_2 \cdot \mathbf{u}$ and $\gamma_2 = \mathbf{b}_2 \cdot \mathbf{u}$, and so on.

Each TA's share consists of a pair of bits.

Threshold secret-sharing with five pairs of 6-vectors: recoverability

Any three TAs jointly can solve a matrix-vector equation with a 6×6 matrix to obtain $\mathbf{u}$, whence they can obtain the secret bits s and t. Suppose, for example, TAs 1, 2, and 3 came together. Then they would solve the equation

$$\left[\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{b}_1 \\ \hline \mathbf{a}_2 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{a}_3 \\ \hline \mathbf{b}_3 \end{array}\right] \left[\begin{array}{c} \\ \\ \\ \mathbf{x} \\ \\ \\ \\ \end{array}\right] = \left[\begin{array}{c} \beta_1 \\ \gamma_1 \\ \beta_2 \\ \gamma_2 \\ \beta_3 \\ \gamma_3 \end{array}\right]$$

to obtain $\mathbf{u}$ and thereby obtain the secret bits. Since the vectors $\mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2, \mathbf{a}_3, \mathbf{b}_3$ are linearly independent, the matrix is invertible, so there is a unique solution to this equation.

Threshold secret-sharing with five pairs of 6-vectors: recoverability

Suppose TAs 1 and 2 go rogue and try to recover s and t. They possess the bits $\beta_1, \gamma_1, \beta_2, \gamma_2$. Are these bits consistent with $s = 0$ and $t = 1$? They are if there is a vector $\mathbf{u}$ that solves the equation

$$\left[\begin{array}{c} \mathbf{a}_0 \\ \hline \mathbf{b}_0 \\ \hline \mathbf{a}_1 \\ \hline \mathbf{b}_1 \\ \hline \mathbf{a}_2 \\ \hline \mathbf{b}_2 \end{array}\right] \left[\begin{array}{c} \\ \\ \mathbf{x} \\ \\ \\ \\ \end{array}\right] = \left[\begin{array}{c} 0 \\ 1 \\ \beta_1 \\ \gamma_1 \\ \beta_2 \\ \gamma_2 \end{array}\right]$$

where the first two entries of the right-hand side are the guessed values of s and t.

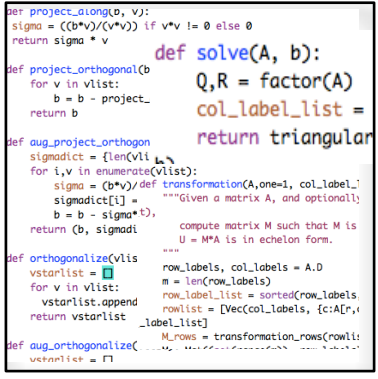
Since the vectors $\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2$ are linearly independent, the matrix is invertible, so a solution exists.

Similarly, no matter what you put in the first two entries of the right-hand side, there is exactly one solution.

This shows that the shares of TAs 1 and 2 tell them nothing about the true values of s and t. The secret is safe.

[8] The Inner Product

Continuing to look inside the black box



We studied Gaussian elimination, which is used in modules `solver` and `independence` when working over $GF(2)$.

We next study the methods used in these modules when working over $\mathbb{R}$.

Fire Engine problem

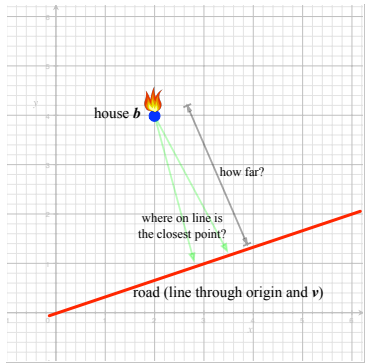
There is a burning house located at coordinates $[2, 4]$!

A street runs near the house, along the line through the origin and through $[6, 2]$—but it is near enough?

Fire engine has a hose 3.5 units long.

If we can navigate the fire engine to the point on the line nearest the house, will the distance be small enough to save the house?

We're faced with two questions: what point along the line is closest to the house, and how far is it?



What do we mean by *closest*?

Distance, length, norm, inner product

We will define the distance between two vectors $\mathbf{p}$ and $\mathbf{b}$ to be the length of the difference $\mathbf{p} - \mathbf{b}$.

