program for Hill cipher succumbs to a known plaintext attack if sufficient plaintext ciphertext pairs are provided. It is even easier to solve the Hill cipher if a chosen plaintext attack can be mounted. Implement in C programming.

import random

def egcd(a,b):

if b==0: return (a,1,0)

g,x1,y1=egcd(b,a%b)

return (g,y1,x1-(a//b)\*y1)

def modinv(a,m):

g,x,y=egcd(a,m)

if g!=1: return None

return x % m

def det2(mat):

return (mat[0]\*mat[3]-mat[1]\*mat[2])%26

def inv2\_mod26(mat):

d=det2(mat)

invd=modinv(d,26)

if invd is None: return None

inv=[mat[3]\*invd%26, (-mat[1])\*invd%26, (-mat[2])\*invd%26, mat[0]\*invd%26]

return inv

def mat\_mult2(A,B):

return [(A[0]\*B[0]+A[1]\*B[2])%26, (A[0]\*B[1]+A[1]\*B[3])%26, (A[2]\*B[0]+A[3]\*B[2])%26, (A[2]\*B[1]+A[3]\*B[3])%26]

def mat\_vec\_mult2(mat,vec):

return [(mat[0]\*vec[0]+mat[1]\*vec[1])%26, (mat[2]\*vec[0]+mat[3]\*vec[1])%26]

def text\_to\_nums(s):

s=''.join(c for c in s.upper() if c.isalpha())

if len(s)%2: s+='X'

return [ord(c)-65 for c in s]

def nums\_to\_text(nums):

return ''.join(chr(n+65) for n in nums)

def encrypt\_hill(pt\_nums,key\_mat):

out=[]

for i in range(0,len(pt\_nums),2):

v=[pt\_nums[i],pt\_nums[i+1]]

c=mat\_vec\_mult2(key\_mat,v)

out+=c

return out

def decrypt\_hill(ct\_nums,key\_mat):

inv=inv2\_mod26(key\_mat)

out=[]

for i in range(0,len(ct\_nums),2):

v=[ct\_nums[i],ct\_nums[i+1]]

p=mat\_vec\_mult2(inv,v)

out+=p

return out

key=None

while True:

k=[random.randint(0,25) for \_ in range(4)]

if det2(k)%2!=0 and egcd(det2(k),26)[0]==1:

key=k

break

plaintext="HELPTHIS"

pt\_nums=text\_to\_nums(plaintext)

ct\_nums=encrypt\_hill(pt\_nums,key)

print("Actual key matrix:", key)

print("Plaintext:", plaintext)

print("Ciphertext numbers:", ct\_nums)

p0=[pt\_nums[0],pt\_nums[2]]

p1=[pt\_nums[1],pt\_nums[3]]

P=[p0[0],p1[0],p0[1],p1[1]]

C=[ct\_nums[0],ct\_nums[2],ct\_nums[1],ct\_nums[3]]

Pinv=inv2\_mod26(P)

if Pinv is None:

print("Selected plaintext pairs not invertible; attack needs different pairs.")

else:

recovered=mat\_mult2(C,Pinv)

print("Recovered key matrix:", recovered)

decrypted=nums\_to\_text(decrypt\_hill(ct\_nums,recovered))

print("Decrypted with recovered key:", decrypted) 