This means that we must define the length of a vector.

Instead of using the term "length" for vectors, we typically use the term *norm*.

The norm of a vector $\mathbf{v}$ is written $\|\mathbf{v}\|$

Since it plays the role of length, it should satisfy the following *norm properties*:

Property N1 $\|\mathbf{v}\|$ is a nonnegative real number.

Property N2 $\|\mathbf{v}\|$ is zero if and only if $\mathbf{v}$ is a zero vector.

Property N3 for any scalar α, $\|\alpha\,\mathbf{v}\| = |\alpha|\|\mathbf{v}\|$.

Property N4 $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (triangle inequality).

One way to define vector norm is to define an operation on vectors called *inner product*.

Inner product of vectors $\mathbf{u}$ and $\mathbf{v}$ is written

$$\langle \mathbf{u}, \mathbf{v} \rangle$$

The inner product must satisfy certain axioms, which we outline later.

No way to define inner product for $GF(2)$ so no more $GF(2)$ ☹

From inner product to norm

For the real numbers and complex numbers, we have some flexibility in defining the inner product.

This flexibility is used heavily, e.g. in Machine Learning.

Once we have defined an inner product, the norm of a vector $\mathbf{u}$ is defined by

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

For simplicity, we will focus on $\mathbb{R}$ and will use the most natural and convenient definition of inner product.

This definition leads to the norm of a vector being the geometric length of its arrow.

For vectors over $\mathbb{R}$, we define our inner product as the dot-product:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v}$$

Properties of inner product of vectors over $\mathbb{R}$

- *linearity in the first argument:* $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$
- *symmetry:* $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$
- *homogeneity:* $\langle \alpha \mathbf{u}, \mathbf{v} \rangle = \alpha \langle \mathbf{u}, \mathbf{v} \rangle$

For inner product = dot-product, can easily prove these properties.

From inner product to norm

We have defined $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v}$ and $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$ Does this norm satisfy the *norm properties?*

Property N1: $||\mathbf{v}||$ is a nonnegative real number.

Property N2: $||\mathbf{v}||$ is zero if and only if $\mathbf{v}$ is a zero vector.

Property N3: for any scalar α, $||\alpha\,\mathbf{v}|| = |\alpha|||\mathbf{v}||$.

Property N4: $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (triangle inequality).

Write $\mathbf{v} = [v_1, v_2, \ldots, v_n]$. Then $\|[v_1, v_2, \ldots, v_n]\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$

1. Sum of squares is a nonnegative real number so $\|\mathbf{v}\|$ is nonnegative real number.
2. If any entry v_i of $\mathbf{v}$ is nonzero then sum of squares is nonzero, so norm is nonzero.
3. Proof of third property:

$$
\begin{aligned}
||\alpha\,\mathbf{v}||^2 &= \langle \alpha\,\mathbf{v}, \alpha\,\mathbf{v} \rangle && \text{by definition of norm} \\
&= \alpha\,\langle \mathbf{v}, \alpha\,\mathbf{v} \rangle && \text{by homogeneity of inner product} \\
&= \alpha\,(\alpha\,\langle \mathbf{v}, \mathbf{v} \rangle) && \text{by symmetry and homogeneity (again) of inner product} \\
&= \alpha^2 ||\mathbf{v}||^2 && \text{by definition of norm}
\end{aligned}
$$

Thus $||\alpha\,\mathbf{v}|| = \alpha\,||\mathbf{v}||$.

We skip the proof of fourth property.

Norm is geometric length of arrow

Example: What is the length of the vector $\mathbf{u} = [u_1, u_2]$?

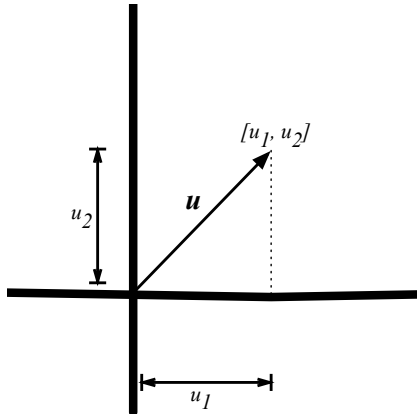
Remember the *Pythagorean Theorem*:

> *for a right triangle with side-lengths a, b, c, where c is the length of the hypotenuse,*
>
> $$a^2 + b^2 = c^2$$

We can use this equation to calculate the length of $\mathbf{u}$:

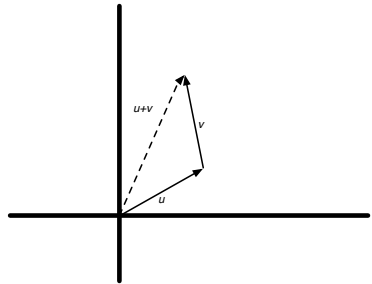
$$(\text{length of } \mathbf{u})^2 = u_1^2 + u_2^2$$

So this notion of length agrees with the one we learned in grade school, at least for vectors in $\mathbb{R}^2$.



Orthogonality

Orthogonal is linear-algebra-ese for *perpendicular*.
We'll define it so as to make Pythagorean Theorem true.
Let $\mathbf{u}$ and $\mathbf{v}$ be vectors. Their lengths are $||\mathbf{u}||$ and $||\mathbf{v}||$.
Draw the corresponding arrows, and the arrow for $\mathbf{u}+\mathbf{v}$
The arrow for $\mathbf{u}+\mathbf{v}$ is the "hypotenuse". (The triangle is not necessarily a right angle.)
The squared length of the vector $\mathbf{u}+\mathbf{v}$ (the "hypotenuse") is



$$
\begin{aligned}
||\mathbf{u}+\mathbf{v}||^2 &= \langle \mathbf{u}+\mathbf{v}, \mathbf{u}+\mathbf{v} \rangle \\
&= \langle \mathbf{u}, \mathbf{u}+\mathbf{v} \rangle + \langle \mathbf{v}, \mathbf{u}+\mathbf{v} \rangle && \text{by linearity of inner product in } 1^{st} \text{ argument} \\
&= \langle \mathbf{u}, \mathbf{u} \rangle + \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{v}, \mathbf{u} \rangle + \langle \mathbf{v}, \mathbf{v} \rangle && \text{by symmetry and linearity} \\
&= ||\mathbf{u}||^2 + 2\langle \mathbf{u}, \mathbf{v} \rangle + ||\mathbf{v}||^2 && \text{by symmetry}
\end{aligned}
$$

The last expression is $||\mathbf{u}||^2 + ||\mathbf{v}||^2$ if and only if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.
We therefore define $\mathbf{u}$ and $\mathbf{v}$ to be *orthogonal* if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.
Pythagorean Theorem for vectors: if vectors $\mathbf{u}$ and $\mathbf{v}$ over the reals are orthogonal then $||\mathbf{u}+\mathbf{v}||^2 = ||\mathbf{u}||^2 + ||\mathbf{v}||^2$.

Properties of orthogonality

To solve the Fire Engine Problem, we will use the Pythagorean Theorem in conjunction with the following simple observations:

Orthogonality Properties:

Property O1: If $\mathbf{u}$ is orthogonal to $\mathbf{v}$ then $\mathbf{u}$ is orthogonal to $\alpha\,\mathbf{v}$ for every scalar α.

Property O2: If $\mathbf{u}$ and $\mathbf{v}$ are both orthogonal to $\mathbf{w}$ then $\mathbf{u}+\mathbf{v}$ is orthogonal to $\mathbf{w}$.

Proof:

1. $\langle \mathbf{u}, \alpha\,\mathbf{v}\rangle = \alpha\,\langle \mathbf{u}, \mathbf{v}\rangle = \alpha\,0 = 0$
2. $\langle \mathbf{u}+\mathbf{v}, \mathbf{w}\rangle = \langle \mathbf{u}, \mathbf{w}\rangle + \langle \mathbf{v}, \mathbf{w}\rangle = 0 + 0$

Example: $[1,2]\cdot[2,-1] = 0$ so $[1,2]\cdot[20,-10] = 0$

Example:

$$\begin{array}{rcl} [1,2,1]\cdot[1,-1,1] & = & 0 \\ [0,1,1]\cdot[1,-1,1] & = & 0 \\ \hline ([1,2,1]+[0,1,1])\cdot[1,-1,1] & = & 0 \end{array}$$